UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Designing and presenting digital nudges on mobile phones

Building an app based on system requirements and usability heuristics

David Kristoffersen

INF-3990 Master's Thesis in Computer Science, December 2022

To Me.

For reaching the end of the tunnel

“Bad programmers worry about the code.
Good programmers worry about data structures and their relationships.”
–Linus Torvalds

“A good programmer is someone who always looks both ways before crossing
a one-way street.”
–Doug Linder

Abstract

The environment is progressively affected by global warming and pollution, whereas fossil fuel transportation is one of the major causes[1]. This thesis describes a system that aims to support users in choosing environmentally friendly transportation alternatives. The system uses digital nudging to motivate behavioral change in a non-intrusive manner.

This project focuses on the presentation of nudging in a mobile environment. Mobile applications reside in a complex environment with many constraints and limitations. The applications also communicate and influence the end users based on architectural and front-end components. Such applications should thus follow strict guidelines to ensure a robust, extendable, and reusable foundation. Furthermore, the applications should utilize various techniques based on psychological effects and user experience (UX) principles to stay competitive in the current market.

This project presents a selection of psychological requirements designed for nudging. Additionally, the project creates a novel set of usability heuristics designed for nudging. The project implements an Android app based on the requirements and heuristics. The app lays the foundation for future extensions of front-end designs and nudging.

Acknowledgements

During the writing of this thesis, several people have given me support. Though there are a select few I would like especially thank. For their support, guidance, inspiration, and for giving me hope during this long project. Thanks to your support, I have finally completed my master's thesis.

To my supervisor, Prof. Randi Karlsen, for guiding me throughout this project. What stood out to me was your positivity and empathy. Sometimes I thought I did not make enough progress, feeling ashamed and dreading the next meeting. But you always turned it around, making me feel optimistic and relieved. After all my meetings, I felt you wanted me to succeed at my own pace. Thank you for being patient with me, giving me advice, and being my supervisor.

To my institute adviser, Jan Fuglesteg, for always being a reliable support during the study. Some issues during the study greatly help with having a person like Fuglesteg. A person that is always there and easy to talk to. A person that is passionate and genuinely wants to help the students. Thank you for all the help during my entire study.

To Simon, for being a good friend and my inspiration as a student. At the start of our study, we somehow got to know each other through a study group. It was a start, but it quickly turned to us working closer together. I remember you always bugging me with new tech and wanting us to try it out. Jumping on Vim and Linux in the first semester was quite a ride. Your genuine interest and love for tech were truly inspiring, and they spread over to me. So thank you for being my friend and inspiration over these years.

To my mom and dad, Elisabeth and Bernt-Harald, for always being there and supporting me during good and challenging times. The university has been tough from the start, and I never felt I had an advantage over others. I had to work hard for it and sacrifice, and sometimes the pressure was almost overwhelming. Regardless of how I felt, though, I always had your continuous and loving support. You have always been there to listen and give advice, to comport when in need, and to congratulate my accomplishments. University really has been a rollercoaster. Without you as my parents, I cannot imagine

getting as far as I did, and for that, I am truly grateful. Thank you for always being there for me and taking care of me.

And finally, to my siblings, friends, and others that have supported me. Thank you, whether minor or a lot; I appreciate all your support.

Contents

List of Figures

List of Tables

Glossary

anthropogenic caused or produced by humans

baseline grid a 4dp to 8dp grid

color role a color tone associated with a component or an element

color scheme a specification of color tones

component a template for future use

container an enclosed area containing various content

element a text, an icon, or an image

icon theme a specification of stroke and fill, cornen radius and color of icons

Material Shorthand for Material Design

Material Design Android's default desgin system

padding space between components

shape scheme a specification of the angle and curvature of corners and edges

surface a flat plain that illustrates elevation with light and shadow

theme a specification of the color and shape of components and elements

type scale a specification of type attributes and application

List of Abbreviations

dp density-independent pixel

GHG greenhouse gas

IPCC Intergovernmental Panel on Climate Change

ODS Open Distributed Systems

PPI pixels per inch

sp scalable pixel

UI user interface

UX user experience

1 Introduction

Climate change is affecting the globe and has worsened dramatically over recent years. The increase in global temperature is one of the main reasons for the changes in the climate. The temperature has had periods with a decrease, such as from 1998 - 2008. However, the last century has had an overall increase in temperature[2]. In 1990 the Intergovernmental Panel on Climate Change (IPCC) released a report together with the World Meteorological Organization and United Nations Environment Programme that stated:

> . . . most of the observed increase in global average temperature since the mid 20th century is very likely due to the observed increase in anthropogenic greenhouse gas concentrations.[3]

Quote 1.1

Anthropogenic climate change (ACC) is the theory that humans are the leading cause of climate change. Data from the International Energy Agency (IEA) shows that fossil fuels account for approximately 62% of global greenhouse gas (GhG) emissions[1]. Furthermore, oil and gas account for about 60% of fossil fuel emissions[1].

Data from the Mitigation of Climate report from the IPCC shows that the transport sector accounts for about 14% of global GhG emissions[4]. Transportation methods such as cars, buses, and airplanes are accountable for this emission. The IPCC report further specifies that road transportation accounts for 72%

of emissions within the transport sector[5]. Transportation methods such as cars and buses are thus the first emission sources to reduce to have notable changes.

1.1 Motivation

As climate changes worsen, the incentives to reduce GhG emissions are exponentially expanding. The individual cannot contribute much to public and business-related contributions. However, individual contributions by larger groups of people can amount to more notable reduction rates. Private and collective transportation methods produce varying levels of GhG emissions. Many circumstances decide the pollution level, such as road quality, road route, and the number of passengers. In cities, however, private transportation such as cars usually pollutes more than collective alternatives such as buses, trains, and trams. Similarly, walking and biking produce no pollution, thus considered the greenest alternative.

Groups of people can reduce individual GhG emissions by switching the accustomed transportation methods to a greener alternative. Most people would need an external push to change habits. However, the push cannot force people to choose; instead, the push must be a non-intrusive motivation, also known as a nudge.

1.2 Nudging

This research contributes to a project revolving around nudging. Nudging is a method to influence individuals' choices in a non-intrusive manner. Thus, nudging is a tool used to achieve behavioral change. Digital environments can utilize nudging through the use of digital nudges.

1.3 Goal

This thesis explores the implementation of digital nudges utilizing a user experience (UX) environment. The research contributes to the Open Distributed Systems (ODS) nudge project and discusses the following:

"*How can mobile interfaces present digital nudges?*"

Designing a nudge related front end application requires answering the subsequent questions related to nudge theory and UX:

1. *How can mobile interfaces utilize motivational incentives?*

2. *How can mobile interfaces utilize principles regarding user experience?*

The project should integrate nudges into an application in a non-intrusive manner. The application should be transparent about how it uses nudging on its users and handles sensitive user data. The application should also have a user interface (UI) following UX best practices, particularly for nudging. This thesis will explore the required theoretical, visual, and functional aspects to answer the research questions.

1.3.1 Benefits

The reduction of pollution from greener transportation will contribute to fighting climate change. However, the most notable changes will be apparent in large cities. Areas with a higher traffic concentration and industry expose humans to polluted air over time. Reduced air quality can have a variety of implications for the health of the residents. Kampa and Castanas published a study on how polluted air can affect health that stated the following:

> Human health effects can range from nausea and difficulty in breathing or skin irritation, to cancer.[6]

Quote 1.2

Reduction of traffic in such areas can alleviate some of these health risks. People who choose to walk or take their bike to a destination will also have the added health benefit of being more physically active. Said people would also benefit economically by not paying for tickets and fuel.

1.3.2 Sustainability

A concern related to nudging is that people might get used to long-term nudges, and they could either normalize the nudge or grow tired of it. The latter case means nudging can only utilize short-term effects, which have been one of the central positions for nudging critics[7]. A study on food waste in the hospitality industry evaluated the effectiveness and sustainability of social norm nudges and pre-commitment nudges[8]. However, the study found that they sustained the same effect even after being implemented over a period. Thus, the ODS nudge project can support stable effectiveness throughout the life cycle.

1.4 Method

The research conducted in this project follows a *qualitative* research method, which entails understanding human behavior and opinions[9]. The research aims to develop hypotheses around human interaction and develop the project as a computer system. The project utilizes *interpretivism* as its philosophical assumption by studying humans' perspectives and behavior related to the research phenomenon[9].

The project uses pre-existing knowledge and research related to the phenomenon, thus utilizing the *applied research* method. The project builds an understanding of a problem and solves it through novel applications and technologies[9].

The research follows an *abductive* approach, which expresses preconditions with limited data sets. The preconditions are requirements based on the pre-existing knowledge described in the above research methods. The hypothesis can use the requirements to narrow down the areas to conclude from within the phenomenon[9].

1.5 Challenges

The project faces challenges related to the core concepts and the application's design. Nudging can emphasize the positive aspects and the need for action. However, in this project, the action's consequence is not immediate, as it can take decades to notice environmental changes. Such actions may feel less urgent to the user and thus can lose motivational ground.

People have various habits and react differently to behavioral change. The

application must be able to display relevant information to each user according to their preferences.

The application must support detecting that the user acted through methods such as user feedback. The success of a nudge is, after all, dependent on user action.

1.5.1 Limitations

This thesis explores the ODS nudge project's front-end aspects, whereas the rest of the stack remains incomplete. Thus it is not possible to perform practical user testing of the application.

1.5.2 Delimitations

Regarding infrastructure, the project limits itself to the stack's front-end part, using only the mobile operating system, Android. The project limits the number of mobile devices that can install the application. The implementation sets the minimum Android API level to 29, equivalent to the Android operating system version 10. Thus, only Android phones with Android 10 or higher can install and run the application.

The project does not implement any infrastructure related to a remote back end. The project also does not implement methods to collect usage metrics due to no practical user testing. Finally, the project primarily focuses on architectural implementation. The project regards the design of content within pages as a secondary priority.

1.6 Contribution

The project achieves to set a baseline for future theoretical research and technical implementation within front-end development. The project provides an application prototype with the baseline of a robust architecture. The architecture design is a practical demonstration of connecting front-end components to nudge theory. The project presents a set of selected psychological requirements designed for nudging. Furthermore, the project contributes with novel usability heuristics designed for nudging. See section 7.1 for more details.

1.7 Context

This work belongs to the Open Distributed Systems (ODS) group at UiT the Arctic University of Norway. The research group focuses on next-generation applications, data exchange, and data analysis. However, the group centers on middleware. The group works on several projects revolving around, among others, the Internet of Things (IoT), personalization, and nudging[10]. The ODS nudge project explores how to motivate people to choose environmentally friendly transportation methods. The project utilizes nudges as non-coercive means to achieve behavioral change towards the goal of the nudges.

1.8 Outline

The structure of the rest of the thesis is as follows:

Chapter 2 - Background Provides general information about the psychology behind persuasion and nudging and how digital systems use them. The section follows up with UX and UI theory, emphasizing design systems. The section also discusses ethics in nudging and related work.

Chapter 3 - Method Describes the utilized research methods used for this thesis.

Chapter 4 - Design Presents the design of the architecture. The section describes the representation of nudges and their connection to the application.

Chapter 5 - Implementation Presents the implementation details and architecture of the prototype.

Chapter 6 - Discussion Discusses the limitations and progress.

Chapter 7 - Evaluation Reflects on the project's contributions and goal completion.

Chapter 8 - Conclusion Provides the conclusion of the thesis and ends with future work.

2 Background

This chapter outlines the technical concepts required to understand the implementation of the project. The chapter begins, in 2.1, by introducing the psychology behind persuasion. The following section, 2.2, connects the concept of nudging to the former section. The section begins with an overview of nudging. The first subsection, 2.2.1, describes the digitalization of nudging, and the second subsection, 2.2.2, describes smart nudging and how they utilize user profiles. The following subsection, 2.2.3, bridges over to persuasion in a digital environment. The last subsection, 2.2.4, describes the process of designing nudges within the ODS nudge project. The third section, 2.3, describes design principles in user experience (UX) emphasizing design systems. The fourth section, 2.4, outlines the ethical considerations related to persuasive systems such as nudging. The final section, 2.5, gives an overview of related work.

2.1 Persuasion

Persuasion is an old concept that dates back to social psychology[11]. The purpose of persuasion is to prompt behavioral change through various forms of interaction. The decision and action are the basis for behavioral change, thus being an essential aspect of persuasion. Research on the psychology of human persuasion aims to discover fundamental principles and models that can trigger decisions and actions.

2.1.1 Six Principles of Persuasion

The book "Influence: The Psychology of Persuasion" discusses the psychology behind human decision-making and how it is considered effortful[12]. Cialdini describes six principles that can be beneficial in persuasion. However, whether the principles are applied ethically depends on the persuader's intentions.

Reciprocity: *Represents detriment in inequality.*
Giving a favor can trigger the feeling of imbalance, thus prompting the desire to settle the debt. Some restaurants utilize reciprocity by giving out sweets next to the bill to increase the chance of tipping.

Scarcity: *Represents value in rarity.*
Displaying limited availability and quantity can increase interest in the service. Stores can use scarcity by highlighting the remaining amount of items.

Authority: *Represents value in credibility and reputation.*
Connecting services to professionals within the field can increase the desire and trust for the service. Recommendations by credible sources can result in indirect authority, which is helpful in job interviews.

Commitment and consistency: *Represents value in keeping a consistent identity.*
Asking for small favors, growing over time, gives the donor a feeling of gradual identity change, increasing the likelihood of using the service. Stores can utilize commitment by selling free samples of items, so recipients identify as users of the items, thus being more likely to buy the item later.

Liking: *Represents the combined value of generosity, cooperation, and relatability.*
Generosity generally means being nice to others, such as giving compliments, which can increase likeability. Cooperation represents how humans, as herd animals, naturally respond positively to genuine collaboration. Relatability represents how similarities in most areas can form the perception of an attachment, resulting in acceptance and inclusion. People in working environments can utilize liking by being genuinely friendly and open to cooperating with coworkers. They can also emphasize their similarities in various areas related to work and personal life.

Consensus: *Represents value in conforming to the crowd.*
Highlighting the choice by the majority gives the instinct to follow the norm. Digital solutions can utilize consensus by recommending an option stating that 9 out of 10 recipients chose it.

2.1.2 Psychological Effects: Cognitive Bias

People make decisions every day, both unconsciously and consciously. Psychology defines the two types of choices as systems 1 and 2, with the former operating automatically and efficiently and the latter operating on complex and effortful tasks. It is possible to persuade people to change through system 2[13]. However, active and complex thinking affects the effectiveness of persuasion, such as triggering critical thinking and realizing possible consequences of behavioral change. Thus it is considerably easier to target system 1 in persuasive systems.

An effective technique to persuade humans is taking advantage of their subconscious preferences and behavior to trigger the desired change. Psychology defines such preferences as cognitive bias. Blanco described cognitive bias as:

> Cognitive bias refers to a systematic (that is, nonrandom and, thus, predictable) deviation from rationality in judgment or decision-making.[14]

Quote 2.3

Framing Framing is the conscious phrasing of information to support the targeted behavior; thus, the same information can result in two completely different behaviors. Stores can utilize a value-oriented framing effect to highlight the worth of a sale, such as displaying the value saved on an item instead of the percentage[15].

Priming Exposing a person to a stimulus, such as words and ideas, can alter the behavior from later stimuli. The human mind keeps the recent ideas fresh in memory and responds faster upon reacting to similar ideas. Exposure to the word yellow can produce a faster response to a banana than unrelated ideas like a car or a house[15].

Overconfidence People tend to overestimate their capabilities, especially when comparing themselves to others. People often underestimate time until completing projects and tasks, thus conversely showing overconfidence in their abilities[15].

Social Behavior often follows social norms, usually perceived as the ideal behavior. Not conforming to such norms can result in social isolation and ridicule. Norms affect daily activities such as fashion in varying scenarios[15].

Status quo bias Change requires energy and conscious decision-making. Thus, people tend to prefer to keep the current state the same. Highlighting an option with a minor deviation from the current status quo usually appeals more than opposing options[15].

Middle-option bias People tend to select the middle option upon visually presenting a series of options. A study shows how the bias combines the compromise- and response-order effect. The former effect suggests avoiding the extremes of a selection of choices, thus compromising with the option with the median value. The latter effect emphasizes the importance of the choice order. The study proposes that selections with more than two choices naturally favor the middle option. The significance of a middle position outweighs that of a median value. Combining the two effects results in the middle-option bias, often utilized in online stores upon buying a product[16].

Hyperbolic discounting Benefits gained earlier are valued higher than later ones. Online stores and subscription services often offer exclusive discounts on their services for first-time users[15].

Loss-aversion People usually value a loss or disadvantage more than a gain or advantage. A travel plan can highlight the cost of using a car instead of traveling by bus[15].

Decoy-effect The effect adds a third option to increase another option's perceived value. The new option's value is significantly less than another option, making it asymmetrically dominated. In cinemas, selling popcorn usually utilizes this effect by adding a medium-sized popcorn close to the same price as the large size[16].

2.2 Nudging

Nudging is a form of persuasion oriented towards influencing individuals' choices in a non-intrusive manner[17]. Thaler and Sunstein described nudges as:

> To count as a mere nudge, the intervention must be easy and cheap to avoid. Nudges are not mandates. Putting fruit at eye level counts as a nudge. Banning junk food does not.[17]

Quote 2.4

Both nudging and persuasion use positive reinforcement as central factors to motivate behavioral change. However, nudging emphasizes the importance of the individual's freedom of choice.

2.2.1 Digital Nudging

Various fields, such as food stores, apartment buildings, and digital environments, utilize nudges. The latter uses what is called digital nudging, described by Meske and Potthoff as:

> . . . a subtle form of using design, information and interaction elements to guide user behavior in digital environments, without restricting the individual's freedom of choice.[18]

Quote 2.5

The same rules for normal nudges apply to digital nudges regarding freedom of choice and unobtrusiveness. However, using design and interaction elements adds a layer of new complexity.

2.2.2 Smart Nudging

Digital nudges do not focus on specific users or user groups, thus creating occasional, irrelevant nudges for particular users. Smart nudges aim to remove redundant nudges through personalization, described by Karlsen and Andersen as:

> . . . digital nudging, where the guidance of user behavior is tailored to be relevant to the current situation of each individual user.[19]

Quote 2.6

2.2.2.1 User Profile

Smart nudges can only work upon gathering and studying personal information to specialize the nudges toward users' needs. The system combines and stores the information in a user profile. The following list describes the essential fields in such a profile:

- **Personal data** Core user data such as name, gender, and age
- **Intention/goal** The primary goal the user wishes to achieve
- **Capabilities** Physical and psychological
- **Interests** User topic interests
- **Device information** Personalization of presentation info
- **Context** The current situation while using the system. Any relevant real-time data gets aggregated and combined into the context, e.g., bus schedule and arrival estimation. Further elaborated on in section 2.2.4
- **History** A log of every time a user has interacted with a nudge in the system
- **Nudging history** All nudge prompts from the system get logged, regardless of if the user responded or not

2.2.2.2 Nudge Components

Karlsen and Dalecke describe smart nudges as having four main components; activity, content, influence, and presentation[15]. Each nudge gets assigned a specific activity, e.g., taking a trip outside or eating vegetables. The content refers to practical, helpful information to perform the activity, such as weather forecasts. Behavioral change is hard; people need a push to perform the activity and achieve change. Smart nudges combine nudging with cognitive biases to create influence types that prompt the push; see table 2.1. The final component describes the visual and interactive parts of the nudge.

Type	Effect	Description
Salience/remind	Framing with others	Emphasize consequence and commitments
Simplification	Framing	Remove unnecessary complexity
Disclosure	Anchoring	Transparency and openness
Implementation intentions	Priming	Point out user intentions and identity
Affect	Framing and priming	Evoke positive emotions
Commitment	Framing and priming	Precommitment strategies
Defaults	Status quo bias	Default options favor the goal
Informing	Framing to engage system 2	Info of action consequence
Loss aversion	Loss aversion	Loss unless acted upon
Social	Social norms	Inform others' choices
Priming	Priming and framing	Association through cues
Incentives	Hyperbolic discounting	Motivate behavior

Table 2.1: Influence types, based on the work from Karlsen and Dalecke[15]

2.2.3 Persuasive Systems

Digital nudges cover a specialized spectrum of persuasion within digital environments, emphasizing design elements and freedom of choice. Persuasive systems act as a broader definition with the sole focus of changing behavior with non-coercive means, described by Oinas-Kukkonen and Harjumaa as:

> ... computerized software or information systems designed to reinforce, change or shape attitudes or behaviors or both without using coercion or deception.[20]

Quote 2.7

Persuasive systems such as Fogg's functional triad describe how to incorporate persuasion into digital environments[21]. However, software development needs requirements translated into working system features. The requirements are either functional or non-functional. Non-functional requirements describe qualities the system must include, whereas functional requirements describe how the system should behave.

