EECS3311- Project Report

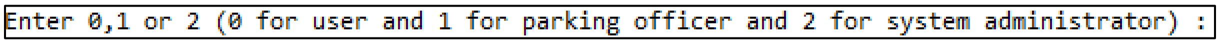
Rag Pratheek madhu

215567274

The implementation of express parking has strictly followed the design built in the midterm with some changes to the initial design. The major change from my initial design
is removing **EnformentOfficer** class and having the **AdminGUI** class directly be associated
with **ParkingAuthority** class. This change happened because in the GUI of the program, all the attributes of **EnformentOfficer** class are included and it is directly associated with **ParkingAuthority** class. This makes **EnformentOfficer** class somewhat a redundant class, hence removed. Another change that was made is the links between the classes, during the implementation, there was no inheritance used, instead association or direct association links were used throughout the design. The reason for these changes in the links between classes is purely based on the implementation preference but not necessarily the problem in the design. Inheritance was complicated whereas the association was easier to implement. The information between classes is passed through an array list containing **Users** class as its elements. This array list is manipulated based on the class the program is in and writes to the database ("user.csv") after the manipulation. This is the way inheritance is changed to association or direct association. During the implementation, there are more classes than indicated in the class diagram because the GUI needing a new window to perform certain actions. In order to create a new window, a new class is created just to perform some of the tasks. This is why there are a lot more classes than indicated in the class diagram. However, the final idea is the same with some minor changes because of the GUI.

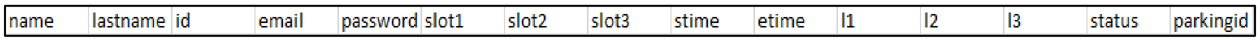
Step-by-step process of the program

The program starts with the input to the **console** asking for 0,1 or 2 as each integer refers to something (as shown below). Input zero means the program is entering as the customer, input one means the program is entering as the parking officer and two means system administrator.



```
Enter 0,1 or 2 (0 for user and 1 for parking officer and 2 for system administrator) :
```

Any of these options chosen will interact with the database (read or write). The database is called user.csv where it has 15 columns separated with sections like name, last name, email, and more as shown below.



name	lastname	id	email	password	slot1	slot2	slot3	stime	etime	l1	l2	l3	status	parkingid

Each column will be read or write based on the class and required action.

If 0 is pressed in the console:

A new window will pop up asking the user to register or sign in as shown in *figure 1* below.

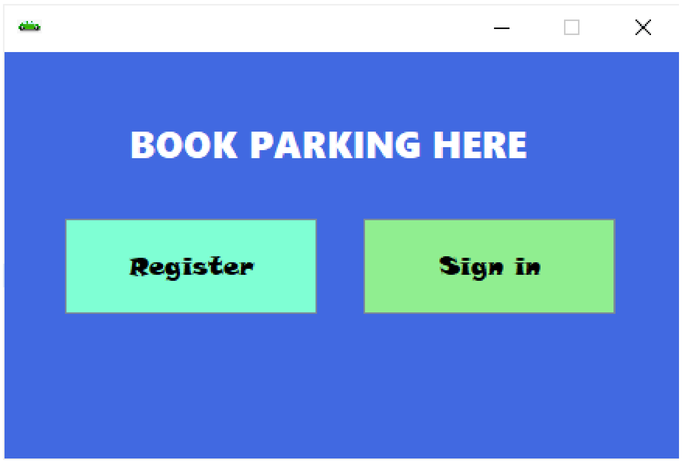
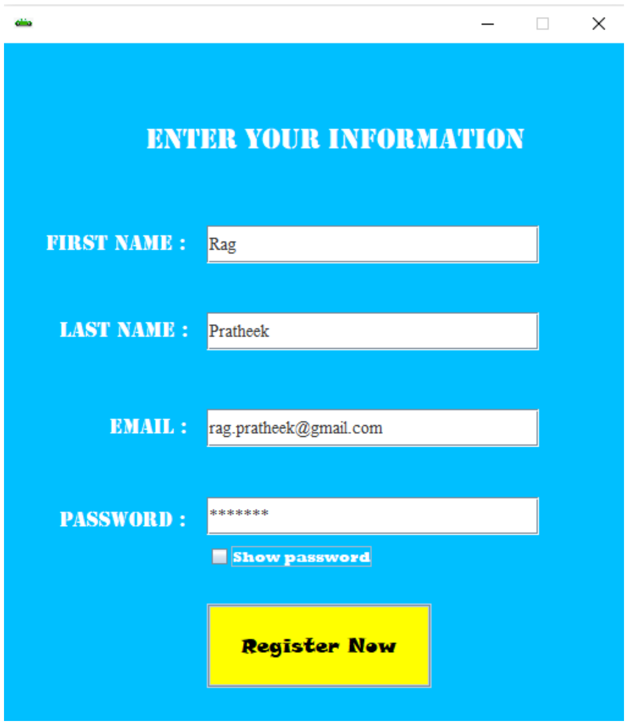


