Express Parking: Software Design Final Project

Nourin Abd El Hadi

216107021

EECS3311

April 24th, 2021

Contents

[Design 3](#_Toc70185543)

[Test Cases 3](#_Toc70185544)

[Assumptions 6](#_Toc70185545)

[Class Diagram Comparison 6](#_Toc70185546)

# Design

When I started creating this project, I used the class diagram I created during the midterm as a template on how to design the program. I quickly realized that I heavily underestimated how many variables and methods I would need to run the program. Though the base is similar, it was definitely very different because I was able to visualize the GUI of the program and all the components needed to run smoothly. The use of JavaFX and SceneBuilder made it a lot easier to create only necessary methods as I could see what the user would see easily. I also realized that I did not create a customer object, instead I relied on my csv files to read and write customer information to be shared with my classes. During signup and login of the user, their email, first name, and last name are stored as global variables to help authenticate bookings, cancellations, etc. Based on the way I created this project; I used the state design pattern where most of the pages are derived from the LandingPageController class. There are 4 states that derive from that class, and that is where the bulk of what the user can do occurs.

# Test Cases

After doing some research on Junit testing with JavaFx, I realized that any method that I had to switch to new page required the parameter “ActionEvent event”, which I unfortunately did not know how to deal with. Because of that I am unable to create test cases in Java, but I did test my program manually using the below inputs for every section.

**4.1**:

REQ-1: I inputted 10 different officers with different emails and generated an ID and then compared the column with their IDs using a text comparator and it passed.

REQ-2: Once an email is entered by the system administrator and the email already exists in the file, the officer is then removed (If [test@mail.com](mailto:test@mail.com) is entered, the user will be deleted). If the admin enters an email that does not exist, a warning message pops up (if [tst@mail.com](mailto:tst@mail.com) is entered, an error message shows) and the user can try to enter another email.

REQ-3: If the ID that is generated is equal to one already in the file, then a new number is generated and checked.

REQ-4: All information is stored to officers.csv (comma separated)

**4.2:**

REQ-1: Three “John Smith” entries were inputted with different emails (can be found in signups.csv) with no issue.

REQ-2: If two identical emails are inputted, the system throws an error, and the user can enter a new email or go back to sign in.

REQ-3: All user information is stored in sigups.csv

**4.3:**

REQ-1: If an email does not exist in signups.csv, the system throws a warning to sign up (if [lola@mail.com](mailto:lola@mail.com) is entered, a warning is thrown).

REQ-2: If an incorrect username or password combination are entered then an error is thrown (if [mail@mail.com](mailto:mail@mail.com) with password123 is entered, an error is thrown)

**4.4:**

REQ-1: Users can never reach the “Book a Parking Space” page without signing in.

REQ-2, REQ-4: If there is a null entry, the system presents the user with a warning to fill in all the spaces.

REQ-3: The file “parkingSpaces.csv” is read and compared with the entry to make sure it does not exist already as taken (inputting “44444” and “66666” will throw an error).

REQ-5: A customer can input up to 3 spaces to reserve, but if more than 3 spaces are associated with their email then an error is thrown (if the user email@mail.com tries to book another space, an error will occur, but the user ail@mail.com can book 1 more space).

REQ-6: A random generated 5-digit number is stored in parkingSpaces.csv and is also outputted to the user for them to keep.

REQ-7: The same as REQ-3.

**4.5:**

REQ-1: The user cannot reach the cancellation page without being signed in.

REQ-2: No null entries are accepted; a warning will appear.

REQ-3: If the time is below 0 the cancellation will not go through.

**4.6:**

REQ-1: The user cannot reach the Payment page without being signed in.

REQ-2: No null values will be accepted; a warning will occur.

REQ-3: There are three payment types; PayPal, debit, and credit. These options appear after a correct input is detected (for example [mail@mail.com](mailto:mail@mail.com) needs to enter 12345 and 77777 for the payment options to appear or else an error is thrown).

REQ-4: A time is added.

REQ-5: Countdown.

REQ-6: The user can enter 1-3 parking spaces and pay for them at the same time (for example [mail@mail.com](mailto:mail@mail.com) can enter 12345 and 77777 and pay for both).

REQ-7: Payment information is confirmed, and an alert is shown.

**4.7:**

REQ-1: Information is viewed using the customers login information, so a person that is logged in cannot see the view booking page. They are also limited to only seeing their own bookings.

REQ-2: Information is shown in table format.

REQ-3: A label will be shown in the column “Expiry Status”

**4.8:**

REQ-1: The manage parking page cannot be reach unless an officer has logged in.

REQ-2: An error is thrown if a parking spot that does not exist is attempted to be removed.

REQ-3: There are more than one parking space in the parkingSpaces.csv file.

REQ-4: Same as REQ-2.

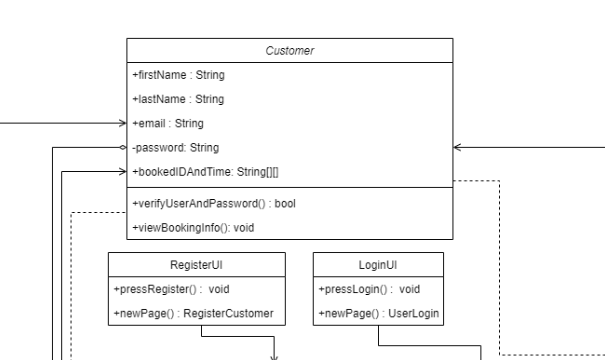
**4.9:**

# Assumptions

I made few assumptions while creating this project. I assumed that the password for the officers was their unique ID as there was no instruction about entering a password when the officer was being signed up. I also assumed that the System Administrator was automatically signed in and can access this information from their end. I also assumed the price for parking is $0.05 per minute and that the user enters the time they wish to park in minutes.

# Class Diagram Comparison

In the class diagram from my midterm, I wanted to create a customer object that is surrounded by all the features the customer can use.



But in my new class diagram, I based the features around the landing page and made it a hub.