The requirements have four main categories; *primary task*, *dialogue*, *social support*, and *system credibility*[20]. The *primary task* relates to the system's core functionality, which is essential for a working product. *Dialogue* relates to how the system communicates with the users through different effects. *Social support* relates to how the system incorporates how multiple users can interact, collaborate and compete. Finally, *system credibility* ensures a more persuasive product through credible information. Table 2.2 describes the requirements of each category:

	Non-functional	Functional
Primary task	Personalization	The system personalizes content to the individual user according to their needs and preferences
	Tailoring	The system tailors content toward the relevant user groups. Tailoring factors include interests, usage context, and personality
	Self-monitoring	Performance and status tracking that guides towards accomplishing user goals
	Reduction	Simple tasks generally result in increased persuasion and ease of fulfillment instead of complex tasks and behavior. Thus, reducing complex tasks into simpler ones
	Tunneling	Some tasks might require a series of user actions. Such tasks can guide the user to completion using a logical flow of actions, thus increasing persuasion and reducing annoyance.
Dialogue	Reminders	The system increases the likelihood of goal achievement through reminders
	Suggestion	Recommend action related to behavior at an opportune moment
	Praise	Give positive feedback to the users
	Rewards	Virtual rewards such as trophies and achievements increase persuasion
	Liking	The system has an attractive look and feel
Social support	Social learning	Show progress and statistics of other users
	Social facilitation	Motivate users through the recognition of other users using the system at the same time
	Social comparison	Enable the possibility of comparing user data and accomplishments with other users
System credibility	Trustworthiness	The system provides transparent and truthful information
	Surface credibility	The system has a competent look and feel
	Verifiability	Easy methods to verify the accuracy of content, such as reference links

Table 2.2: System requirements, based on the work from Oinas-Kukkonen and Harjumaa[20]

2.2.4 Design Process

The ODS smart nudge architecture utilizes a circular directed graph, from defining the foundational goal of the system, to evaluating the nudge results. Figure 2.1 illustrates the architecture.

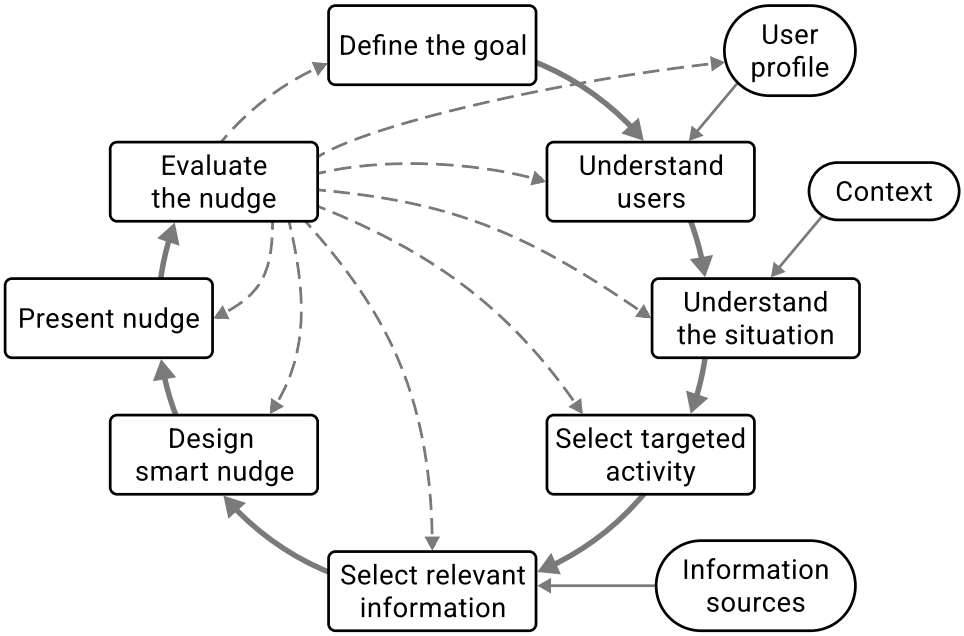


Figure 2.1: ODS smart nudge architecture

The first step of the system is to define a goal that can drive the rest of the steps in the right direction. The goal represents a long-term vision followed by the system and its users. The following two steps aggregate user and situational data through user profiles and context information. The aggregated data enables the system to select the appropriate activity, such as taking the bus to work. Before designing the nudge, the system gathers the necessary external information needed to create the nudge, such as bus schedules. Designing the nudge relates to combining the appropriate cognitive bias' and influence types to maximize the persuasive effect. Presenting the nudge creates the visual and interactive components of the system. The presentation stage needs to consider design principles such as usability heuristics, see section 2.3.4, to ensure a quality product. The last step evaluates the nudge based on the entire cycle's results. The evaluated data propagates back into each step, so the next cycle improves on earlier mistakes and successes.

2.3 User Experience

Digital environments build the front-end components based on the overarching concepts; user interface (UI) and user experience (UX). Both concepts relate to users interacting with digital environments. Obi described user interface (UI) as:

> UI is focused on how a product's surfaces look and function.[22]

Quote 2.8

UI handles all visual and interactive components of digital environments. Individual UI components can follow best practices, though they do not account for the experience of using the system as a whole. Obi described UX as:

> 'User experience' encompasses all aspects of the end-user's interaction with the company, its services, and its products.[22]

Quote 2.9

UX handles a broader field than UI, so some designers argue that UX is a superset of UI[23]. UX relates to the overall experience of using the system.

The following sub-sections, 2.3.1 and 2.3.2, cover aspects of mobile environments and their inherent constraints. Sub-section 2.3.3 describes in-depth design systems emphasizing the design system this project uses. The last sub-section, 2.3.4, briefly covers design principles within usability heuristics.

2.3.1 Mobile Design

UX principles and areas of importance change dramatically between mobile systems and desktop computers. Mobile systems cover, for the most part, mobile phones, tablets, and wearables. Conversely, mobile UX refers to the experience of interacting with such devices[24]. The nature of the hardware, software, and usage differences in mobile systems drives designers and developers to adapt more suitable UX and UI principles.

2.3.1.1 Screen Types

Mobile applications usually follow a set of base screen types defining the main structure of the app. Every app has four essential screens: the splash, onboarding, login, and home screen.

Splash screens They show the first images upon opening an app, with the primary objective of waiting until the app finishes initializing[25]. Starting an app requires initializing various frameworks and programs, which can be noticeable depending on the application. The splash screen displays an idle and straightforward UI while the user waits for everything to load. The screen makes the initialization process seem clean, without lag and stuttering.

Onboarding screens They aim to teach the app's functionality and motivate usage. Opening an app for the first time presents the user with new information and unfamiliar controls, so the user might feel confused and unwilling to use the application. Onboarding can guide the user at the start through the app's functions, also referred to as the nickel tour[26]. Progressive onboarding can use tooltips and guided tasks to further help the user after getting used to the app. Onboarding can also present the benefit of using the application to motivate the user to start using the application.

Login screens They show essential login information such as username, password field, and the possibility to sign up as a new user[25]. Most modern applications store login information such that second-time launches log in automatically.

Home screens They act as the first screen where users can utilize the app's core functionality. The home screen can navigate to the rest of the screens, which vary depending on the application. Modern apps usually keep the most utilized functions easily accessible on the home screen[25].

2.3.1.2 Notifications

Systems can utilize notifications to communicate information related to the application to the user[27]. The content and purpose of the information vary and shape different types of notifications. Push and pull notifications differentiate when delivering information to the user. The first invented method, pulling, works through the client-side polling for new information from the server side. The newer method, pushing, works through the server sending updated information to the client. Pushing omits useless client requests and ensures the first request arrives at the client; thus, most situations prioritize pushing. However, the design and content of notifications do not depend on it using pushing or pulling.

The following paragraphs detail four essential areas of the structure and usage of notifications. The first paragraph describes where notifications can display on the phone and where they can redirect the user. The second paragraph covers how long notifications can remain active. The following paragraph describes how to categorize notifications based on their type of content, and finally, the last paragraph covers the methods of granting permission to receive notifications.

Location

Mobile apps operate with notifications in two distinct areas on the device; inside and outside the app. The core aspects of the two types differentiate to such a degree that the rest of the thesis will refer to the former as push notifications and the latter as in-app notifications. Push notifications usually prompt the user without user input beforehand. Meanwhile, in-app notifications more often depend on user input[28]. Upon clicking on some push notifications, they can redirect the user to the app. A seamless experience ensures that the app page corresponds to the notification content[29]. A third notification type, although less critical, lists a log or history of previous notifications. The mobile operating system and the app can show such a log, whereas it would usually display within a notification history page in an app.

Persistence

Non-persistent notifications, also called pop-up notifications, will automatically disappear after a certain amount of time. Most push notifications are non-persistent, usually displayed as a message box at the top of the screen[28]. Persistent notifications remain active on the screen until user input of some kind. Some push notifications, such as status dots and widgets, remain silent without interfering with the user. However, persistent, in-app notifications usually stem from the system reacting to user input. Such notifications may block further progress until dismissed by the user[28].

Content Purpose

Transactional notifications contain information about the upcoming activity, such as reminders. The user already knows about the activity; however, the notifications can help prepare for the activity[28]. Non-transactional notifications contain status and progress-related information. The user can skip such information to complete the upcoming activity, though it might help motivate further application usage.

Some apps divide their content into channels based on the importance levels; high, medium, and low. The highest level shows time-sensitive and critical information such as alerts, error messages, and notifications requiring user input. The medium level shows helpful information that does not need immediate action, such as warnings, success messages, and requests for feedback. Finally, the lowest level shows passive notifications such as status and informational messages[28].

Permission

Modern apps require user permission to send push notifications, whereas many apps use the onboarding stage to request the user to "opt in" for notifications[30]. The user can always "opt out" of receiving notifications from the system or the app itself. Non-critical notifications may enable the user to opt out of that type on the notification itself.

2.3.2 Mobile Constraints

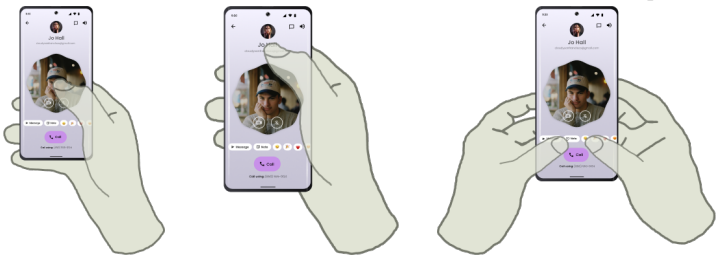
Mobile systems face challenges related to their nature of using smaller hardware and their users moving the devices frequently. Resulting in unique constraints, compared to stationary systems, most prominently within how the users interact with the device and the environment they use it within.

2.3.2.1 Interaction

Mobile systems have less screen space than other systems and thus result in unique interaction constraints through touching and reading the device. Users can hold a smartphone in one hand, both hands, or cradled position. Hoober[31] expands the three positions into six holding methods, shown in figure 2.2. Users change methods depending on the context, so most users eventually use all six methods. However, on average more than 50% of users hold the phone with both hands, whereas 75% of users only touch it with one thumb[31].

The data suggest high thumb usage compared to other fingers, which raises the importance of adjusting design elements accordingly. The thumb moves with a sweeping motion and a limited angle, leading to different levels of reachability in the device screen seen in figure 2.3. The green area reflects where a thumb would naturally move and inflict less stress on the hand.

(a) Cradled **(b)** Hold and touch **(c)** Two hands - landscape



(d) One hand - first order **(e)** One hand - second order **(f)** Two hands - portrait

Figure 2.2: Smartphone hold methods

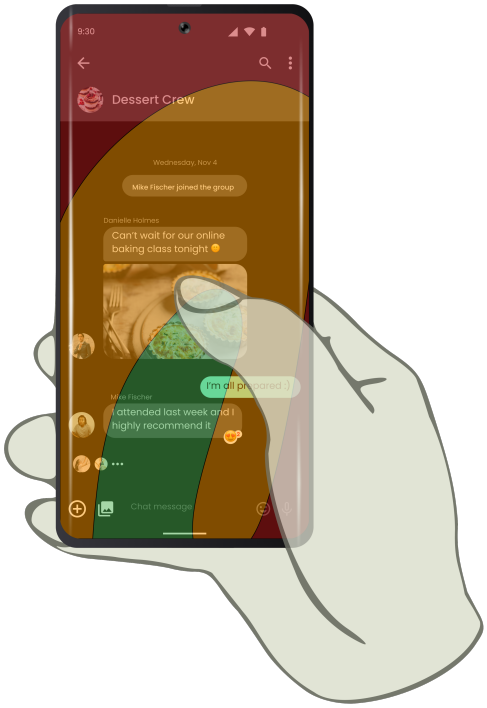


Figure 2.3: Smartphone thumb reach

The figure suggests prioritizing elements on the sweeping line of the thumb. However, users also hold the device in two other methods; two hands result in greater reach on both sides of the screen, and the cradle method extends thumb mobility and reach. Thus, figure 2.4(a) reflects how users typically touch and read closer to the screen's center, diminishing importance towards the edges. This pattern contradicts the classic F-pattern reading method in larger screens, also referred to as scanning information[32], see figure 2.5. Desktop websites can display much more information, leading to smaller text fonts and more scrolling. Such sites tend to use the F-pattern as a structural basis. The users reading the site skim the text through "scanning" while guided by the F-pattern.

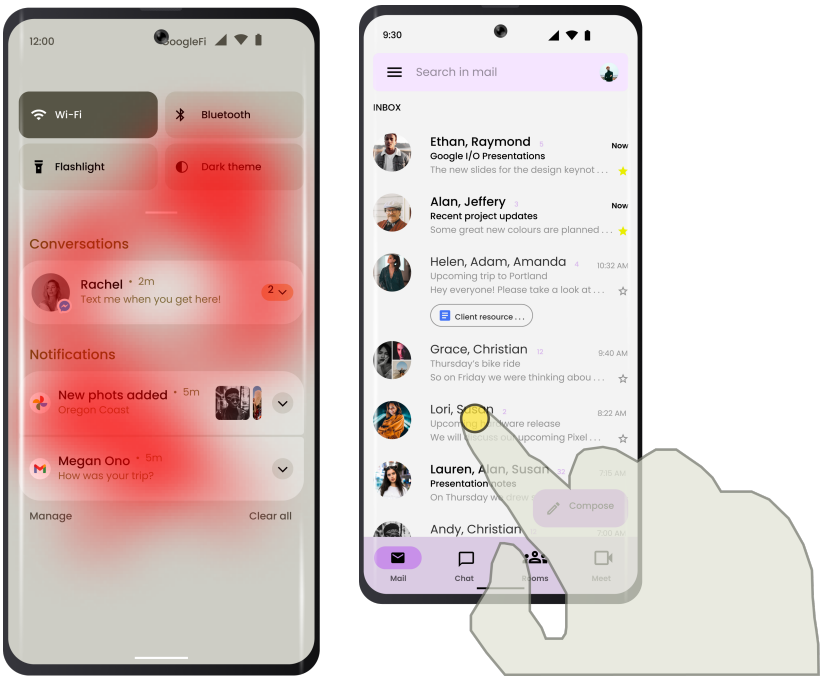
Desktop systems can use external input methods such as a mouse and keyboard, whereas users of mobile devices obscure the bottom edge with their hand, see figure 2.4(b). The touch area figure reflects the hidden area; some consider this area the hardest to reach.

The average human's fingertips have a width of 1.6-2cm, with thumbs up to 2.5cm[33]. Thus, fingertips lack the precision of computer mice, so designers should follow the $1x1cm$ rule. The clickable components should have a minimum diameter of $1cm$ to ensure that users can accurately interact with the system.

Reducing the size and centralizing components naturally leads designers to integrate minimalism in mobile design. Fewer and more visible components clean up the UI and give the system a more modern look. Section 4.3.2.1 further describes minimalism.

2.3.2.2 Environment

Mobile devices should function while moving outside of optimal working conditions. Bad lighting from the outside, such as sunshine, can impair the application's experience; therefore, the designers should avoid susceptible color combinations and schemes. Moving outside also results in the possibility of no accessible network, resulting in some applications malfunctioning or unnecessary mobile data usage.



(a) Hand touch area **(b)** Hand obscuring screen

Figure 2.4: Smartphone hand usage

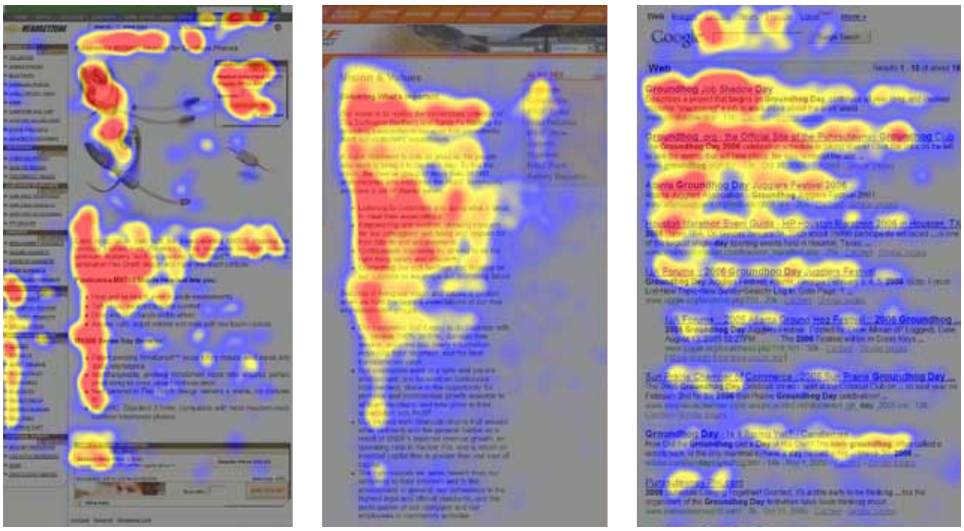


Figure 2.5: F-shape pattern[34]

2.3.3 Design System

This section defines design systems and details the default design system used by Android, called Material Design.

A design system in its purest form describes a set of UX and UI design standards and guidelines. Designers use a design system to ensure visual consistency and reduce redundancy between UI elements. Design systems also improve communication and collaboration between designers through a shared visual language[35].

This project uses the default design system used by Android, called Material Design or just Material. The design system incorporates the standard definition of a design system and includes components and tools to streamline development and collaboration. Material defines the best practices of UX and UI design for Android developers, thus being a well-established base for creating new and improving existing applications. Material is open-source, so everyone can contribute and follow along with changes to the system[36]. The latest version of Material, called Material You, focuses on expressive and adaptable designs, further described in section 2.3.3.2. The following subsections detail the most significant structure of Material by using information from the Material Design guidelines documentation[36].

2.3.3.1 Structure

The structure covers the root building blocks necessary for modern design systems. The following sections cover the design system's environment, followed by the definition of component-based design. The next section covers the various methods to generate layouts within the screen. Finally, the last section covers how to utilize navigation.

Environment

The environment of Material Design contains *surfaces* that act as the base for all other structures[1]. The surfaces are flat plains that illustrate elevation with light and shadow. Although the surfaces exist in three dimensions, they all have a depth of 1dp; see paragraph 2.3.3.1(Layout). Material uses vectors to effectively renders surfaces with infinite resolution, such that although the shape and size can vary, the resolution stays the same. Surfaces can also apply opacity and temporary layers to make content less prominent, called *scrims*.

1. https://material.io/design/environment[36]

Material differentiates the elevation and importance of surfaces through the position on the z-axis. Users perceive higher elevation as physically closer and thus more significant than other surfaces. Material uses virtual light and shadow to imitate surface elevation. Material creates realistic shadows with a combination of single-source lights, called *key lights*, and lights from all angles, called *ambient light*. The former creates sharp and directional *key shadows*, whereas the latter creates soft and diffused *ambient shadows*.

Figure 2.6: Surface elevation

Component

Material uses three methods to display visual content, the first being surfaces as described above, whereas the two others are *elements* and *styling*[2]. Surfaces contain elements representing any core type of content, such as text, icons, and images. Styling can modify the color and shape of surfaces and elements, referred to as theming in section 2.3.3.2. A *container* represents an enclosed area containing various content, acting as an effective grouping method. Material uses containers for more freedom and precision in styling and positioning surfaces and elements. Designers can set default styling and position values for containers and their content to create a *component*, thus using it as a template for future use. Material specifies components that support its design principles, such as buttons, dialogs, and banners.

Layout

The layout defines the position and spacing of components, emphasizing consistency across different environments[3]. Most applications split the layout into three logically separated regions; body, navigation, and app bar. The regions usually display content with similar functionality, such as the navigation region containing components related to navigating to other pages. The largest surface area contains the body and displays the application's main functionality. The app bar region extends the body by grouping components and actions to

2. https://material.io/components[36]
3. https://material.io/design/layout[36]

enable users to perform everyday actions faster and more intuitively. Regions containing various content can highlight similar elements by grouping them closer together and containing them away from unrelated content using boundaries or spacing. Material uses *alignment* by positioning an element relative to a component's boundaries.

The pixel density, pixels per inch (PPI), varies based on screen size and resolution, so element size and positioning using PPI can break UI consistency. Material preserves the consistency with density-independent pixels (dp) that scale relative to the PPI. Android also uses scalable pixels (sp), functioning equally as dp, exclusively for fonts.

Material uses grids split into *margins*, *gutters*, and *columns*, to ensure responsive content size and positioning, as illustrated in figure 2.7. The columns show where to place content, while the margins and gutters add space outside and between the columns, though generally, space between components relates to *padding*. Margins and gutters usually have a set dp size, while a screen size percentage calculates the column size. Elements can position along the *baseline grid* for finer granularity when the responsive layout grid is too coarse-grained, illustrated in figure 2.8. The baseline grid provides a 4dp to 8dp grid, depending on the component type.

