**Project Report**

**Haolun Huang**

**Student Number : 213813340**

**CSE login: allen95**

**Abstract:**

**Toronto has 7500 single space parking meters, collecting fees from these meters one by one manually can be painful and time-consuming.**

**In order to make the whole process easier, I designed and implemented the GUI application Haolun’s Parking Helper.**

**This application employs MVC design pattern and provide multiple useful functions for Customers, Parking Enforcers and Admin.**

**With the help of Haolun’s Parking Helper:**

* **Paying manually at parking meters is no longer required, customers can pay easily with this app in the car.**
* **Checking parking space status with eyes is no longer required, parking enforcers can check the space’s status online with only a few clicks.**
* **Collecting fees from parking meters one by one manually is no longer required, all fees paid through the app will be sent to business account automatically. Admin can verify all payment status easily.**
* **Putting something on the space to mark it disabled is no longer required, parking enforcer can disable/enable the parking space online with a few clicks.**

**Outline:**

1. **Business rules assumptions**
2. **Database**
   1. **Database format**
   2. **Database logic**
      1. **Load**
      2. **Write**
3. **Test cases**
   1. **Test Customer Registration**
   2. **Test User Login**
   3. **Test Book a parking space**
   4. **Test payment**
   5. **Test change payment status**
   6. **Test cancel a parking space**
   7. **Test View Booking**
   8. **Test Manage Parking Enforcement Officer**
   9. **Test Manage Parking Spaces**
4. **Differences between initial design and current design**
5. **Business rule assumption**

* **Only Space 1 – 10 are open for testing as we are using csv file to mimic database. If all 7500 spots are open for testing, reload and rebuild database will take much more time.**
* **Customer cannot book a parking space in advance. Like most parking lot in Toronto, customers have to park their car first then use the app to book and pay. i.e. Each space can have either one booking or no booking.**
* **Customer can cancel a booking when it is not paid. Once the booking is paid, it can no longer get cancelled.**
* **Each customer can book up to 3 spaces.**
* **Parking enforcer can only search and view bookings, they can not make any changes to bookings.**
* **Enforcer accounts use their Unique as password.**
* **Admin account use email admin and password admin to login.**

1. **Database**
   1. **Database format**

**Database is the most important thing in my application, most functions need to load data from database or write data into database. Unfortunately, we can not use a real database at this time. Instead, we use 4 csv files to mimic a sql database.**

* 1. **Database Logic**

**There are four csv files which are Users.csv, ParkingSpaces.csv, Bookings.csv and Enforcers.csv. They behave just like sql tables, each of them have different columns and a key value to identify each row.**

**These csv files are manipulated in Database class only, no other classes have access to csv files directly. Class Database provide multiple function to load data from or write data into csv files.**

**To load from csv files, call ReloadUserDatabase(), ReloadSpaceDatabase(), ReloadEnforcerDatabase(), ReloadBookingDatabase(). These functions read data from csv files row by row and instantiate corresponding object with values of each row, e.g. read data from Users.csv, instantiate a User object once a whole row is loaded. After instantiating object, the function will put the object into HashMap with key (email is User objects’ key). So we can select each row from the Map just like we run select query in SQL database. To conclude: Reload functions are used to maintain objects and corresponding HashMaps.**

**To write data into csv file, call RebuildUserDatabase(), RebuildSpaceDatabase(), RebuildEnforcerDatabase(), RebuildBookingDatabase(). Since it is very difficult to change a specific data with Java IO, we rewrite the whole database every time object is changed. Rebuild functions are called each time a change is made to objects. E.g. when a customer makes a booking, this object’s bookingID field is changed, call Rebuild functions to rewrite the corresponding csv files to apply changes. To conclude: Rebuild functions are used to maintain Objects and Database so that they always have identical values.**

1. **Test Cases**

**Since there is no suitable library in Java libraries to test JavaFX GUI applications and the project requirements did not specify that we can use external library to test (the requirement only specifies we can use external library for UI).**

**Thus, I decide to use Manual Based Testing and Record and Replay techniques to test my application.**

**For each test, I will specify the steps and expected output. Then start the application, follow the given steps and compare the outputs/database with expected outputs/database. The whole process will be recorded and submitted with project for reviewing.**

