university of groningen

faculty of mathematics and natural sciences

Advanced Software Architecture

Train ticketing system

Aida Baxhaku (3242528)

Petar Hariskov (3226735)

Alexandra Matreata (3179265)

Eric Rwemigabo (3040356)

January 13, 2017

version 1.0.0

Revision history

Contents

List of figures

List of tables

Glossary

ET	Easy Ticketing

Chapter 1

Introduction and goals

1.1 Introduction

Traveling by train is one of the most common means of transport in the Netherlands and the Western Europe. The majority of people use trains on a daily basis, to go to school, work, etc. It is quite easy and comfortable, but one common problem, that everyone may relate to is the discomfort of train tickets. The printed tickets might get lost, or a passenger might forget to check in, another one is too late to buy one, etc.

However, we have come up with an alternative train ticket, which will facilitate the journey of many passengers: Easy Ticketing (ET). This is a new, promising and innovative technology that suits practically everyone who owns a smartphone. The idea behind ET is very simple: you download an app in your smartphone, open your personal account and connect it to your favorite payment method, and ET will do everything else automatically.

This means, every train will have beacons, which will connect to the passenger's phones via bluetooth, it will check the passenger in, it will automatically calculate the ticket fee and it will reconnect when the passenger leaves the train. There will be a beacon per every gateway. The beacon can detect the mobile in a diameter of 20m.

The aim of this new technology is to facilitate the journey of train passangers, and take the ticket purchasing to the next level. We plan to implement ET firsly in Groningen and after the initial success the aim of our company is to cover the whole Netherlands.

In this document we are presenting the architecture of our system. In the next chapters you will be introduced to the Stakeholders of the systems and their concerns, the requirements, functional and non-functional. In section 3, we will briefly analyze the business context and the benefits that ET will brings in terms of business. In section 4, the solution strategy is presented, in terms of our main key drivers. Section 5 outlines the building block view, whereas chapter 6 and 7 introduce the runtime and deployment view. The architecture of our system will be concluded with the design decisions and quality scenarios.

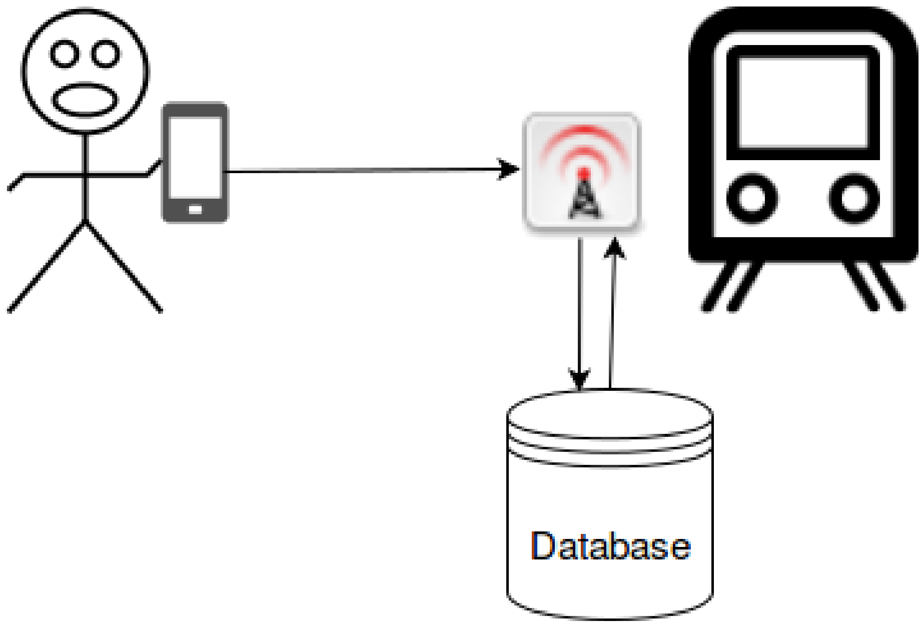

Figure 1.1: Overview of the System

1.2 Quality goals

The top three goals of the architecture of Easy Ticketing whose fulfillment is of highest importance to the major stakeholders (as agreed between them) are listed below:

Security In our system, we could view the importance of security in two major aspects: Firstly, hackers might hack the system and steal money from the account of the train company and secondly, passengers might find a way to fake their tickets. In both cases, this is unacceptable for our customer under any circumstance. Therefore the system must be designed in such way that potential data leaks, which are in theory inevitable, will not result in access to the main database or create damage to the company.

Reliability Customers are expected to rely on the functioning of Easy Ticketing. Problems with availability of the service are almost as undesirable as security issues, therefore availability must be taken into account throughout the whole system architecture. Breakdown of the system could lead to financial loss for the company, which is in any case not acceptable.

Compatibility Is an important driver for our system. Passengers may be in possession to different smartphones, which may result in incompatibility with the beacon, incompatible frequencies of bluetooth. It is very important to provide the passengers the possibility to purchase the ticket, for as many phone versions as possible.

Chapter 2

System scope and context

This section outlines the main relations between the system and its environment, external systems or entities with which it interacts. The business and technical contexts in which the system will perform as well as different inter-component communication solutions will be presented in this chapter.

2.1 Business context

The system focuses on simplifying the management of information related to train traveling and ticket payment for its users. The system will provide an account for each user which will store information regarding travel distance, destinations, payments, etc. This will help both users travelling by train and the companies providing the travelling services by keeping track of all these components in a centralized manner.

The system will present the user with the choice of making an automated payment for each trip through the connection between the mobile app and the beacon in the train cart. Alternatively, an external payment system will be presented to the user I every train station where they can log in and perform the transaction.

A second important target for our system will be represented by companies providing travelling services by train, which may include governmental institutions or different private companies.

2.2 Business drivers

This chapter describes the business forces acting on the system: most important revenue streams and costs and give a justification for the system.

2.2.1 Revenue streams

The main revenue stream for the system will consist of the travel service providers. In order to use the Easy Tickting system, they will have to pay a monthly fee which will include storage and maintenance costs.

