**PDA Flexible Ticket System**

3rd .Semester Project

Team5, PROI3 IT-Project

|  |  |
| --- | --- |
| **Supervisor(s):** |  |

Svend Mortensen – DIST3

Ole Rydahl – NET3

Roger Munck Fairwood – COMP3

|  |  |  |
| --- | --- | --- |
| **Student name** | **Student number** | **Signature** |
| **James Testmann**  **Lukas Janocko**  **Shicheng Dai**  **Sudhir Chaurasiya**  **Aimo Suikkanen** | **S071954**  **S135272**  **S133342**  **S137239**  **S082577** |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1. Introduction 2](#_Toc403806189)

[**1.1** **Problem Formulation** 3](#_Toc403806190)

[2. Requirements specification 4](#_Toc403806191)

[**2.1** **Functional requirements** 4](#_Toc403806192)

[**2.2** **Non-functional requirements** 5](#_Toc403806193)

[3. Problem analysis and solution 5](#_Toc403806194)

[**3.1** **System Description** 5](#_Toc403806195)

[**3.1.1** **Main Server** 6](#_Toc403806196)

[**3.2** **System Distribution** 7](#_Toc403806197)

[3.3 Domain model 10](#_Toc403806198)

[3.3.1 Class responsibilities 10](#_Toc403806199)

[3.4 Design model 10](#_Toc403806200)

[**3.4.1** **Use Case Diagram** 11](#_Toc403806201)

[**3.4.2** **State chart and activity diagrams** 20](#_Toc403806202)

[3.4.3 Solution classes/Interfaces 24](#_Toc403806203)

[4. Testing 24](#_Toc403806204)

[4.1 Unit Tests 24](#_Toc403806205)

[5. Conclusion 24](#_Toc403806206)

[5.1 Product oriented conclusion 24](#_Toc403806207)

[5.2 Process oriented conclusion 24](#_Toc403806208)

[6. Literature 24](#_Toc403806209)

[7. Appendices 24](#_Toc403806210)

[7.1 Glossary 24](#_Toc403806211)

[7.2 Test results 24](#_Toc403806212)

[7.2.1 Unit Tests 24](#_Toc403806213)

[7.3 Source code Java 25](#_Toc403806214)

# Introduction

This project intends to develop a prototype for a flexible ticket system.

Can be used in buses and trains. The system must operate so that customers are not physically must check in and out on buses and trains. This must be done by the customer’s PDA.

To communicate with the bus / train and automatically check in without the customer having to do anything. This communication will be through Bus / train LAN connection.

Servers and train computers should be located on our virtual servers (“Goonhilly”) and use relevant DBMSs – PostgreSQL. It is however acceptable to use a number of a team’s portable computers as bus / train computers.

After completion of travel, the system itself could calculate the price of the journey made, and withdraw this amount from the customer Account.

During the journey the customer should always be able to see its provisional Zone structure.

It will also develop a travel plan, which customers can use to calculate

the quickest route to their destination. This itinerary must be kept updated with delays and cancellations, so the customer will always get the fastest route based on the current traffic situation.

Customers should be able to log on to a website where they will be

register before they can use the system. This website also

Customer to view account balance, fullness money in the account and correct Personal information.

## **Problem Formulation**

Project tells the details about the new ticketing system for making the life easier to without doing physical interaction.

Public ticketing system where customers come to be recorded automatically by the buses or train. By doing this facilitates the Customers know that, they should not have to think about buying a ticket or scan in and out in each station.

For this to be done, there are a number of major issues to be

answered. It includes:

* How to register customer PDAS?
* Where to users' information stored once they have been registered of a bus or a train?
* How do we handle many trains / buses close to each other?
* The customer is registered on the correct transport?
* Where and how is the payment handled?
* How is a route calculated?
* How should the total distributed so that the system is as effective as Possible?

**1.2 The Problem Boundary**

The system is being developed is a prototype. The goal is to establish a system

------More details will come after our project work as way we want----with James---

The PDA part is not an app, but also simulated on a PC.

# Requirements specification

## **Functional requirements**

**Criteria:**

• Give the customer/client an easy and user experience.

• Design the system to require minimal user interactions.

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| R1 | The customer must be able to create a user via the website. |
| R2 | Client’s PDA to registration by BTC at out and input by train/bus |
| R3 | Customer must be able to log out /in its user |
| R4 | The customer must be able to deposit money into their user account |
| R5 | Client must be able to see his balance from his user account |
| R6 | The customer must be able to correct its information on the website |
| R7 | The client must be able see the price of its provisional travel at any time |
| R8 | The Customer must present a ticket on his PDA |
| R9 | The customer shall be automatically deducted from the price of the trip |
| R10 | The customer should be able to find the fastest trip e.g. from A to B from travel plan |
| R11 |  |
| R12 |  |

A customer needs to see and correct the information and requirements are designed.

