Final project documentation

Mobile Computing

Exchange student

Yeray Navascués Trincado

Fabian Günter ErsäBer

22/01/2025

Index of contents

[1. Introduction and motivation 2](#_Toc1678186470)

[1.1 Motivation 3](#_Toc2017414427)

[1.2 Goal 3](#_Toc296393454)

[2. Requirements analysis 3](#_Toc387177889)

[2.1 Storyboard 4](#_Toc752228880)

[2.2 Personas 4](#_Toc449763063)

[2.2.1 Persona 1 5](#_Toc1287802890)

[2.2.1 Persona 2 6](#_Toc1413768180)

[2.3 User stories 7](#_Toc573772239)

[2.4 Scenarios 10](#_Toc1178701972)

[2.5 Requirement specifications 11](#_Toc1332516303)

[2.5.1 Functional requirements 12](#_Toc1820182811)

[2.5.2 Non-Functional requirements 16](#_Toc885926181)

[3. Conceptual model of the solution 17](#_Toc2071038884)

[3.1 User flow diagram 18](#_Toc1828784753)

[3.2 ER diagram 18](#_Toc1881013627)

[3.3 Mockups 19](#_Toc298490986)

[3.3.1 Initial screen 20](#_Toc1816921947)

[3.3.2 Login screen 20](#_Toc696659724)

[3.3.3 Action booking (Customer) 21](#_Toc1904328051)

[3.3.4 Profile sp 22](#_Toc1508773323)

[3.3.5 Action fullfill (Business) 23](#_Toc1148197179)

[4. Design decisions and implemetation 24](#_Toc1939504540)

[4.1 Plattaform used(VS,AS) 25](#_Toc606508425)

[4.2 Packages used (tools) 25](#_Toc1471077002)

[4.2.1 Flutter 26](#_Toc1182968814)

[4.2.2 Dart 26](#_Toc1053028533)

[4.2.3 Firebase (Web storage) 26](#_Toc1968718938)

[4.2.4 Communication (chatting) 27](#_Toc109637919)

[5. Results 28](#_Toc291301752)

[6. Conclusion 28](#_Toc1063556880)

# Introduction and motivation

## 1.1 Motivation

The motivation behind the app arises from the challenge people face when trying to book registered service providers in their local area. Currently, there is no simple, direct solution to quickly find and book trusted providers nearby. The goal of the app is to fill this gap by offering a user-friendly platform where individuals can easily connect with local service providers. The process should be seamless, requiring just a few simple and fast clicks to share the necessary contact information and any additional details. By eliminating obstacles such as complex forms or lengthy searches, the app ensures that users can efficiently secure the services they need, saving time and effort in the process. Whether it’s for urgent repairs, regular maintenance, or any other service, the app aims to streamline the booking experience and make it as convenient as possible for everyone.

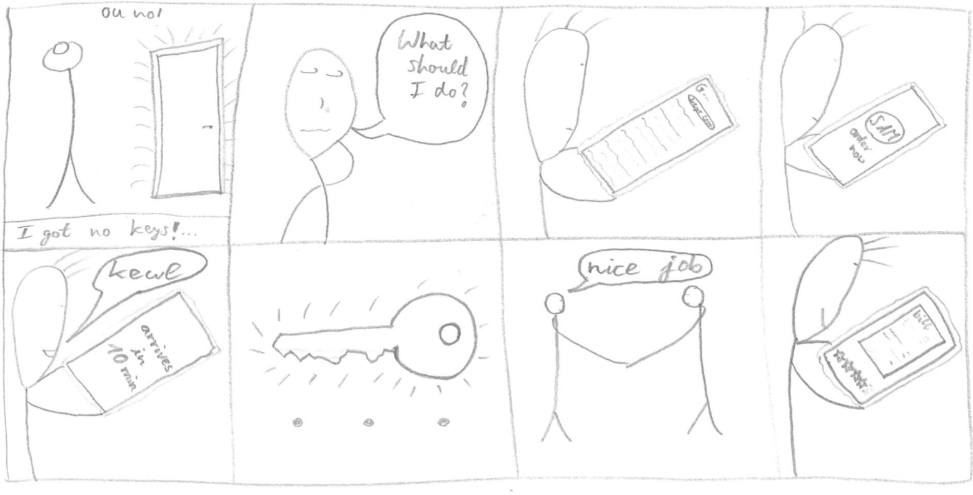
## 1.2 Goal

Our goal behind the application is the lack of a direct and straightforward way to book a registered service provider in the local area. A few simple and quick clicks should be enough to share the necessary contact and all other data.

# Requirements analysis

## 2.1 Storyboard

Defining the needs and expectations of users was a key step in shaping the application. Booking local service providers often involves unnecessary complexity, which can be a challenge for individuals of all ages and backgrounds. The application is designed to simplify this process, offering a seamless experience that caters to a diverse audience, regardless of technical expertise or familiarity with similar platforms.



## 2.2 Personas

Personas help to better understand the needs and challenges of specific user groups. For this application, two primary target groups have been identified. The first group consists of individuals that need a service. The second group includes qualified workers which are willing to give a service as their profession. The application is designed to be accessible to people from various backgrounds, ages, and levels of technical proficiency. Two personas have been created to represent these groups: Markus Müller, who embodies the client side concerning the roles, and Aaron Schimmzt, representing the service provider side.

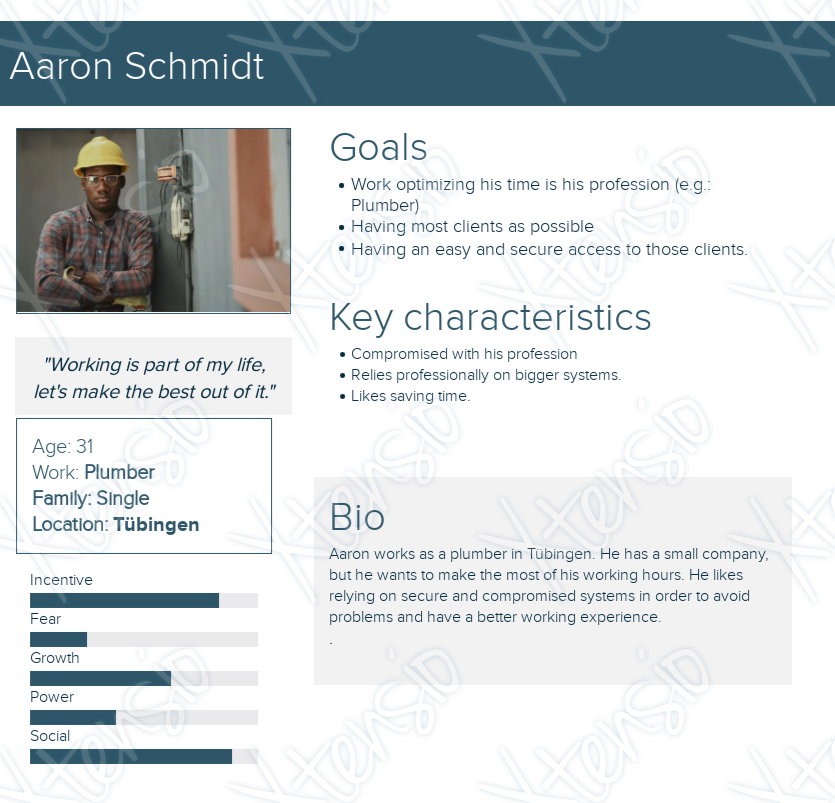
### 2.2.1 Persona 1

Markus works as a professor at the Hochschule Reutlingen. He also does shifts in a company, he is busy most of the time. He doesn't like dealing with problems by himself, he prefers professionals doing this tasks for him. Markus doesn't like wasting his time, so if there is way he can save, he will.