Depending on how well the system usage evolves over time, once enough travel service providers decide to use the system, a small monthly fee can be charged to the user as well. Also, the mobile app functionalities can be improved over time and be accessed by the user over a pay-by-use scenario(e.g. check distances travelled over the last period of time, check fees, see a list of possible price reductions for similar journeys).

2.2.2 Costs

The costs for the system will be divided in two main groups:

- **development costs:**
  these costs will be comprised of installation costs(hardware equipment such as beacons and local servers will be installed and configured inside the trains, central server configuration, backup servers configuration) and software development costs(for the mobile application, databases, computational algorithm for fee calculations).

- **maintenance costs:**
  maintenance operations will include constant monitoring of the hardrware equipment(remote monitoring and repairs for components inside the trains and a local team for the central server) and a team for additional functionalities(for the mobile app, optimisations of the computational algorithm, etc).

2.3 Domain model

Figure 2.1 outlines the most significant entities taking part in or being used by the Easy Ticketing system. The system components and their interactions are presented alongside the main external entities needed for different functionalities of the system. The figure also introduces the main actors responsible for the use cases presented in chapter 4.

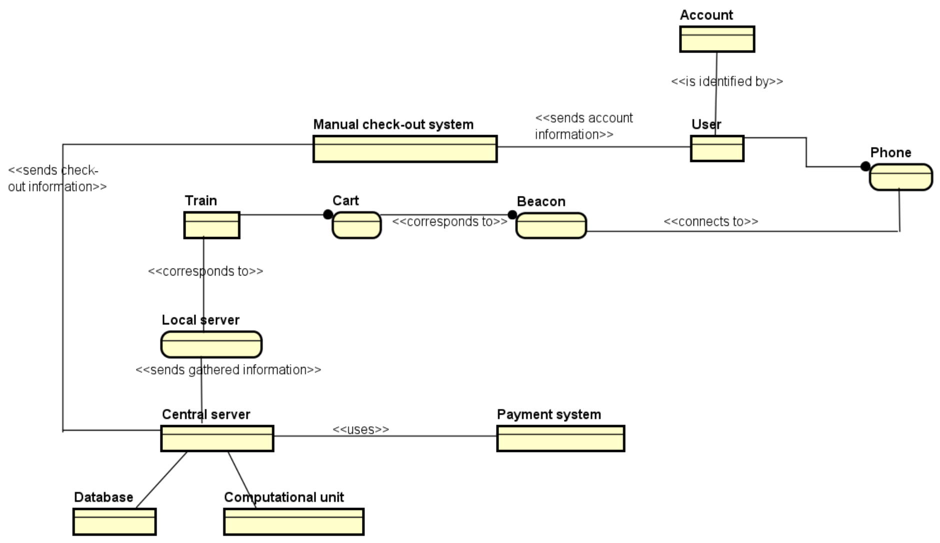

Figure 2.1: Domain model

2.4 Technical context

The system will need to perform in a specifically designed technical context consisting of three main environments: a central server collecting all information and performing necessary computations, the setup inside the trains themselves (comprised of beacons and small servers which will collect information per train and send it to the central one every time the train leaves the station) and a login system in each station allowing users to manually check-out of their trip and perform payment.

A set of measures will need to be taken into consideration for the system to be able to operate inside this context by using these different components and environments. These measures will be related to the key attributes required of the system as follows:

- **security**: since the system will have access to sensitive information (such as location, financial transactions), every connection between components of the system will need to be secured and trusted. The main connections which will need to be verified each time they are created will be: beacon to mobile application, beacon to server inside the train, small local server to central server.

- **reliability**: for the system to be considered reliable, the main components should be provided with a back-up in case of failure, such as the central server, the beacons in each cart, etc. Also, users should be presented with alternative scenarios in case the main desired activity flow gets interrupted (e.g. possibility to manually check-out of a trip using login system in train station in case of beacon to phone communication being severed)

- **compatibility**: the system will need to be compatible with at least the main and most commonly used mobile OS and frequencies

2.5 External interfaces

Channel	Beacon $\Rightarrow$ Mobile app
Description	Beacons detect mobile phones in their proximity having installed the application and initialise a connection.
Connection	Bluetooth
Protocol	iBeacon
Frequency	The train has left a station and a mobile phone is in range of the beacon.

Table 2.1: Interface - Beacon to phone

Channel	Beacon $\Rightarrow$ Local train server
Description	Beacons send information gathered every time a train departs from a station.
Connection	Bluetooth
Protocol	iBeacon
Frequency	The train has left a station.

Table 2.2: Interface - Beacon to server

Channel	Local server ⇒ Central server
Description	The local server situated inside the train sends information gathered periodically to the central server if an internet connection is possible.
Connection	Internet
Protocol	UDP
Frequency	Once every 20 min or when a connection is possible.

Table 2.3: Interface - Local server to central server

Chapter 3

Requirements

In this section the main Stakeholders and the functional and non functional requirements of Easy Ticketing will be introduced.

3.1 Stakeholders

The following stakeholders are described along with their concerns. These concerns eventually determine the key drivers of the service.

3.1.1 Represented stakeholders

Passengers are the direct users of Easy Ticketing. The typical target customer is the traveler who uses the train systems regularly. Passengers are mainly concerned with the reliability and the availability of the system as well as the security often their data and the regular payment of the ticket. If the service is not available then they risk not having a ticket and eventually getting a fine. Another concern of the passengers could be the incompatibility of their phone with the beacon, which could risk the validity of their ticket.

Customers are the owners of the Easy Ticketing. The typical target are the large train companies, operating in the Netherlands. The companies are mainly concerned with the reliability of the system, since many passengers could end up not paying, given that there is a breakdown of the system. This could result in monetary loss for the company. Another main concern of our customer is the security. If the system is hacked the company loses money and this is certainly something that the company wants to avoid.

Developers and maintenance team is responsible for building the software part of our system, test and debug it afterwards. They are mainly concerned with the maintainability, testability and portability of the system.