So that the customer can find the information that is needed. This means that

System to meet all the requirements a customer has to make a journey.

## **Non-functional requirements**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| NFR1 | BTC(**BUS TRAIN SERVER)** will be developed in Java |
| NFR2 | Main Server to be developed in Java |
| NFR3 | Proprietary protocol to be developed in java |
| NFR4 | Algorithms for travel plan and price calculator to be developed in Java |
| NFR5 | The database will be developed in PostgreSQL. |
| NFR6 | The site should be developed with JSP and servlets |
| NFR7 | RMI to be used for the server |
| NFR8 | server must be on the DTU server |

# Problem analysis and solution

## **System Description**

The system shall consist of six parts:

* Database
* Main server
* Bus / train computer (BTC)
* Website
* PDA
* Proprietary protocol (between BTC and PDA) -------- not done yet…..RMI

### **Main Server**

* Main server to process all the requests coming from BTC.
* The server must be multithreaded so it can process multiple requests simultaneously. (not sure have to ask with James)
* Main server are on the DTU servers.

**3.1.2 BTC**

* Each bus or train will be installed with a BTC (Bus Train Computer).
* When the bus / train stops at a bus stop will BTC record all the customers who have gone on board the bus / train.
* For this communication, there is developed a proprietary protocol.
* BTC stores cached user information and sends to the main server.
* Main server verify all the customers and sends this information back to BTC.

**3.1.3 PDA**

PDA is the customer's smartphone / pocket computer which is used to register

On buses and trains.

And show status to conductor.

**3.1.4 The website**

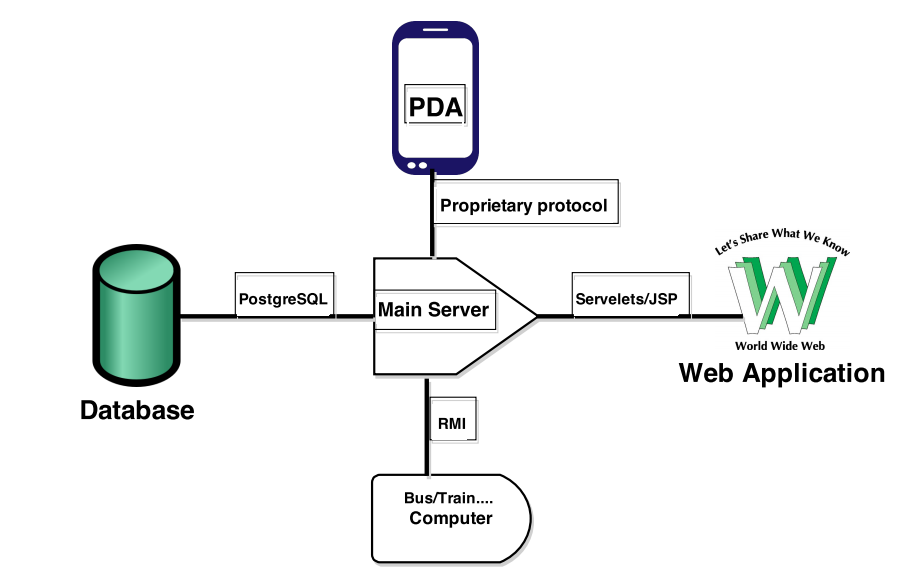
The website is developed with servlets and JSP (Java Server Page).

On the website, customers can create a user, deposit money into their account,

Change user info, see the travel history, see balance and use A\* to find the

Fastest route between two stops.

## **System Distribution**

****

**3.1.1 A \* algorithm**

To purpose of using the A\* algorithm to find the the fastest / shortest route between two points.

--------More details will come after the discussion with James……..

**3.1.2 Database analysis and Design**

As our lot of discussion to DBMS finally we reached on the specific point. The problem statement, it appears that the database must be accessible from many different applications through the server, making the design of the database is very important for instance, to avoid duplicate entries and tables are the same. The database is set up on the server set available at the DTU “Goonhilly". It is made with PostgreSQL as it is a good free database service. In the Other hands to select Goonhilly, The server is preloaded with two database handlers, MySQL and PostgreSQL. The PostgreSQL is preferred as this database is being taught in depth during the project.

To reduce strain on one database, more databases are created for the purpose having one database with the customer information, one database with the travel history of every customer and one database containing station names.