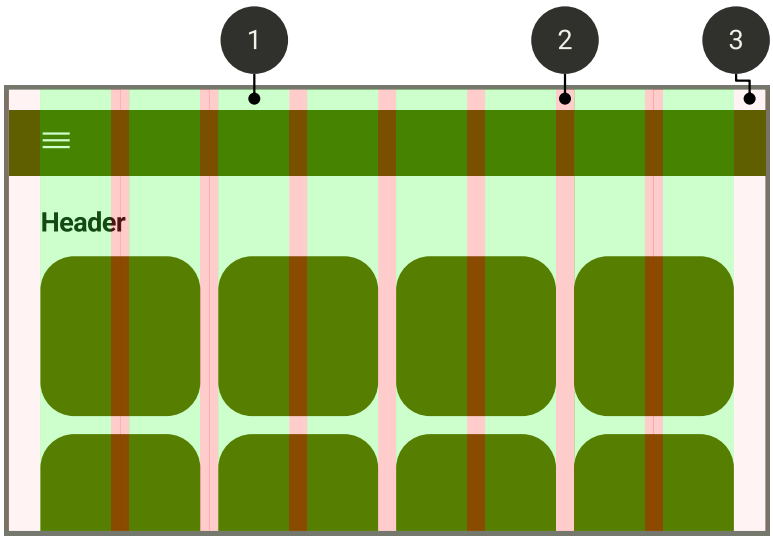


Figure 2.7: Responsive layout grid
(1) Column (2) Gutter (3) Margin

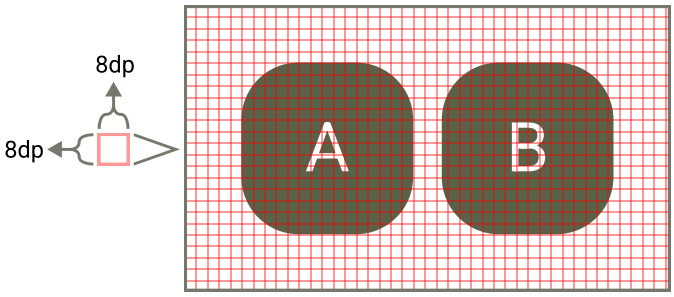


Figure 2.8: Baseline grid

Responsive systems support various screen dimensions by adapting to the position and size of components. The layout can add additional components or swap with more informative ones upon switching to larger screens. A third option can introduce *density* by decreasing padding to create a more compact layout. Higher-density layouts can result in more components and information being visible. However, Material limits elements from reducing their size to lower than the minimum touch target size of 48dp, further described in paragraph 4.3.2.2.

Navigation

Applications with several screens use a hierarchy to organize the screens into different levels vertically and horizontally. Material uses navigation to move between screens using three directions; *Lateral*, *forward*, and *reverse*[4]. The lateral direction moves across the same level, which is helpful for top-level navigational components such as drawers and bottom bars. The forward direction moves the hierarchy downwards or follows a sequence of action movements. The reverse direction moves opposite to the forward, either upward the hierarchy or backward the history of previously visited screens. Most clickable components, such as buttons and search components, use forward navigation, whereas search fields enable direct access to deeper levels of the applications.

4. https://material.io/design/navigation[36]

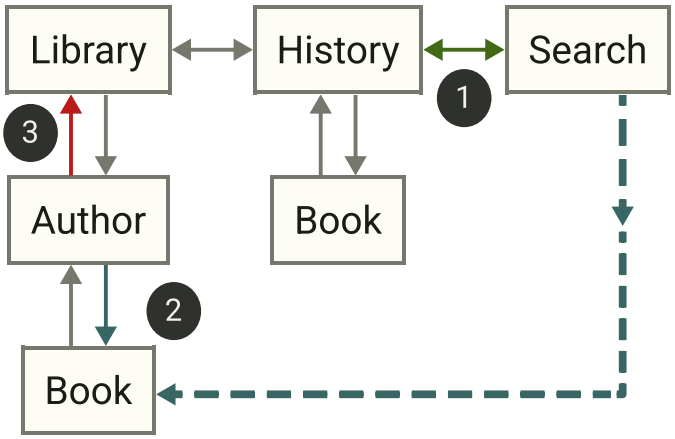


Figure 2.9: Navigation direction
(1) Lateral (2) Forward (3) Reverse

2.3.3.2 Theme

Material uses Material Theming to design themes reflecting the application's brand[5]. A theme determines the color and shape of components, text, and images, effectively changing the application's entire UI. Material theming can customize the default design system values to create one or several themes for the application, most commonly a light and dark theme.

Creating a successful brand for an application requires the combination of color schemes, shape schemes, type scales, and icon themes that complement each other consistently throughout the UI. Each scheme and the relevant attributes should express uniqueness so that users can distinguish and recognize the brand.

Color

Material uses *color schemes* to define the color tones used within a theme[6]. A color scheme uses a foundation of five *key colors* mapped to a series of *color roles* representing the base colors associated with most elements within the UI. The key colors map the primary, secondary, and tertiary *accent* colors to components and containers. While the *neutral* colors, neutral and neutral variant, map to backgrounds, surfaces, and outlines. The color scheme may include additional colors, such as error states or custom colors.

5. https://material.io/design/material-theming/overview.html[36]
6. https://m3.material.io/styles/color/overview[36]

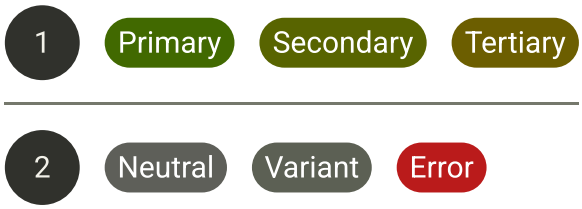


Figure 2.10: Key colors
(1) Accents (2) Neutral and error

Each key color generates a tonal palette with 13 tones of color, changing the brightness from black to white. Figure 2.11 illustrates the conversion of the primary accent, tone 40, to a tonal palette.

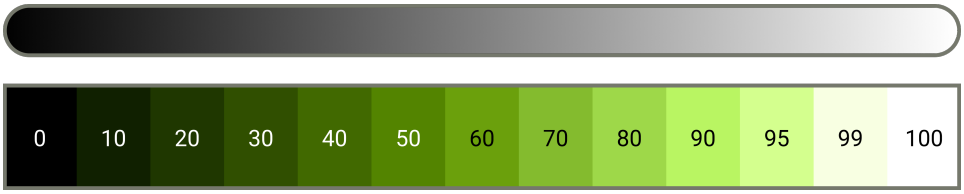


Figure 2.11: Tonal palette

Material selects sets of color tones from the tonal palette and maps them to color roles, usually with a background color and a matching color on top. Figure 2 illustrates the mapping of the primary accent. Usually, a light color scheme can automatically generate its dark counterpart and vice versa. Material You emphasizes the usage of personal and dynamic color schemes, using custom colors and wallpapers to generate schemes automatically.

Figure 2.12: Color roles

Shape

Material uses *shape schemes* to define the angle and curvature of corners and edges[7]. A shape scheme can highlight components' hierarchy, state, and motion differences through the four metrics illustrated in figure 2.13.

7. https://material.io/design/shape[36]

1. **Category** Divides the components into their appropriate size differences; small, medium, and large

2. **Family** Defines rounded or cut corners

3. **Size** Defines absolute or percentage-based corner scaling

4. **Symmetry** Defines symmetrical or asymmetrical shapes

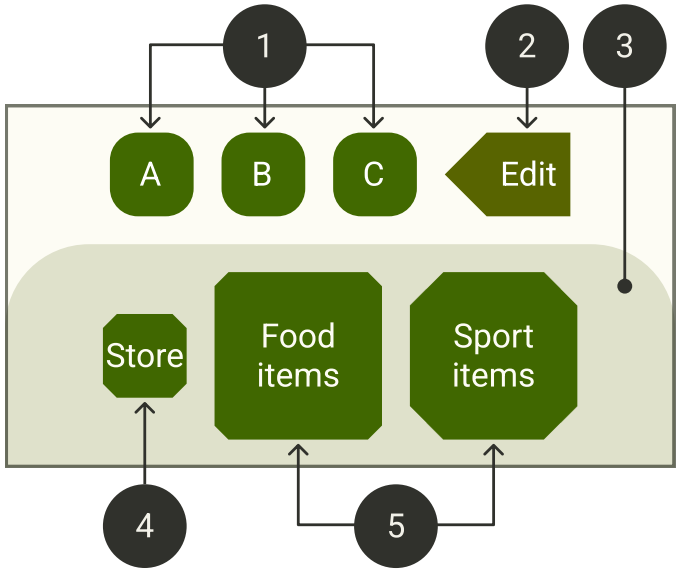


Figure 2.13: Component shapes
(1) Similar (2) Related (3) Separate (4) Cut corner
(5) Scaled absolute(left) and percentage(right)

Typography

Material uses *type scales* to define the attributes and applications of typefaces[8]. A type scale determines a set of styles for the most common text applications. Material generates type scales with 13 styles using the computer font service *Google Fonts*[9]. Each style contains a *scale* category describing the use case of such a style, such as “Header 1” or “Subtitle 2”. Furthermore, each category defines the associated typeface and general text attributes, whereas Android utilizes the “Roboto” typeface[10]. Figure 2.14 shows an example type scale using the categories “Header 3” and “Button”.

8. https://material.io/design/typography[36]
9. https://fonts.google.com
10. https://fonts.google.com/specimen/Roboto

Scale Category	Typeface	Weight	Size	Case	Letter spacing
H3	Roboto	Regular	32	Sentence	0px
BUTTON	Roboto	Medium	14	All caps	1.25px

Figure 2.14: Type scale

Iconography

Material uses *icon themes* to define icons' stroke, fill, corner radius, and color[11], as illustrated in figure 2.15. An icon theme can simultaneously combine the four attributes except outlined and filled.

(a) Outlined **(b)** Filled **(c)** Rounded **(d)** Two-tone

Figure 2.15: Icon attributes

2.3.3.3 Communication

Material utilizes communication to teach fundamentals of the application and continuously convey the application status quo, referred to as system state[12].

System Feedback

The system conveys state information to ensure users understand that their actions have been performed[13]. Specific components can utilize different states depending on how and when the user interacts with them, such as buttons having states for being idle, hovered over, or pressed. Other aspects of the UI can display the state through loading mechanisms, such as a loading symbol for updating pages or placeholder components while waiting for the components to load.

11. https://material.io/design/iconography[36]
12. https://material.io/design/communication[36]
13. https://material.io/design/platform-guidance/android-haptics[36]

Informational Pages

Certain application pages specialize in teaching the user how the application works, examples being the splash and onboarding screens[14]. Applications can also provide informational pages dedicated to guiding the user if problems should arise when using the app, such as help and feedback pages.

Notifications

Notifications notify users of useful information based on three metrics; In-app or push, persistence, and silent or non-silent[15].

In-app notifications use dialogs and banners to display persistent and critical information that will not disappear unless the user interacts with them. Non-persistent notifications, called pop-up notifications, use snackbars to display system status and feedback.

Silent notifications display persistent information through badges, widgets, and app shortcuts. Badges indicate if notifications exist in the app, whereas widgets and shortcuts act as hybrid information capable of performing more complex tasks.

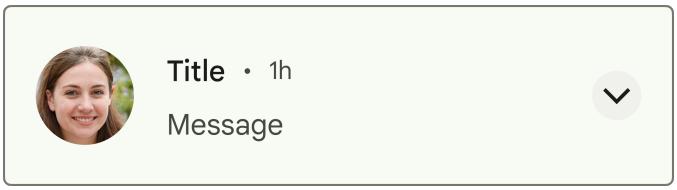


Figure 2.16: Collapsed push notification

Push notifications display non-silent and non-persistent information, although they can be persistent because they do not disappear from the notification list until dismissed. They typically drop down from the top of the screen as rectangular boxes with the information split into a header, content, and actions. The header contains the app icon, the app name, supportive text, a timestamp, and an optional expand icon. The content shows the central message of the notification through a title, the body, and a large icon if needed. Action enables user input through buttons and input fields. Additional styling with background and coloring can emphasize the importance level of the notification[37].

14. https://material.io/design/communication/onboarding[36]
15. https://material.io/design/platform-guidance/android-notifications[36]

Figure 2.17: Expanded push notification

2.3.4 Usability Heuristics

Usability heuristics provide a set of principles that aim to improve the overall development lifecycle and ensure sufficient software quality. In 1993, Nielsen published the book "Usability Engineering" as one of the first pieces of literature on usability evaluation[38, 39]. Later, in 1994, they released one of the first renditions of usability heuristics divided into seven categories[40]. Finally, Nielsen and Norman created an updated version for modern systems within UI design, divided into ten categories[39]. Section 4.3.2 describes a novel set of heuristics inspired by Nielsen and Norman, designed to fit the nudge architecture.

2.4 Ethics

Ethics in digital systems covers a set of moral guidelines used to regulate the development of the systems. Common ethical problems, such as privacy, discuss the interaction between the user and the system. Digital nudging systems interact with humans, thus the users, and aim to change their behavior. Thus, making ethics an essential topic to such systems. The following sections discuss three ethical concepts; 2.4.1 Transparency, 2.4.2 Coercion and freedom of choice, and 2.4.3 Privacy.

2.4.1 Transparency

Transparent digital systems disclose the purpose of the system and how it interacts with its users. The system should make the information freely available and easily accessed by the users.

This ODS nudge project heavily relies on tracking information, particularly location data. A transparent system would explain why it needs the location information and how it will use it. Additionally, the system should offer details on how it handles data related to the user and its organization. Commonly through privacy policies and terms and conditions, explained further in section 2.4.3.

A typical critic of nudging argues that nudging decreases humans' autonomy[41, 42]. Engelen and Nys discussed different meanings of autonomy. This section discusses "Autonomy as Liberty", which focuses on preserving the freedom of choice[43]. Critics thus claim that nudges can decrease humans' preference in choice, and such nudges indirectly make their choice. An online study in 2019 investigated whether transparency could counteract the reduction of autonomy. The study found that nudges with transparency kept high autonomy and retained the same level of persuasion as those without transparency[44]. The system should thus implement transparency by containing easily accessed information on how the system uses nudging and the purpose of nudging its users.

2.4.2 Coercion and Freedom of Choice

Persuasive systems avoid coercion or deception, as described in quote 2.7(page 14). The Oxford dictionary defines coercion as:

> the action of making somebody do something that they do not want to do, using force or threatening to use force.[45]

Quote 2.10

Digital systems that utilize coercion would act with ill intent aiming to achieve behavioral change without users' consent. Nudging systems seek to preserve the users' freedom of choice, as described in quote 2.5(page 11). One could argue that enforcing behavioral change implies removing the freedom of choice from the users. Thus, nudging systems could strengthen users' freedom of choice by avoiding coercion.

2.4.3 Privacy

In 2016 Lee *et al.* published the article "An Ethical Approach to Data Privacy Protection" which describes data privacy as ". . . access, use and collection of data, and the data subject's legal right to the data"[46]. Data usage affects both the user and the system's owner. Applications provide privacy policies to protect the users' rights[47]. The policy includes information such as how the system stores the user data, who can access it, and how to delete it. On the other hand, applications provide terms and conditions to protect the organization's rights[47]. The users must accept the terms as a set of conditions for using the system and its data.

Data privacy connects to the legal field where laws enforce system data management regulations[46], such as the General Data Protection Regulation (GDPR)[48]. Systems that utilize third-party services should particularly disclose how they share data with the services. Additionally, such services might require the system to obtain a software license. The system can disclose the usage of the service to the user on an informational acknowledgment page.

Nudging systems revolve around changing human behavior and monitoring their choices. They must store and manage a certain amount of sensitive user data. Thus, nudging systems must carefully follow the ethics of data privacy. The ODS nudge project can store user profile information, such as the user's name and email address. The project also relies on location tracking, resulting in a remote system routinely receiving sensitive user data.

2.5 Related Work

The project presented in this thesis broadly covers three areas; *nudging*(2.5.1), *mobile user experience*(2.5.3), and the use case of *green transportation*. My project primarily centers its theory and design on the first two. This section discusses related work covering similar theory as the first two areas. The section will also include some related work centered around *persuasive systems*(2.5.2) to compare with my project's usage of system requirements.

2.5.1 Nudging

In 2018 Schneider *et al.* published an analysis of using nudging with interface design in a mobile environment[16]. They described people as having cognitive limitations, such as cognitive bias, that can influence decision-making. My project describes systems 1 and 2 in section 2.1.2. My project describes system 1 as a passive thought process, similar to Schneider *et al.*'s description of the cognitive limitation.

They created a nudge design process cycle with four steps; define the goal, understand the users, design the nudge, and test the nudge. Compared to the ODS nudge architecture, they share similarities, particularly their design step. They used the cycle with a three-step program to select the user choice to influence, then the bias to use, and finally, the UI-related element that can utilize the bias. They analyzed the results of three experiments from researchers at the University of Liechtenstein[49, 50, 51]. The experiments used a reward-based crowdfunding mobile application where they tested the bias *decoy*[49], *scarcity*[50], and *middle-option bias*[51]. Finally, the results showed an increase in persuasion in all three experiments.

My project covers a broader set of cognitive biases and other nudge-related theory, such as the system requirements. It also researches more in-depth UX and related principles. However, my project did not perform experiments and can thus not compare results to their findings.

2.5.2 Persuasive Systems

The following related works use the persuasive system design published by Oinas-Kukkonen and Harjumaa and their system requirements[20]; see table 2.2(page 15).

In 2012 Langrial *et al.* published a study and evaluation of mobile applications utilizing system requirements[52]. The study used scientists specialized

in persuasive systems to evaluate the experiments. They created a map of requirements for the features supported within 12 different mobile applications. The results showed that the two requirement categories, *primary task* and *system credibility*, had comprehensive coverage in the application. Within those categories, the applications mainly supported the *self-monitoring* and *personalization* requirements. Though, most of the applications lacked support for *dialogue* and *social support*[52]. My project emphasizes how it includes all four requirement categories, further described in chapter 4. Langrial *et al.* also pointed out how none of the applications supported the *tailoring* requirement[52]. The ODS nudge project considers such a requirement essential, which also my project includes.

In 2021 Torkamaan and Ziegler published a new framework similar to the system requirements[53]. They created a framework inspired by the system requirements and other frameworks[54, 55, 56, 57, 58, 59, 60, 61]. They split their framework into the components; user and system. The user component contained beliefs, perceptions, status, goals, and intentions. Meanwhile, the system components contained content for recommendations and persuasive elements. My project primarily uses the system requirements as a base for other nudge-related theory. Their framework uses components from systems requirements and other frameworks and groups them into one unified framework. Their framework notably includes information such as the user's status, compared to the ODS nudge project's user profile information. For the implementation, they presented their framework in an Android recommender application relating to mobile health[53]. They discussed some design choices connected to their framework, though they did not go in-depth about UX. My project addresses the application's connection to the nudge-related theory, such as the system requirements, and the principles within UX.

2.5.3 Usability Heuristics

In 2019 Maulana and Suzianti published an analysis of using usability heuristics to redesign a mobile application[62]. They redesigned an Indonesian retail store app to increase user interaction and improve the UX. They used the same design system as my project, Material Design, as a basis for their heuristics. Similarly to my project, they created ten usability heuristics for redesigning the app. Most of the heuristics follow similar purposes as my project's heuristics, with some exceptions. Maulana and Suzianti used a heuristic dedicated to helping users recognize, diagnose, and recover from errors[62]. My project effectively combines that heuristic with the heuristic of error prevention. Their heuristics does not include my project's 4th heuristic of personalization. Their system thus does not focus on principles such as personalization and tailoring, as opposed to the ODS nudge project. They performed experiments on the

redesigned app with 30 people, comparing the old design with the new one. The result showed an increase in the effectiveness of the UI. Thus, the users spend less time doing the same tasks within the application[62].

3

Method

This chapter outlines the methodologies used within this project. All the methods chosen use theory from the methodology publication by Håkansson[9]. The publication separates methodologies into a spectrum of stages from the start of research until the final presentation of results. The first stage splits the methods into the two methods described by Newman *et al.* as:

> Quantitative methodology and qualitative methodology are considered to be polar opposites.[63]

Quote 3.11

The *quantitative* method focuses on rigorous scientific testing of sizeable numerical data sets. This project researched using the *qualitative* method that differentiates through using limited non-numerical data sets that focuses on human meanings, opinions, and behavior. The project aimed to develop a hypothesis related to the research phenomenon. The project utilized the philosophical assumption of *interpretivism* by using a phenomenon that revolves around studying humans' perspectives and behavior. The following section describes methods selected based on the project's use case, 3.1, and the connection between the literature study and the design, 3.2.

3.1 Method and Approach

The project revolves around motivating humans to choose more environmentally friendly transportation. The phenomenon connects to the research question and its sub-questions, described in section 1.3. Thus, in the context of this project, the phenomenon asks how an application can utilize nudging to achieve such behavioral change within a digital environment. The phenomenon is a practical problem based on existing research using the *applied* research method. The project aimed to solve the problem by developing a novel technology as a mobile application.

The project did not rely on an existing data set, thus utilizing the *abductive* approach. The approach further defines a set of preconditions based on the current knowledge of the phenomenon. The preconditions act as a necessary basis for the hypothesis. They narrow down the areas to research within the phenomenon—for example, the sub-questions to the research question.

3.2 Strategy

The project used *action research* by developing a practical solution to the phenomenon. This stage started with a preliminary literature study before designing a solution, referred to as the planning stage of action research.

The study covered two theoretical areas; psychology connected to nudge theory and mobile design theory. The project started by researching psychological aspects of persuasion. The aspects created a base to support the related nudge theory. The nudge theory covers the categorization of different nudge types and their relation to persuasion. The nudge theory also describes persuasive systems as a broader term covering digital environments. The persuasive systems represent a set of system requirements that connects to both nudge theory and persuasion. After the nudge theory, the project researched mobile design theory. The project focused on best practices using design systems and user experience (UX) principles in a mobile environment.