Architects will be the team that will design the system. Their main concern is satisfying the needs of all the stakeholders and finding the best solution. They determine the feasibility of desired properties and functions, and guarantee a satisfactory end-product.

3.1.2 Non represented stakeholders

Hardware companies are the companies that provide beacons for our system. In our case they are non represented stakeholders, since they do not have much say

Hardware maintenance team maintains the physical side of the system, in our case beacons and takes care of handling malfunctions.

Ticket inspectors are the employees of the Train companies, that check the passengers, whether they have an available ticket or not. The are mostly concerned with the security and the reliability of the system.

Third party payment systems are companies that provide a paying system for the passengers.

3.2 Key Drivers

In order to determine the most important key drivers for our system, the stakeholders are required to give points to the key drivers according to what they consider as the most important driver for the system. Since the weight of the decision differs among our stakeholders, they are given different amount of points. The train companies (customer) have 150, the architects have 120, the developer team has 100, and the passangers 80. Considering the distribution of the points, the top 3 will be chosen as key drivers of our system. table 3.1 below shows how the different stakeholders used there points.

	Security	*Reliability*	*Compatibility*	*Performance*	*Maintainability*	*Scalability*
Train Companies(Customer)	50	50	20	30	0	0
Architects	50	30	20	20	0	0
Developer team	0	10	10	20	50	10
Passengers	40	10	30	0	0	0
Total	**140**	**100**	**80**	70	50	10

Table 3.1: Key driver selection

Security is concerned with the management of possible risks that may affect our system. Since ET is working with delicate data which involves payments and bank account records The stakeholders came to the agreement that security is the most important key driver of the system. It is mostly affected by the ability of the system to survive attacks, threats, the algorithms used, etc. The customers(train companies) and passengers are mostly concerned with security.

Reliability is measured as the probability that a system will not fail and that it will perform its intended function for a specified time interval. This can be guaranteed by redundancy, implementing an extra database, increase in resources, implementation of more than a single point of failure for different parts of the system, etc. Passengers are mostly concerned with the reliability of the system as they don't want to risk not having a ticket, while train companies do not want to get any economical loss.

Compatibility is the ability of the software to work with other systems. In the case of ET, the system should be able to adapt to three different types of the operating system of the smartphones: iOS, Android, Windows. The stakeholders that are mostly concerned with this, are the passangers and the train companies.

Maintainability is the **web:software** ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.

Performance is concerned with how long it takes the system to respond to an event. The performance of our system is affected by:

- Latency (time between the arrival of the stimulus and the systems response to it)
- Deadlines in processing
- Throughput of the system (the number of transactions the system can process in a second)
- Jitter of the response (the variation in latency)
- Miss rate (the number of events not processed because the system was busy to respond)
- Data loss (data that was lost because the system was busy)

Within our system context, the performance which brings most interest to the stakeholders is the performance of ET during the event itself. This is related to the detection of risky individuals, matching them to their profiles in the database and fight detection. The stakeholders that are mostly concerned about performance are the architects and the customers.

Scalability is ability **web:df** of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged.

3.3 Functional requirements

FR-1	Must	There should be at least 2 different ways of payment.
FR-2	Must	The beacons shall connect to a phone's bluetooth and application.
FR-3	Must	The system MUST/WILL/HAVE TO charge a person based on account/travel location.
FR-4	Must	The system should verify which cart sends which data (or train actually).
FR-5	Must	Phones should be verified based on accounts.
FR-6	Must	The system should provide a back-up to log out of the trip in case battery dies via touch-screen panels in stations.
FR-7	Must	The beacons will start connecting to mobile devices once the train is 300m away from the station.
FR-8	Must	Phone application should have beacon verification.
FR-9	Must	The train will collect data from the carts/beacons and send them to the server after leaving a station.
FR-10	Must	The system should charge on trip exit.
FR-11	Must	Once a connection between a beacon and a phone is established, the system must send a notification to the user's phone.
FR-12	Must	Application provides a user interface enabling the user to check his account, update amounts.
FR-13	Must	The beacons register both logged in and not logged in users and match the account with the phone even if the user logs in later on during the journey.

Table 3.2: Functional requirements

3.4 Non-functional requirements

3.4.1 Security

NF-1.1	Must	One account is allowed to login at one smartphone at one time.
NF-1.2	Must	Users will need to authenticate in order to login.
NF-1.3	Must	The data information provided by the costumers will be safe and secure.
NF-1.4	Must	Every request from a subserver component to the main repository must be verified so that the repository knows at all times who sent the request.

3.4.2 Reliability

NF-1.1	Must	The databases must be available 99.9999% of the time during a train journey.
NF-1.2	Must	The system must be fault tolerant.
NF-1.3	Must	The beacons inside the carts will be operational at all times when the train is working.
NF-1.4	Must	The Server/database will be 99.999% available , with backups .

3.4.3 Compatibility

NF-1.1	Must	The system should be compatible with the 3 main os (ios, android, windows).
NF-1.2	Must	Compatible frequencies.

3.5 Constraints

A constraints is **web:bus-doc** considered as an element, factor, or subsystem that works as a bottleneck. It restricts an entity, project, or system (such as a manufacturing or decision making process) from achieving its potential (or higher level of output) with reference to its goal. We are going to analyze the constraints of our systems in three different categories: organisational, business and technological.

