350 project

Khalil baydoun

Jad Sarkis

Salah el dine Mneimneh

OUTLINE

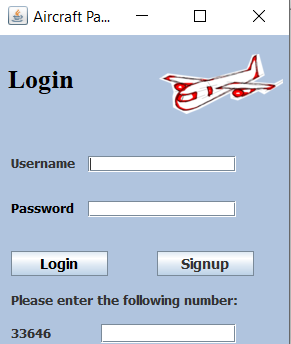
1. GUI.
2. Database.
3. Server and client communication.
4. multithreading.
5. functions and classes.
6. Difficulties faced.

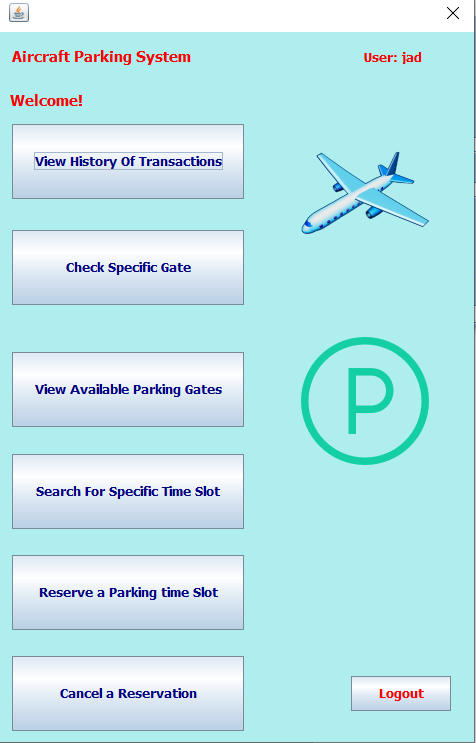
**GUI:**

We built the Graphical user interface using window builder which is the drag and drop GUI building application as a plug in in eclipse. the images in the GUI are presented as references in the code and present in the resource file in order for them to be present in the project.

Each functionality such as the Reserver or the Gate Searcher has its own GUI.

as an example this is our login page GUI for the app where the client can either log in or sign up and each one of those buttons will take us to another GUI upon success.



and here you see the main page GUI on the top right corner

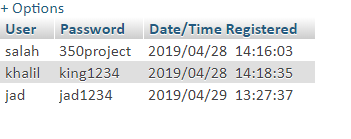
we print the username (to be explained later in Multithreading)

and each of the buttons is linked to a function which also directs

us to other GUIs.

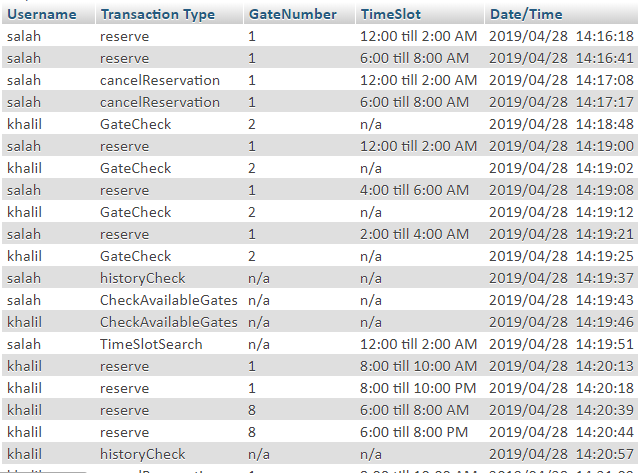
**Database:**

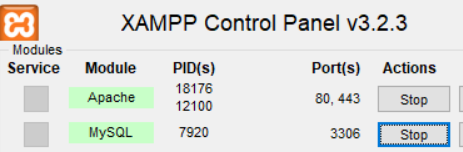
The database and tables needed were built using phpMyAdmin and mysql. We needed for this project three tables, one to contain the users as well as the date they signed up and their respective passwords called users



the second table called gates is used to store the Gates and their time slots to indicate if they are free or booked and if booked it shows what username booked that time slot.



and finally the third table called transactions to keep track of the history of each transaction or request done by the clients.

this database is done on the local host and

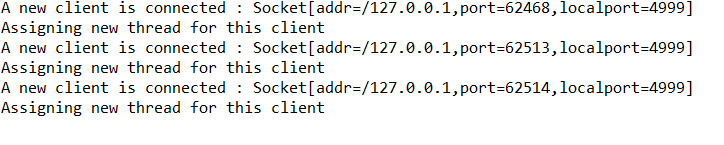
we open its server using XAMPP which opens the

apache server that this database runs on.

**Server and client communication:**

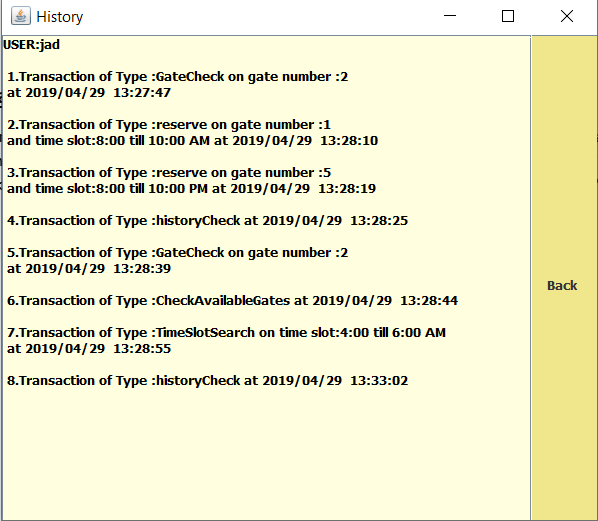
in our project the client sends commands to our server and our server opens a connection to the database server. All requests that the client does are in the classes of the GUIs upon each request the server reads the text sent by the client along with the needed information and then it executes the request by communicating with the database server either to query a table for a certain value(such as if a gate is “free” or “booked” ) or inserts information into the table. this communication is established using mysql JDBC drivers. The server opens a socket to each client as a means of establishing the connection.