The CustomerInformationDB contains a unique customer id, the first name, last name and email. The customer id is auto generated for each added customer. A precondition is that emails are personal and unique, therefore the email is also a unique identifier, making sure a customer is not signed up with two accounts. This gives the option of two customers with identic names can sign up without interfering with each other’s account.  
A customer could sing up with two different emails, and disregard the purpose of a unique email as a customer identifier. This could be prevented by using social security numbers, but in this project there is no option for requesting a check on a social security number.

The stationDB contains a name for each station, but also a unique station number, making it possible for two stations to have the same name, but still be uniquely identified.

The TravelDB is going to be the largest of the databases, containing information about date, start time, start station, end time, end station, number of zones, cost of travel, customer id and travel number.  
The date and travel number are mainly used for statistics.  
The start and end station is the station number, used to calculate the number of zones travelled.  
The start and end times gives the travel duration, which is then compared to the number of zones travelled and the appropriate price can then be calculated. Zones is calculated as   
|EndStation - StartStation|= Zones. The Price is calculated to be 20, - pr. Zone.  
The customer id is used to tie a customer to a specific travel, making it easier to calculate price on demand.

----More details will come after the discussion with **James**….or may be have to modify the some of the things on the DBMS after our code will be work perfectly because according to James

**3.1.3 Database package**

|  |  |
| --- | --- |
| **Class:** **DataBaseHandling** | **Collaboration** |
| This class is the interface implemented by DatabaseHandler class. | Customer Collect Thread clientDatabaseHandler |

|  |  |
| --- | --- |
| **Class:** **DatabaseHandler** | **Collaboration** |
| This class has all the methods that retrieve / store data in the database. | DataBaseHandling |

|  |  |
| --- | --- |
| **Class: PostgresqlConnection** | **Collaboration** |
| This class is making connection between to our system for process all the data. | DatabaseHandler |

|  |  |
| --- | --- |
| **Class:** **User** | **Collaboration** |
| This class is the interface implemented by UserImp class. | This class has field, constructor, getter and setter for making available for doing the process in DatabaseHandler class. |

## Domain model

... (Explain what the domain model is and how the classes were found in the model)

### Class responsibilities

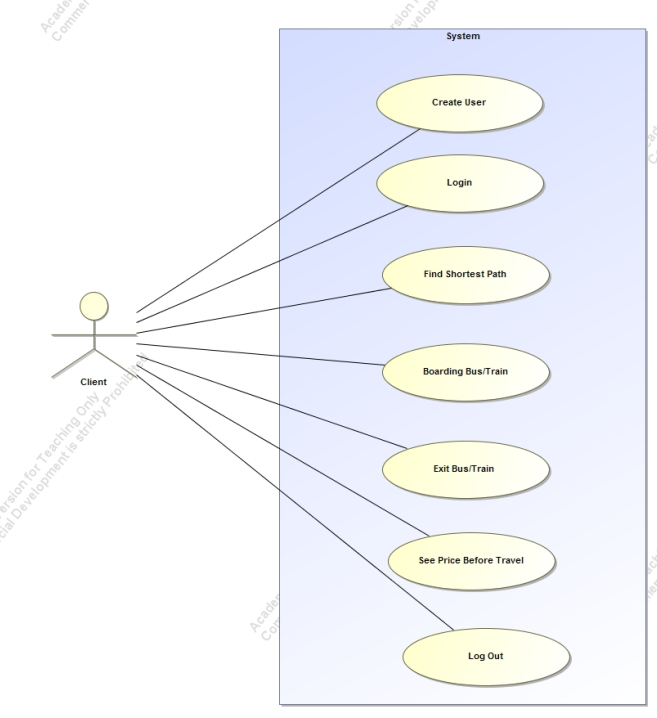
|  |  |
| --- | --- |
| **Class name: xx** |  |
| **Responsibility:**  xxx | **Collaboration:**  **xxxx** |

## Design model

... (Describe the design model and how it was determined i.e. consideration of inheritance, interfaces and design patterns in order to achieve a design having a low coupling and a high cohesion.