### 2.2.1 Persona 2

Aaron works as a plumber in Tübingen. He has a small company, but he wants to make the most of his working hours. He likes relying on secure and compromised systems in order to avoid problems and have a better working experience.



## 2.3 User stories

User stories outline the software requirements from the perspective of the application's users. Based on the personas that represent the typical user roles, specific requirements for each user have been defined. For each of these requirements, acceptance criteria have been established to ensure the users’ needs are fully met.

1. As a customer I want to access the app directly from the website of the service provider because I always use the browser first to see what is nearby.
2. As a customer I want to have the possibility to cancel the service for free, if the service provider is delayed so I can book another service provider.
3. As a user of the app, I want to be able to log in so I can be able to operate from my account.
4. As a service provider I want to get certain data I need from my customers so that I can come prepared and know where to go.
5. As a service provider I want to customize the questionnaire for different services so I can prepare myself better for the exact service and bring the right tools with me.
6. As a service provider I want to have a picture of the issue so I can prepare better to solve it.
7. As a customer I want to choose the importance of the problem so that if it is an emergency the service provider knows that and will hopefully come quickly.
8. As a customer I want to know the arrival time before I accept the service so I can choose if it suits me.
9. As a service provider I want to know how important the service is so I can hurry up if it is an emergency.
10. As a customer I want to have different possibilities to pay for the service so that I can use a process I have already used before and do not have to get a new app.
11. As a customer I want to know if the service request is accepted so that I am sure the service provider will arrive and fix my problem.
12. As a customer I want to know the arrival time of the service provider so that I can organize my daily schedule around it.
13. As a service provider I want to influence the estimation of the costs of the drive because I want to integrate my individual costs.
14. 14 As a service provider I want to notify the customer if I am not on time so that the customer is informed.
15. 15 As a costumer I want to have the possibility to cancel the service for free, if the service provider is delayed so I can book another service provider.
16. 16 As a service provider I want to have the possibility to make offer of a discount so the customer might not cancel the service despite my delay.
17. As a service provider I want to get all travel expenses paid by the customer if the customer cancels the service before the end of the arrival time because I may already be on my way.
18. As a service provider I want to be paid for extra waiting time if the customer is not at the agreed place at the agreed time, because I want to get paid for my time.
19. As a service provider I want to cancel the service if the costumer is not at the agreed place at the agreed time so I can accept the next order.
20. As a service provider I want to have the exact bill with all costs so that my documents are complete.
21. As a customer I want to notify the service provider if I am not on time so that the service provide is informed and has the option to cancel the service or to accept the delay.
22. As a service provider I want to have the possibility to accept the delay from the costumer and be paid for my extra waiting time so that so that my time waiting is paid.
23. As a service provider I want to have the possibility to deny because of the delay from the costumer so that I can take other orders.
24. As a service provider I want to have the possibility to accept the delay from the costumer for free because in some situation I do not want to be that exact.
25. As a customer I want to be notified if the service provider cancels the service so I can choose another service provider.
26. As a customer I do not want to pay the travel expenses if the service provider cancels the service before the arrival time because the ordered service was not provided and it was not my fault.
27. As a service provider I want to have the possibility to cancel the service at every time and not get paid because maybe something happened and I could not get there.
28. As a service provider I want the customer to be notified automatically if I am close to the arranged place so the customer can come out and I can start working immediately.
29. As a customer I want to track the service provider on his way to me so I can better estimate the arrival time.
30. As a service provider I want to share the account with my employers so that anybody can work on their own and have the possibility to accept orders.
31. As an user I want to have the bills directly prepared in the correct format for the tax office so that I do not have to invest time in processing the forms.
32. As an user I want to have the possibility to download all documents as a pdf file so that I can use it for the tax office.
33. As a service provider I want to close the order
34. As a colourblind user of the app I want to be able to understand the app and all features without needing to see colour because I want to be able to use all available features.
35. As a user of the app I want my shared data to be to be save because I do not want them to be public.
36. As a user of the app I want the app to be available at all times so that I do not have to wait if I want to do something in the app.
37. As a new user of the app I want all features to be understandable so that I do not have to invest a lot of time in understanding the app.
38. As a user of the app I want the app to be aesthetically pleasing so I can enjoy my time on the app.
39. As a user of the app I want the app to respond quickly to my actions/requests so that I do not have to wait when I want to do something in the app.
40. As a user I want to get to my goal with as little steps as possible so that I do not get confused and are done rather quickly.

## 2.4 Scenarios

Scenarios serve as practical examples that illustrate how users interact with the application in real-life situations. They provide context to understand user behaviors, goals, and challenges while using the app. By detailing these interactions, scenarios help identify requirements, refine features, and ensure the application aligns with the users’ needs and expectations.

“Send a request”

1. The customer will indicate that they want to send a request to a service provider to get help for a problem they have.

2. The customer will press the button for a new request form.

3. The system will present a form with two sections and a button. The first section will be hidden but expandable. The second section will be a text field.

4. The customer will click on the button with the upside-down triangle.

5. The system will expand the section and show the customers personal information (name, address, etc.)

6. The customer will press the pen symbol next to the address.

7. The system will open the keyboard, magnify the text field with the address and show a cursor at the end of the address.

8. The customer will delete the address and update it to another address.

9. The customer will click outside of the keyboard and the text field area.

10. The system will close the keyboard and demagnify the text field.

11. The customer will fill the text field with information about the problem they need help with.

12. The customer will click the button with the arrow symbol under the text field.

13. The system will save the information and send the information to the service provider.

14. The system will show a pop-up message that confirms that the request was send with button that says “OK”.

15. The customer will press the button that says “OK.

16. The system will close the request form section.

## 2.5 Requirement specifications

Functional requirements were derived from the analysis of Personas, User Stories, and Use Cases. These requirements outline the specific features and actions the application must support to meet user needs. Each functional requirement is defined with a name, ID, description, success criteria, and priority, ensuring clarity and alignment with the application’s core purpose.