Figure 1

Two buttons will take the user to two different windows, if the register is pressed the program will ask the user for *first name, last name, and email (figure 2).*



ENTER YOUR INFORMATION

FIRST NAME : Rag

LAST NAME : Pratheek

EMAIL : rag.pratheek@gmail.com

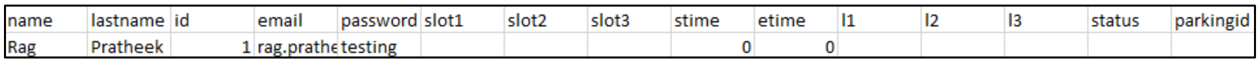
PASSWORD : *******

Show password

Figure 2

After pressing "Register Now" the program will check if the user entered the correct information, especially checks if the user entered a unique email (already exists in the database). If the user did not enter a unique email error message will pop up telling the user to retry. This is requirement **4.2** in the text.

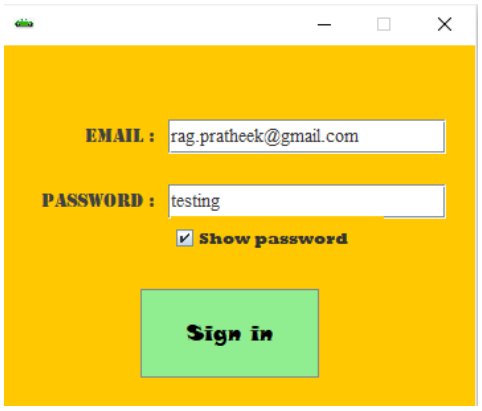
If all information is correct the user will be registered writing the information to the database with a unique id assigned to it. Soon after pressing the "Register now" the program will return back to *figure 1*.



name	lastname	id	email	password	slot1	slot2	slot3	stime	etime	l1	l2	l3	status	parkingid
Rag	Pratheek	1	rag.prathe	testing				0	0					

Information in the database

Now Users can register again with a different email or sign in this time. If the user pressed "Sign in " button, they will be prompted to enter login information such as email and password


EMAIL : rag.pratheek@gmail.com

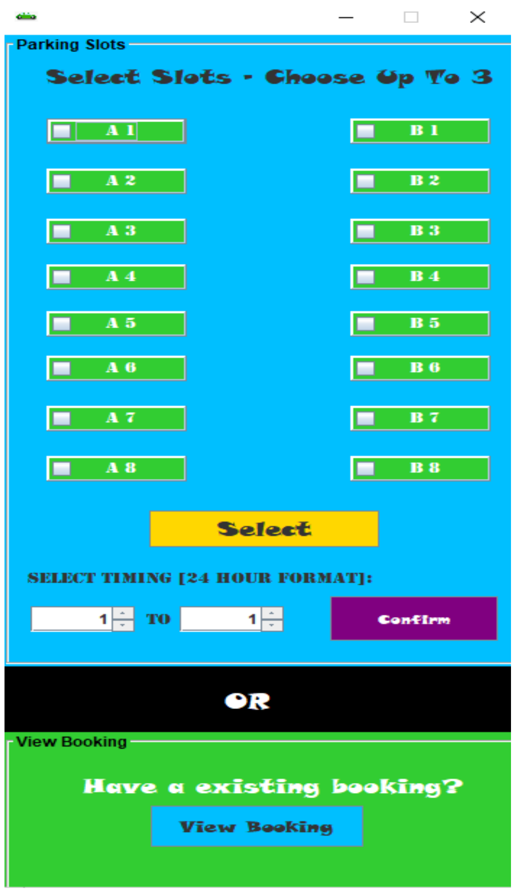
PASSWORD : testing

☑ Show password

Figure 3

If the email and password do not match in the database, the program will display an error message and tell the user to retry. If the user enters their correct information they will be granted access to the application. This is requirement **4.3** in the text.