The design stage connected the literature study to an application design. The process combined the nudge theory with the practical features of a mobile application's architecture. Thus, specific architectural components in an implemented system can support various theoretical principles within nudging. Meanwhile, the mobile design theory defines a set of principles referred to as usability heuristics. They aim to provide a robust method of designing content while conforming to best practices within UX and user interface (UI).

The project implemented the design as an Android app while conforming to robust architectural requirements. The process ensured a basis capable of supporting the envisioned nudge theory within the design. The heuristics also ensured that development always utilized designs conforming to best practices within mobile design theory. The implementation completed the literature study and design by testing the theory in a practical application. Finally, the project discussed and evaluated the implemented applications. The section described the positive aspects and missing features within its design and architecture.

3.3 Quality Assurance

The project did not perform the data collection and analysis stages, though the research followed a robust approach to the literature study, design, and implementation. The result ensured an application and theoretical base that other projects and future development can efficiently utilize. Thus, providing *transferability* within the quality assurance of the research material. Additionally, the project conforms to *confirmability* by using no personal assessment while conducting the research and designing the application.

4

Design

This chapter outlines the project's design, effectively connecting the background theory while presenting a technical architecture for the application. The chapter begins, in 4.1, by explaining the aspiration and minimum features required for the project. Section 4.2 describes the requirements related to implementing digital nudging, and what requiremnts this project focuses on. Section 4.3 describes how to include the nudges in the design, connecting the background theory with a feature set. Section 4.4 presents the design system principles accounted for during the design stage. The final section, 4.5, covers the technical architecture of the application.

The following sections outline novel models created for the ODS nudge project. Some aspects of the models use inspiration from other sources, though they were created from the ground up as novel solutions to the aspiration of this project. Section 2.5 describes how other solutions solved similar problems as this project.

4.1 Use Case

The project uses an Android app to cover a nudging system's most significant design and presentation-related aspects. It conforms to the use case of motivating its users to choose more environmentally friendly transportation. Covering the use case requires a well-defined nudge definition, described in section 4.2, that can cover the practical features required. The features nudge the user through notifications of various types, described in section 4.3.1.6, and through practical usage of the app itself. As the use case centers around transportation, the central aspect of the app supports registering current and future trips from one place to another. Supplementary information and statistics add to a polished product while improving the users' motivation, further discussed in section 4.3.1.

4.2 Digital Nudge

This project contributes to the ODS nudge project, which works with smart nudging systems. As quote 2.6(page 11) describes, smart nudging uses digital nudges tailored to the user. However, the ODS nudge project does not have an existing back end to supply the application with user-tailored information. Thus, this project implements an application that uses digital nudges in the user interface (UI). The definition of a digital nudge described in quote 2.5(page 11) defines three requirements for digital nudges.

1. **Presented** The nudge has to interact with the user, commonly using visible or audible elements.

2. **Behavioral guidance** The nudge shall motivate behavioral change in the user

3. **Freedom of choice** The nudge shall at all times retain the user's freedom of choice

Requirement 1 does not specify specific types of elements or locations in a front-end environment that can count as nudges. Thus, a nudge does not need to emphasize a particular activity; simply changing behavior is enough. The app's notifications and functionality, combined with user experience (UX) principles, can nudge the users. Requirement 2 emphasizes the need for psychological principles as a backbone for motivating change, which in this project would relate to persuasion and persuasive system theory. Finally, requirement 3 enforces using the two first requirements to follow a user's choice-friendly approach.

4.3 Design Structure

The ODS nudge architecture consists of a cycle of components from designing a goal to presenting and evaluating nudges. This project concerns the front-end part through designing and presenting the nudges to the end users, shown as the green part in figure 4.1. All communication between the back end and front end resides in the blue part, the *bridge*. Section 6.3.3 discusses how future development could implement an HTTP client that communicates with a remote back end, thus effectively acting as the *bridge* part of the figure. The project implements the two green stages through an overall design architecture, as shown in figure 4.2.

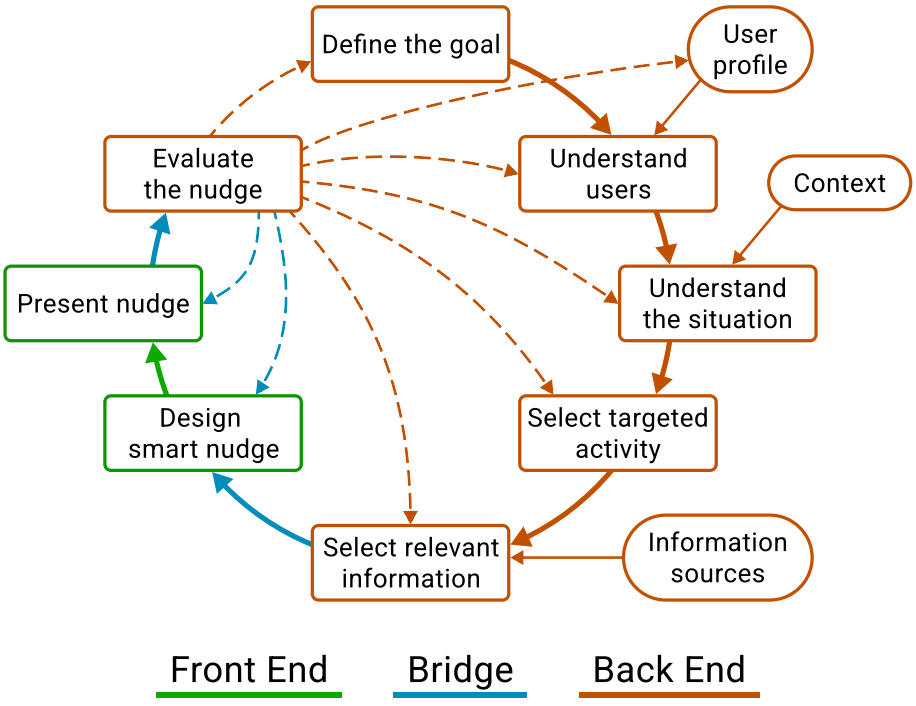


Figure 4.1: The front-end part of the ODS nudge project

Design The stage defines the project's requirements, split into functional and non-functional, presenting an application feature set and theoretical principles. The *Features* part represents practical features required within the application, such as the capability of registering or planning trips. On the other side, the *Theory* part acts as a combination of psychological principles based on nudge theory and non-functional system qualities, such as *reduction* and *tunneling*. The blue arrow between *Features* and *Theory* illustrates that the feature set complements the theory and vice versa. The feature set describes practical features needed to fulfill the use case, which indirectly results in using theoretical principles—for example, the **help page** and **app page** in settings using qualities from *system credibility*. Similarly, the theory expands the feature set based on the relevant principles to nudge-related projects, see subsection 4.3.1. Such as the *social support* system qualities directly influencing the feature **social page**.

Presentation The stage defines the usability heuristics of the application's visual components, see subsection 4.3.2. The blue arrow from theory and up illustrates how specific theory determined some chosen heuristics. As an example, the system quality *simplification* determined **minimalism**.

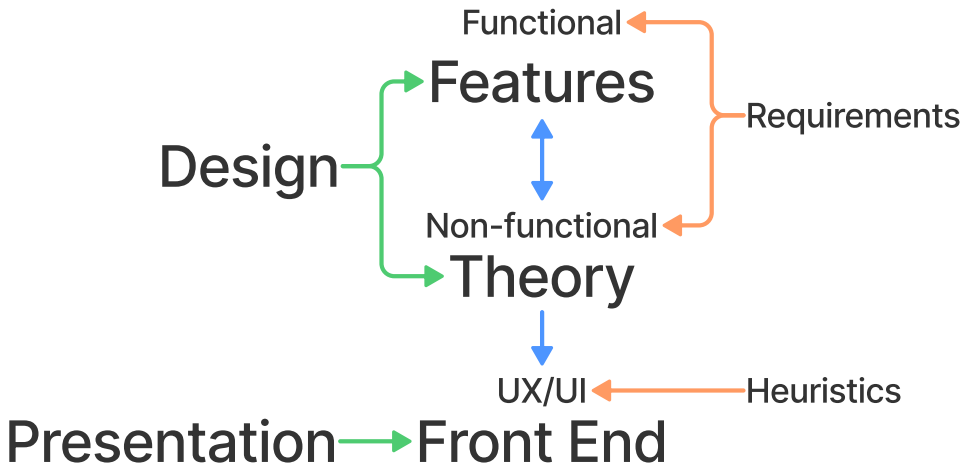


Figure 4.2: Overall design structure

4.3.1 Design Requirements

The design stage connects the system requirements to the psychological principles behind nudging, shown as the design part in figure 4.2.

Figure 4.4 visualizes a feature set for functional requirements of the application, such as the ability to edit user account information on a settings page. The different colors separate different stages of the app, mainly through different navigation stages and notification types. The inner boxes represent placeholders for various categories of data shown in that location.

The *Locked phone* and *Unlocked phone* stages represent using the phone outside the app and show methods to access the app from there. A locked phone can access the app with a lock screen **push notification**. In comparison, an unlocked phone can most commonly access the app through the **app icon** or **push notifications**. The notification stage can happen inside and outside the app, thus simply depicting notification categories.

The remaining four stages show different stages of using the app. The home stage covers the most frequently used and practical pages, while the menu stage covers supporting information related to various app usage. The settings stage presents changeable preferences and system information, and finally, the notification stage categorizes app notifications.

The following section presents how each category of the feature set connects to various non-functional system requirements, subsequently connecting nudge theory.

4.3.1.1 Bridge

The bridge stage, shown in figure 4.3, defines three possible scenarios before reaching the app home screen. **Onboarding** displays a series of screens informing of the app's basic functionality and ensures access to the required permissions to use the app.

Figure 4.3: Bridge feature stage

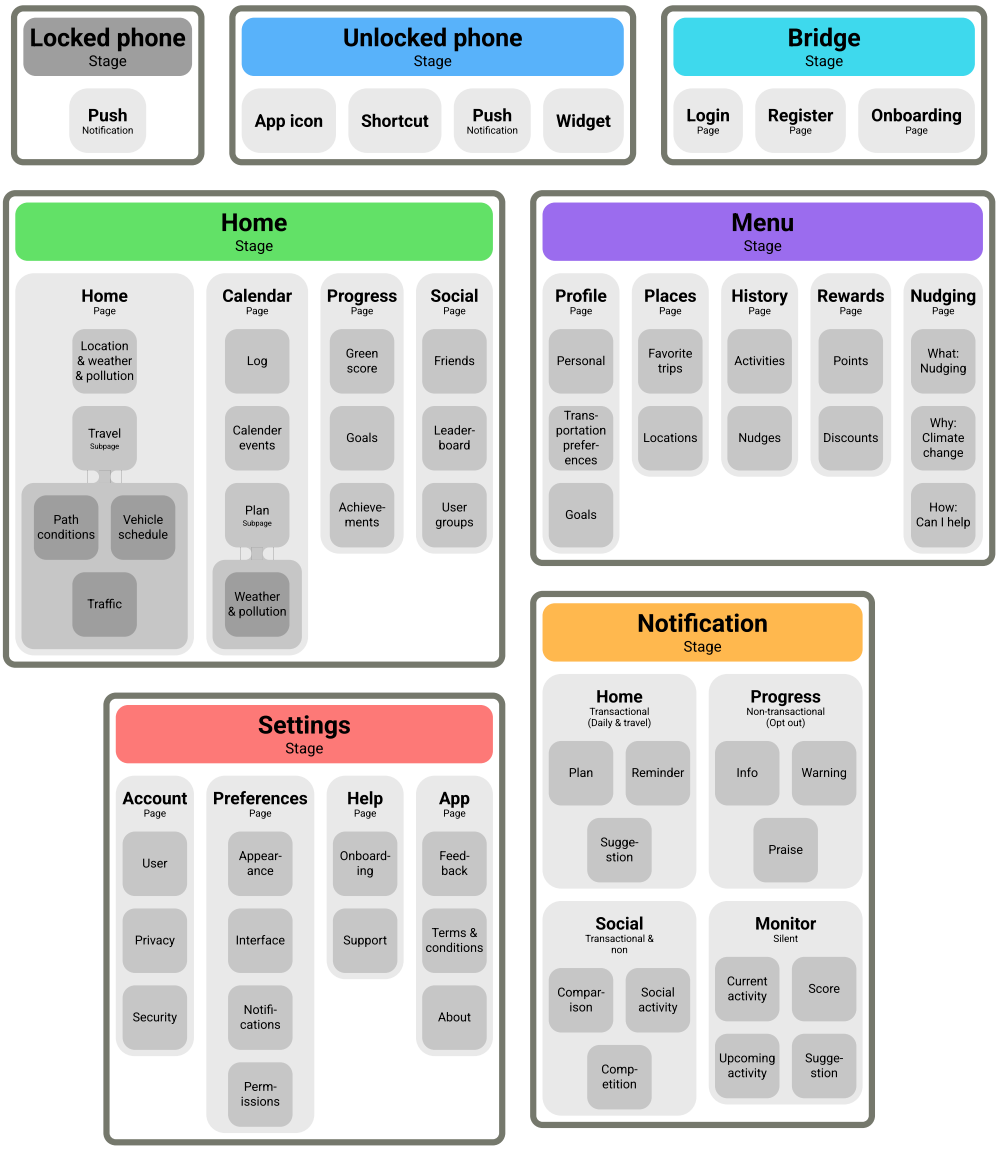


Figure 4.4: Feature map

4.3.1.2 Connection to System Requirements

The following stage figures follow the color codes according to the legend in figure 4.5. The green, yellow, purple, and blue colors represent the four system requirements categories described in table 2.2(page 15). Meanwhile, the brown color represents relevant user data to display and store. Boxes that have multiple colors apply to all the respective categories.

The bottom part of the figure shows how three principles from the six principles of persuasion, see section 2.1.1, connect to the described categories. The *dialogue* category connects to the principle of *liking*, *system credibility* to the principle of *authority*, and *social support* to the principle of *consensus*.

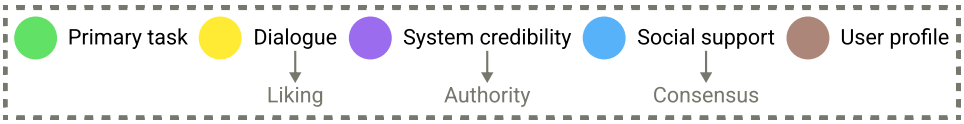


Figure 4.5: Feature stage legend

4.3.1.3 Home

The home stage, shown in figure 4.6, displays the most commonly used activities when using the app. Each column represents a top-level navigation destination. The **home page** contains the primary usage of the project, registering new trips using the **travel sub-page**. Registering new trips requires information such as **path conditions** where the road can have maintenance. There might be a delay in the **vehicle schedule** when using public transportation. Finally, when using roads, the **traffic** can affect the user choice. The page also contains relevant information about the user's current **location**, such as **weather** and city **pollution**. Such information can utilize the *priming* by visualizing relevant imagery, such as sunshine at the user's current location. The page facilitates the non-functional quality *tailoring* by only displaying relevant trip information according to previous usage data and *user profile* information. The page also applies *reduction* and *tunneling* by guiding the user through creating a trip with minimal input requirements and without any other visual cluttering. The page finally applies the *simplification* quality by enabling easy access to the core functionality on the first app page. The **calendar** page displays a **log** of previous trips, upcoming trips as **events**, and the option of planning a future trip using the **plan sub-page**. The sub-page uses the *commitment* nudge influence type that, in turn, uses the cognitive bias of *framing* and *priming*.

The **progress page** concentrates on facilitating *praise* by showing progress towards personally set **goals** and app **achievements**. **Goals** implement three categories, as a data point in the *user profile*, a form of *self-monitoring* in the primary task category, and *praise*, as mentioned above. A general **green score** is a single source of the users' combined progress towards more environmentally friendly activities. Finally, the **social page** facilitates various social support requirements such as *social learning*, *social facilitation*, and *social comparison*. The **groups** and **friends** allow for showing and comparing progress while simultaneously encouraging competition with **leaderboards**.

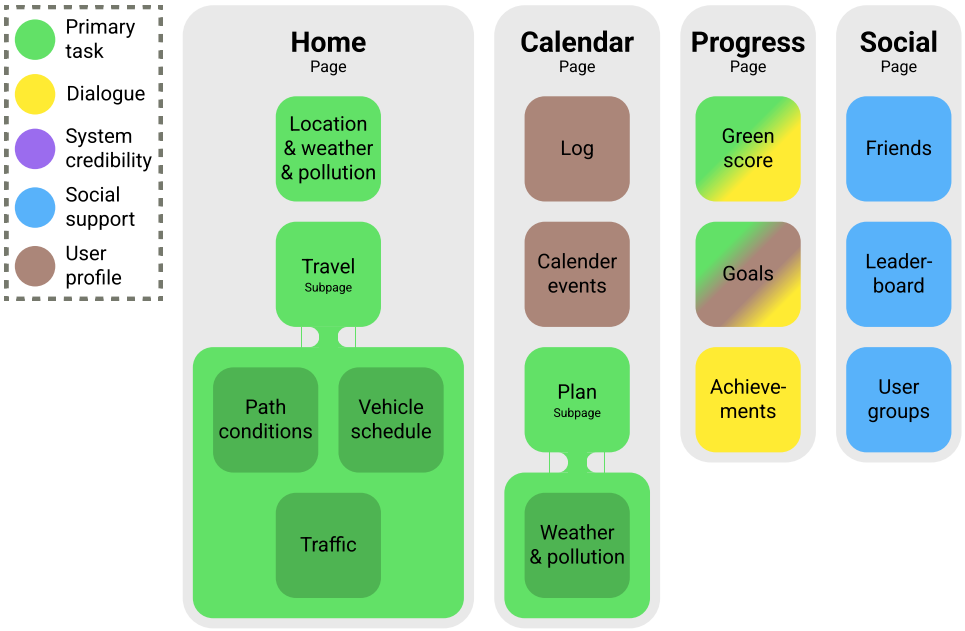


Figure 4.6: Home feature stage

4.3.1.4 Menu

The menu stage acts as a set of pages providing useful second-hand information that supports various non-functional qualities, shown in figure 4.7. The **profile page** shows a combination of user-related data and *personalization* related to setting **personal** preferences, with the addition of the ability to edit **goals**. The **places page** and **history page** add to the *user profile*, with the **places** showing **addresses** and **history** showing lists of previous nudges acted upon as **activities** and all received **nudges**.

The **rewards page** similarly applies the *rewards* quality. The **points** act as an internal app currency used for brand merchandise or converting to other currencies. Meanwhile, **discounts** can apply to local transportation, such as city bikes or public transportation. The application can offer early discounts, thus the *hyperbolic discounting* effect, to achieve the *incentives* system quality. Finally, the **nudging page** gives an overview of the project's aspiration, thus a description of the concept of nudging, **why** the project centers around climate change, and **how** the users can use the application to stay green. The page conforms to system credibility through the *verifiability* of the science and purpose of the project. Therefore, it also conforms to the quality *informing* and *authority* with expert-backed sources.

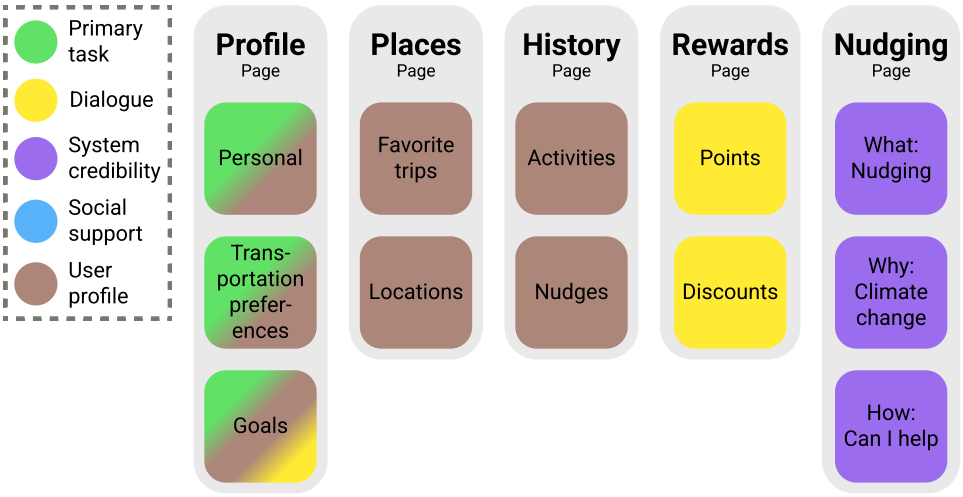


Figure 4.7: Menu feature stage

4.3.1.5 Settings

The settings stage shows both changeable preferences and system-related information, shown in figure 4.8. The **account page** contains core *user profile* data, with the option to toggle public visibility in **privacy** and password management in **security**. The **preferences page** contains the core usage of *personalization* through adaptable **appearance** with themes, technical **interface** changes such as language, management of **notification** frequency, and toggles for **permissions**. The **help page** is an always available path to **support** and show **onboarding** information about the application. Meanwhile, the **app page** relates to system and organization information. To be precise, the **terms and conditions** contain terms for the organization and a privacy policy for the user. Overall the help and app pages conform to the *disclosure* quality by ensuring a transparent system.

