**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Implementation and Testing Report of Software System**

**For**

**Edinburgh city bike-hire scheme**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_**

Created by:

Demetra Charalambous s1452672

Kendeas Theofanous s1317642

**Table of Contents**

* Introduction………………………………………………………………………………..3
* Class diagram……………………………………...……………………………………..
* Target Tasks
* Use Cases Description…………………………………………………….4
* Classes Description…………………………………………………………5
* Functional Requirements Summary………………………………..
* Tests Description…………………………………………………………….
* Supplementary Tasks
* Use Cases Description…………………………………………………….4
* Classes Description…………………………………………………………5
* Functional Requirements Summary………………………………..
* Tests Description…………………………………………………………….
* Appendix for logging events………………………………………………………..

**Introduction**

Following the Requirements and Software Design Document for “ride-Edi”, the new self-serve bike-hire scheme in Edinburgh, this is the Implementation and Testing Report for the Software System. This includes the implementation decisions we made according to our design and the provided implementation, as well as the tests made to check the functionality of the software.

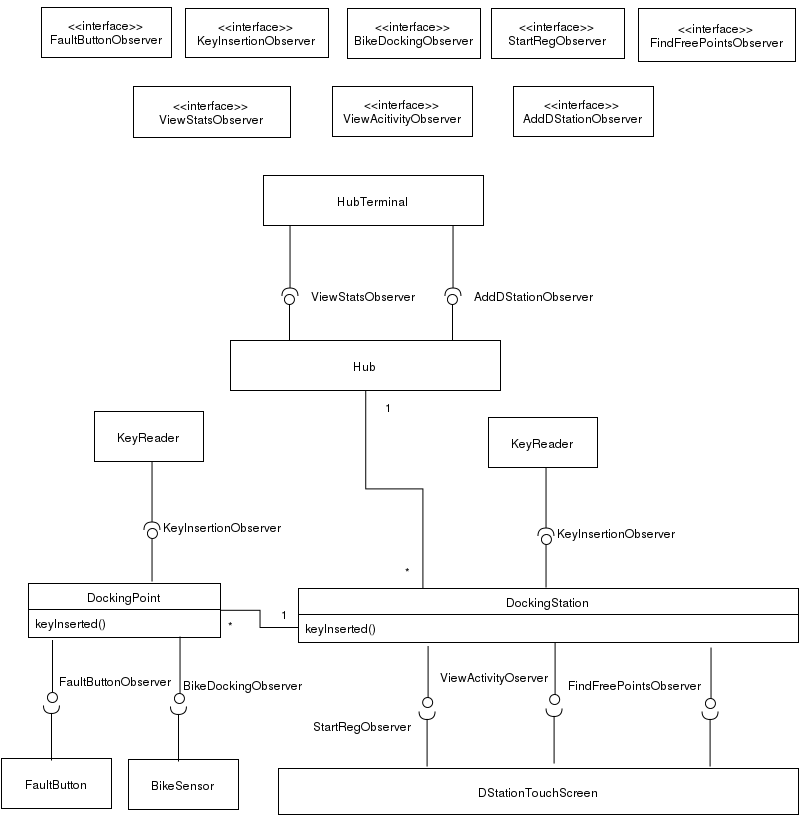
Overall description of the system:

A new self-serve bike-hire scheme is implemented for Edinburgh city, raising awareness about environmentally friendly methods of transport. The name of this new project is “ridE-Edi” and the logo was introduced in the Requirements Document.

The software system is distributed in three areas, the Hub System, the Docking stations and the Docking points.

* At the Hub System, the overall configuration of the system takes place. The Operations Hub is the centre of control and operation of the bike-hire scheme. There are staff members working there, ensuring the successful operation of the whole system, making necessary changes to its configurations and using the data collected to move bikes from/to docking stations.
* Docking stations located in different areas around the city will give citizens the opportunity to hire a bike for a short period of time. Each docking station is linked with the Hub System and equipped with a terminal. Bike users can interact with the terminal, which provides a number of different options to the users, depending on what they would like to do.
* Bikes will be available at the docking points at each docking station. Citizens must follow a specific procedure to hire/return a bike. Charges will be made to the user’s bankcards, depending on the amount of time that the bike has been hired.