* 1. **Test Customer Registration and User login (Testing them separately does not make any sense)**
     1. **Test1 - Steps**
* **Start the application**
* **Click button Create Account**
* **Click sign in again to test button**
* **Click sign up again and enter personal information**
* **Click sign up, then click sign in on the sign up success page**
* **Enter the info just entered, sign in as Admin/Enforcer/Customer to test authentication**
* **Once logged in, clicked log off, then register a different account**
* **Log in again as Admin/Enforcer/Customer with the credential used for registration to test authentication.**
* **Once logged in, test access to all pages available.**
* **Test add a booking, view booking and cancel**
* **Once done, log off. Close Application.**
  + 1. **Test1 - Expected Outputs Pass 59.0 Coverage**
* **Should not be able to login as enforcer or admin with the credentials**
* **Users.csv file should have the credential info used for registration**
  + 1. **Test2 – Steps**
* **Start the application**
* **Login with the both accounts registered in last test to confirm the accounts are still there after restarting application**
* **Log off**
* **Input a wrong email address, left password empty then login**
* **Input a wrong email address, input any password then login**
* **Input a correct email address, left password empty then login**
* **Input a correct email address, input wrong password then login.**
* **Finally input the email address of first account and password of second account, then login Register a third account with different password**
* **Repeat above steps for admin and enforcer**
* **Login with any account, test access to all available pages.** 
  + 1. **Test2 – Expected Outputs Passed 72% Coverage**
* **Should still be able to login with both accounts after restarting**
* **Should not be able to login in all steps which does not enter matching credentials**
  1. **Test Book a parking space, view booking and cancel booking (Again testing them separately does not make any sense)**
     1. **Test1 – Steps / Expected Passed 72.1%**
* **Register a new account**
* **Login with new account and go to Book a space**
* **Enter lot number 0, choose any hour and Search / Cannot search**
* **Enter lot number 11, choose any hour and Search / Cannot search**
* **Enter lot number 1, choose any hour and Search then book /Book successfully, refresh database the booking should be there**
* **Enter lot number 1 again, choose any hour and search then book /Cannot book**
* **Go to my booking /the booking just made should be there, all information should match**
* **Logoff and login as enforcer with test ID and login as admin, go to manage booking and search the book /the booking should be there**
* **Logoff, login as the customer which booked the space, go to my booking /the booking should show**
* **Cancel the booking, go to book a space and search for lot number 1 /the space should be available now, refresh database, all booking info should be gone.**
* **Go to my booking again / the booking should disappear**
* **Logoff and login as enforcer, search for the booking /booking does not exist**
* **Logoff and login as any customer, book two random spaces. Book one of them for 0.5 hour and one of them for 2 hours.**
* **Logoff and close application**
  + 1. **Test2 – Steps / Expected Passed 72.2% coverage**
* **Change System time to an hour later**
* **Log in as customer who makes 2 bookings in last test**
* **Go to my booking. / No bookings should show because we moved time an hour forward. They are all expired.**
* **Go to book a space. Make two more bookings.**
* **Go to my bookings. / There should be 3 bookings**
* **Go to book a space. Try to book a space which is not occupied. / Should not pass as there is already 3 bookings for this customer object**
* **Log off, log in as enforcer and admin, search for the 3 bookings. / Should be able to retrieve information for all 3 booking**
* **Log off, log in as customer and cancel 1 bookings / Refresh database, the 1 booking should disappear**
* **Go to book a space, book another space /book should pass as 1 booking has been cancelled**
* **Cancel all 3 bookings and logoff / Refresh database, all bookings should be gone**
  1. **Test payment and change payment status**
     1. **Test 1 – Steps / Expected Pass 77% coverage**
* **Register a new account, log in and book 3 spaces.**
* **Go to my booking to pay, enter cardholder name which is different from the name of this account. / Pay should not pass**
* **Enter correct credentials and Pay. Log off / Pay should pass**
* **Go back and check my booking / The status should be Processing**
* **Log off and log in as admin**
* **Go to manage payment**
* **Search a booking ID that does not exist. / Search will fail**
* **Search for the booking ID just paid. / booking will show**
* **Click Verify.**
* **Search for the same booking ID. Verify again. / will show paid and can’t verify again**
* **Logoff and log in as the customer. Go to my bookings. / The status should be Paid now**
* **Logoff and log in as enforcer. Go to manage bookings and search for the booking. / The status should be Paid**
  + 1. **Test 2 – Steps / Expected Pass 91.1% Coverage**
* **Register and Log in as a new customer and book 3 spaces**
* **Pay all spaces.**
* **Logoff and log in as admin, verify all 3 payments.**
* **Logoff and log in as a new enforcer, search all 3 bookings. / all should be paid**
* **Logoff and log in as the customer, go to my booking. / all should be Paid**
* **Remove the previous add enforcer**
  1. **Test Manage Parking Enforcer and manage parking spaces**
     1. **Test1 – Steps / Expected Pass 90.7% Coverage**
* **Login as Admin, go to Add an enforcer**
* **Enter information and add the enforcer / Refresh database, enforcer should be there**
* **Logoff, log in as the enforcer just created. / Able to log in**
* **Test access to all available pages. / have access**
* **Logoff, register a new customer and login**
* **Add a booking, logoff and login as the enforcer to search booking / booking should show**
* **Logoff and login as customer, pay the booking. Logoff and login as admin, verify the booking,**
* **Logoff and login as enforcer, search the booking and check the status. / Should be paid.**
* **Login as admin and remove this enforcer /should pass**
  + 1. **Test2 – Steps /Expected Pass 94.5% coverage**
* **Register a new customer account and login**
* **Book a space and pay**
* **Login as Enforcer, search for this space**
* **Try to disable it / will not pass since the space is in use**
* **Login as admin, verify the payment**
* **Login as Enforcer, search for the space**
* **Try to disable it / will not pass**
* **Search for another space and disable it / will pass**
* **Login as customer and try to book this page / will not pass as it is disabled**
* **Login as enforcer and enable the space**
* **Login as customer and book this space /will pass as it is enabled**
* **Login in as admin and remove this enforcer with wrong ID / should not pass**
* **Remove this enforcer with correct ID / should pass**

1. **Differences between initial design and current design**

**I did not strictly follow my initial design(factory design pattern, since we actually implementing the GUI in our project). I employ a different design pattern (MVC combine Event-driven Pattern) in my report.**

**When I was coding the project, I can tell factory design pattern is not a good choice for GUI application because it’s typically a Model-View-Controller design process. So I decided to use MVC design pattern which turned out to be a good choice later.**

**With MVC design pattern employed, the whole structure of the project is very clear. All model classes (User, Database…) are all put in package model. All UI fxml files and their controllers are put in package View. But if factory design pattern is employed, all files are likely to be put in the same package which will be a mess.**

**The developing process also becomes more enjoyable. I simple follow the MVC design pattern. Code Model classes first, then set up View (fxml) file, and finally develop their controllers. If factory design pattern is used, I could end up developing all classes at the same time, writing methods where needed. It could be a disaster.**

**To conclude, MVC design pattern is definitely the best pattern to use when developing GUI application.**