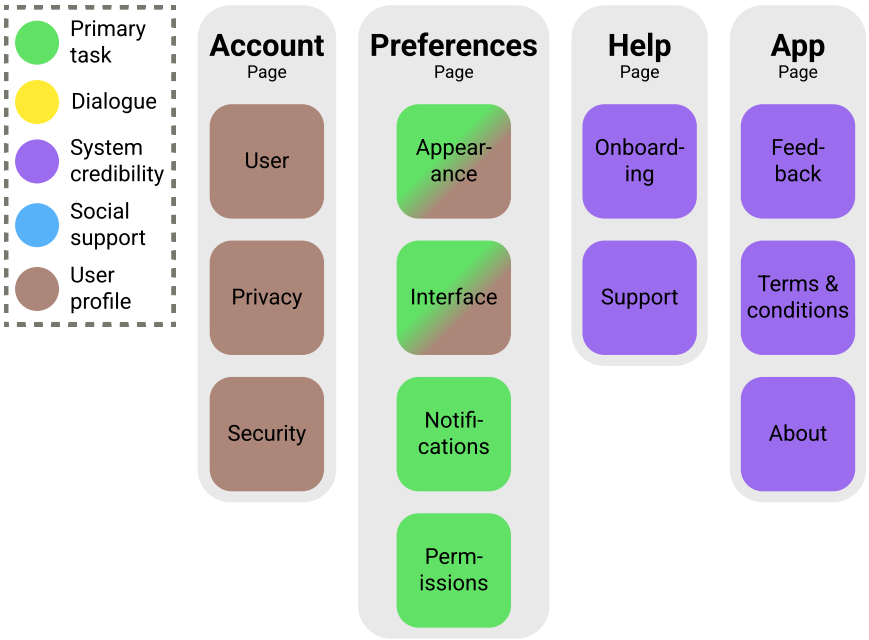


Figure 4.8: Settings feature stage

4.3.1.6 Notification

The notification stage shows how to present nudges divided into categories based on persistence and content purpose. Thus split into transactional or not and silent or not; see paragraph 2.3.1.2(Persistence). Figure 4.9 shows the notification nudges split into four categories.

The **home** and **progress** notifications act as *dialogue* qualities, thus communicating with the user, whereas some apply specific qualities. **Home** differs from **progress** by using transactional notifications, thus more practically oriented, such as travel-related information. On the other hand, the **progress** uses non-transactional notifications, thus less critical information that the user can opt out of receiving. The **reminder** type uses *salience/remind* by emphasizing the commitment to an existing plan, resulting in the *framing* effect. The **plan** type uses *implementation intentions* by predicting user actions, such as taking a particular transportation method to work. The **info** and **warning** types utilize the *informing* quality in the same manner as the help and app pages, emphasizing consequences rather than general information. Finally, the **praise** type is a natural usage of the *praise* quality.

The **social** category acts similarly to **home** and **progress**, although it splits into its category to focus solely on social aspects. **Social** notifications generally display non-critical information and thus focus on informing social-related status. The notifications utilize the *social* quality by showing information such as user groups or friends statistics.

Monitor notifications generally use the *self-monitoring* quality by monitoring **current** and **upcoming** activities and showing the most relevant progress statistics, such as the **green score**. Optionally the notification can contain silent **suggestions** for trips.

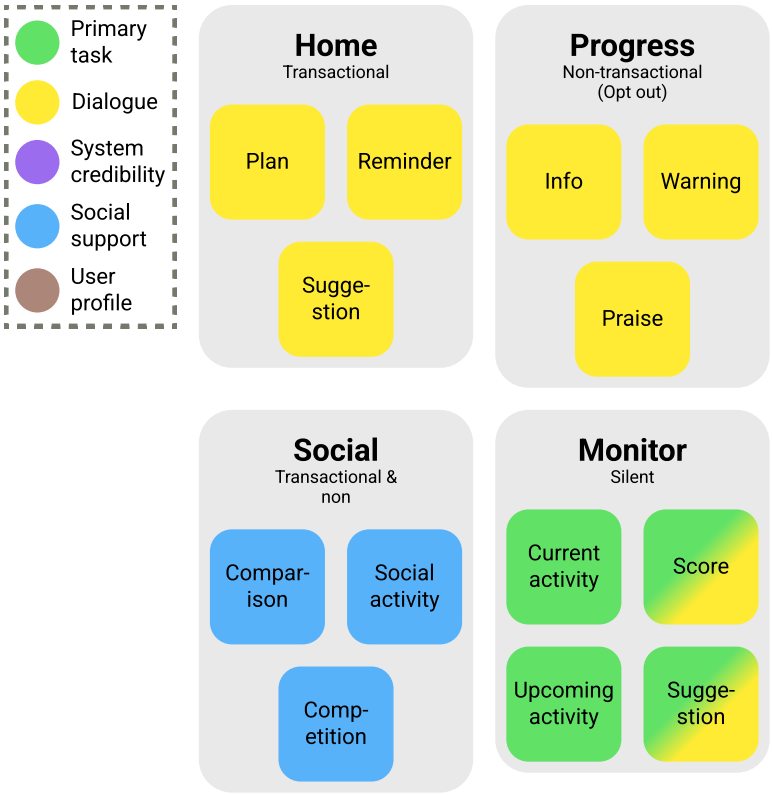


Figure 4.9: Notification feature stage

4.3.2 Usability Heuristics

The presentation stage of the ODS nudge architecture presents the implementation of the UX and UI through the visual requirements, referred to as heuristics.

The section below presents a novel set of 10 heuristics designed for the ODS nudge project. The heuristics take inspiration from the work of Nielsen and Norman[39, 40]. They split into two categories; visual and functional heuristics. Visual heuristics primarily handle the placement and method to render the visual aspects of UX and UI components. In comparison, functional heuristics handle the usage of said components and how they affect the user and the system.

4.3.2.1 Visual Heuristics

1. **Minimalism**
   Minimalism applies the quality of *simplification* to the entire application when done correctly. Minimalism reduces visual clutter by only presenting core visual components at opportune moments. A suitable application structure through menus and navigation can help separate and categorize such components. Multimedia and icons can also reduce clutter by replacing or supporting text.

2. **Recognition over recall**
   Recognition uses similar methods to minimalism, though they differ in intention. Visual cues and logical navigation can enable users to recognize patterns naturally through recall instead of memory. Thus, users do not have to memorize structure and data to efficiently use the application, reducing the memory load required to use the application.

3. **Consistency**
   Consistency in visual design can relate to any group of components and elements that share visual attributes. Though it, in many ways, represents the same values as a design system, as described in section 2.3.3. The structure and theme primarily cover the most important aspects of consistency when applied within a design system.

4. **Personalization**
   Personalization enables the users to customize the application according to their preferences, either in appearance or interface. Appearance customization changes the application's visuals, most commonly through themes or fonts. Interface customization, however, changes the application's functional behavior, such as the default home page or language.

5. **Real-world familiarity**
   Applications can use language and visuals that connect to the real world. The language can reflect local dialects, while the visuals can represent contextually appropriate objects or places, such as an image of the view

from a destination. The use of real-world familiarity uses the *affect* influence type to evoke positive emotions within the user, with the basis of effects *framing* and *priming*.

4.3.2.2 Functional Heuristics

6. **Efficiency**
   Efficient systems aim to reach a destination page quickly while retaining a logical direction flow. Navigation components such as menus and drawers effectively achieve such movements, though they cannot jump to locations out of the scope of the visible menus. Content search and interpretation fix that issue by showing results to inner locations through a list of search results or automatically guessing the following appropriate location with AI.

7. **User control**
   User control ensures that the user always feels that they are controlling the actions in the application. By supporting the reversal of actions through cancellation or back buttons, the user can always control the flow of operations.

8. **System status**
   Systems can affirm the state change after the user's actions through visual, audible, or haptic feedback. Commonly, visual cues utilize splash screens or loading components to visualize actions currently in progress.

9. **Help and documentation**
   Help and documentation ensure users can always find answers upon needing help understanding something about an application. Onboarding help to give an overview of the functionality of an app, as described in section 2.3.1.1. Technical support and documentation help with more specific or in-depth problems that could arise.

10. **Error prevention and handling**
    Error prevention ensures that the user can use the application without fail, either physically or mentally. Physical prevention includes using component touch targets large enough to be pressed by fingers and considering how users generally hold mobile devices. Mental prevention includes the application safely handling invalid user input and avoiding any form of crashing or stalling.

4.4 Design System

This project accounted for some of Material Design's principles during the design stage. Notably, navigation and color theming.

Navigation Figure 4.4(page 48) illustrates the various stages of the architecture's design. The design's primary navigation occurs between the stages bridge, home, menu, and settings. The navigation combines all three directions; *lateral*, *forward*, and *reverse*; see paragraph 2.3.3.1(Navigation). The lateral direction occurs when navigating within a stage between the top-level components—for example, navigating in the home stage from the social page to the calendar page. The forward direction occurs when navigating to a sub-page or a deeper-leveled stage. Stages closer to the home stage generally reside higher in the navigation tree—for example, navigating from the home stage to a menu page and then from the menu page to settings results in navigating two times forward. Finally, the design aims to use a back button that can trigger the reverse navigation direction. From the example of forward navigation, pressing the back button twice from the settings page results in navigating back home.

Color theming In October 2021, Material Design launched a color theme generator called "Material Theme Builder"[64]. The theme builder sets one or more key color accents, as shown in figure 2.10(page 29). Though, only putting the primary accent is sufficient. From there, the builder generates tonal palettes and color roles based on the accents; see figures 2.11(page 29) and 2.12(page 29), respectively. Finally, the builder generates a light and dark theme. This project will use the builder to export the themes into the application.

4.5 Architecture

The application uses a novel architecture built upon the Android operating system specifically designed for the ODS nudge project. The architecture follows the official Android architecture components guidelines, illustrated in figure 4.10. The architecture uses a two-layer approach where the *UI layer* handles the visual elements and their state, and the *data layer* handles data persistence. The *UI state* uses *ViewModels* as state holders and coordinators between the UI layer and the data layer. ViewModels initializes *BG Work*(background work) through the *WorkManager*[1]. The following terms describe the connection between the layers:

- **State** Data used by the UI layer
- **Event** Events that the user produces when interacting with the app
- **App data** Application data from data sources
- **Action** Any query or request that the repository forwards to the appropriate data source
- **BG Thread** A thread that runs outside of the main UI thread

The following sections use information and terminology from the official Android developer guidelines[65].

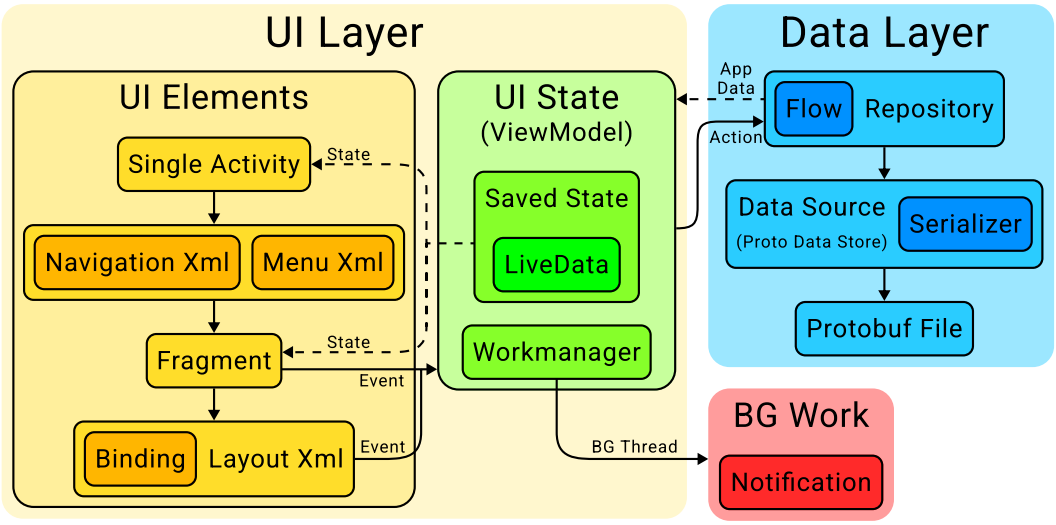


Figure 4.10: Technical design architecture

1. https://developer.android.com/topic/architecture[65]

4.5.1 Terminology

The overall architecture utilizes the following core concepts:

View Views is the core mechanic for rendering visuals onto the screen, utilized through XML[2].

UI controller UI controllers, i.e., activities and fragments, handle all logic directly connected to the views, thus controlling the UI[3].

Lifecycle-aware components Android automatically handles lifecycle states connected to UI controllers. States such as *created*, *resumed*, *paused*, and *destroyed* determine the existence and visibility of their UI elements. Components can, in turn, connect to a controller to automatically perform actions based on lifecycle state changes within the respective controller[3].

Separation of concerns The principle of separation of concerns forces certain parts of the program to operate in isolated areas. In the context of Android architecture components, the principle defines that UI controllers must strictly operate on UI elements. It also defines that state holders, such as ViewModels, and components within the data layer, must operate only in their area[4].

Unidirectional Data Flow (UDF) *UDF* enforces that state and events flow in opposite directions. Thus, the state flows towards the UI elements, while events flow towards the data layer. Conforming to *UDF* ensures data consistency and complements the following term, *SSOT*[4].

Single Source Of Truth (SSOT) *SSOT* enforces only one owner capable of changing data, i.e., the data source[4].

2. https://developer.android.com/reference/android/view/View[65]
3. https://developer.android.com/topic/architecture/lifecycle[65]
4. https://developer.android.com/topic/architecture[65]

4.5.2 UI Layer

The UI layer describes how the architecture interacts and saves data strictly related to the visual elements, such as the text element of a title. Thus, concerning data such as the UI state, not the data persisted to disk. The layer splits into the two sub-layers described in the following sections; firstly, focusing on elements in the UI and secondly on the UI state, thus conforming to the principle of separation of concerns.

4.5.2.1 UI Elements Sub-layer

The elements sub-layer uses a single activity with multiple fragments, shown in figure 4.11. The activity is the top-level UI controller and parent lifecycle owner of all other lifecycle-aware components. If the top-level UI controller dies, all other UI-bound components die. Both fragments and the activity listen to state changes from the UI state sub-layer.

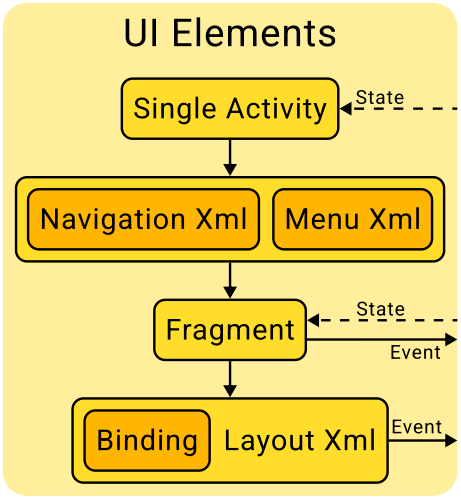


Figure 4.11: UI elements

The activity uses navigation UI combined with menu components to navigate to fragments[5]. The fragments use view binding to connect to the layout XML files, where the actual views render visuals to the screen[6]. Finally, events triggered by the user result in calls up to the UI state sub-layer, coming directly from the layouts or the fragments.

5. https://developer.android.com/guide/navigation[65]
6. https://developer.android.com/topic/libraries/view-binding[65]

4.5.2.2 UI State Sub-layer

The UI state sub-layer acts as a state holder and coordinator between the UI layer and the data layer, as shown in figure 4.12. Primarily, ViewModels performs all tasks in this sub-layer. The ViewModel saves the state using non-persistent data storage with the ViewModel and saved instance state[7]. The state gets stored in the ViewModel as *LiveData*, which essentially acts as lifecycle-aware data[8]. The ViewModel receives application data updates, such as the user's first name and age, from the data layer. Afterward, the ViewModel converts the data to LiveData if necessary. The ViewModel handles user events, though usually forwarded to the data layer as an appropriate action.

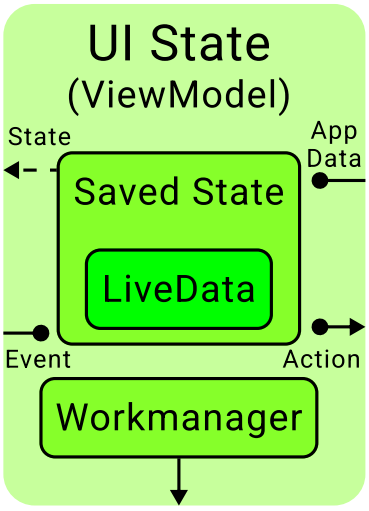


Figure 4.12: UI state

Finally, the ViewModel also initializes the WorkManager, which starts up background work, such as a task for sending notifications.

4.5.3 Background Work

Background work runs as tasks initialized by WorkManager. The tasks can run in three modes: immediate, long-running, and deferrable. Immediate work runs right after WorkManager starts the tasks. Long-running works the same way as immediate, except that the work usually runs for an extended period, such as tracking tasks. Lastly, deferrable tasks run after a scheduled time[9].

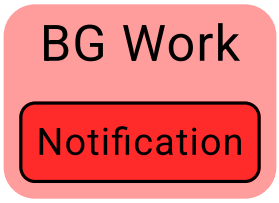


Figure 4.13: Background work

This architecture primarily utilizes background work for notifications, as shown in figure 4.13. The notifications can utilize all three task types, though primarily long-running tasks, such as tracking location during a trip.

7. https://developer.android.com/topic/libraries/architecture/saving-states[65]
8. https://developer.android.com/topic/libraries/architecture/livedata[65]
9. https://developer.android.com/topic/libraries/architecture/workmanager[65]

4.5.4 Data Layer

The repository coordinates input and output data from the data sources and the UI layer. The UI layer can listen to changes in the data sources through *flow* objects stored in the repository[10]. This architecture relies solely on local persistent data storage, thus not needing to call any remote back-end, shown in figure 4.14.

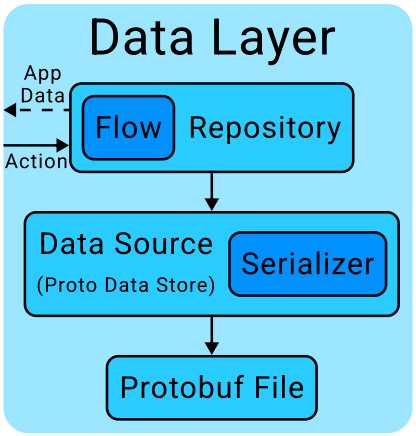


Figure 4.14: Data layer

The scale of the project does not enforce a data source as complex as structured data, e.g., an SQL database. Additionally, the limited hardware size of mobile devices limits the storage capabilities. Modern systems increase in complexity and size; therefore, they must optimize parallelly to reduce storage usage for cheaper devices. The architecture thus implements the proto data store for persistence, which utilizes Protocol Buffers to serialize structured data[66]. The serializers essentially compile the data into highly space-efficient data to store or send to remote sources. Thus, relevant to smaller-scale mobile phone architectures.

10. https://developer.android.com/kotlin/flow[65]

5

Implementation

The implementation of this project consists of creating an Android app with the name *Plancite*. The name combines the words “planet” and “incite” as a metaphor for motivating people to take care of the planet and its environment. Plancite utilizes all the architectural components described in section 4.5. Plancite contains all the pages from the stages; *Bridge*, *Home*, *Menu*, and *Settings*, presented in figure 4.4(page 48). It also renders most of its components using the design system Material Design, discussed in section 2.3.3.

5.1 Technical Tools

This project implemented the application using the development environment Android Studio. Android Studio is an IDE that primarily develops Android applications, running them directly on hardware devices or emulators. This project tested the application using hardware and emulators. The project ran on the mobile device “Pixel 6 Pro”, using the Android operating system versions 12 and 13.

Android uses an API versioning system with its tools and frameworks. Older versions of Android only support some of the newest API versions. The project focused on using modern versions of tools and frameworks that use more recent versions of the API. Therefore, the project set the minimum supported API version to 29, equivalent to the Android operating system version 10.

Consequently, only Android devices running on version 10 or higher can install and run this application.

5.2 App Icon

Plancite has a novel icon designed to fit the aspiration and name of the application, shown in figure 5.1. The standard app icon shows the figure of a person with green thoughts shaped inside the letter “P” as the first letter of the name Plancite. The leaves grow from the ground using the brown color, symbolizing thinking of nature and our planet Earth. Android 12 and upwards added support for dynamic color themes across the entire device, with one feature enabling monochrome-themed icons. Therefore, Plancite also supports a monochrome version of its icon, shown in the context of a blue theme in figure 5.1(c).