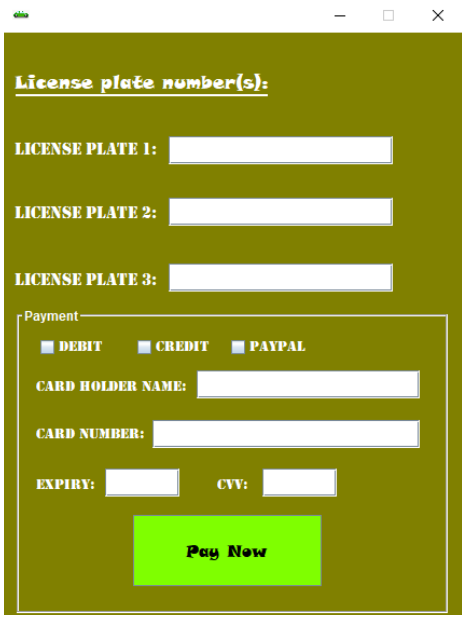
When the user presses "Sign in" (assuming correct information entered), they will now see a new window to book parking slots or **view their booking** (if any).



The user can pick the slots from A1 to A8 and B1 to B8. If the user picked more than 3 slots and pressed "Select", the program will display an error message that will pop up telling users to select 3 or fewer slots. If the user selects a slot that is taken, an error message will pop up telling them to retry and pick a different slot. When users select 3 or fewer slots they have to "Select" to confirm the slots. After that user must select timing in 24-hour format and click "confirm" to book the slots. This is requirement **4.4** in the text.

Figure 4

After clicking “Confirm”, new window will appear asking for license plate (s) and payment information.This is requirement **4.6** in the text.


License plate number(s):

LICENSE PLATE 1:

LICENSE PLATE 2:

LICENSE PLATE 3:

Payment

DEBIT CREDIT PAYPAL

CARD HOLDER NAME:

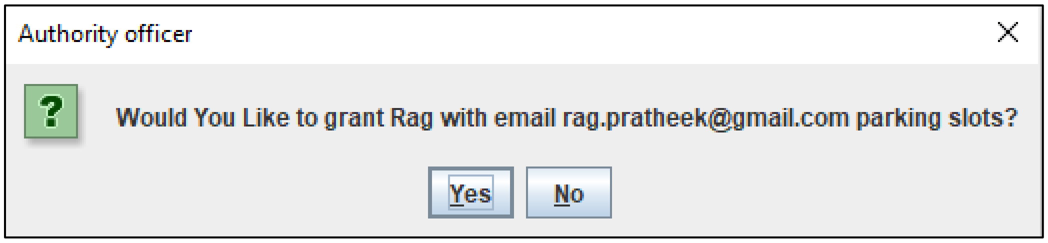
CARD NUMBER:

EXPIRY: CVV:

Pay Now

Figure 5

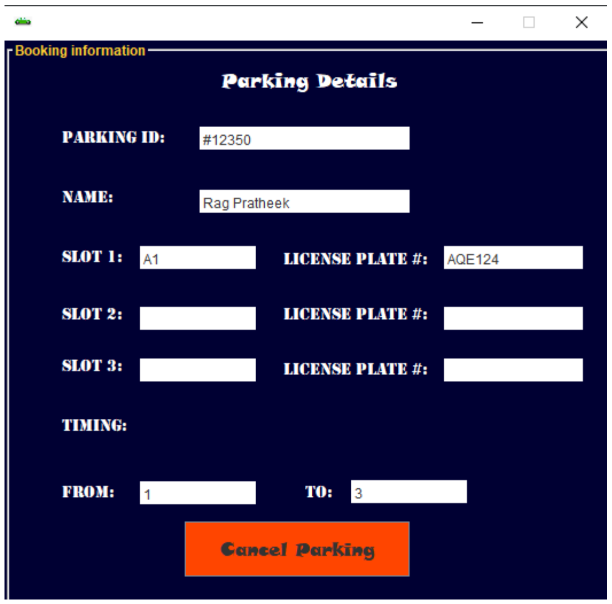
After pressing “Pay now” with all the information filled in, a new message will appear which is intended for Authority /parking officer. The message shown below is assumed to be displayed to Authority Officer/parking officer, **not** the user.


Authority officer

Would You Like to grant Rag with email rag.pratheek@gmail.com parking slots?

Yes No

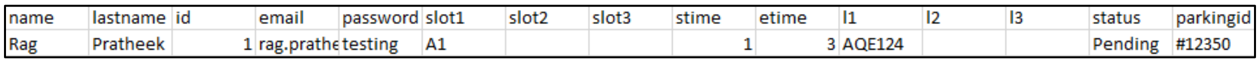
If the Authority officer/parking officer presses “No” request is denied (requirement **4.8 cancel request** feature) and the program returns to the booking slots window. If the Authority officer/parking officer presses “yes” the request is granted (requirement **4.8 grant request** feature) and booking information will be displayed. This is requirement **4.7** in the text.