Only the most important classes, interfaces, attributes and methods are shown in the diagram)

### **Use Case Diagram**

****

|  |
| --- |
| **UC1:** *Log in* |
| **Brief description:**  Customer should be able to login. |
| **Actors:**  *Customer* |
| **Preconditions:**  *The users can see that in the homiside of log in website.* |
| **Main flow:**  1. This use case starts when the user wants to log into the system. 2. The correct username and password is entered. 3. The system checks if password is equal to the corresponding user. 4. If the password is correct, then the user has the permission to log in.  5. The user comes to the homeside. |
| **Post conditions:**  *Users log in after entering the right password and username.* |
| **Alternative flows:**  1. The user enters the wrong code. 2. The system checks if password is equal to the corresponding user. 3. The user is told that the password is wrong and that the password must be entered again. |

|  |
| --- |
| **UC2:** create User |
| **Brief description:**  *If users have no account, users can register accounts.* |
| **Actors:**  *Customers* |
| **Preconditions:**  *The customer will be connected to the website.* |
| **Main flow:**  1. This use case starts when a customer wants to use the travel card. 2. The customer chooses to create. 3. The customer enters their information. 4. The system stores all information in the database. 5. The system may provide instructions to the user is created |
| **Post conditions:**  *The customer is created and the customer's data is stored in the database.* |

|  |
| --- |
| **UC3:** Updating customer information |
| **Brief description:**  The customer can be able to update customers' information. |
| **Actors:**  customer |
| **Preconditions:**  The customer is logged in on the website. |
| **Main flow:**  1. Customers can change their information. 2. The system will find the customer information from the database. 3. The system changes the information in the database. 4. Customer will be informed that the information is updated. |
| **Post conditions:**  Customer has updated its information. |
| **Alternative flows:**  1. The customer presses cancel. 2. The customer will be notified that the update is interrupted. |

|  |
| --- |
| **UC4:** Boarding the bus or train |
| **Brief description:**  A customer should be automatically registered when boarding a bus or train. |
| **Actors:** *Customer* |
| **Preconditions:**  The customer's PDA is enabled. |
| **Main flow:**  1. This use case starts when a customer embarking on a bus or train. 2. BTC detects that the client is on board. 3. The system retrieves the customer's initial travel and caches this in BTC server. |
| **Post conditions:**  The customer is detected boarding on the bus or train. |
| **Alternative flows:**  If customer the PDA.  Customer will not able to connect to the system.  Customer will have to go to the further registration process. |

|  |
| --- |
| **UC5:** See preliminary price of travel |
| **Brief description:**  Before customer starts travelling, customer can check the price of where the destination is |
| **Actors:**  *Customer* |
| **Preconditions:**  The customer is on a bus or train. |
| **Main flow:**  1. This use case when the customer chooses to see it’s provisional. 2. Prices will be displayed on the website 3. Journey preliminary cost will be shown on the PDA |
| **Post conditions:**  Customers are still on the bus or train. |
| **Alternative flows** |

|  |
| --- |
| **UC6:** see balance |
| **Brief description:**  A customer should be able to see his balance. |
| **Actors:**  *Customer* |
| **Preconditions:**  The customer is logged in to the site |
| **Main flow:**  1. This use case starts when the customer chooses to see his balance 2. The website takes the customer's balance from the database 3. The balance is displayed on the home website. |
| **Post conditions:**  The customer is the home page |
| **Alternative flows:** |

|  |
| --- |
| **UC7:** Find the shortest route |
| **Brief description:**  A customer should be able to find the quickest route on the website |
| **Actors:**  *Customer* |
| **Preconditions:**  The customer is logged in to the site |
| **Main flow:**  1. This use case starts when the customer chooses to find the fastest route in the the website 2. The customer selects the two stops he will soon find route 3. The website will find the fastest route and displays it on the website |
| **Post conditions:**  The customer is on the website. |
| **Alternative flows:** |

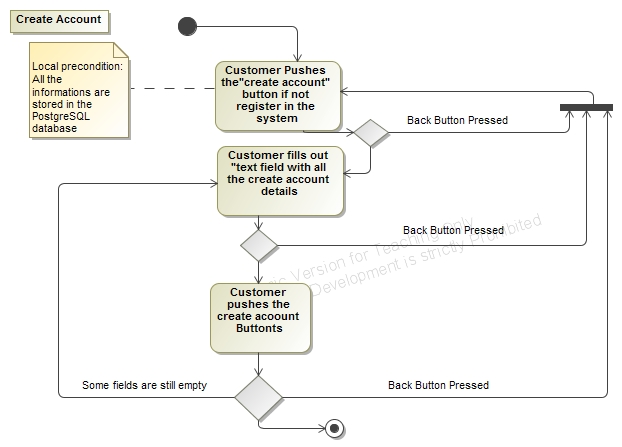
|  |
| --- |
| **UC8:** Viewing ticket on PDA |
| **Brief description:**  When a customer is in travel, the customer must be able to show his ticket to the conductor |
| .  **Actors:**  *Customer* |
| **Preconditions:**  The customer is on a bus or train. |
| **Main flow:**  1. This use case starts when the customer chooses to see his ticket. 2. BTC server sends a valid ticket for PDA 3. The ticket can be displayed on the PDA |
| **Post conditions:**  The customer is still on the bus or train. |
| **Alternative flows:** |