(a) Standard **(b)** Monochrome **(c)** Monochrome with example color theme

Figure 5.1: App icons

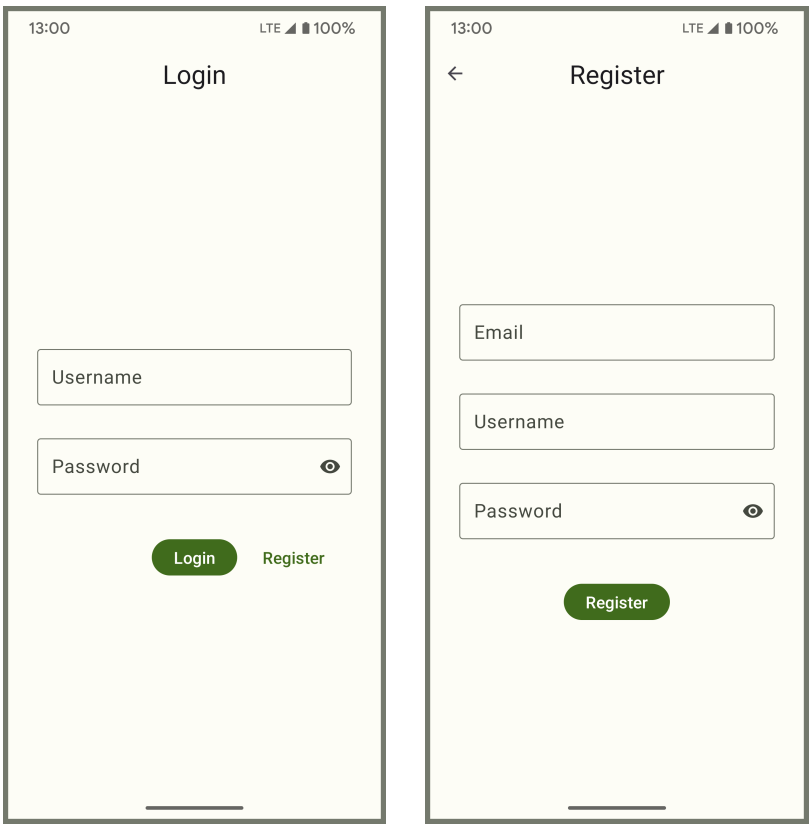
5.3 App Launch

This section and the following sections describes each stage of the implementation. Each section shows screenshots of the application with figures of the architectural components used in the respective stages. The figures, such as 5.3, show the used components with colors and those not used as grayed out.

Launching the application through the app icon can result in two destinations; the *login page* or the *home page*. The login page marks the start of the bridge stage. The user can either log in or register as a new user, as shown in figure 5.2. Login redirects directly to the home page, and registering redirects to the *onboarding pages*. Registering a user persists the email, username, and password to disk using the data layer. Section 5.7 further described how the architecture persists the data.

Figure 5.3 illustrates the architectural components used when launching the app. The activity starts the app by fetching the login state from a ViewModel called `EntryViewModel`. The ViewModel handles events and states related to users logging in and registering. The ViewModel returns the state as *LiveData* to the activity and uses navigation to move to either the login or home page. Users could have opened the application beforehand and navigated to a different destination, such as the *calendar page*. Such a scenario would result in the activity navigating to the topmost destination on the navigation stack instead of the home page. This flow of actions ensures that the application continuously checks the users' login state when opening the app and redirects to the appropriate page.

The activity acts as a top-level view controller and runs as long as the application runs. Every stage of the architecture uses the activity in some way or another. So the following sections gray out the activity for simplicity's sake.



(a) Login page **(b)** Register page

Figure 5.2: Login and register page

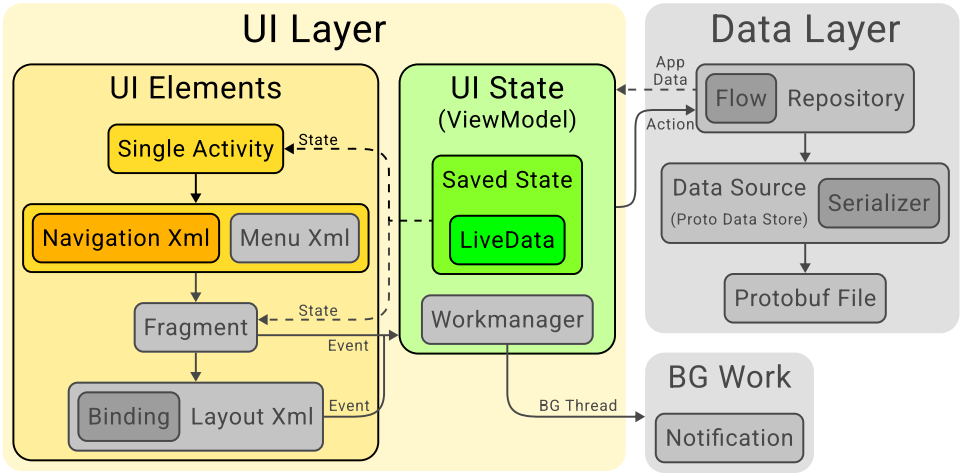
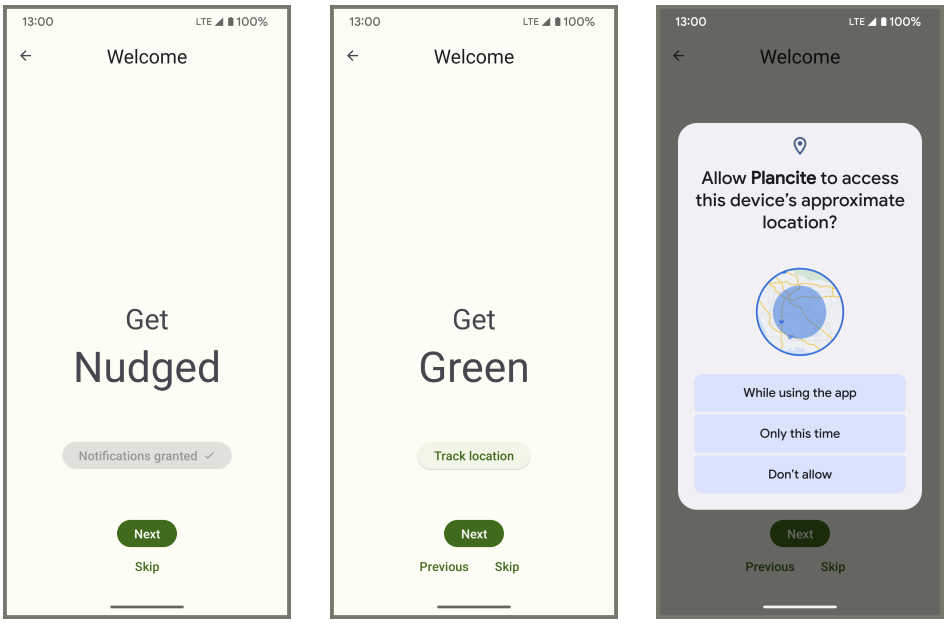


Figure 5.3: App launch implementation

5.4 Onboarding

The onboarding pages, shown in figure 5.4, present the option to allow permission for notifications and location tracking. The pages can navigate back and forth using two bottom buttons. Figure 5.5 illustrates the architecture, where each page represents a fragment with a layout binding. Figure 5.4(a) shows the notification request button as already granted. Thus, the user had previously granted the application notification access. In comparison, figure 5.4(b) shows the application lacking location access.

The application does not currently persist any state data related to permissions due to the Android operating system saving such information. Therefore, the fragments retrieve the current active permissions on each app launch. The ViewModel uses a custom-made permission helper to request permissions. The helper simplifies requesting multiple permissions and retrieving their results. The ViewModel finally returns the permission state as LiveData to the fragments.



(a) Notification page **(b)** Location page **(c)** Accept permission

Figure 5.4: Onboarding pages

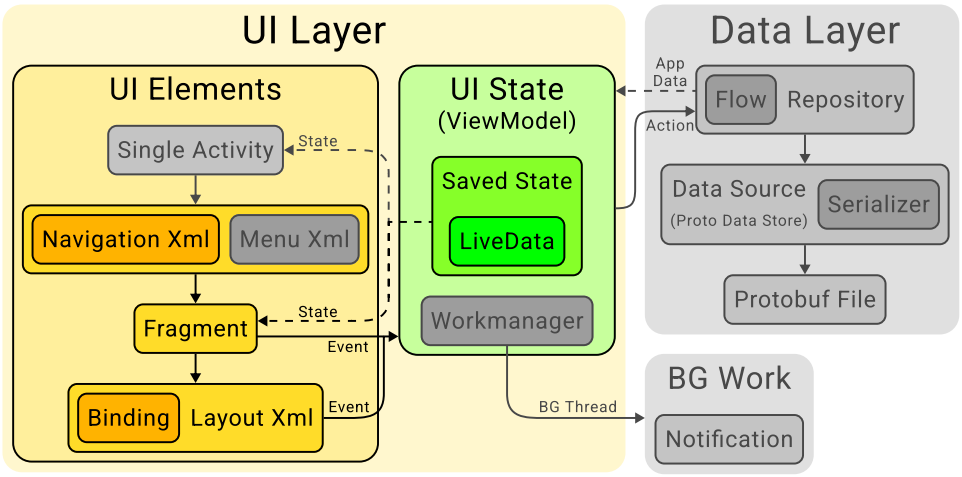


Figure 5.5: Onboarding implementation

5.5 Home

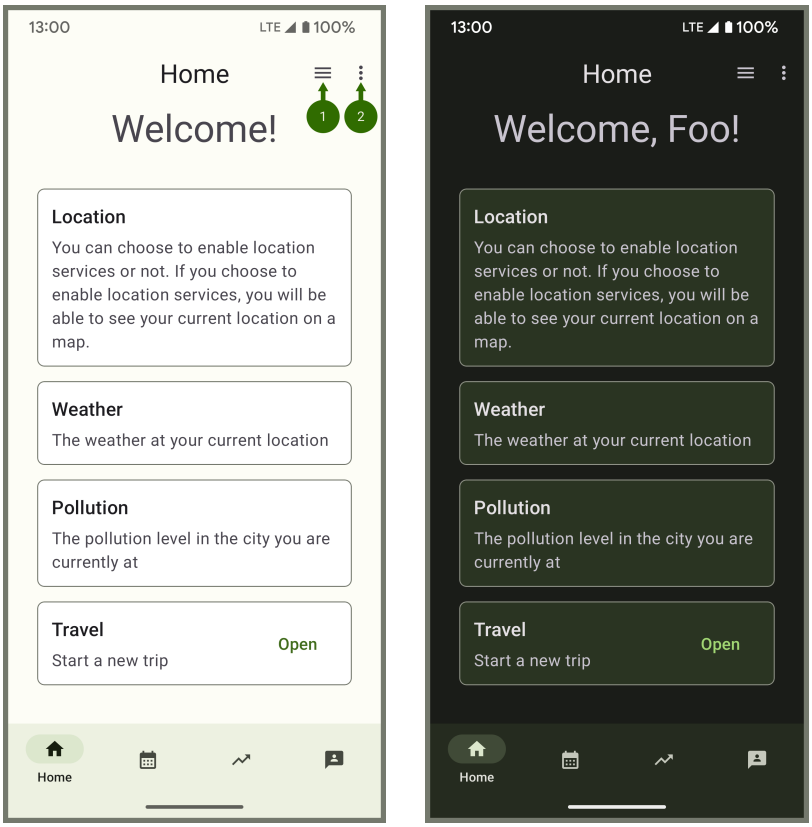
The home stage has four main pages with icons on the bottom navigation; home, calendar, progress, and social. Figure 5.6 presents the default home page after logging in or completing the onboarding pages. The figure illustrates the support of light and dark system theme support. Figure 5.6(a) shows the app's appearance when the Android system has the light theme enabled, while figure 5.6(b) shows the dark theme. The theme follows the color guidelines in section 2.3.3.2, ensuring an appropriate color contrast ratio and harmony between components.

The home and *menu* pages present design cards describing the data the application should render at that location. The cards use a container with an outline, a solid background color, a title, and content. So the cards act as placeholders for future implementation of incomplete content. The `Weather` card, as an example, has the title "Weather" and the content as a text element.

Figure 5.6(a) shows two green circles at the top right, (1) and (2). They point to the *hamburger* and *kebab* menu icons, respectively. The hamburger icon opens up the menu drawer; see section 5.6. The kebab icon grants the user access to the help page from anywhere in the app.

The home page uses most of the components in the UI layer, as illustrated in figure 5.7. The bottom navigation uses the menu XML to generate its item as pages. The navigation framework saves in memory the current destination on a stack. As long as the application does not terminate entirely, each launch will return to its current destination. The bottom navigation only shows up in the home stage. The architecture achieves this by using a separate navigation structure specialized to the home stage. The architecture also preserves the back stack with the home stage as the top-level destination. So if the user presses the back button enough times, as shown in section 5.6, they will eventually return to the home stage. Finally, pressing the system back button on the home page closes the application.

The home stage fetches application data from the data layer through a ViewModel. Currently, only one visible data shows up on the pages. Figure 5.6(b) shows how the title will include the user's first name when the data layer contains the name.



(a) Home page light theme
(1) Hamburger menu (2) Kebab menu

(b) Home page dark theme

Figure 5.6: Home page

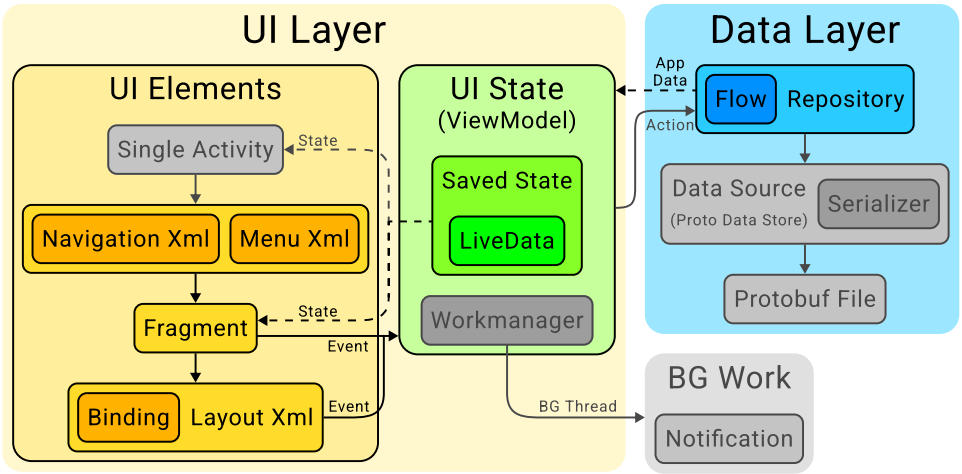
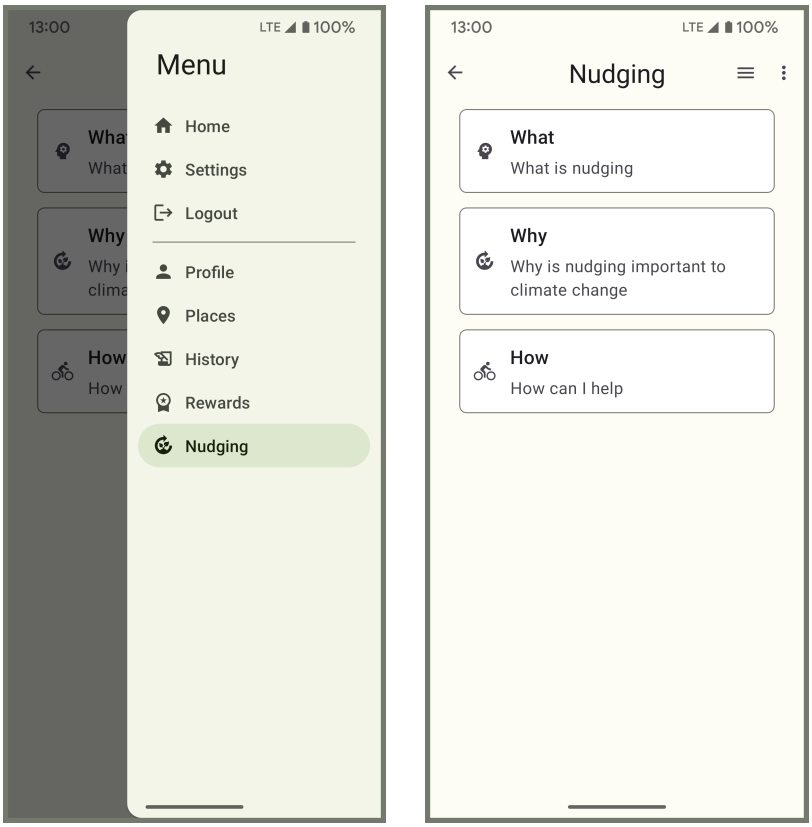


Figure 5.7: Home implementation

5.6 Menu

The menu presents a side navigation drawer with the menu pages as well as a button to home, a button to settings, and the ability to log out, as shown in figure 5.8(a). The drawer uses menu XML to generate the items. Each menu page contains cards as described in the previous section, shown in figure 5.8(b). The back button in the top left corner appears on all non-top-level pages. The *nudging* page is one destination away from the home stage. So pressing the back button from there navigates directly back to home.

Figure 5.9 shows that the menu pages send events to the data layer but do not receive any state. Currently, none of the cards use any state information. Though, the logout button changes the login state of the user. So clicking it propagates the user event to the data layer and persists the state change to disk. The activity listens on the login state. When the state changes to “logged out”, the activity removes all destinations from the navigation stack and navigates to the login page.



(a) Menu drawer **(b)** Nudging page in menu

Figure 5.8: Menu drawer with one of its pages

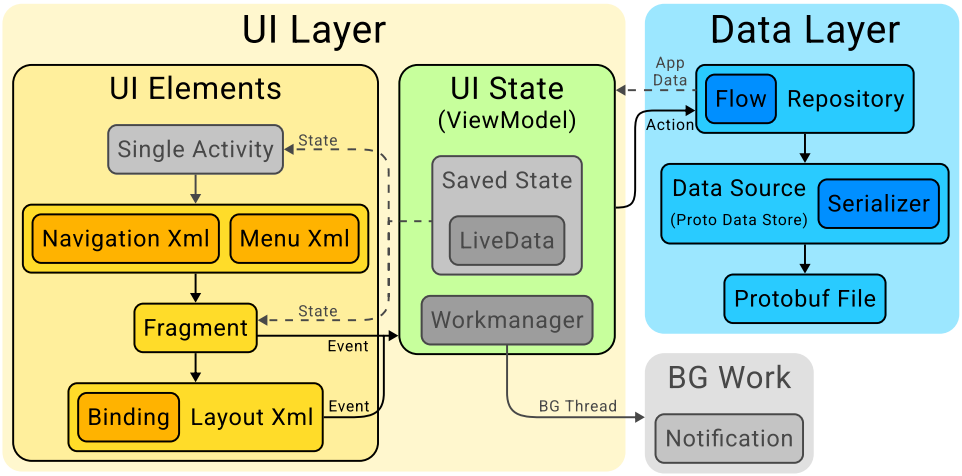


Figure 5.9: Menu implementation

5.7 Settings

The top level of the *settings page* has four buttons for its four pages, shown in figure 5.10(a). Figure 5.10(b) and 5.10(c) present the *account page* and *preferences page* with mock user data filled in. All the fields in the settings pages connect to the data layer in the architecture, thus persisting the data to a local data storage, as shown in figure 5.11.

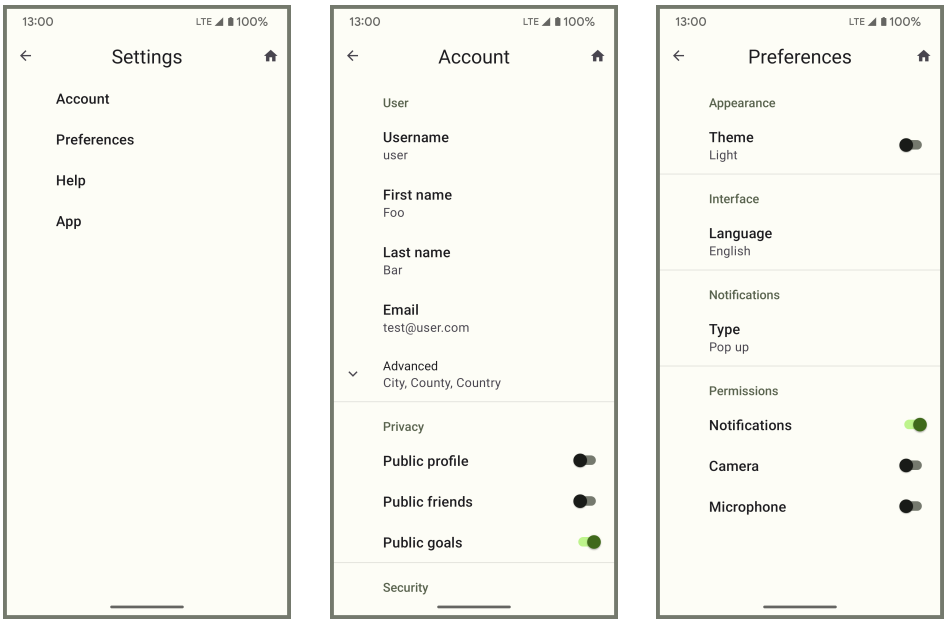
Each settings page connects to its respective repository, such as the account page connecting to the account repository. The repository contains an instance of the relevant data sources. Currently, the project only uses local data storage with the *Proto DataStore*. The DataStore uses protocol buffers to serialize the structured data onto a file locally on disk. Each DataStore persists its data to one file, whereas each file uses a `.proto` file to define the schema of its content. The schema follows the syntax of the protocol buffer language. The account repository contains an *account* DataStore that also serializes data to a file called `accountProtoStore.pb`. When an action asks for specific application data, the data layer reverses the data flow. The data source deserializes the data source and propagates the data to the repository. The repository finally converts the data to a flow object listened to by ViewModels.

The repository approach can easily extend to multiple data sources, such as using structured data or a remote database. The Android architecture components recommend the library "Room" to store structured data with *SQLite*[1]. Similarly, they recommend the library "Retrofit" as an HTTP client to connect to remote servers[67]. Figure 5.12 illustrates extending the data layer to use both room and retrofit.

The data stays stored for as long as the user does not uninstall the app or register a new user. New registrations trigger a call to a ViewModel to reset all stores. The action goes through all repositories and deletes all saved data within their respective sources.

Figure 5.11 shows that the settings pages do not rely on layout XML files. That is because the pages use a preferences settings library to generate the UI of the pages. The library simplifies managing settings and persisting the data to disk.