### 2.5.1 Functional requirements

|  |  |
| --- | --- |
| Requirement | Access to the app directly from the website of the service provider |
| Number | 1 |
| Type | functional |
| User Story Reference | 1 |
| Description | Customers can access the app with the link on service providers website. Therefore it is helpful to have a finish code package which service providers can use uad integrate in their website easily. The code includes the SAM Logo which is everywhere pressable an leads to the app. It should have also a description how to integrate the code into the own website so the service providers can integrate it in a fast way. (install app first) |
| Definition of Done/Success criteria | Description of the integration for the code in the website finished code package includes the pressable Logo the logo leads to the app |
| Level of importance | high, must have |
| Colour Code | red |

|  |  |
| --- | --- |
| Requirement | Overview over all closed orders with |
| Number |  |
| Type | Non functional or functional |
| User Story Reference |  |
| Description |  |
| Definition of Done/Success criteria |  |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code | Green: Done, Yellow: in progress, Red: not touched/not working |

|  |  |
| --- | --- |
| Requirement | Input field and send button for request |
| Number |  |
| Type | Functional |
| Epic | Request Form |
| User Story Reference | 4 |
| Description | The customer should be able to describe their issue/request in a text field. The text field should be able to contain up to 2000 characters. Under the text field there should be a button with the symbol of an arrow. It is not possible to press the button if nothing was written in the text field. With a click on the button the information should be send to the service provider. The screen with the text field will be closed automatically and the recorded information will be saved. |
| Definition of Done/Success criteria | - text field which can hold up to 2000 characters  - arrow-button which cannot be pressed if the text field is empty  - link in the arrow button which will lead to saving and sending the information to the service provider  - after the arrow button was clicked the form is closed |
| Level of importance | High, must have |
| Colour Code |  |

|  |  |
| --- | --- |
| Requirement |  |
| Number |  |
| Type | Functional |
| Epic | Request Form |
| User Story Reference | 4 |
| Description |  |
| Definition of Done/Success criteria |  |
| Level of importance | High, must have |
| Colour Code |  |

|  |  |
| --- | --- |
| Requirement | New request should be list in open request for the customer |
| Number |  |
| Type | functional |
| User Story Reference | 4 |
| Description | The customer describes his issue or request in the form and klick on the button to send it to the service provider. In this moment the form closed and should be shown in the list with open request in the customer view. |
| Definition of Done/Success criteria | - with a klick on the send button from the form the request should be shown in the list with all open requests in the customer view  - it is possible to open the form and to see all given information |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code | Green: Done, Yellow: in progress, Red: not touched/not working |

|  |  |
| --- | --- |
| Requirement | New request should be listed in a new request for the service provider |
| Number |  |
| Type | functional |
| User Story Reference | 4 |
| Description | Is there a new request for the service provider it should be shown under the order ‘new requests’, like a new message. So the screen automatically switches to the request. It should be possible to open the form and to see all information. |
| Definition of Done/Success criteria | - the request should be shown in the list with all new requests in the service provider view  - it is possible to open the form and to see all given information |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code | Green: Done, Yellow: in progress, Red: not touched/not working |

As a customer I want to have different possibilities to pay for the service so that I can use a process I have already used before and do not have to get a new app.

|  |  |
| --- | --- |
| Requirement | Customer can pay charges with PayPal |
| Number |  |
| Type | Functional |
| User Story Reference | 10 |
| Description | To pay for the services of a service provider the customer should be able to choose between different options for payment one of them being PayPal. The customer will have an overview of which charges they are accepting and will be able to select PayPal as their payment method. They will have to press a button ton be guided to PayPal where they will have to give their final acceptance of the charges. After the charges were accepted in PayPal they are guided back to the app where they are presented a screen which says: “Payment successful”. From there they can access the other features of the app. |
| Definition of Done/Success criteria | * Overview on charges * Buttons to choose between payment methods one of them being PayPal * Button to accept presented charges * Link in said button to PayPal with amount of money that is charged * Finish screen stating “Payment successful” from which they will have access to other features of the app |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code |  |

|  |  |
| --- | --- |
| Requirement | Customer can pay charges with SEPA direct debit mandate |
| Number |  |
| Type | Functional |
| User Story Reference | 10 |
| Description | To pay for the services of a service provider the customer should be able to choose between different options for payment one of them being SEPA direct debit mandate. The customer will have an overview of which charges they are accepting and will be able to select SEPA direct debit mandate as their payment method. When SEPA is selected as payment method a formular appears which includes: name, bank and IBAN. The customer will be able to press a button labelled “accept charges” after all 3 fields are filled with content that makes sense. After they accepted the charges they will be guided to a screen stating “SEPA direct debit mandate successful”. From there they can access the other features of the app. |
| Definition of Done/Success criteria | * Overview on charges * Buttons to choose between payment methods one of them being SEPA direct debit mandate * Formular with name, bank and IBAN to fill in which only appears when choosing SEPA direct debit mandate * Button to confirm information which can only be pressed when all fields (name, bank and IBAN) are filled with content, that makes sense (name: characters, no numbers; bank: ?, IBAN: country code has to be conclusive with length, 3rd and 4th position are numbers) * Finish screen stating “SEPA direct debit mandate successful” from which they will have access to other features of the app |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code |  |

|  |  |
| --- | --- |
| Requirement | Notification of an accepted service request to the customer |
| Number |  |
| Type | Functional |
| User Story Reference | 11 |
| Description | After the customer sent a request for a service they |
| Definition of Done/Success criteria |  |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code | Green: Done, Yellow: in progress, Red: not touched/not working |

### 2.5.2 Non-Functional requirements

|  |  |
| --- | --- |
| Requirement | The design of the app is inclusive for colourblind people |
| Number | 2 |
| Type | Non functional |
| Epic | Design criteria |
| User Story Reference | 34 |
| Description | The user should be able to understand and use all features without needing to see colour. All buttons should be labelled in a way that is understandable in addition to them being colour coded. Colours that colour blind people could see as the same colour should never be used to show opposite functions or information. |
| Definition of Done/Success criteria | * all functions and buttons must be additionally labelled with symbols or writing * red and green as well as orange and yellow and yellow and blue must never be displayed as opposite functions/information * all features and the app in itself must be understandable even if the design is perceived in greyscale |
| Level of importance | High, must have |
| Colour Code | Red |