In this document, target tasks and supplementary tasks are described separately with the class diagram below referring to both types of tasks.

**Class diagram  
  
  
  
  
  
  
  
  
  
  
**

**Target tasks**

Target Use Cases Description

1. Add new docking station

The operator at the hub is responsible to add the details of a new docking station to the system. This operation happens at the hub Terminal, using the provided input/output devices.

1. Add Bike

A staff member with unique ID “staff123” adds a bike at a free docking point. It might be a bike that was removed for bike service or a completely new bike.

1. Register User

New bike users can register to the system by entering their personal details on the touch screen in any docking point. They then insert their card so that the system can get their bankcard details and provided that the card is valid the user is added to the system.

1. Hire Bike

At every available docking point the user can hire a bike by using their key, as long as they are registered in the system and have not already hired a bike. After the user inserts their key, the system unlocks the bike and flashes an OK light to show that the bike was successfully hired.

1. Return bike

At every free docking point, users can return their bikes. The system recognizes the bike by its unique bike ID and if the return is successful it locks the bikes and flashes an OK light.

1. View User Activity

By using the touch screen at every docking terminal, the user can see the details of every trip they have made such as: start time, start and end location, as well as the charge for each trip.

1. View Occupancy

The occupancy of docking stations is shown on the hub display in regular intervals. High Occupancy is over 85% and Low below 15%.

Classes Description

Bike Hire scheme given to us includes most classes needed for the implementation of the scheme. In our design, we implemented a few more classes, namely User, Bike and Hire to be able to complete the target tags. Below, the classes created regarding the target tasks are described in detail.

**User Class**

* We implemented a User class having as private attributes the unique keyID of the user, their personal information and their banking details. We also created a list, storing each trip made by a user.
* It has getter methods necessary for usage by the hub system. A Boolean was added to confirm that a user has actually hired a bike and we are excluding the possibility of a user to hire more than one bike at a time.

**Bike Class**

* We implemented a Bike class having in its constructor the argument bikeID.
* We added some private attributes to handle the cases of hiring/returning, reporting faults and bike being added/removed.

**Hub Class**

* The Hub class is responsible for storing the users and bikes. Each of those is a HashMap where the key is their unique ID and the value is a User or Bike class accordingly.
* The Hub also includes a HashMap of the bike hires currently taking place. The purpose of this is to make hiring and returning bikes possible. Once hire takes place at a docking station, this action is added in the bike hires HashMap. (Use Case 4)
* When a user returns a bike, the bike is recognized as indeed hired and is then removed from the HashMap. (Use Case 5)
* A HashMap of docking stations in the Hub system is already provided to us and this makes sure all docking stations are in the hub system. Once an operator adds a new docking station on the Hub Terminal, our implementation adds this information in the docking station Hash Map. (Use Case 1)

**Docking Station Class**

* We modified the Docking Station class to also implement the View Activity Observer and the Bike Docking Observer, to handle the input events from the docking station touch screen and the bike sensor. We changed the constructor to include the hub as its first argument and set a Hub variable as a private attribute to make the connection with the Hub class.
* Once the touch screen triggers the start registration method, the start registration received is called by the Start Registration Observer and the Register User use case takes place, adding the user to the hub system. (Use Case 3)
* When the touch screen triggers the view Activity method, the view activity received method is called by the View Activity Observer and the View User Activity Use Case takes place, showing the user’s activity on the touch screen display. (Use Case 6)

**Docking Point Class**

* We changed the constructor to include a docking station variable, indicating the location of the docking point we are referring to.
* We implemented the functionality of a key insertion at a docking point. The key Inserted method takes as an argument a keyID. It checks, through the hub, whether the keyID belongs to a user or a staff member. If it belongs to a user, this indicates the docking point that a user wants to hire the docked bike, therefore the Hire Bike Use Case takes place, such that the bike is unlocked and the ok light is flashed. (Use Case 2)
* If the bike sensor detects a bike, the observer calls the bikeDocked method of the docking point. The docking point identifies the bikeID through the docking station and ultimately through the hub. If the bike was hired then, the ok light is flashed and the bike is locked and set as returned. (Use Case 4)

Functional Requirements Summary

1. There exists an **Operations Hub**, which is responsible for running the Hub System
   1. In the case where a new station is added, the operator at the hub is responsible to register the new station in the hub system.
   2. The status of the stations is shown by a large wall display