**MultiThreading:**

this functionality allows us to open multiple clients and be connected to the same server but with different port numbers. This allows the server to be able to manage multiple requests from different clients without interruption. For example, in the image below we see multiple clients connecting to the server each being assigned a different port.

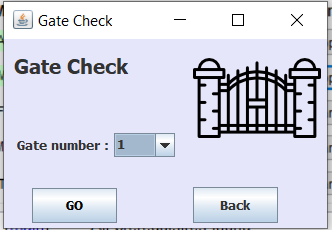
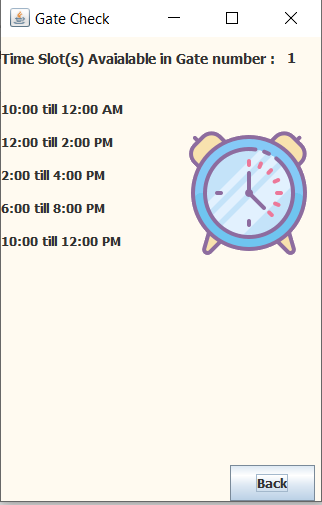
**Functions and Classes:**

In functions and classes, we worked on writing code to build GUIs for each button that was available to us in the main page. Those buttons are “View History of Transaction”, “Check Specific Gate”, “View Available Parking Gates”, “Search for Specific Time Slots”, “Reserve a Parking Time Slot”, and “Cancel a Reservation”. Each time we press one of those buttons, we will be directed to another GUI.

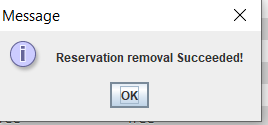
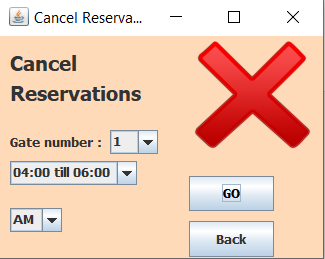


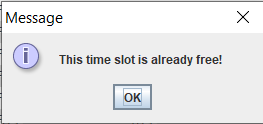
For example, the picture above shows the GUI we are being directed to once we click on “View History of Transaction”. It shows us the previous actions the user “jad” has done including time reservation and cancelation. The “back” button will direct us back to the main page.

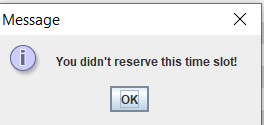
Moreover, if we click on “Check Specific Gate”, we will be directed to a GUI where we are asked to pick the gate that we want to see available. Once we have chosen the gate, the program will direct us to another GUI that will give us the times available. The pictures below are captures of the GUIs just described.

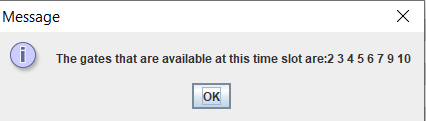
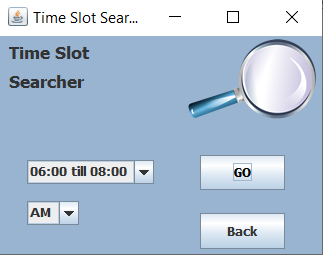


Furthermore, clicking on the “Cancel Reservation” button will take us to a GUI. In it we will have to chose the gate and the times. If we pick a gate and time that we haven’t reserved and that another user has reserved, a GUI will pop up with the message “You didn’t reserve the Time Slot!”. If we pick a time and gate that is free or that no one has reserved, the pop up message will say: “This Time slot is already Free!”. But if we chose a gate and time previously reserve, the program will cancel the reservation and show the message, “Reservation Removal Succeeded!”.

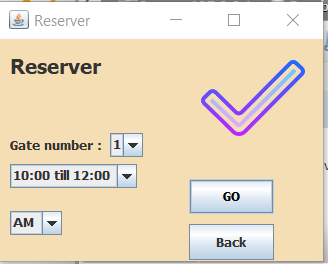
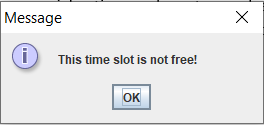
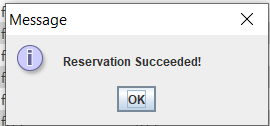






If we press on “Search for a Specific Time Slot”, the GUI that will pop up will ask for a time. Once we press “Go”, a message where we are given the available gates at the specified time will pop up.

Chosing the option “Reserve a Parking time Slot” will direct us to a “reserver” window where we need to pick a time and a gate number. If the chosen time and gate number is free, the reservation will become successful and it will be saved in the database. If the chosen time and gate number is not free, a message will pop up stating that the place is already taken.



Finally, the button “View Available parking Gates” will direct us to a Message box that will state the available gates in increasing order.



**Difficulties faced:**

During the building of our project, we faced a considerable number of difficulties. To begin with, we found it particularly difficult to link the windows together. Meaning for each time we clicked a button, telling the program to go a specific GUI was challenging. Moreover, it was required for each client to only use one socket for the communication in order to maintain a persistent connection. Making sure that it was only one socket and not more was a challenging task. Lastly, the history window gave us a hard time. Indeed, when the previous actions a user has made was too large, the info written on the window exceeded the page capacity. This forced us to add a scroll up/down on the screen to view all the available histories of the user.