1. https://developer.android.com/training/data-storage/room[65]



(a) Top level settings page **(b)** Account page in settings **(c)** Preferences page in settings

Figure 5.10: Settings pages

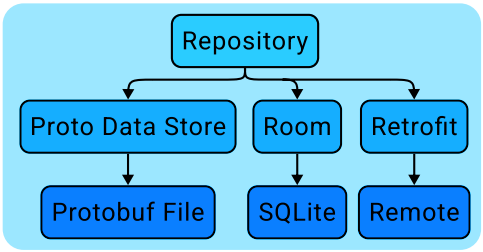


Figure 5.12: Multiple data sources

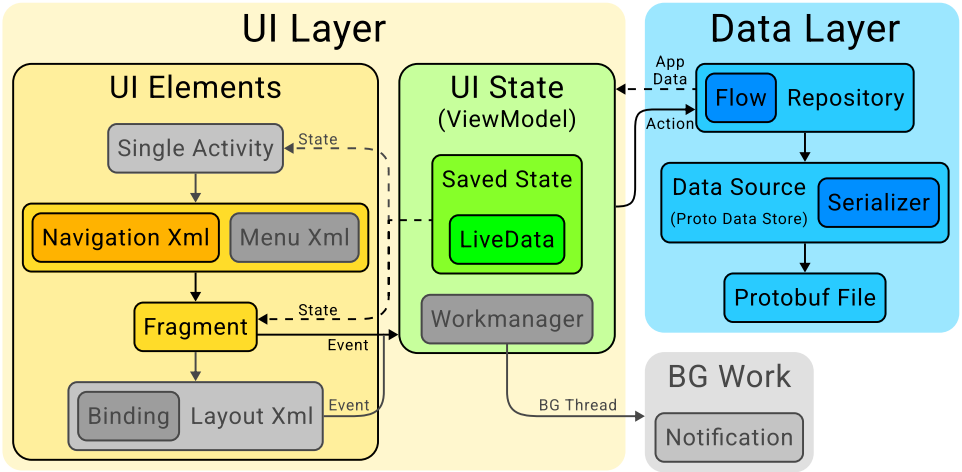


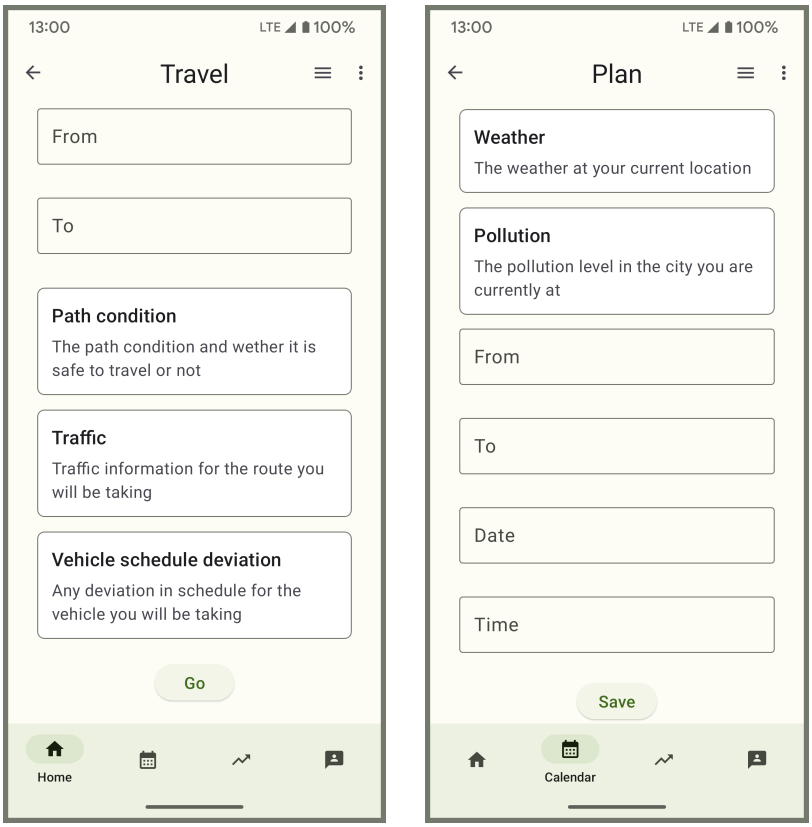
Figure 5.11: Settings implementation

5.8 Notifications

Figure 5.13 represents the two sub-pages *travel* and *plan*, reached from, respectively, home and calendar. The sub-pages act as the two methods to register new trips, starting immediately or in the future. The *Date* and *Time* field in figure 5.13(b) opens up a system date picker and time picker, respectively. The buttons *Go* and *Save* launch the notifications presented in figure 5.14.

The ViewModel contains an instance of WorkManager, as shown in figure 5.15. When the user clicks one of the two buttons, the ViewModel dispatches background work through the WorkManager. The background work runs on a thread outside of the main UI thread. This project only launches background work specialized in sending notifications. The workers utilize a custom-made notification library to send system notifications. The library simplifies sending notifications and declaring different notification types and their content.

Figure 5.14(a) presents the *travel* notification. It uses immediate background work, which displays on the screen immediately after clicking the *Go* button on the travel sub-page. Figure 5.14(b) presents the *Reminder* notification. It uses deferrable background work, which in this case, displays on the screen after a timer of 10 seconds. Figure 5.14(c) shows the reminder notification expanded in the Android system notification dropdown. The expanded version includes the application name “Plancite” as well. These notifications confirm the functionality of notifications and background work by using WorkManager.



(a) Travel sub-page **(b)** Plan sub-page

Figure 5.13: Home sub-pages

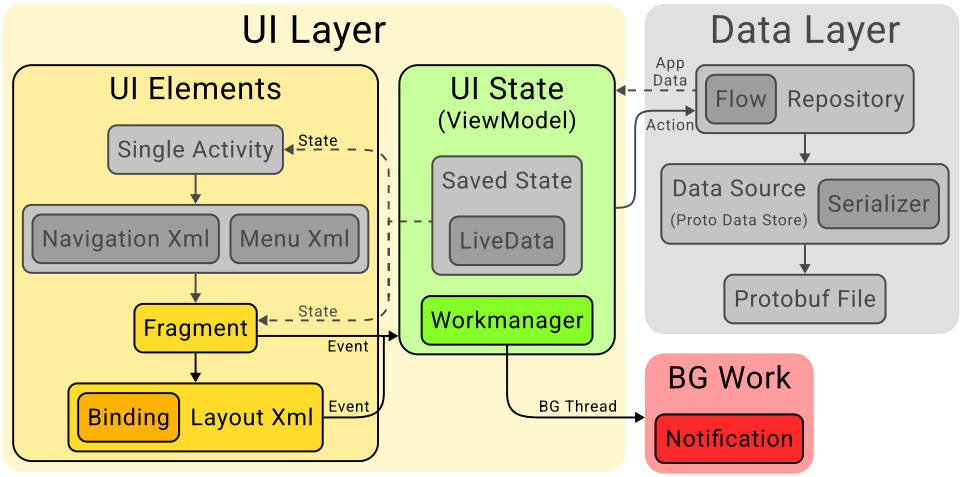
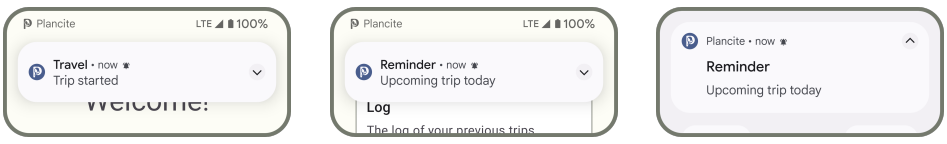


Figure 5.15: Notifications implementation



(a) Travel **(b)** Reminder **(c)** Reminder in system dropdown

Figure 5.14: Notifications

5.9 User Experience

The application utilizes several user experience (UX) principles and best practices. Notably, the device hold method, grids, color themes, typography, and iconography.

Hold method Section 2.3.2.1 describes various methods people use to hold their devices. Notably how they often use their thumb to touch their device. Approximately 90% of people worldwide are right-handed[68]. So this application focuses on design optimized for right-hand usage. The application places the hamburger menu and the side menu navigation drawer on the right side of the screen. Consequently, right-handed people can more easily access the menu items with their thumb when holding the device in the cradled position or one hand, see figures 2.2(page 21) and 2.3(page 21).

Grids The application places components using the baseline grid and the responsive layout grid, further described in paragraph 2.3.3.1(Layout). All components align to the baseline grid of 8dp; see figure 2.8(page 27). Therefore, all margin and padding variables in the application use dp with a value of a factor of 8. All the pages use a single margin size of 16dp, thus following the guidelines of the responsive layout grid's margins; see figure 2.7(page 26). The application currently uses a one-column layout for most of its pages, with the settings page as an exception.

Color theme The application implements a generated light and dark color theme with the Material Theme Builder. The theme connects to the current system theme, such that if the Android system theme uses light, the application will automatically also use its light color theme. The application imported the builder theme as XML resources.

Typography The application uses a type scale with the default typeface of Android applications, Roboto, see figure 2.14(page 31). All text components use regular *weight*, 0px letter spacing, and the "Sentence" case type. Primarily, only the size attribute differs between the type styles. For example, the floating action buttons use text components that are 20 scalable pixels (sp), such as the travel page's *go* button. While the title text components use a slightly larger size attribute as 32sp.

Iconography The application uses default icons shipped with the IDE Android Studio. The icons follow an icon theme with the filled icon type, as shown in figure 2.15(b)(page 31). The icons use the Apache License, Version 2.0[69], making them free to use and alter their source code.

6 Discussion

This chapter discusses the advantages of the implementation and how to improve on its existing and missing features. Section 6.1 discusses the advantages of the structure of the codebase and how it matters for future development. Section 6.2 discusses how the application conforms to the usability heuristics and what heuristics can improve. The current state of the application does not allow user testing due to the lack of the three application requirements discussed in section 6.3. The application also can improve on various areas by introducing additional features, discussed in section 6.4. Finally, section 6.5 discusses how to use the application's architecture components in other nudge-related projects.

6.1 Code Structure

The codebase follows a highly organized structure that tightly connects to the architecture. The code splits the pages into separate logical units into several abstraction levels. Most pages implement a fragment file and a layout file connected with view binding. Each stage of the application shares ViewModels as a method to communicate with other fragments and the data layer. For example, the home stage uses a ViewModel called `MainViewModel`, where all pages, such as calendar and social, connect to that ViewModel. The repositories split into separate files and connect their data to the relevant data sources. Similarly, the protocol buffer schemas also use separate files.

The files follow consistent naming to improve organization and structure. An example flow of operations looks like this. The account settings page uses the fragment, `AccountFragment`, to send an event to the ViewModel, `SettingsViewModel`. The ViewModel forwards the action to the repository, `AccountRepository`. The repository updates the data source, `AccountDataSource`, which serializes the data with the schema from the `account.proto` file.

The application uses custom-implemented managers for permissions, notifications, and logging. They act as utility functions split outside the user interface (UI)-related and data-related files. They enable customization and simplification of using the respective libraries and system functions.

The organization of the code structure results in a codebase that developers can easily extend and modify. Logically split units and actions also result in easier debugging during development.

6.2 Heuristics Conformity

The project described a novel set of usability heuristics to guide the development of the front-end aspects of the application. This section will discuss the usage of each heuristic and how the application can improve on them in future development.

1. **Minimalism**
   The architecture depends on a relatively complex structure with many pages. The project conforms to minimalism by logically separating the various destinations into navigational structures. The structures include a bottom navigation bar, a side navigation drawer, and separate settings pages. The structure enables a cleaner visual layout that more easily can display only the core content relevant to each page. The application uses design cards as placeholders for future content on each page, as described in section 5.5. Whereas on each page, the cards display in a one-column grid. Future implementations of the cards could use complex grids of content with multiple columns and rows. Such content could conform to minimalism by using consistent column sizes and gutter padding sizes, as described in paragraph 2.3.3.1(Layout).

2. **Recognition over recall**
   The application uses its logical navigational structure to reduce memory recall. Notably, icons in the bottom navigation and side navigation improve pattern recognition. Thus, the application does conform to the heuristics of recognition over recall, though it can improve in two areas.

Firstly, the settings pages should include icons on the top-level setting page. Secondly, future development of the design cards should utilize multimedia when possible to naturally distinguish the content.

3. **Consistency**
   The application primarily conforms to the consistency heuristics through its usage of the design system. The design ensures consistency between shared visual attributes of the components. Particularly theming of typography, iconography, color, and shape covers a large part of those shared attributes. Together they define a brand for the application. Thus, users can uniquely identify and distinguish the application from others.

   The application uses all of the theming areas except for the shape. Shape schemes could further distinguish the application from others using the same design system, for example, using asymmetric shapes with one cut corner. Currently, the iconography depends on a publically available set of icons, thus possibly using the same icons as other applications. Designing custom icons can be tedious, though it would drastically increase the brand's uniqueness. Finally, the color theme uses the color theme builder with a custom primary color accent. Future work could include the support of adaptive color themes, thus automatically adapting to the Android system theme. Though, only Android 12 and upwards support adaptive color themes.

4. **Personalization**
   The project included the settings page *preferences* with the categories *appearance* and *interface* to support the heuristic of personalization. The pages aim to support the personalization of visual and functional aspects of the application. Currently, they contain toggles related to the application theme and language, though they do not do anything yet. Future work should connect the settings to actions that change the application's behavior. Additionally, the interface could assign options to change the UI layout, such as changing the default home page.

5. **Real-world familiarity**
   Currently, the application conforms to familiarity through a couple of icons referencing real-world objects, shown in table 6.1. Future work could improve familiarity through the design cards. For example, the location card could add an image of the city from the user's current location.

Icon	Home	Calendar	Profile	History	Rewards
Real-world	House	Calendar	Person	Pen and paper	Medal

Table 6.1: Real-word familiar icons

6. **Efficiency**
   The architecture splits the content logically so that the most helpful content lies closer to the home page. Thus, the application conforms to efficiency by making navigation to common actions faster. Future work could improve the heuristic by including a search bar on the top that can directly move to deeper locations, such as one of the settings pages.

7. **User control**
   The application supports user control through a back button at the top left corner. The back button enables users to navigate backward from anywhere in the application. The button acts as a system back action, closing the app by pressing back at the home page. Future development could add action flows with several steps of actions. Such actions could add cancellation logic with buttons, further emphasizing user control.

8. **System status**
   When opening apps, Android displays a splash screen by default. The screen displays the app icon with a plain background. Additionally, the bottom navigation bar shows a short animation on clicking one of the items. Those two feedback methods act as visual feedback conforming to the heuristic of presenting the system's status. Devices that enable sound on touch system-wide also support audible feedback. Each button click and menu item click in the application result in a sound. Future development that displays tracking information or data from a remote server can update in real-time. The pages displaying such information can include loading components to conform to system status further.

9. **Help and documentation**
   The application uses onboarding pages to introduce the user to the primary usage of the application. The pages only offer the option to set notification and location tracking permissions. Future development can add information regarding why the system needs those permissions and optionally explain common usage of the application. The system uses the kebab menu icon to have an always-available button on the *help* settings page. The settings page offer help in the form of support and a shortcut to open up the onboarding pages.

10. **Error prevention and handling**
    The side navigation menu displays on the right side of the screen to accommodate thumb use and right-handed people. The application also supports mental error prevention using simple e-mail formatting and number formatting rules in the relevant text fields. Future development could further improve handling by accounting for left-handed people. The implementation could use a layout with the menu on the left and a respective toggle in the interface category of the preferences settings page.

6.3 Application Requirements

The application has a strict implementation of the architecture, emphasizing navigation. However, the following sections describe areas needed for a working system capable of performing practical user testing.

6.3.1 Design Cards

The home and menu stages contain cards as placeholders for missing content, described in section 5.5. The first step to improve the application would be to go through each card by designing and implementing the content in the card description. The cards broadly describe the content to render, however, not the design. The content varies based on the card, with some considered harder to implement than others.

The *What* card on the top of the nudging page in the menu should have a relatively straightforward implementation. The card says: “What is nudging”. The card is strictly informational, thus only representing simple text. The developer can implement the card by adding text elements describing how the application uses nudging toward the users. The developer could include supporting images to convey the message further.

On the other hand, the *Location* card at the top of the home page requires more effort. The card says: “You can choose to enable location services or not. If you choose to enable location services, you will be able to see your current location on a map”. The developer can use a third-party service, such as google maps, to implement the card. Including new third-party services in architecture usually requires more effort than using existing components. The map also requires permission to track the user’s location. The implementation can include various supportive text elements related to the user’s current location.

6.3.2 Background Work

The application can use long-running background work to enable tasks such as tracking GPS location. The task can use the location to suggest transportation alternatives from the user's current location.

Currently, only notifications run using background work. Two example notifications connect to the Travel and Plan page, as described in section 5.8. The application lacks the implementation of the four notification types presented in figure 4.9(page 53). Additionally, the application can implement various importance levels, see section 2.3.1.2, to further distinguish the notifications from each other.

6.3.3 Back End

Currently, the architecture utilizes non-structured data with *Protocol Buffers* yielding primarily space-efficient storage. However, future additions to the application might require storing more complex and extensive data sizes. A solution would be to implement local database storage using the *Room* library connecting to an *SQLite* database[1].

The ODS nudge project does not have an existing back end to connect the application to, thus the reason for the architecture utilizing exclusively local data storage. However, after the ODS nudge project implements the back end, this application can connect to it using the HTTP client library *Retrofit*[67].

The architecture uses data sources as neutral components for data persistence. Thus, the application can implement Room and Retrofit as additional data sources without restructuring the architecture, as visualized in figure 5.12(page 73).

6.4 Additional Features

The application also can improve in various areas by introducing additional features. Some features do not prevent a usable system, though their advantages can result in a more polished end product other than only supporting the bare minimum functionality.

1. https://developer.android.com/training/data-storage/room[65]

6.4.1 Third-Party Services

Generally, users should have relatively few apps performing the same task. The application can improve the user experience (UX) by connecting to third-party services commonly used in the Android environment, thus omitting that responsibility from the application. A calendar app ensures that users connect their scheduled events to the application. The application can use the contacts app to connect contacts to users' friend lists or invite new users. Finally, the application can use web links to the default browser app to show external information commonly related to support or to provide statistics from credible sources.

6.4.2 Architecture

The current architecture design deliberately left out two structures due to their size and complexity. The first one, being multi-module-based architecture, splits the entire codebase into a series of logically separate modules, also referred to as libraries[2]. The structure decouples components enabling a more straightforward reuse of components while simultaneously relaxing the dependency between components. When the complexity and size of the application increase, the structure generally solves problems related to reliability and scalability.

The second structure, Jetpack compose, utilizes *Composables* as a replacement for the Views for rendering visuals[70]. Composables yield improvements in accelerated development through more intuitive and fewer lines of code[3]. The Composables, e.g., composable functions, render the UI programmatically instead of using XML. The framework, released in 2021[71], is relatively new compared to Views and thus still has features under development. However, given the advantages of composables, a future effort to restructure the framework would be beneficial.

6.4.3 Security

The account sub-page of settings can enable two-factor authentication, currently only storing the toggle as true or false. The application can improve security by setting the two-factor through third-party one-time code generation services such as authentication apps. Moreover, hardware that supports biometrics can use authentication, with fingerprint or face recognition.

2. https://developer.android.com/topic/modularization[65]
3. https://developer.android.com/jetpack/compose/why-adopt[65]

6.5 Usage in Other Projects

The application uses an architecture specifically designed for the use case of this project. However, other nudge-related applications can utilize components from the application. The following sections discuss other use cases with the reusability of the architecture, section 6.5.1, and the content explicitly designed towards nudging, section 6.5.2.

6.5.1 Reusable Architecture

The architecture uses navigational components with menus neutral to the application context. Most modern mobile apps use a navigational structure of one to two layers, with an additional settings menu on the side. Thus, other applications can utilize the same file structure regardless of the use case. The main migration effort would involve replacing the number of top-level pages in each menu and renaming the destinations. The current state of the application also uses neutral content for the settings pages, thus making them eligible for cross-application usage.

6.5.2 Content Designed for Nudging

The architecture uses some of the pages presented in section 4.3.1 specifically to support various principles within nudge theory. The pages related to the *primary task* quality, such as the *home* page, use content connected to the application's use case—in this project relating to green transportation. Other pages, such as the *social*, *progress*, and *rewards* pages, use content relatively neutral to the use case for the application. Most settings pages also utilize nudge theory to some degree with neutral content. Thus, other nudge applications can reuse those pages and fit them into their use case. For example, the score parameter on the social page could present the user's activity level in a fitness-related application.

7 Evaluation

This chapter looks back on the progress and accomplishments of the project. It starts with section 7.1 by highlighting the contributions to the ODS nudge project and other possible nudge-related applications. Section 7.2 reflects on initial research-related questions and experiences made during the process of working on the project.

7.1 Contribution

The project's contribution includes theoretical research and technical implementation. The contributions create a basis for future development and research within the same project and other nudge-related projects.

Theoretical Presents a connection between the front end and nudge theory, presented as design requirements in section 4.3.1. The figures referred to as feature maps/stages, aim to provide a simple overview of the connection for future research. The simple design of the figures also makes future addition and modification straightforward. Additionally, the project presents novel usability heuristics that provide guidelines to appropriate user experience (UX) and user interface (UI) principles related to nudging, see section 4.3.2. The heuristics guide the developer toward designing the application's content.