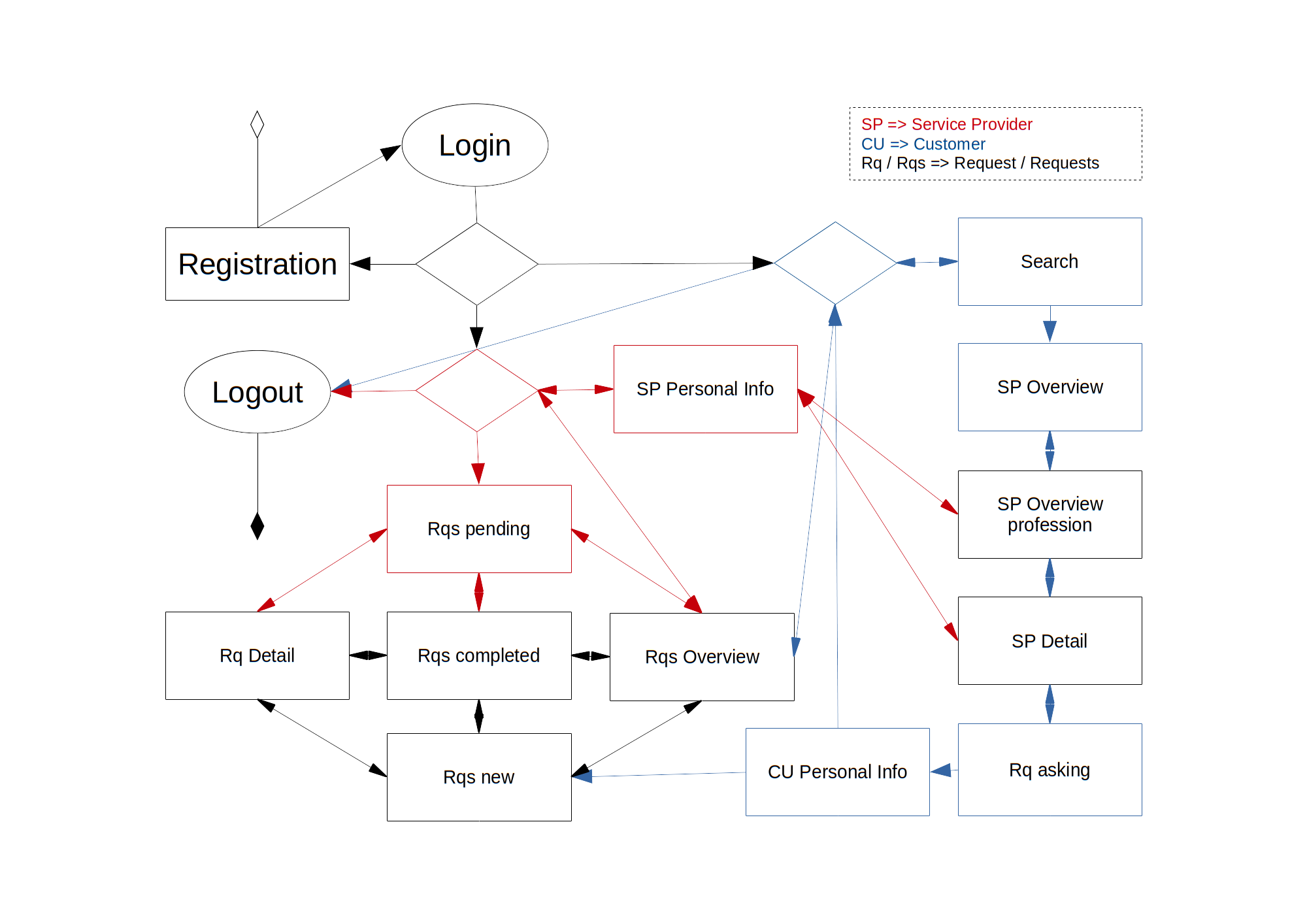
|  |  |
| --- | --- |
| Requirement | Structure of the dashboard |
| Number |  |
| Type | Non functional |
| User Story Reference | 2 |
| Description | The app has much information which has to be structured. The information can be classified into three groups - personal data, closed services and open services. The access to the personal datas should be over a round person icon in the right corner on top. All orders, closed and open should be shown as a list but it is possible to switch on the top between closed and opened orders. |
| Definition of Done/Success criteria |  |
| Level of importance | Priority, label (must have/should have/nice to have) |
| Colour Code | Green: Done, Yellow: in progress, Red: not touched/not working |

# Conceptual model of the solution

The foundation of the development process is the conceptual model, designed to describe and visualize the application and its core functions. This step involved the use of various diagrams and, in particular, mock-ups to create a clear representation of the application’s structure and features.

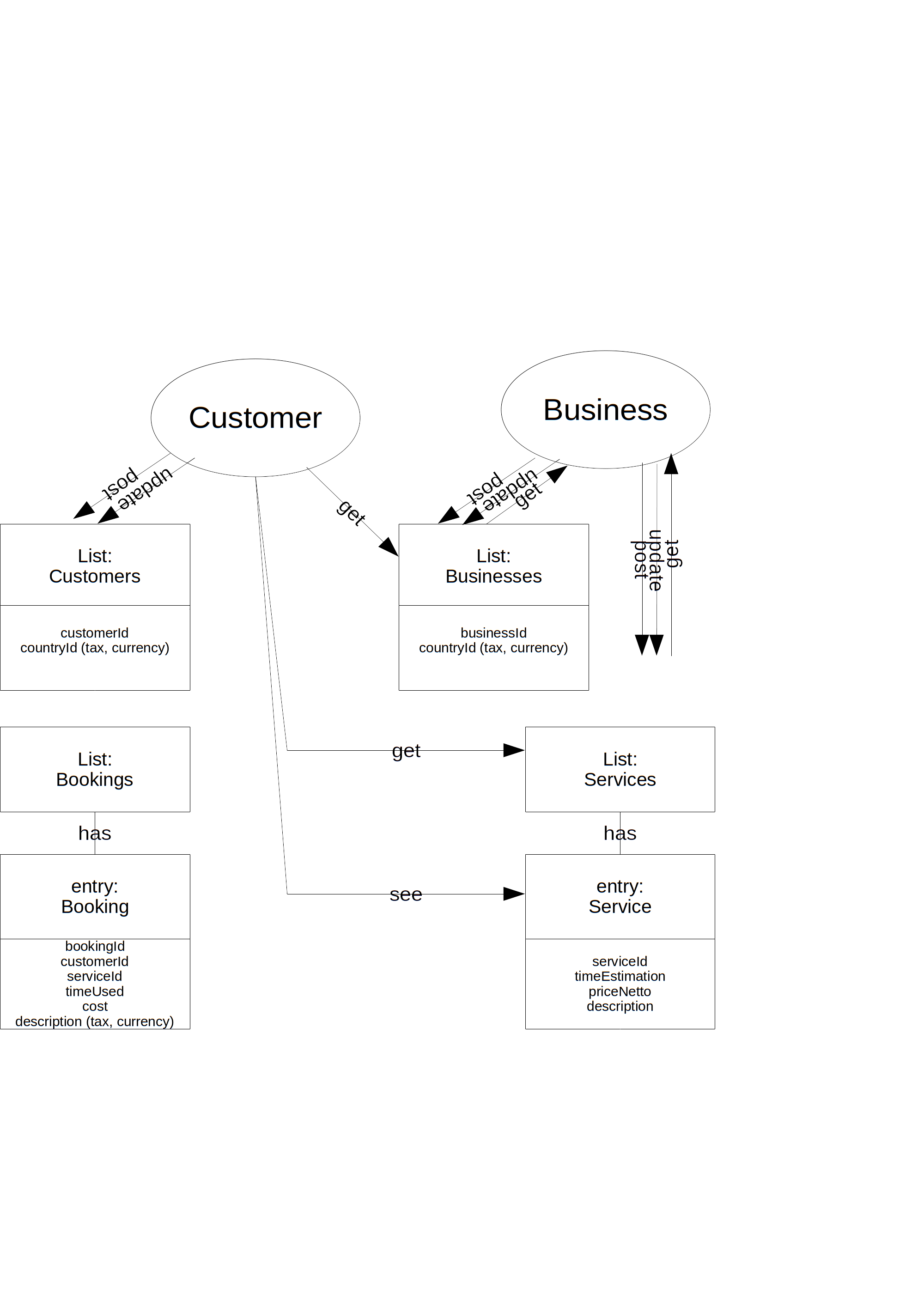
## 3.1 User flow diagram

The flow diagram provides a visual representation of the application's processes and user interactions. It outlines the steps users take to complete key tasks and how different components of the system connect and function together. By mapping out these workflows, the flow diagram ensures clarity in the application's design and helps identify any potential bottlenecks or inefficiencies in the process.



## 3.2 ER diagram

The Entity-Relationship (ER) diagram is a visual representation of the application's data structure. It illustrates the entities within the system, their attributes, and the relationships between them. This diagram provides a clear overview of how data is organized and interconnected, serving as a foundational tool for database design and ensuring the system’s data flows are efficient and well-structured.