3.5.1 Organisational Constraints

3.5.2 Business Constraints

3.5.3 Technological Constraints

Chapter 4

Main Use Cases

The following diagram displays the use cases of the system. The main use cases will be further described in use case scenarios:

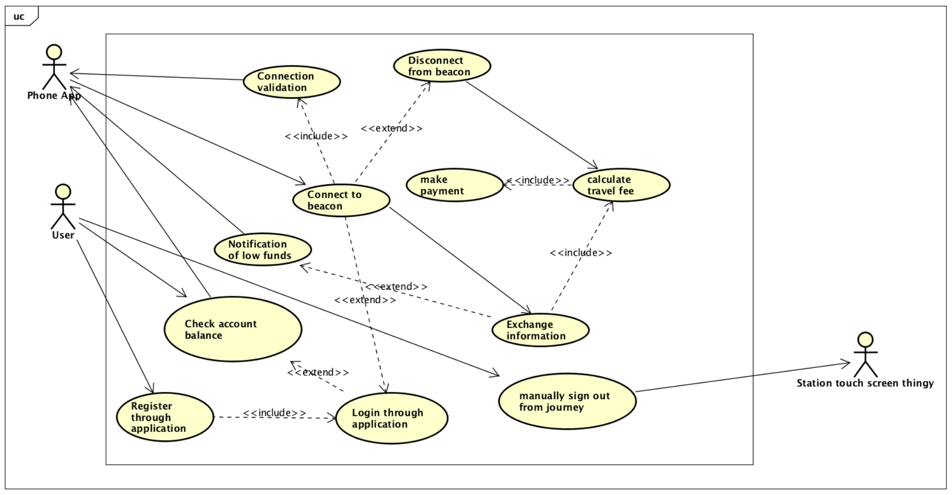

Figure 4.1: Main use cases

Usecase	Connect phone to beacon
Main actor	Phone app
Goal	Connect to the beacon
Preconditions	• Bluetooth is switched on • App is installed • Train has left the station
Main scenario	• The beacon discovers the phone • The phone receives a notification from the beacon • The beacon saves the account information locally
Exception scenarios	• The app doesn't get the notification because the phone is off • The app doesn't get the notification, due to a communication failure • The beacon fails to detect the phone • The app is installed but the owner doesn't have an account
Postconditions	• Phone is connected • Information exchange between phone and beacon
Related requirements	NFR2, FR2, NFR3

Table 4.1: Main use cases - use case 1

Usecase	Disconnect from beacon
Main actor	Beacon
Goal	Register a phone disconnection
Precondition	• Bluetooth is switched on • Use case 1
Main scenario	• The user leaves the train • The beacon checks available connections • The beacon sends gathered information to local server • Local server sends information to main server • Main server computes distance travelled by user
Exception scenarios	• Phone battery dies before user leaves the train • The beacon malfunctions and cannot check for connections
Postconditions	Travel distance and price are computed
Related requirements	NFR3, FR3, FR10, FR9

Table 4.2: Main use cases - use case 2

Usecase	Calculate travel fee
Main actor	Server
Goal	Charge user for travelled distance
Precondition	• Usecase 1 • Usecase 2
Main scenario	• Main server receives information regarding users which are still connected • Main server compares accounts currently connected with accounts connected for previous stop • Users which were connected previously and are not connected anymore are identified • Travel distance and fee are calculated for identified users
Exception scenarios	• Information cannot be transmitted between local and main servers • Account is not identified
Postconditions	• Notification regarding payment is sent to user • Payment is made
Related requirements	FR3, FR9, FR10, FR13

Table 4.3: Main use cases - use case 3

Chapter 5

Solution strategy

This chapter aims to discuss and present a concrete solution space for the problems described in chapter 2. The solutions provided will directly follow the main key drivers of the project and focus on the problem space using views limited only to these most important attributes.

5.1 Security

Different security measures will be taken into account so that the information transferred between different components is not visible to external parties and remains consistent throughout the exchange.

In this manner, the security mechanisms will focus mostly on the interface levels:

- **Beacon to phone:**
  Beacons taking part in the system will be uniquely identifiable via its id and mac address. When a phone connects to a beacon, the beacon's identity will be checked against a list of valid beacons by the mobile app.

  Only if the beacon is found in the list, the connection is validated and the mobile app sends the account information to the beacon, thus keeping account information secret from external parties.

- **Beacon to local server:**
  Similarly to the beacon verrification performed by the mobile app once a connection between the beacon and the phone is established, another verification will be performed at the level of the local server.

The local server will check the identity of each beacon sending it information against a list similar to the one available to the mobile app.

As a second security mechanism, before sending information to the server, the beacons will also check the identity of above mentioned server based on a hashCode.

- **Local to central server:**
  Every time the main server receives information from a local server, the identity of the local server will be checked using the hashCode mentioned for the previous connection.

  Also, before sending any information to the central server, every local server will ask the central server to identify itself using a cryptographic key.

5.2 Compatibility

The compatibility of the system mainly refers to mobile phones compatibility. The system will take into consideration different types of OS and different frequencies. While not all of these possibilities are taken into consideration, the system will focuss on the most used ones:

- OS compatibility(Compatibility-NF-1.1):
  - Android
  - iOS
  - Windows
- frequency: the beacons will send out 10 packages every second(Apple recommendation for optimal retrieval of packages**web:beacons**)

5.3 Reliability

The reliability of the system takes into consideration 3 main characteristics: availability, fault tolerance and recoverability**iso**

- **Availability:**
  Both the local and central servers will be provided with backup servers. The information stored in the local backup servers will be updated every time the train leaves a station, while the central backup server

will be updated more frequently, namely every time the central server receives information from one of the local servers.

This will allow the backup servers to be fully up to date and to minimize information loss. The backup servers will remain in a passive mode and only become functional once their corresponding main server fails to respond. Once the main server is repaired, information handling will be switched to it and the backup server will return to passive mode.

Every time a main server receives information(local server from beacons and main server from local one), it will notify the senders of having received the packages through a heartbeat signal. If this signal fails to be sent, the backup server will be started.

- **Fault tolerance:**
  Fault tolerance will be achieved by the system using resource redundancy and duplication of information. By using the backup servers and updating their information regularly, a copy of the information will be stored at all times minimizing information loss. Also, the beacons will be placed in such a manner as to have intersecting range areas, allowing at least two beacons to monitor a specific location in the same time.

- **Recoverability:**
  By enabling components to keep track of received information packages, the system state is known at each point in time, which allows a very fast detection of defective components. By minimizing error detection time, the recoverability time is also minimized.