Technical Implements an Android app using a robust architecture. Future development can easily extend the architecture, such as efficiently adding more data sources, as discussed in section 6.3.3. The architecture additionally decouples components. Overall, the architecture enables other nudge-related projects to utilize separate navigational structures and single pages, discussed in section 6.5. The application presents a working prototype utilizing various psychological requirements and usability heuristics. Additionally, the application utilizes best practices in UX based on the guidelines of Material Design, further described in section 2.3.3.

7.2 Reflection

The following subsections reflect on the start of the project with its initial research questions and how the process shaped the project.

7.2.1 Research Questions

The project started with the overall question from section 1.3:

Question: *"How can mobile interfaces present digital nudges?"*

Section 4.2 expands on the definition of digital nudges in a front-end setting. The section concludes with how most aspects of applications can utilize nudging, including inside the app and notifications. This application uses design cards as placeholders for future content development, which can represent nudges in the application. Furthermore, section 4.3.1 discusses how applications' architectures can include aspects of nudging. The section describes how a set of system requirements link to the architectural components of the application. The system requirements then link to psychological principles such as theory within nudging. The project furthermore specified sub-questions related to the overall question.

Sub-question 1: *"How can mobile interfaces utilize motivational incentives?"*

The research found principles such as the six principles of persuasion, cognitive bias, and nudge influence types. They describe how to influence people, though not in a digital environment like mobile interfaces. The project chose the system requirements as an optimal approach to connect the principles to the interface. The system requirements describe a set of non-functional qualities and functional requirements relating to

psychological principles and features of a digital environment. Such as discounts relating to the *incentives* quality and the *hyperbolic discounting* effect. Thus, the application effectively uses motivational incentives by connecting nudge theory and system requirements. Section 4.3.1 presents how the project uses a selection of the system requirements through the architecture components.

Sub-question 2: "*How can mobile interfaces utilize principles regarding user experience?*"

The research started by studying various principles within UX. However, the project needed a method to categorize the principles while specializing them in a specific context. The project chose usability heuristics as a framework to provide guidelines to the UX within mobile environments. The project created novel usability heuristics inspired by other sources[39, 40]. The project explicitly designed the heuristics for mobile nudging systems. Each heuristic serves as general guidelines, whereas the developer can implement them in their context according to their needs. Thus, the mobile application utilizes principles within UX through novel usability heuristics. This project emphasizes using the *consistency* heuristic, implemented using the design system, Material Design. The design system further provides UX guidelines, such as guidelines for a set of visual and interactable components. For example, this project uses the bottom navigation bar as a design system component. Section 4.3.2 presents the connection of the rest of the heuristics to nudge theory, such as minimalism relating to *simplification*.

7.2.2 Experiences

This thesis was written with architecture and structure in mind. The research phase started by studying the theory, then compressing the information into a logical structure. A sizeable amount of time was spent figuring out how to make the information tie into the upcoming design of an application. I had yet to gain experience with most of the tools and frameworks in the development and design process. Thus, additional time had to be allocated to learning the technology. However, the tools were explicitly chosen as the most up-to-date and recommended within the field. The structural thinking throughout the project certainly made progress at times slower than preferred. However, it resulted in a robust architecture and thorough design specification connecting the theory to the application. Furthermore, it prepared me for the future with modern tools and frameworks used in front-end development.

8

Conclusion

This project aimed to research how nudges can integrate into the front end of a mobile application. The literature study started as a two-fold process. The first part focused on the psychology behind nudging and its connection to persuasion and persuasive systems. Then, the second part focused on front-end theory using user experience (UX) and user interface (UI).

The front-end theory centered on two areas; common mobile principles and design systems. The first area applied common principles in a mobile environment. Most prominently, applying principles within three areas; screen types, notifications, and interaction. The screen types described standard mobile pages, such as the home and onboarding pages. The notifications described their usage as a communication to the user. Additionally, they could split into different types based on the notification content and how long it stays visible to the user. Finally, the interaction described various methods people hold mobile phones and touch their screens. The second area focused on the theory behind this project's design system: Material Design. The design system provided a set of standards and guidelines that ensures visual consistency and reduce redundancy between the UI elements.

The project continued after the literature study by learning the required technical theory to design and implement the application. The design stage aimed to connect the theory studied to the application design and architecture. The design used figures to illustrate how users interact with the application. The project systematically chose each page and sub-page displayed to the users to support various theory. For example, the social page supports qualities within social support. Additionally, the design stage supported the design with novel usability heuristics specifically designed for nudging. The project implemented an application following the architectural specifications in the design. The specification strictly followed best practices from the official Android guidelines, thus resulting in a architecture that simplifies future development.

8.1 Future Work

The robust architecture lays the foundation for future development. The decoupling of components and pages enables efficient modification and extension of the codebase. The architecture also provides components and pages that other nudge-related projects could reuse. Future work should focus on creating a complete system that supports all the primary features presented in the design architecture. After a complete system, the project can begin user testing and evaluate the results.

Completing the system The project has to implement features before performing meaningful user testing. Sections 6.3 and 6.4 discussed these features. However, the required effort and priority of the features vary. Both the design cards and notifications do not depend on the architecture. Future development can thus implement them parallelly along the other features. Third-party services, like Google Maps integration, should have relatively low priority. Developers should introduce them when implementing design cards that need them.

The ODS nudge project does not have a functioning remote back-end that can communicate with the application. Future development could implement a simple server sending mock data as a temporary solution. The mobile application would then expand the data layer with the retrofit library, as discussed in section 6.3.3.

Architecture restructure Introducing a multi-module architecture and Jetpack compose requires restructuring a large percentage of the codebase. It would require time and effort, though it would have several advantages, such as more efficient development.

User testing The next step of the ODS nudge project architecture revolves around testing and evaluating the nudges, as shown in figure 2.1(page 16). The project can deploy the application to Google Play under closed or open testing. Compared to open testing, closed testing has the advantage of using fewer and specifically selected users. The experiments should test the effectiveness of nudges within the application and its notifications. Developers should also run several experiments with design variations. The evaluation can then compare the results from each variation and iterate the design accordingly.

Reusing components This project designed the architecture with extensibility and reusability in mind. The implementations ensure possible integration of the architecture, single components, and single pages, further discussed in section 6.5. Thus, future work can test the architecture's reusability by integrating components into other nudge-related projects. The other projects could test the components in a different use case than green transportation. A future multi-modular architecture could use its modules as external libraries. Other projects could then easily import the libraries to use in their codebase.

References

[1] Köne, A. Ç., & Büke, T. (2010). Forecasting of co2 emissions from fuel combustion using trend analysis. *Renewable Sustainable Energy Rev.*, *14*(9), 2906–2915. https://doi.org/10.1016/j.rser.2010.06.006

[2] Kaufmann, R. K., Kauppi, H., Mann, M. L., & Stock, J. H. (2011). Reconciling anthropogenic climate change with observed temperature 1998–2008. *Proc. Natl. Acad. Sci. U.S.A.*, *108*(29), 11790–11793. https://doi.org/10.1073/pnas.1102467108

[3] Houghton, J. T., Jenkins, G. J., & Ephraums, J. J. (1990). *Climate change*. https://www.osti.gov/etdeweb/biblio/6041139

[4] Victor, D. G., Zhou, D., Ahmed, E. H. M., Dadhich, P. K., Olivier, J. G. J., Rogner, H.-H., Sheikho, K., & Yamaguchi, M. (2014). Introductory chapter. In C. of Working Group Iii to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Edenhofer, R. P.-M. O., Y. Sokona, E. Farahani, S. Kadner, K. Seyboth, A. Adler, I. Baum, S. Brunner, P. Eickemeier, B. Kriemann, J. Savolainen, S. Schlömer, C. von Stechow, T. Zwickel, & J. C. Minx (Eds.), *Climate change 2014: Mitigation of climate change*. Cambridge University Press.

[5] Sims, R., Schaeffer, R., Creutzig, F., Cruz-Núñez, X., D'Agosto, M., Dimitriu, D., Meza, M. J. F., Fulton, L., Kobayashi, S., Lah, O., McKinnon, A., Newman, P., Ouyang, M., Schauer, J. J., Sperling, D., & Tiwari, G. (2014). Transport. In C. of Working Group Iii to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Edenhofer, R. P.-M. O., Y. Sokona, E. Farahani, S. Kadner, K. Seyboth, A. Adler, I. Baum, S. Brunner, P. Eickemeier, B. Kriemann, J. Savolainen, S. Schlömer, C. von Stechow, T. Zwickel, & J. C. Minx (Eds.), *Climate change 2014: Mitigation of climate change*. Cambridge University Press.

[6] Kampa, M., & Castanas, E. (2008). Human health effects of air pollution. *Environ. Pollut.*, *151*(2), 362–367. https://doi.org/10.1016/j.envpol.2007.06.012

[7] Bucher, T., Collins, C., Rollo, M. E., McCaffrey, T. A., De Vlieger, N., Van der Bend, D., Truby, H., & Perez-Cueto, F. J. A. (2016). Nudging consumers towards healthier choices: A systematic review of positional influences on food choice. *Br. J. Nutr.*, *115*(12), 2252–2263. https://doi.org/10.1017/S0007114516001653

[8] de Visser-Amundson, A., & Kleijnen, M. (2019). Nudging in food waste management: Where sustainability meets cost-effectiveness. In *Food Waste Management: Solving the Wicked Problem* (pp. 57–87). Palgrave Macmillan. https://doi.org/10.1007/978-3-030-20561-4_3

[9] Håkansson, A. (2013). Portal of research methods and methodologies for research projects and degree projects. *DIVA*, 67–73. https://www.diva-portal.org/smash/record.jsf?pid=diva2:677684

[10] Karlsen, R., Håkansson, A., Andersen, A., & Yu, W. (2021). Ods – open distributed systems | research group. *Department of Computer Science, UiT The Arctic University of Norway*. https://site.uit.no/ods

[11] Briñol, P., & Petty, R. E. (2012). *The history of attitudes and persuasion research*. https://www.researchgate.net/publication/271503858_The_history_of_attitudes_and_persuasion_research

[12] Cialdini, R. B. (1993). Influence: The psychology of persuasion. *ResearchGate*. https://www.researchgate.net/publication/243742535_Influence_The_Psychology_of_Persuasion

[13] Kahneman, D. (2012). Of 2 minds: How fast and slow thinking shape perception and choice [excerpt]. https://www.scientificamerican.com/article/kahneman-excerpt-thinking-fast-and-slow

[14] Blanco, F. (2017). *Cognitive bias*. https://doi.org/10.1007/978-3-319-47829-6_1244-1

[15] Karlsen, R., & Dalecke, S. (2020). Designing dynamic and personalized nudges. In *WIMS 2020: Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics* (pp. 139–148). Association for Computing Machinery. https://doi.org/10.1145/3405962.3405975

[16] Schneider, C., Weinmann, M., & Vom Brocke, J. (2018). Digital nudging: Guiding online user choices through interface design. *Commun. ACM*, *61*(7), 67–73. https://doi.org/10.1145/3213765

[17] Thaler, R. H., & Sunstein, C. R. (2009). *Nudge: Improving decisions about health, wealth, and happiness* (Vol. 47). https://www.researchgate.net/publication/235413094_NUDGE_Improving_Decisions_About_Health_Wealth_and_Happiness

[18] Meske, C., & Potthoff, T. (2017). The dinu-model – a process model for the design of nudges. *ResearchGate*. https://www.researchgate.net/publication/317661783_The_DINU-Model_-_A_Process_Model_for_the_Design_of_Nudges

[19] Karlsen, R., & Andersen, A. (2019). Recommendations with a nudge. *Technologies*, *7*(2), 45. https://doi.org/10.3390/technologies7020045

[20] Oinas-Kukkonen, H., & Harjumaa, M. (2009). Persuasive systems design: Key issues, process model, and system features. *Communications of the Association for Information Systems*, *24*(1). https://doi.org/10.17705/1CAIS.02428

[21] Fogg, B. J. (2002a). Persuasive technology: Using computers to change what we think and do. *Ubiquity*, *2002*(December), 2. https://doi.org/10.1145/764008.763957

[22] Obi, C. (2022). Ui vs. ux: What's the difference? | usertesting blog. https://www.usertesting.com/blog/ui-vs-ux

[23] Ui vs. ux: What's the difference? | usertesting blog. (2022). https://www.usertesting.com/blog/ui-vs-ux

[24] Ketterman, S. (2019). Mobile ux design constraints, best practices, and working with developers. *Toptal Design Blog*. https://www.toptal.com/designers/ux/mobile-ux-design-best-practices

[25] Arhipova, A. (2020). Mobile ui design: 15 basic types of screens. *Tubik Blog: Articles About Design*. https://blog.tubikstudio.com/mobile-ui-design-15-basic-types-of-screens

[26] Chapin, B. (2018). First impressions – a guide to onboarding ux. *Toptal Design Blog*. https://www.toptal.com/designers/product-design/guide-to-onboarding-ux

[27] Shen, S. (2018). Toasts or snack bars? — designing organic notifications. *Medium*. https://uxdesign.cc/toasts-or-snack-bars-design-organic-system-notifications-1236f2883023

[28] Santiago, S. V. (2020). A comprehensive guide to notification design. *Toptal Design Blog*. https://www.toptal.com/designers/ux/notification-design

[29] Basu, S. (2021). Guidelines for push notification ux - bootcamp. *Medium*. https://bootcamp.uxdesign.cc/guidelines-for-push-notification-ux-bde2bae9666e

[30] Push notification ux: The full guide 2022 (updated). (2022). https://uxcam.com/blog/push-notification-guide

[31] Hoober, S. (2022). Design for fingers, touch, and people, part 1 :: Uxmatters. https://www.uxmatters.com/mt/archives/2017/03/design-for-fingers-touch-and-people-part-1.php

[32] Nielsen, J., & Norman, D. (2020). How people read online: New and old findings. https://www.nngroup.com/articles/how-people-read-online

[33] Nielsen, J., & Norman, D. (2019). Touch targets on touchscreens. https://www.nngroup.com/articles/touch-target-size

[34] F-shaped pattern for reading web content (original eyetracking research). (2019). https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered

[35] Nielsen, J. (2021). Design systems 101. https://www.nngroup.com/articles/design-systems-101

[36] Material design. (2022). https://m2.material.io/design

[37] Android notifications. (2022). https://material.io/design/platform-guidance/android-notifications.html#anatomy-of-a-notification

[38] Nielsen, J. (1993). *Usability engineering*. Morgan Kaufmann. https://doi.org/10.1016/C2009-0-21512-1

[39] Nielsen, J., & Norman, D. (2022). Usability engineering : Book by jakob nielsen. https://www.nngroup.com/books/usability-engineering

[40] Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 152–158). Association for Computing Machinery. https://doi.org/10.1145/191666.191729

[41] Hausman, D. M., & Welch, B. (2010). Debate: To nudge or not to nudge. *Journal of Political Philosophy*, *18*(1), 123–136. https://doi.org/10.1111/j.1467-9760.2009.00351.x

[42] Bovens, L. (2009). The ethics of nudge. In *Preference change* (pp. 207–219). Springer. https://doi.org/10.1007/978-90-481-2593-7_10

[43] Engelen, B., & Nys, T. (2020). Nudging and autonomy: Analyzing and alleviating the worries. *Rev.Phil.Psych.*, *11*(1), 137–156. https://doi.org/10.1007/s13164-019-00450-z

[44] Wachner, J., Adriaanse, M., & De Ridder, D. (2020). The influence of nudge transparency on the experience of autonomy. *Comprehensive*

Results in Social Psychology, 1–15. https://doi.org/10.1080/23743603.2020.1808782

[45] Coercion noun [[Online; accessed 29. Nov. 2022]]. (2022). https://www.oxfordlearnersdictionaries.com/definition/english/coercion

[46] Lee, W. W., Zankl, W., & Chang, H. (2016). An ethical approach to data privacy protection. https://www.isaca.org/resources/isaca-journal/issues/2016/volume-6/an-ethical-approach-to-data-privacy-protection

[47] Pınarbaşı, A. T., CIPP/E, & LLM. (2022). Privacy policy vs. terms and conditions. *Termly*. https://termly.io/resources/articles/privacy-policy-vs-terms-and-conditions

[48] General data protection regulation (gdpr) – official legal text [[Online; accessed 29. Nov. 2022]]. (2022). https://gdpr-info.eu

[49] Tietz, M., Simons, A., Weinmann, M., & Brocke, J. v. (2016). The decoy effect in reward-based crowdfunding: Preliminary results from an online experiment. *ResearchGate*. https://www.researchgate.net/publication/308611392_The_Decoy_Effect_in_Reward-Based_Crowdfunding_Preliminary_Results_from_an_Online_Experiment

[50] Weinmann, M., Tietz, M., Simons, A., & Vom Brocke, J. (2017). Get it before it's gone? how limited rewards influence backers' choices in reward-based crowdfunding. https://aisel.aisnet.org/icis2017/Peer-to-Peer/Presentations/18

[51] Simons, A., Weinmann, M., Tietz, M., & Brocke, J. v. (2017). Which reward should i choose? preliminary evidence for the middle-option bias in reward-based crowdfunding. *ResearchGate*. https://doi.org/10.24251/HICSS.2017.526

[52] Langrial, S., Lehto, T., Oinas-Kukkonen, H., Harjumaa, M., & Karppinen, P. (2012). Native mobile applications for personal wellbeing: A persuasive systems design evaluation. *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2012*. https://www.researchgate.net/publication/257396607_Native_Mobile_Applications_For_Personal_Wellbeing_A_Persuasive_Systems_Design_Evaluation

[53] Torkamaan, H., & Ziegler, J. (2021). Integrating behavior change and persuasive design theories into an example mobile health recommender system. In *Ubicomp '21: Adjunct proceedings of the 2021 acm international joint conference on pervasive and ubiquitous computing and proceedings of the 2021 acm international symposium on wearable computers* (pp. 218–

225). Association for Computing Machinery. https://doi.org/10.1145/3460418.3479330

[54] Fogg, B. J. (2009). A behavior model for persuasive design. In *Persuasive '09: Proceedings of the 4th international conference on persuasive technology* (pp. 1–7). Association for Computing Machinery. https://doi.org/10.1145/1541948.1541999

[55] Fogg, B. J. (2002b). Persuasive technology: Using computers to change what we think and do. *Ubiquity*, *2002*(December), 2. https://doi.org/10.1145/764008.763957

[56] Ryan, R. M. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *Am. Psychol.*, *55*(1), 68. https://doi.org/10.1037/0003-066X.55.1.68

[57] Ryan, R. M., & Deci, E. L. (2017). *Self-determination theory: Basic psychological needs in motivation, development, and wellness*. Guilford Publications. https://books.google.no/books/about/Self_Determination_Theory.html?id=Bc_DDAAAQBAJ&redir_esc=y

[58] Sundar, S. S., Bellur, S., & Jia, H. (2012). Motivational technologies: A theoretical framework for designing preventive health applications. In *Persuasive technology. design for health and safety* (pp. 112–122). Springer. https://doi.org/10.1007/978-3-642-31037-9_10

[59] Szalma, J. L. (2014). On the application of motivation theory to human factors/ergonomics: Motivational design principles for human–technology interaction. *Hum. Factors*, *56*(8), 1453–1471. https://doi.org/10.1177/0018720814553471

[60] Torkamaan, H., & Ziegler, J. (2018). Multi-criteria rating-based preference elicitation in health recommender systems. In *Healthrefsys '18: Proceedings of the 3rd international workshop on health recommender systems* (pp. 18–23). Ceur Workshop Proceedings. http://ceur-ws.org/Vol-2216

[61] Torkamaan, H., & Ziegler, J. (2019). Rating-based preference elicitation for recommendation of stress intervention. In *Umap '19: Proceedings of the 27th acm conference on user modeling, adaptation and personalization* (pp. 46–50). Association for Computing Machinery. https://doi.org/10.1145/3320435.3324990

[62] Maulana, S. A., & Suzianti, A. (2019). User interface redesign in a point rewards mobile application using usability testing method. In *Iccip '19: Proceedings of the 5th international conference on communication and in-*

formation processing (pp. 43–48). Association for Computing Machinery. https://doi.org/10.1145/3369985.3370001

[63] Newman, I., Benz, C. R., & Ridenour, P. C. S. (1998). *Qualitative-quantitative research methodology: Exploring the interactive continuum*. Southern Illinois University Press. https://books.google.no/books/about/Qualitative_quantitative_Research_Method.html?id=xumf1ABFz8cC&redir_esc=y

[64] Introducing material theme builder - material design. (2021). https://material.io/blog/material-theme-builder

[65] Developer guides. (2022). *Google Inc.* https://developer.android.com/guide

[66] Protocol buffers. (2022). *Google Inc.* https://developers.google.com/protocol-buffers

[67] Retrofit. (2020). *Square Inc.* https://square.github.io/retrofit

[68] de Kovel, C. G. F., Carrión-Castillo, A., & Francks, C. (2019). A large-scale population study of early life factors influencing left-handedness. *Sci. Rep.*, *9*(584), 1–11. https://doi.org/10.1038/s41598-018-37423-8

[69] Apache license, version 2.0 [[Online; accessed 11. Dec. 2022]]. (2004). https://www.apache.org/licenses/LICENSE-2.0

[70] Jetpack compose ui app development toolkit. (2022). *Google Inc.* https://developer.android.com/jetpack/compose

[71] Jetpack compose is now 1.0: Announcing android's modern toolkit for building native ui. (2021). *Google Inc.* https://android-developers.googleblog.com/2021/07/jetpack-compose-announcement.html