## 3.3 Mockups

Mock-ups are essential for visualizing the application prior to implementation. They ensure a cohesive design across all screens and provide a basis for discussing and refining the user experience.

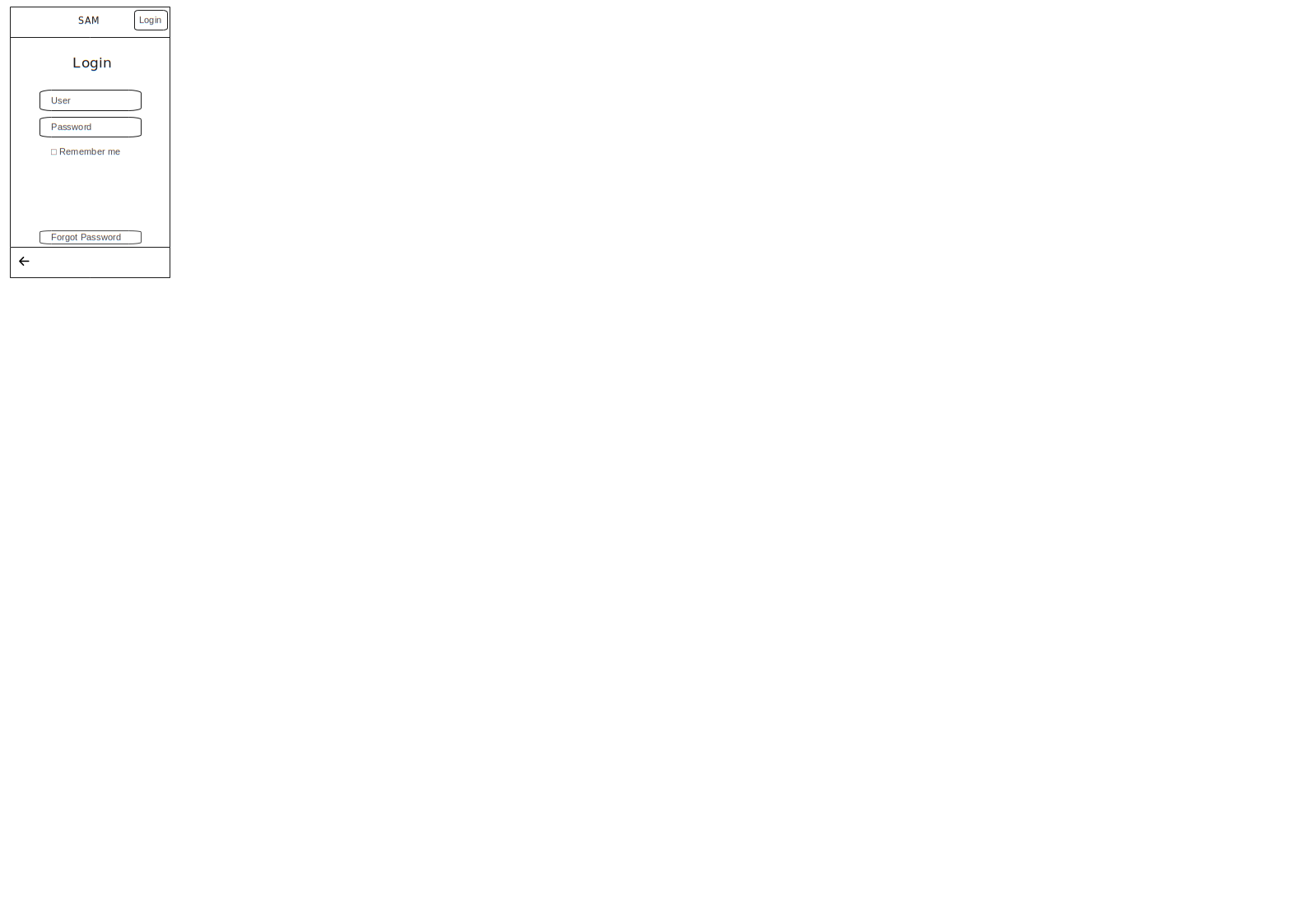
### 3.3.1 Initial screen

This is the welcome screen of the app, featuring the app logo prominently at the center to establish brand identity. Users are presented with two main options for accessing the platform: 'Login or Register (Private)' and 'Register (Company)'. The 'Login or Register (Private)' option is for individual users who wish to either log into their existing personal account or create a new one. The 'Register (Company)' option is designed for business users who need to create an account for their company, allowing them to access business-specific features and services. The layout is clean, intuitive, and easy to navigate, providing a seamless entry point into the app for all types of users.