It shall include percentage of the occupancy of docking points at each station, focusing on whether it is under 15% or over 85%

1. Every **docking station** has a terminal
   1. A user, in order to be able to hire a bike, must first register at the terminal
      1. Users’ personal details must be entered using the touch-screen display
      2. The touch-screen display prompts the user to insert their bank card in the bank card reader
         1. System must ensure the card is valid
         2. System must ensure the correct PIN is entered
   2. The key issuer, then, issues a contactless electronic key to the user
   3. Details of every user trips can be requested at each terminal
      1. Details of each trip include the start time, start and end docking station, trip duration and charges
2. **Docking points** at every docking station
   1. A user can hire a bike at a docking point
      1. The contactless electronic key of the user must be inserted to the key-reader slot
      2. The key must be removed immediately by the user
      3. The docking point checks the keyID of the user through the hub system and the bike is unlocked
      4. A green OK light flashes at the docking point where the bike was unlocked
      5. The system does not allow the user to hire more than one bike at a time
   2. Every bike must be returned by the user at a docking point
      1. The user’s bike must be inserted to a free docking point
      2. The system at the docking point recognizes the ID of the bike, locks it and flashes an OK Light

Tests Description

Firstly, we created a setup configuration that will initialize the system, adding two new docking stations with free docking points. It will register two users in two different docking stations and their personal information is read and if the card is valid, the key issuer issues them a unique key, depending on the docking station they have registered in. We also added two bikes and set a return of a bike, which will help with the functionality of the system.

The tests we implemented are the following:

1) Hire and Return Test

A user will use their key card at a docking point key reader. The hub will identify the user with the given keyID and will unlock the bike and flash the OK Light. This will only happen provided that the user has not already hired a bike without returning it. Then, the same user wants to return the bike at a free docking point. The bike sensor will detect the bikeID of the bike, the hub will identify the bike and the docking point will send the instruction to the bike lock to lock the bike and the OK Light to flash.

2) Add Bike Test

A staff member will insert a bike at a free docking point. The bike sensor will detect the bike. Since the bike was not hired in the first place, the system assumes that it’s added to the docking point. The docking point lock will lock the bike and the OK light will be flashed.

3) Remove Bike Test

The key insertion of a staff key triggers this test. The hub identifies that it’s a staffID, so the Hire Bike Use Case will not take place but the Remove Bike Use Case. The docking point lock unlocks the bike and the OK Light flashes.

4) View User Activity Test

A user requests to view their activity through the day by selecting the appropriate option on the docking station touch screen. This will prompt the user to insert their key and is waiting for them to do so. It will then identify the user and will output their activity on the touch screen display.

5) Report Fault Test

If the bike is faulty, the user when returning a bike will press the fault button. The docking point checks whether the button was pressed in less than two minutes since the time of detection of the bike. The bike is locked and the OK Light is flashed. Since, the fault button was pressed the Fault Light also flashes.

6) View Occupancy Test

The occupancy is viewed on the Hub Display. This is triggered by the clock and it’s displayed in regular intervals.

7) Find Free Docking Points Test

A user, when all docking points at the docking station are full, has the option on the docking station touch screen to view a map of docking stations and the number of free docking points each one contains. They are also given a 15-minute extension on their last 30-minute period.

**Supplementary Tasks**

Supplementary Use Cases Description

1. Remove Bike

A staff member identifies themselves at a docking point by using their unique key and removes the bike, from that specific docking point. The system recognises that the bike has been removed from a staff member, unlocks the bike and flashes an OK Light.

1. Report fault

The same return process is followed as in the target Use Case return bike but here the user presses the faulty button. If the button was pressed within two minutes from the time that was locked by the system then the system reports it as faulty.

1. Find Free Points

The user can find available points in docking stations where they can return a bike. They can do this by identifying themselves using their key ID at the docking station and the system extends their return time by fifteen minutes. We assume that no user will be given an extension when there are available docking points.

1. Charges for hiring bike

Daily at midnight, users’ bankcards are charged for all their trips during the last 24 hours. After each trip, the system calculates the charge amount for each user and at the end of the day it sends these details to a bank server with the bank details (authorisation code) that each user has.