Chapter 6

Software architecture

In this chapter the architecture of the system will be described. The 4+1 model will be used. We will be describing the logical view, the physical view, the process view and the data view. The logical view shows the functional decomposition of the architecture into components. the process view shows behavioral aspects of the system, the data flow view defines the data, the relation between data entities, how the data flows through the system and the measures to ensure reliability and performance. The physical view shows the physical entities and storage units as well as the deployment scenario.

6.1 Logical View

This view shows the connections between structural elements, key abstractions and mechanisms that are used within SFM. At first, an overview of the components is provided. The main components of the system will be displayed in terms of layers. Next, the main components are decomposed a in term of responsibilities and interfaces.

6.1.1 Primary presentation

In order to design the software architecture of ET, the layers pattern is used. This pattern is best suitable for the system since it gives the opportunity to abstract layers from one another, which increases the availability of the system by modularizing the components. The modularization of the components increases also the scalability and the performance of the system. The system has been structured according to the three-layers approach.

The three main components: Mobile app, Beacon Interface and the Local Storage, and Central storage and Payment system can be seen in the figure below.

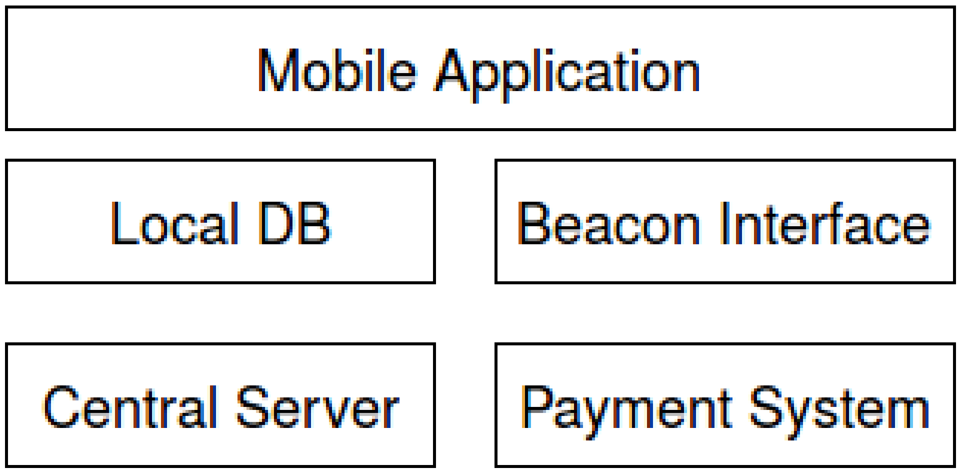

Figure 6.1: Overview of the System

Mobile app is the main component of our system. It will be available for download for the users and it will provide a personalized and unique account for the passengers. They can use this account to login and pay for their ticket, every time they use the trains. This layer will also enable a bluetooth connection with the beacon cart. The data collected from this layer is then sent to the other layers.

Beacon Interface and Local storage This layer creates a connection with the mobile app layer. Its components are the local storage, where the data gathered from the mobile app will be stored, and the beacon interface which establishes the bluetooth connection with the mobile.

Central Storage and Payment system The database and the Beacon Interface are included in this layer. The data that is provided from the Mobile app layer, will be stored in the central database after the mobile phone has connected to the beacon. The calculation of the ticket fees is also done in this layer through the computational center.

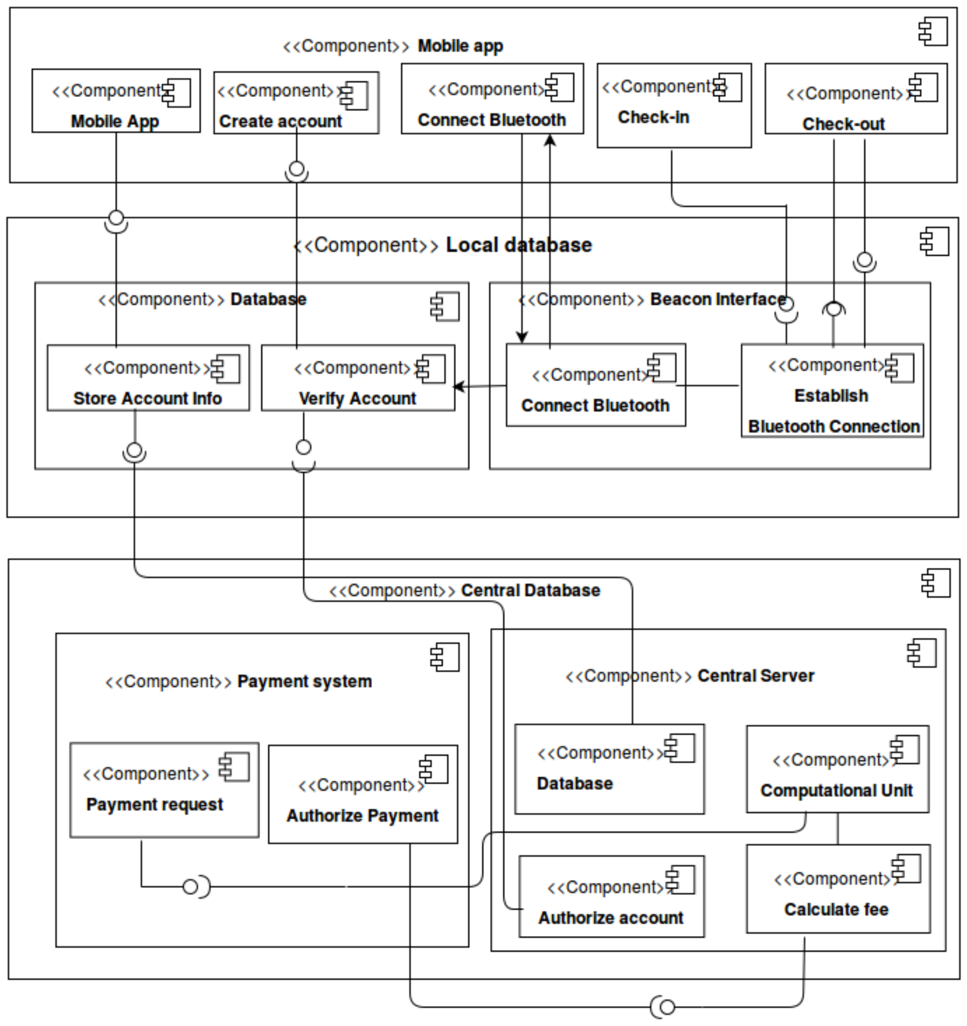
<<Component>> Mobile app
<<Component>> Mobile App
<<Component>> Create account
<<Component>> Connect Bluetooth
<<Component>> Check-in
<<Component>> Check-out
<<Component>> Local database
<<Component>> Database
<<Component>> Store Account Info
<<Component>> Verify Account
<<Component>> Beacon Interface
<<Component>> Connect Bluetooth
<<Component>> Establish Bluetooth Connection
<<Component>> Central Database
<<Component>> Payment system
<<Component>> Payment request
<<Component>> Authorize Payment
<<Component>> Central Server
<<Component>> Database
<<Component>> Computational Unit
<<Component>> Authorize account
<<Component>> Calculate fee