PARKING ID:	#12350
NAME:	Rag Pratheek
SLOT 1:	A1
LICENSE PLATE #:	AQE124
SLOT 2:	
LICENSE PLATE #:	
SLOT 3:	
LICENSE PLATE #:	
TIMING:	
FROM:	1
TO:	3

Figure 6

This is the booking's final information based on the entered data (with generated parking ID). This is also a page that will appear when pressed the "view booking" button in *figure 4*. If the user wishes to cancel the booking then the user can press the "Cancel parking" button. This is requirement **4.5** in the text. If they press the "Cancel parking" button, the program will return back to *figure 4* (the book slots page).



name	lastname	id	email	password	slot1	slot2	slot3	stime	etime	l1	l2	l3	status	parkingid
Rag	Pratheek	1	rag.prathe	testing	A1			1	3	AQE124			Pending	#12350

This is the final information that is written in the database. Notice that status is 'Pending' that is because the system administrator did not approve the payment yet. In order to do that, we need to press 2 in the console to access the program as administrator. Another important note is that slot2, slot3, l1 and l3 (License # 2,3) are empty because the user did not fill in or did not book, hence empty.

<u>If 1 is pressed in the console</u>:

System will enter as Authority Officer/parking officer where he/she has the ability to add or remove parking slots.

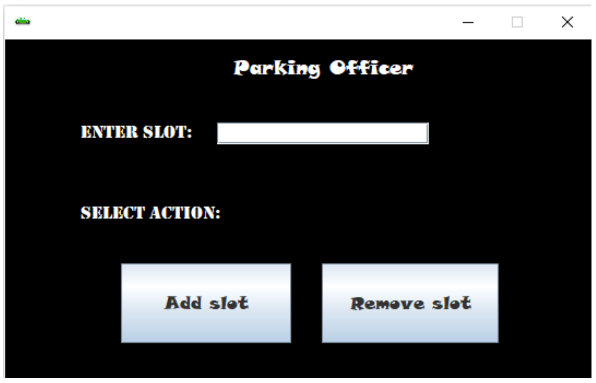
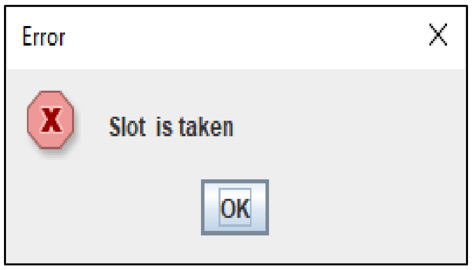


Figure 7

***NOTE:** If any of the GUI looks like this (without attributes), please Minimize it couple of times to see the atttributes (like Figure 6)*

Parking Officer enters the slot number and decides to remove or add slots back to the customer. If "Add slot" is selected then the slot will be added with the rest of the slots and it would not say the slot is occupied by the user. If the parking officer decided to remove the slot, the user will no longer be able to add the specified parking slot. If the slot is occupied then the parking officer is unable to remove the slot (an error message will be displayed).


Error

Slot is taken

OK

This is the what the program says when a user tries to select a removed slot. There are are 16 slots for users and parking officer can decided which slot to remove and which slot to add for the user to use. This is requirement **4.8** in the text.

If 2 **is pressed in the console**:

The program enters as system administrator where it allows them to change payment and add or remove parking officer.