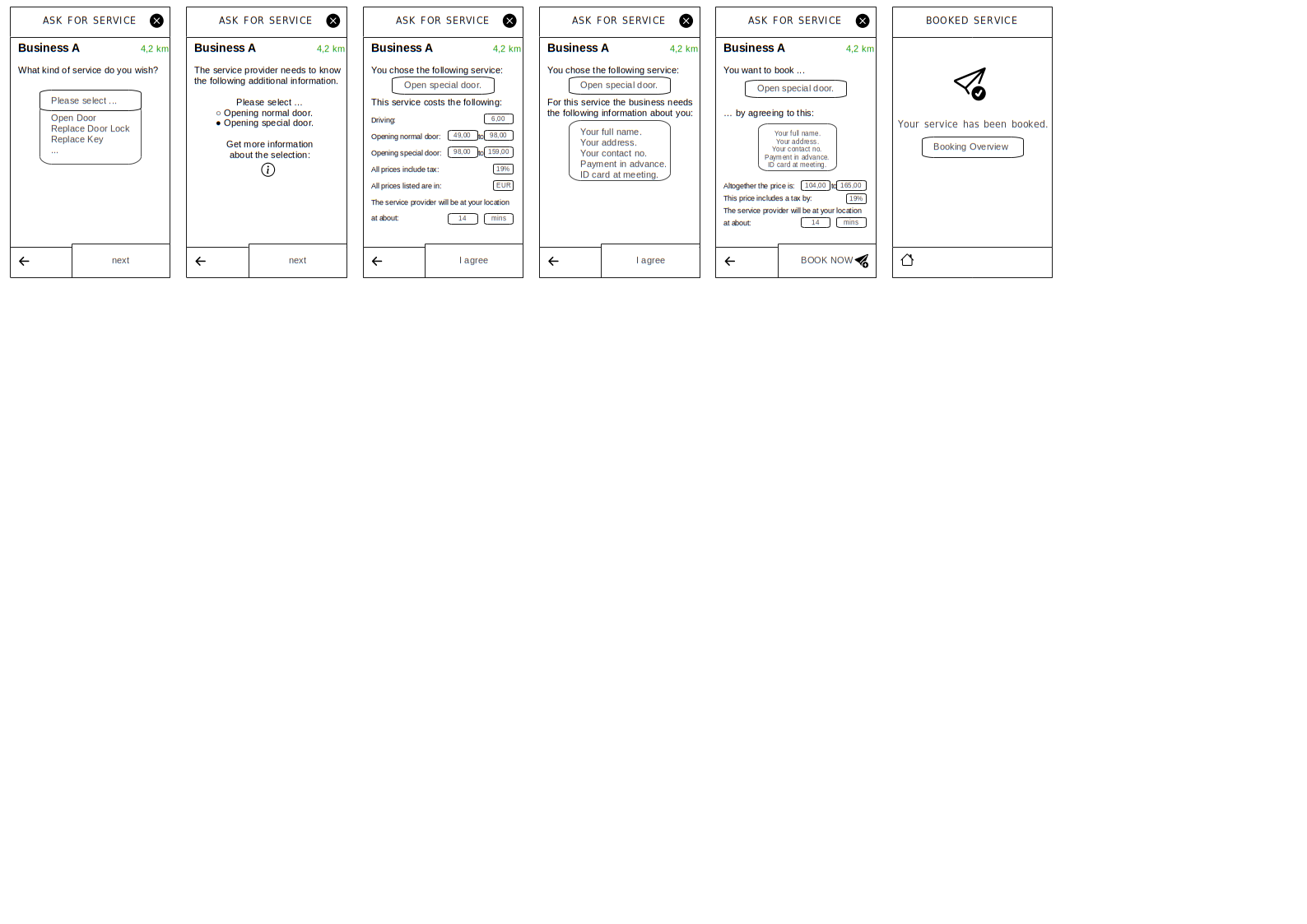
### 3.3.2 Login screen

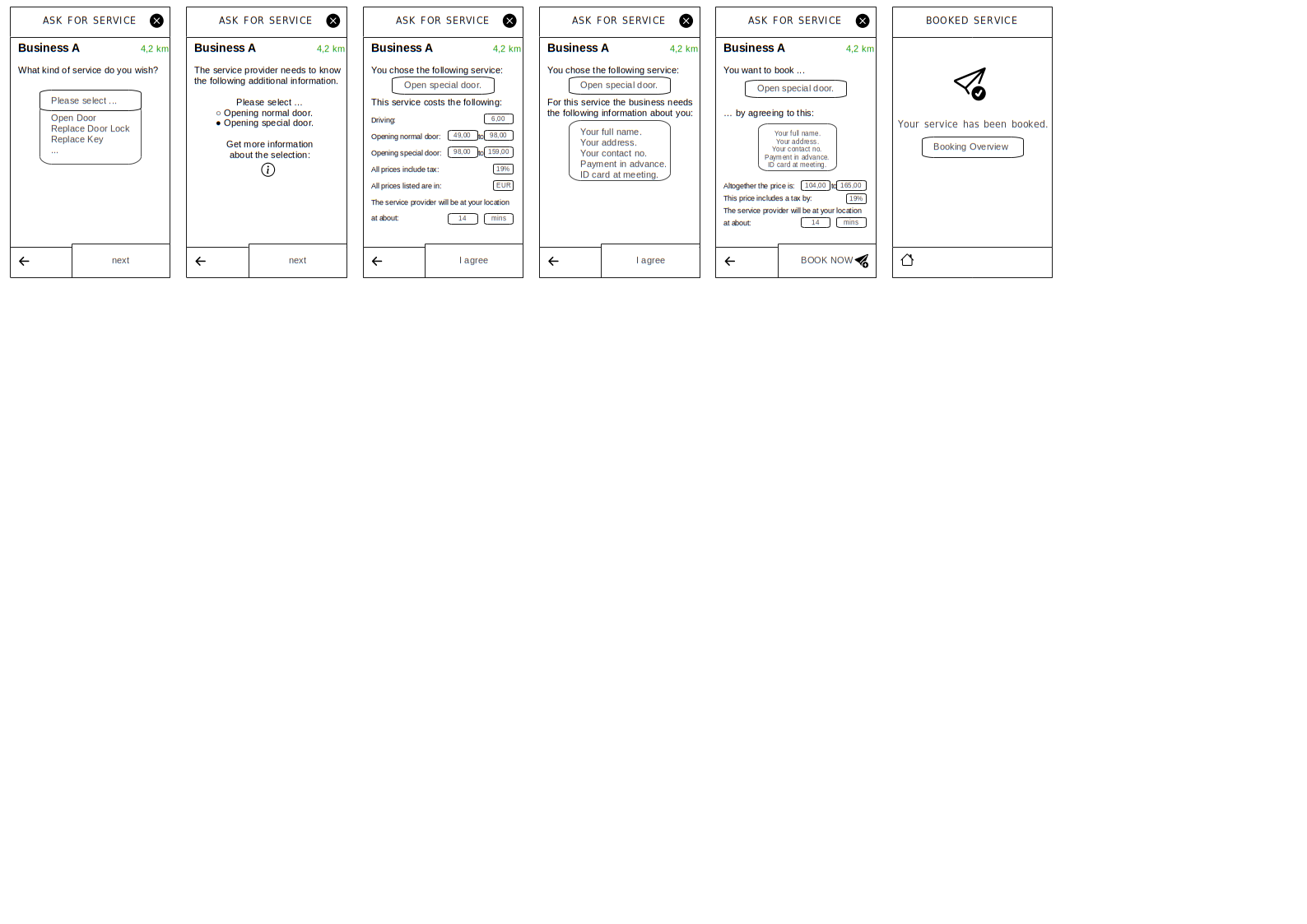
This screen is designed for users to securely log into their account. It prompts the user to enter their username and password to gain access to their personal dashboard. The 'Username' field is where users input their unique identification, while the 'Password' field ensures the security of their account by requiring a confidential passcode. There are also options to recover a forgotten password or to create a new account for new users. The screen provides a clean, user-friendly interface to help users easily navigate through the login process and ensure their information is protected.