Figure 6.2: Logical View

6.1.2 Element Catalog

6.2 Process View

This section describes the main processes performed by the system following some of the main use cases presented in chapter 4. The following system sequence diagrams will help detail every such process by displaying the main information flows between different actors and system components in a chronological manner.

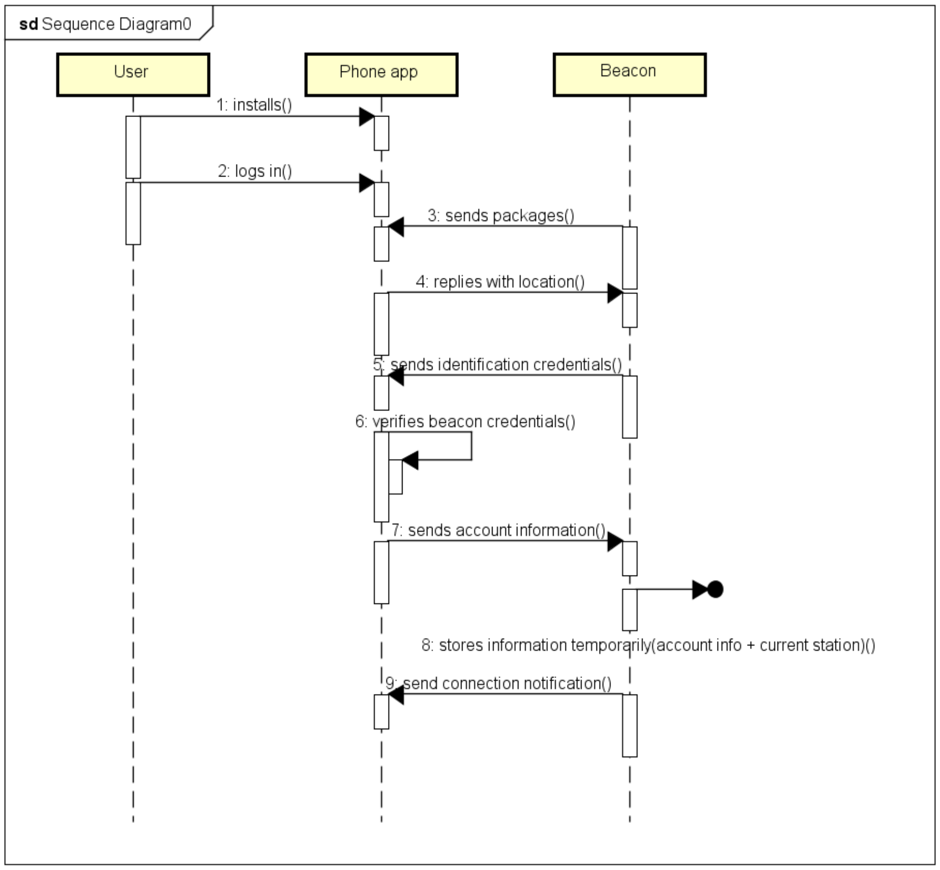
sd Sequence Diagram0
User
Phone app
Beacon
1: installs()
2: logs in()
3: sends packages()
4: replies with location()
5: sends identification credentials()
6: verifies beacon credentials()
7: sends account information()
8: stores information temporarily(account info + current station)()
9: send connection notification()

Figure 6.3: System sequence diagram- use case 1(Connect phone to beacon)

Figure 6.3 is a direct mapping of use case 1(connect phone to beacon).

