Project Description

For our project, we created Book-A-Bus! This app is a bus ticket scheduler designed to assist in booking trips between different cities. The system allows users to insert the information of where and when they want to travel, view all available trips, choose their desired trip, and proceed with the booking process. Users are also given the ability to log in to track their past and upcoming trips. The backend uses MySQL for data storage, while the frontend is implemented using Java and JavaFX, using Scene Builder for the UI design.

The project began with designing the database schema in MySQL. Our team created several tables to store essential data for the trip reservation system, including Passengers, Buses, Reservations, and Departures. The Passengers table stores each traveler's name and email to maintain user records when booking a trip. The Buses table includes the bus ID (bid) and capacity, allowing us to track each bus and the number of available seats. The Reservations table contains the reservation ID (rid), passenger name and email, departure ID (did), and reservation date, helping us manage customer bookings and their associated trip details. Lastly, the Departures table stores key trip information such as did, bid, capacity, date, time, duration, origin (fromCity), destination (toCity), seats taken, and price. This table acts as the central hub for tracking buses, passengers, and travel schedules. The database schema is designed to support efficient queries and updates, enabling the system to retrieve available trips based on user inputs and process new reservations seamlessly.

After setting up the database, the next challenge was establishing a connection between the IntelliJ IDEA and MySQL. After our lesson on JDBC, we eventually learned to use the MySQL Connector/J, a JDBC driver, which facilitates communication between Java and MySQL. Using this driver, we were able to write Java code that connects to the database, executes SQL queries, and processes the results. In order to make the connection more manageable, we implemented a utility class called SQLConnection to handle the connection logic. This class allows other parts of the application to easily access the database without needing to repeatedly write connection code. Utilizing this driver, we were able to write Java code that connects to the database, executes SQL queries, and processes the results.

The user interface (UI) of Book-A-Bus was built using JavaFX. We used Scene Builder to design the layout of the application, which provides a visual interface for placing UI components such as labels, buttons, text fields, etc... This tool allowed us to rapidly prototype the UI, ensuring that the design was both functional and user-friendly. We designed a multitude of pages. Our app begins at a login page. Our login page links the user-entered data (name and email) to a pre-existing passenger. If they do not exist in our system, they will be directed to create an account. When they click the sign-up button on the login page, they will be guided to a separate page to create a new account, allowing us to insert their information into our Passengers table and create a new passenger. From the login/create account pages, the user is then brought to the selection page, allowing them to choose where they want to travel to/from and the date they are looking for. The dates with available trips are highlighted in green. After confirming their selections, the user is brought to the main page. The main page displays available trips in a TableView, where users can select a trip by double-clicking on a row. The table columns include information such as the departure city, destination city, date, time, available seats, and price. Once the user selects a trip, they are taken to a booking page where they can review their trip details and book their trip! This saves their reservation to our Reservations table and brings the user to the confirmation page. Here, the user receives a personalized “You’re All Set” message and their reservation number to confirm their trip was booked. They are also given the option to see all their trips. On their view trips page, the user can toggle between past and upcoming trips associated with their account. Users also have the ability to go to the previous pages and log out.

A significant part of the development was writing SQL queries to interact with the database. One of the queries we implemented was to load available trips. For this query, the system retrieves trips from the Departures table based on user-selected criteria (from city, to city, and date). The query also ensures that only trips with available seats are shown. Another query we implemented was for inserting into the Reservations table, in which a user's reservation details were inserted once they clicked the “Confirm Booking” button on the booking page. Alongside this query, we used an UPDATE statement to change the number of available seats in our Departures table, ensuring our app displays the most up-to-date data when the user is looking for trips. These queries are executed using PreparedStatement in Java, ensuring that user inputs are safely included and preventing SQL injection.