### 3.3.3 Action booking (Customer)

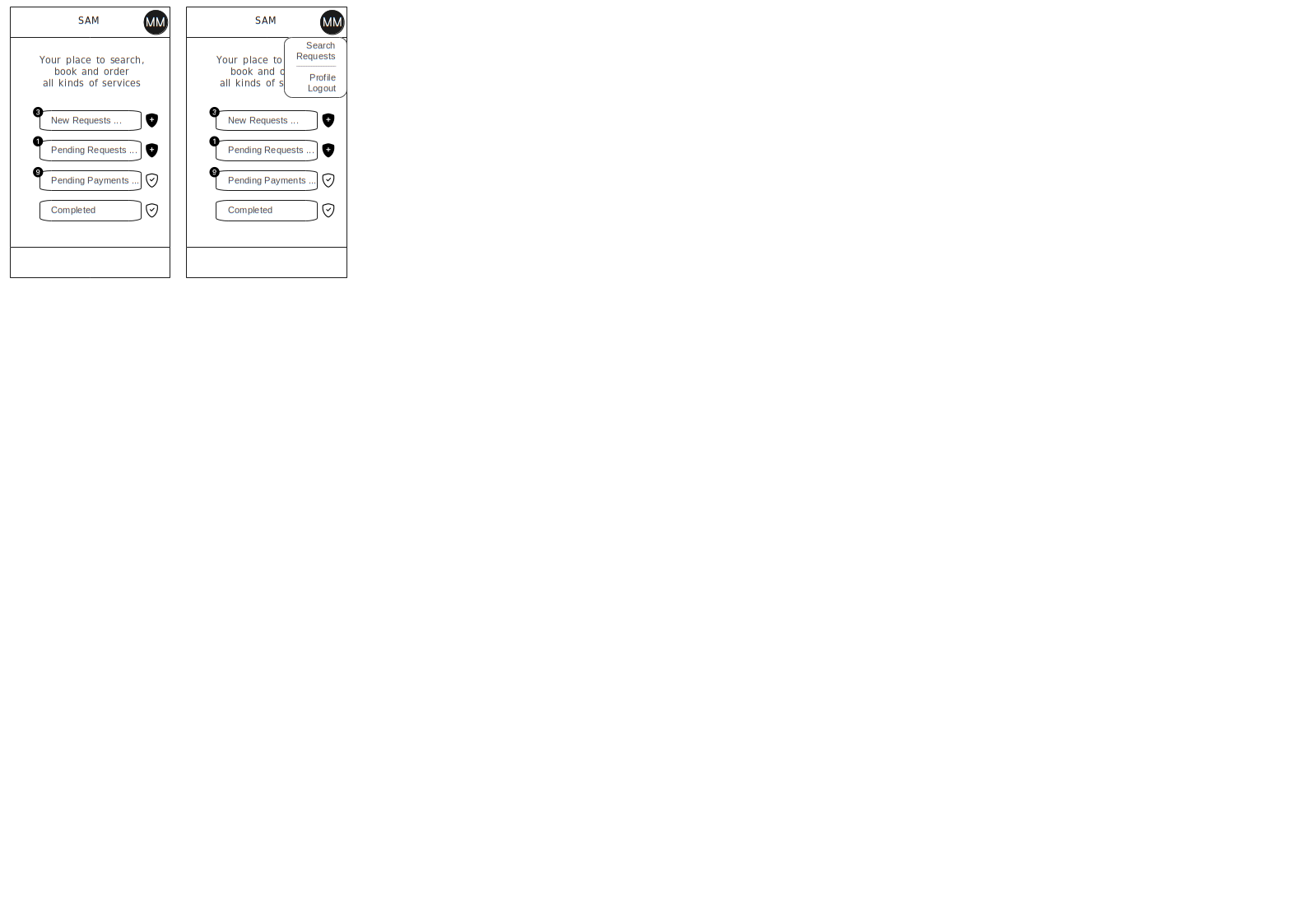
On this screen, the user is prompted to input comprehensive information related to their business requirements. This includes specifying the exact services they require, as well as any relevant details about how these services should be structured or tailored to meet their needs. Additionally, the user is asked to provide information about how the services will be arranged, such as the preferred timing, method of delivery, or any special conditions that may apply to the service setup