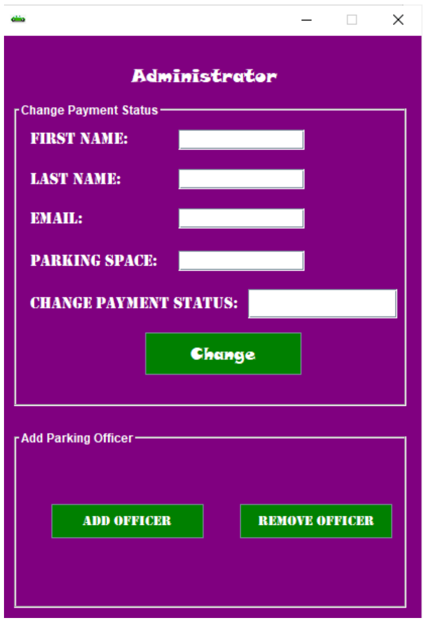
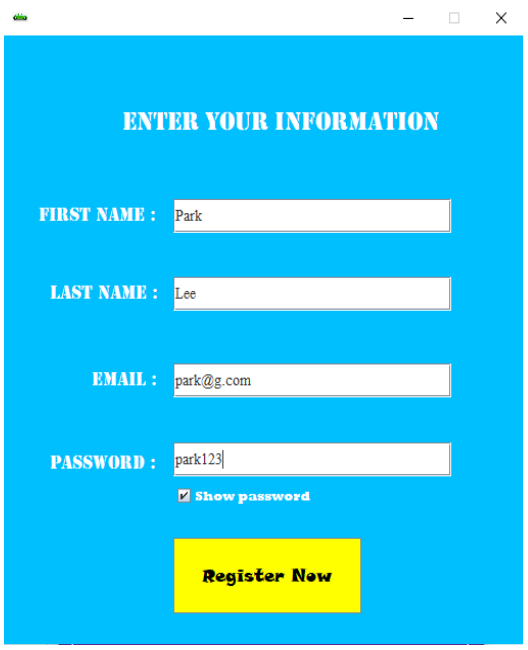


Figure 8

If the administrator wants to change the user's payment status, all they have to do is fill in their information (error message will occur if details are invalid), and press "Change" and the payment status will change to whatever the administrator wrote in the "Change payment status" text field. This is requirement **4.9** in the text.

Administrator can also add an officer by pressing the "Add officer" button. This will lead the program to a new window asking for *first name, last name*, *email*, *and password* and will register the officer by pressing the" register now" button as shown below (*figure 9*). After pressing the register now, the program will go back to the window shown in *figure 8*.



ENTER YOUR INFORMATION

FIRST NAME : Park

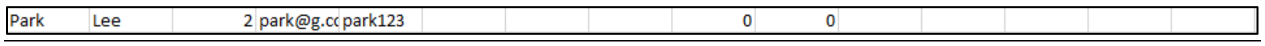
LAST NAME : Lee

EMAIL : park@g.com

PASSWORD : park123

☑ Show password

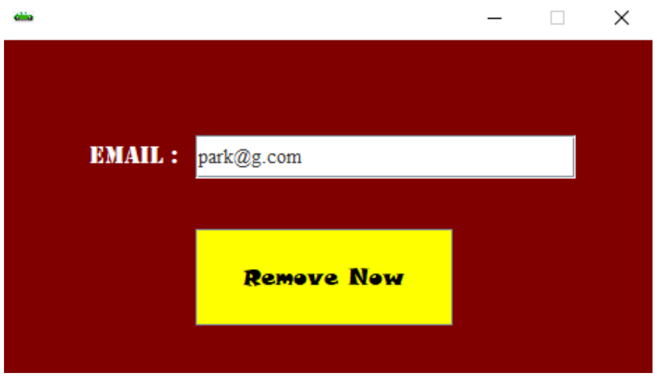
Figure 9



Park	Lee	2	park@g.cc	park123				0	0					

This is the information of the parking officer written to the database.

If the "remove officer" button is pressed, a new window will appear asking for an email. The program will check if the officer exist.If they exists, the program will remove the officer from the database and return to the administrator menus (*figure 8*). This is requirement **4.1** in the text.

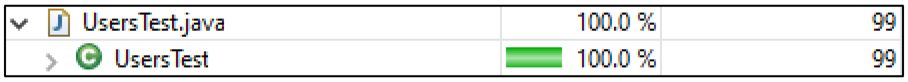


EMAIL : park@g.com

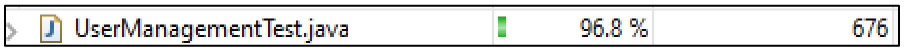
Figure 10

Testing:

This program consists of only two non-GUI classes, UserTest.java and UserManagement.java. The test coverages of these classes are shown below.



UsersTest.java	100.0 %	99
UsersTest	100.0 %	99



UserManagementTest.java	96.8 %	676

Conclusion:

The design pattern used in this program is a state design pattern where each class is a state. The transition between classes occurs when the user clicks a button (i.e sign-in or register). The transition of classes is dictated by input from the users. Another design pattern that was used is observer, the program is always monitoring user input and decides whether to write or read to the database. The observer was useful especially considering the number of times the program reads and writes to the database. The final design pattern that was used in this project is the factory design pattern. The project starts with giving options and based on the input to the console, the program will behave accordingly.