Report

In addition to your presentation, you must submit a project report. There will be two parts.

Part One

Part One is a minimum of 1000 words and requires the following:

- A minimum of 500 words describing your project and the functionality you implemented.
- A minimum of 500 words documenting your design choices while implementing the project. Justify the project structure.

Our project is a cinema booking system. It starts with Auditorium, an object that stores a 2d array of seats and seatprices, a showing name, a date, and a showing time. In the 2d array of seats, a seat is either empty or contains the name of the user that occupies it.

BasicReservation is an object of a reservation. It has the user making it, the movie, the date, the number of people, and the row and column to be reserved.

A user is stored as an object BasicUser. A user contains a username, a password (with secure hashing), a boolean to check if they are an admin, a type (Regular, VIP, Premium), a credit card, a list of their reservations, a list of their transaction history, and a ReentrantLock for security.

Users are managed via UserAccountManager and its methods. This is in order to easily handle user management outside of the database.

ReservationDatabase is the database that stores the events and reservations. The database stores reservations, where each reservation has a movie name, a showing time, an auditorium size, and the availability of each seat in the auditorium. The database also stores users, where each user has a username, type (regular, premium, VIP), a boolean deciding if they are an admin or not, their credit card, their number of reservations, their transaction amount, and their password (uses secure hashing). The database can be viewed from running ViewDatabase.

A client interacts with this through ClientDriverGUI, where they are first asked to login/register for an account. They can then view movies and movie times, and then see the available seats. These seats are updated in real time. Whenever a new seat is taken in that showing, the GUI refreshes to show the updated seating availability. They are then prompted for their card information for payment to buy the seat. Additionally, a user flagged as an admin is able to create new venues with custom seating, add/edit/cancel showings, view all bookings, schedule showings months in advance, and view the database.

The client gets information from the server, Server. The server loads information from the database and sends it back to the client. We chose to exchange information using different object payloads. The object payloads contain different information, and the server behaves differently depending on what object is received. Each action has a different payload associated with it.

Changes to the database from the client are managed via the object ClientService and its methods. The methods send and manage payloads.

The interfaces are to easily show all the elements of the object that implements them. Note that interfaces are not used on multiple objects, rather correspond to a single object.

The tests make sure to check for edge cases. Not every class has a test because many are very basic, and are tested in other classes' tests that use the class as an object.

Part Two

Each team member must write a minimum of 350 words on the following:

- A minimum of 200 words describing your contributions to the project.
- A minimum of 150 words on what you would do differently, if anything, if you were given the opportunity to start over again. If you would not do anything differently, explain why.

- Each team member's section should be labelled with that individual's name.

Logan Dalton

For the project, I contributed the entirety of Auditorium.java, AuditoriumInterface.java, AuditoriumLocalTest.java. I created the non network I/O aspects of ClientDriver.java. I wrote the ReadMe for phases 1 and 2, and I wrote many of the ReadMe class descriptions. I applied many bug fixes across a variety of different classes including the AdminPayloads, ReservationDatabase, ClientService, BasicClient, BasicUser, BasicReservation, CancelPayload, ReservationDatabaseTest, ReservationTest, ReservationPayload. I worked on visual aspects of the slideshow presentation. I created part one of the report. I made sure to attend every group meeting, on time, and interacted with team members to solve problems, to optimize workflow, and to make sure everyone had something to do. I made sure to take ownership of specific tasks, following through with reliable contributions. I made sure to keep communication flowing by asking group members for updates and what they still need to work on, and also asked others what they needed from me. This kept the group on track, and helped get work done efficiently. I helped team members resolve merge conflicts, and avoided conflicts by pulling often and pushing changes directly through github. I reviewed other members' work to find possible bugs and see if they functioned properly.

Some things I would do differently is to encourage team members to communicate what they were doing and what they needed more. I would have liked for us to have given each other more clear directions on what we needed others to do. I would have also liked to have scheduled more team meetings, as this would have combatted these issues significantly. A way to accomplish all of this would be to have a team member behave similarly to a project manager, obtaining and delivering the necessary communication and arranging meetings. Something else I would have liked is for team members to not have pushed directly from their IDE. This caused issues with the build, making it so the project no longer worked in Intellij and needed to be in the same package to function in VSCode (as well as several other bugs in VSCode). Furthermore, this created many strange merge conflicts that needed to be solved. The solution to this would have been to push from the terminal using git, or directly from github.