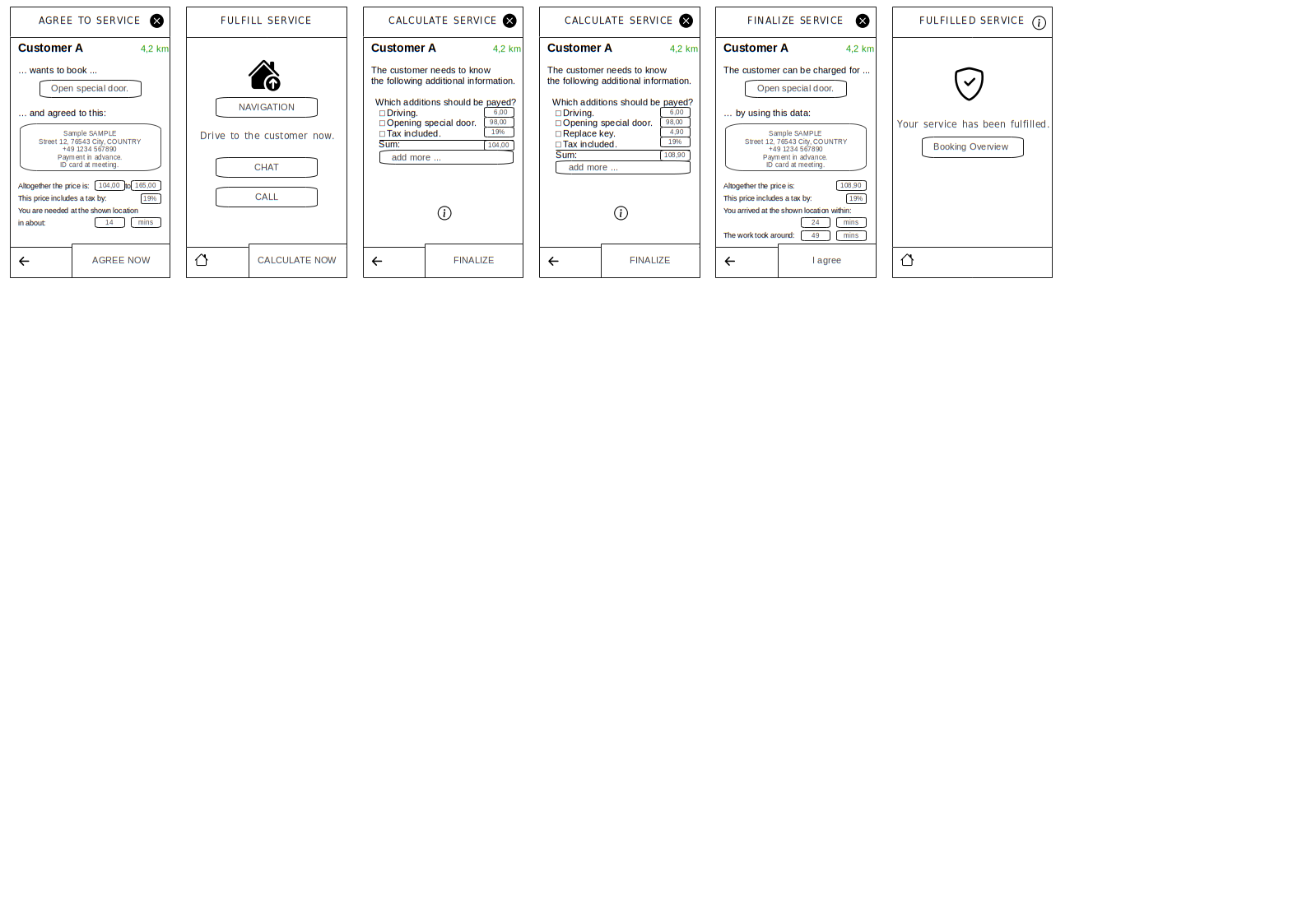
### 3.3.4 Profile sp

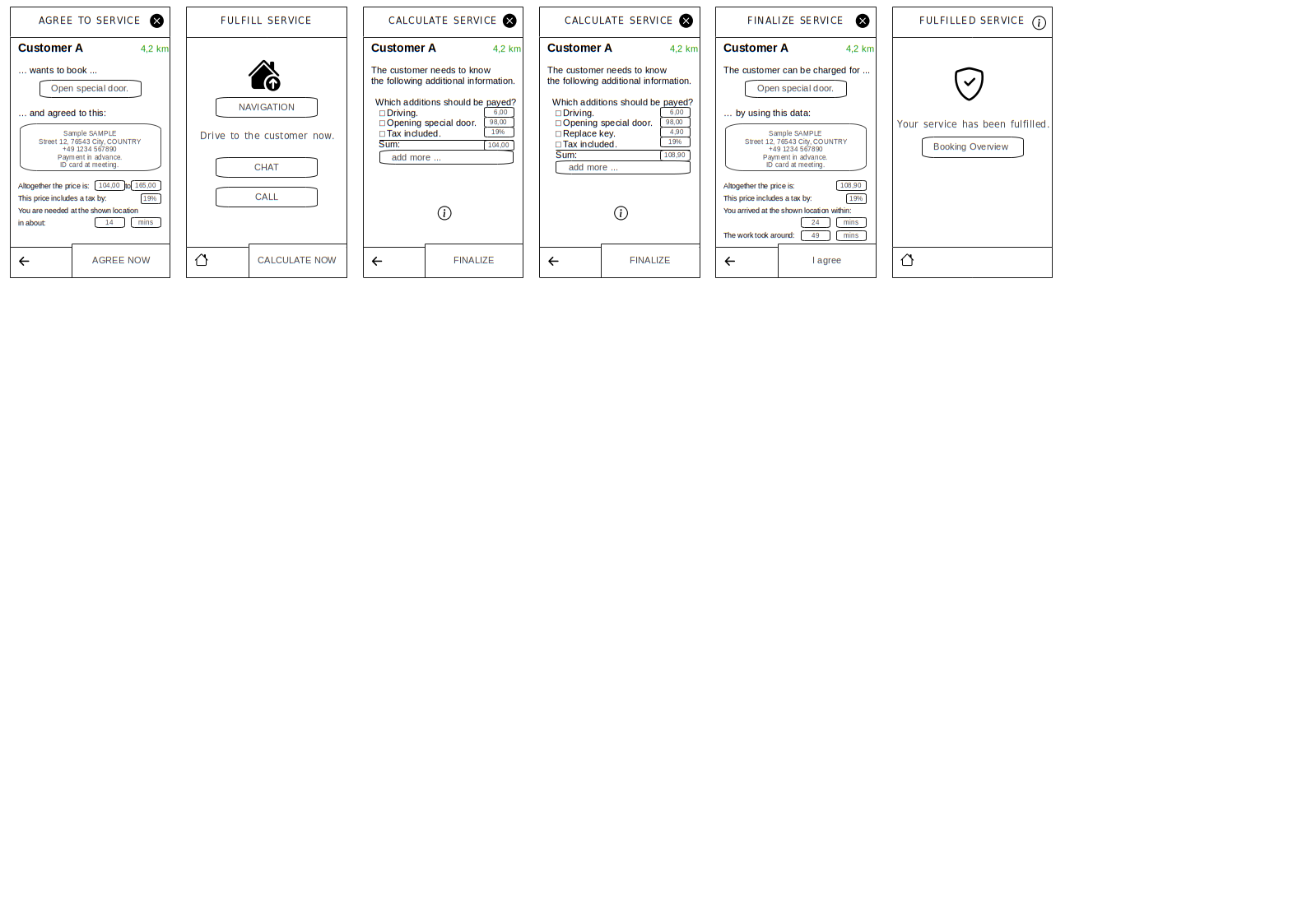
This screen displays the user's profile, offering easy access to various options for managing their requests and payments. The 'New Request' section allows the user to initiate service requests based on their current needs. The 'Pending Requests' section provides a clear overview of any ongoing requests that are still awaiting attention or approval. In the 'Pending Payments' area, users can view any outstanding payments that need to be processed, ensuring that they stay on top of financial obligations. Lastly, the 'Completed' section shows a history of all successfully fulfilled requests, giving the user a complete record of past services provided.



### 3.3.5 Action fullfill (Business)

This series of screens guides the company through the entire process of handling a customer request. First, the company reviews the customer's service needs and preferences. Next, they arrange a communication method—whether by chat or call—to discuss the specifics of the service, clarify any details, and ensure mutual understanding. After that, the company creates a detailed bill, including all services requested, their associated costs, applicable taxes, and any discounts. Once the bill is confirmed, the company and customer proceed to finalize the booking by confirming the service schedule and locking in the details, ensuring everything is set before the service is delivered.





# Design decisions and implemetation

The design decisions and implementation process are critical in translating the conceptual model into a functional application. Design decisions are guided by user needs, technical feasibility, and the overall vision of the project, ensuring a balance between aesthetics and functionality. The implementation phase focuses on bringing these decisions to life, utilizing appropriate technologies and development practices to create a seamless and efficient user experience. Together, these steps ensure the application meets its objectives and provides value to its users.

### 4.1 Plattaform used(VS,AS)

The development process utilized Visual Studio and Android Studio as the primary tools. Visual Studio was employed for backend and cross-platform development, while Android Studio was used for designing and implementing the native Android application. These tools provided robust support for efficient coding, testing, and debugging.

As a coding languaje, we used Flutter as an option

.  

### 4.2 Packages used (tools)

The application leverages various React packages to streamline development and enhance functionality. These packages provide pre-built components, tools, and utilities that simplify tasks such as state management, routing, and API integration. By utilizing these packages, the development process becomes more efficient, ensuring a seamless user experience and adherence to best practices in React development.

### 4.2.1 Flutter

Flutter was chosen as the framework for this project to enable the development of a high-performance, cross-platform application. Its single codebase approach allows seamless deployment on both Android and iOS, reducing development time while ensuring consistency across platforms. With its extensive widget library and robust support for UI customization, Flutter made it possible to deliver a visually appealing and responsive application that meets the project’s requirements.

### 4.2.2 Dart

Dart was employed as the programming language for certain mobile-focused aspects of the project, particularly in conjunction with Flutter. Its fast performance and clean syntax made it an ideal choice for building responsive and efficient mobile features. By using Dart, the development process benefited from its seamless integration with Flutter, ensuring smooth and high-quality implementation of mobile functionalities alongside the main React-based application.

### 4.2.3 Firebase (Web storage)

Firebase was utilized as the web storage solution for the project, offering a reliable and scalable backend for managing user data. Its real-time database and cloud storage capabilities allow for seamless data synchronization across devices, ensuring that users’ information is always up-to-date. Firebase’s easy integration with React also simplified the process of handling authentication, storing user profiles, and managing other dynamic data within the application, making it an essential tool for smooth data management and storage.

### 4.2.4 Communication (chatting)

For real-time communication, the application utilizes Socket.IO to enable seamless chat functionality. This technology allows for instant, bidirectional communication between users and service providers, ensuring that messages are delivered in real-time without delays. By integrating Socket.IO, the application supports a smooth and interactive user experience for chatting, whether it's for clarifying service details or coordinating appointments. This ensures that users can easily communicate within the app, enhancing the overall service experience.

# Results

You can use the heading types for each section

There is the main one:

* Then the second one
* And the third one

# Conclusion

This app was developed to address the challenge of connecting users with trusted local service providers in a simple and efficient way. The implementation focused on creating a user-friendly platform with features like real-time communication, location-based recommendations, and transparent billing.

The process presented challenges, such as ensuring data security, balancing simplicity with functionality, and managing technical complexities. While the app successfully streamlines booking, some drawbacks, like limited accessibility for users without smartphones or service gaps in rural areas, remain.

Despite these challenges, the app simplifies local service booking and empowers both users and providers, with room for future improvements to expand its reach and impact.