### 

)

### **State chart and activity diagrams**

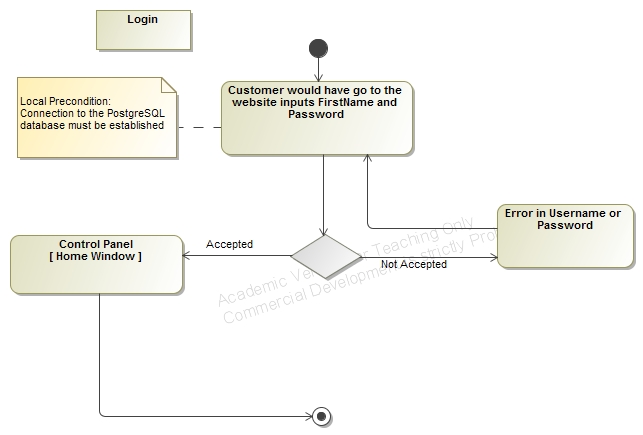
#### Create Account:

****

*Figure - Create Activity Diagram*

The diagram indicates that the Customer will create the account in the PostgreSQL database, according to the information provided by the customer. When all the fields has been filled, the Customer can press the “create” button, and the customer information will be stored in the PostgreSQL database.

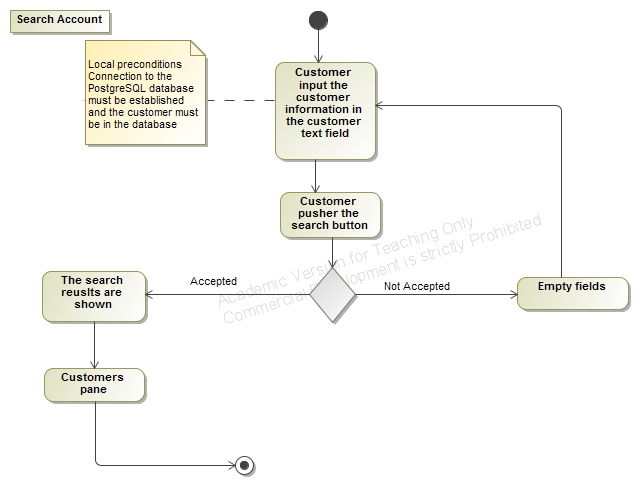
#### Customer Login:

****

*Figure - Login Activity Diagram*

As the diagram indicates, Customer can enter FirstName and password after he/she will press the “Login” button, which calls the PostgreSQL database. After a successful connection, the Customer will hereafter have access to the home window.

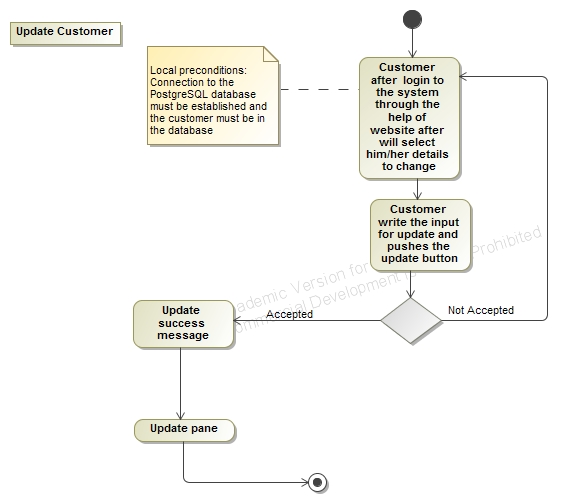
#### Search Account:

****

*Figure - Search Costumer Activity Diagram*

When a Customer wishes to search their account details. He/she will input the proper customer account details in the customer information text field, followed by a click on the “Search” button.

#### Update Customer:



*Figure - Update Costumer Activity Diagram*

When a Customer wishes to update his/her information. He/she will select and change the text fields needed to change, followed by a click on the “update” button.

### Solution classes/Interfaces

… (List and explain the solution classes and interfaces

# Testing

## Unit Tests

... (Describe the test strategies. Include and refer to detailed test results in the appendix)

# Conclusion

## Product oriented conclusion

... (Remember:

* List all ideas for future improvements
* List all known errors)

## Process oriented conclusion

# Literature

# Appendices

## Glossary

## Test results

### Unit Tests

…( Documentation data from Unit Tests)

## Source code Java

…( Use Courier New font and ensure the source code is properly formatted)