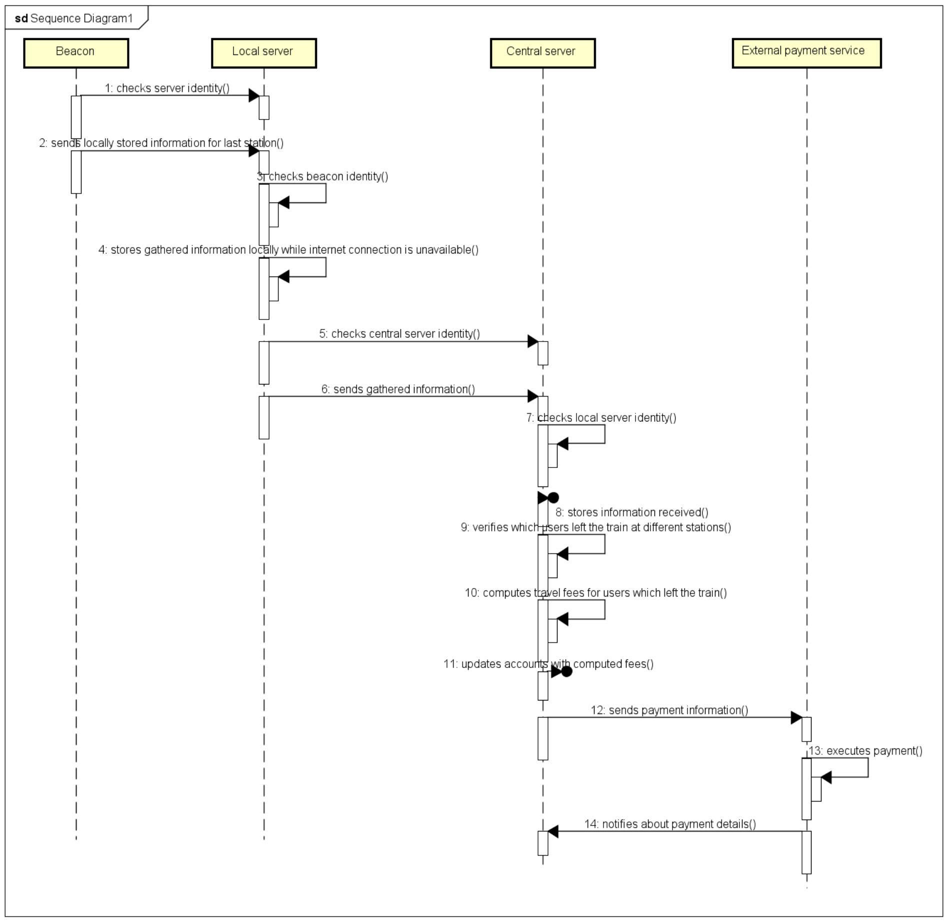
sd Sequence Diagram1
Beacon
Local server
Central server
External payment service
1: checks server identity()
2: sends locally stored information for last station()
3: checks beacon identity()
4: stores gathered information locally while internet connection is unavailable()
5: checks central server identity()
6: sends gathered information()
7: checks local server identity()
8: stores information received()
9: verifies which users left the train at different stations()
10: computes travel fees for users which left the train()
11: updates accounts with computed fees()
12: sends payment information()
13: executes payment()
14: notifies about payment details()

Figure 6.4: System sequence diagram- payment

Figure 6.4 displays the information flow corresponding to use cases "Disconnect from beacon" "Calculate travel fee" and "Make payment". For this process, the system uses a third party payment service.

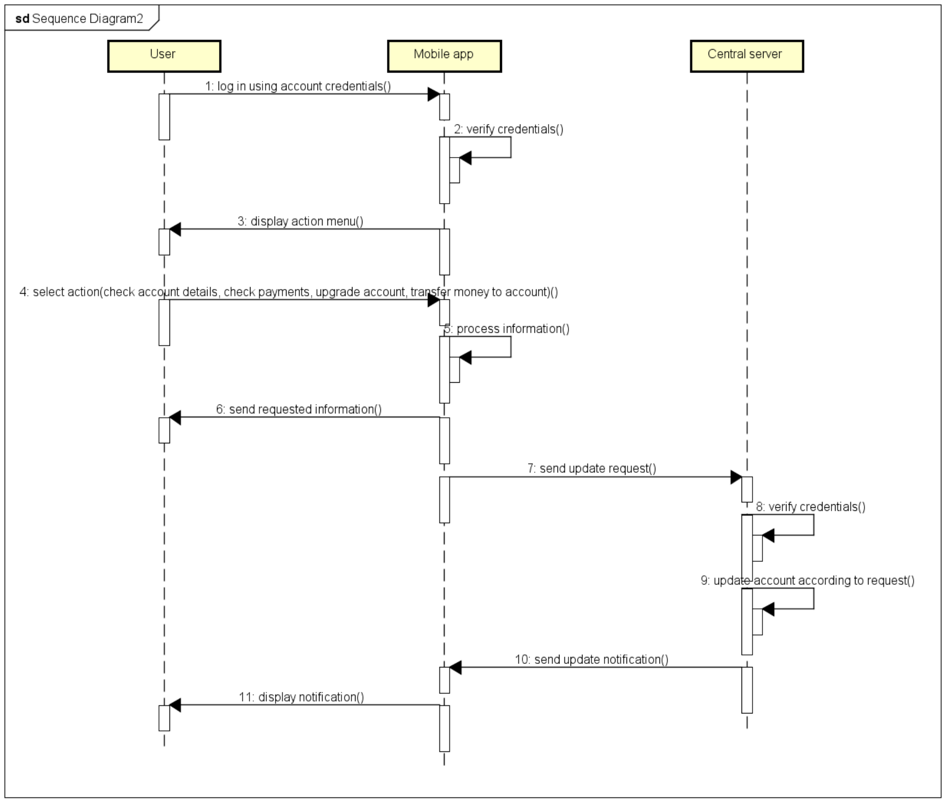
sd Sequence Diagram2
User
Mobile app
Central server
1: log in using account credentials()
2: verify credentials()
3: display action menu()
4: select action(check account details, check payments, upgrade account, transfer money to account)()
5: process information()
6: send requested information()
7: send update request()
8: verify credentials()
9: update account according to request()
10: send update notification()
11: display notification()

Figure 6.5: System sequence diagram- Check account details

Figure 6.5 displays the information flow related to use cases "Register through application", "Check account balance" and "Login through application". This process describes the general outline for different actions the user can perform using the mobile application.