1. View Stats

Staff member requests a report from the system, giving information about how many users used the system today, how much were the total charges for the day and how many bikes were reported as faulty since the first day that the system was adopted.

Classes Description

For the supplementary tasks we also created Fault Light, Fault Button, Bank Server, Find Free Points Observer, Fault Button Observer and View Stats Observer.

**Fault Button Class**

* The Fault Button class we created implements the Abstract Input Devices.
* We also created a Fault Button Observer so that when the fault button is pressed, a method in the docking point is handled. Docking point checks through the clock whether button is pressed in less than 2 minutes that the bike was detected by the bike sensor of the docking point. If that is true, the fault button is activated and the fault light is flashed indicating the fault. (Use Case 6)

**Fault Light Class**

* The Fault Light class we created implements the Abstract Output Devices. It has the method flash and this happens when the Use Case Report Fault takes place. (Use Case 6)

**Bank Server Class**

* A Bank Server class is implemented and it has the method charge user, taking as arguments the bank details of a user and a charge. We left this method empty since this is a bank transaction. (Use Case 8)

**Hub Class**

* We added a private HashMap for daily charges of each user. We implemented a method calculate charge that takes as arguments a user and a trip duration. This takes into account the charge of the first 30 minutes of a trip, being £1, and the charge for the rest 30-minute periods, being £2 each. The possibility of an extension given is handled here so that the user is not charged for an extra 15-minute period. This method is called every time a bike is returned. All trip charges are added to the users in the HashMap. Every day at midnight, the charge user method of the Bank Server is called, and the Charge User Use Case takes place. (Use Case 8)
* The Hub class also has a private attribute total Faulty Occurrences, which indicates the total number of faulty bikes, encountered since the day the system started its operation. This number increases every time a bike is set to be faulty.
* An operator has the option to view statistics at the hub terminal. The hub implements the View Stats Observer to support this operation. The statistics we decided to implement are the number of journeys for each day, the number of users and the total faulty occurrences. (Use Case 7)
* We decided to add a private String StaffID at the hub and this will be the staff ID of the staff members. It will function like a master key, such that all staff will have the same key. Once at a docking point, the staff members will identify themselves and the Remove Bike Use Case will take place. (Use Case 1)

**Docking Station Class**

* In the case that a Docking Station is full, the user has the option to request a map of docking stations with free docking points. A user chooses this option on the docking station touch screen, triggering this Use Case. The hub method show free points is called by the Find Free Points Observer. It returns a list including the location of every docking station and the number of available bikes. This is shown on the docking station touch display. (Use Case 5)

**Docking Point Class**

* We modified the Docking Point class to also implement the Bike Docking Observer and the Fault Button Observer, to handle the input events from the bike sensor and the fault button.
* When the key reader of the docking point recognises the key card of a staff member, this indicates the docking point that a bike is being removed and the Remove Bike Use Case takes place, such that the bike is unlocked and the ok light is flashed. (Use Case 1)
* In the case that the bike detected by the bike sensor of the docking point was not hired, a chain of methods leads adding the bike from the docking point to the hub list of bikes. The ok light is flashed and the bike is locked. (Use Case 11)

Functional Requirements Summary

1) At the **Operations Hub**

a) Reports can be generated by operators when demanded. Reports include statistical analysis of the use of the system showing how many users have used the system on that day, how much were the total charges on that day and the number of faulty bikes reported until that day.

2) At the Terminal of each **Docking Station**

a) If all the docking points at the current station are occupied, the user scans their key and requests for a map showing available docking points. The system must ensure that the user is given a 15 minutes extension on the current 30 minute period.

3) At the **Docking Point**

a) There might be a faulty bike returned. The user must press the fault button at the docking point within 10 seconds of the bike being docked A red light flashes indicating that the bike is out of order.

b) Staff members are responsible for adding bikes at each docking station.

c) Staff member are also responsible for removing bikes from each docking station

4) Through the **Bank Server**

a) Charges apply to the user’s card when hiring a bike. Charges depend on the time period of the hire:

* + 1. £1 is charged for up to 30 minutes hire
    2. £2 extra is added for every further 30 minute period

At midnight the system charges users’ bankcards if the user has completed their trip in the past 24 hours.

Tests Description