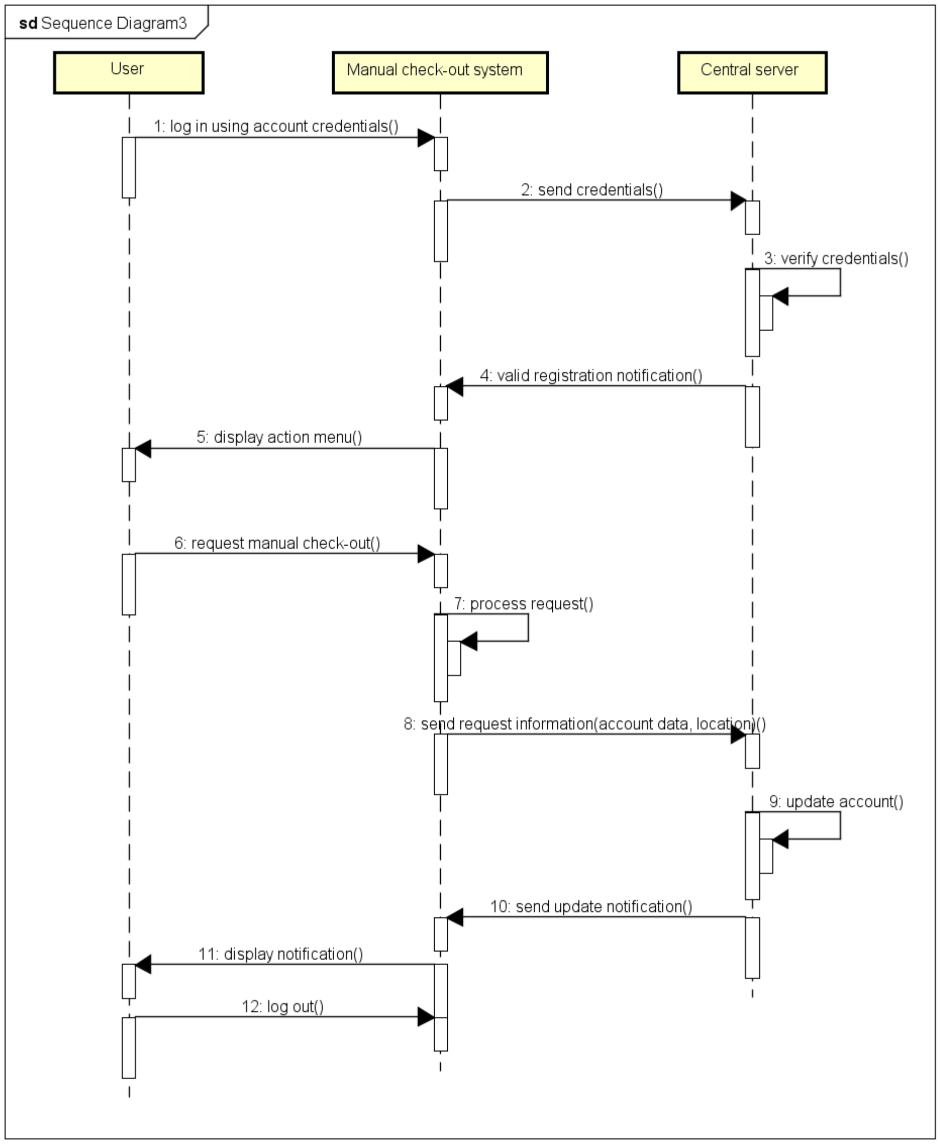
sd Sequence Diagram3
User
Manual check-out system
Central server
1: log in using account credentials()
2: send credentials()
3: verify credentials()
4: valid registration notification()
5: display action menu()
6: request manual check-out()
7: process request()
8: send request information(account data, location)()
9: update account()
10: send update notification()
11: display notification()
12: log out()

Figure 6.6: System sequence diagram- Manual checkout

Figure 6.6 shows the main information flow corresponding to the use case ”Manually sign out from journey”. This use case was designed as an alternative to the main ”Disconnect from beacon” use case which will give users the possibility to manually check out at a particular train station in case of the phone to beacon connection not being available.

6.3 Physical View

Chapter 7

Architectural patterns

Chapter 8

Design decisions

Chapter 9

Evaluation

9.1 Utility tree

The output of the utility tree is a list of scenarios that serves as a plan for the remainder of the ATAM evaluation. It shows the evaluation team where the most important points are and where to examine the architectural approaches and risks. The utility tree is composed of the key drivers of the system, which in our case are security, reliability, and compatibility.

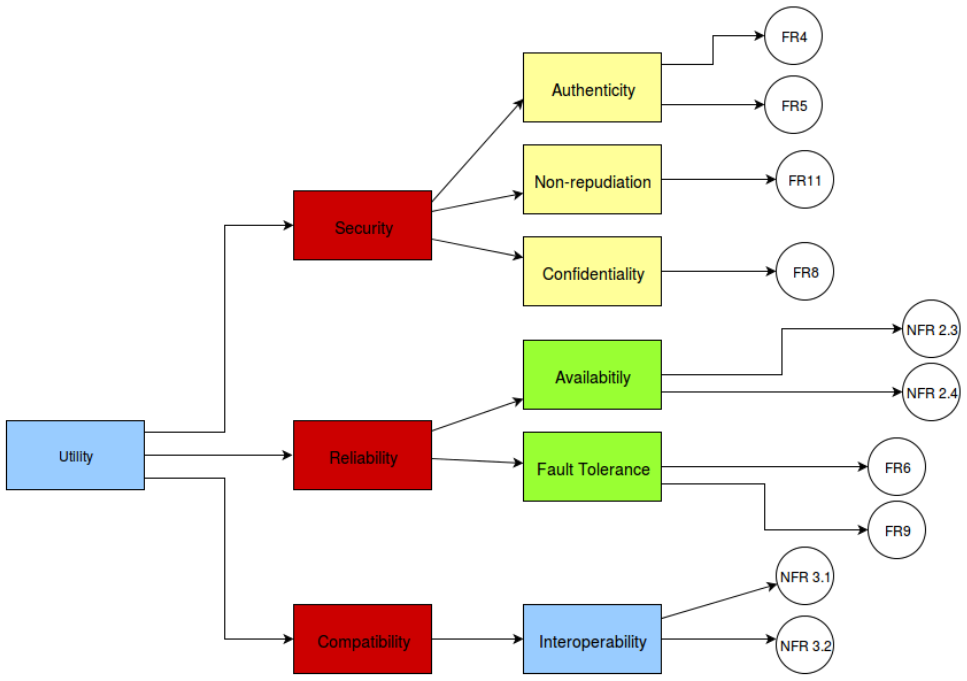

Figure 9.1: Overview of the System

9.2 Key driver verification

9.2.1 Security

9.2.2 Compatibility

9.2.3 Reliability

Notes

Appendix A

Time tracking