Design diagrams for the MSD App and Comment in the Discussion Forum

Driving School APP

By

Owen Jones

Student No: C21377551

Mobile S/W Design

Diagrams of the App

1. Purpose of the app

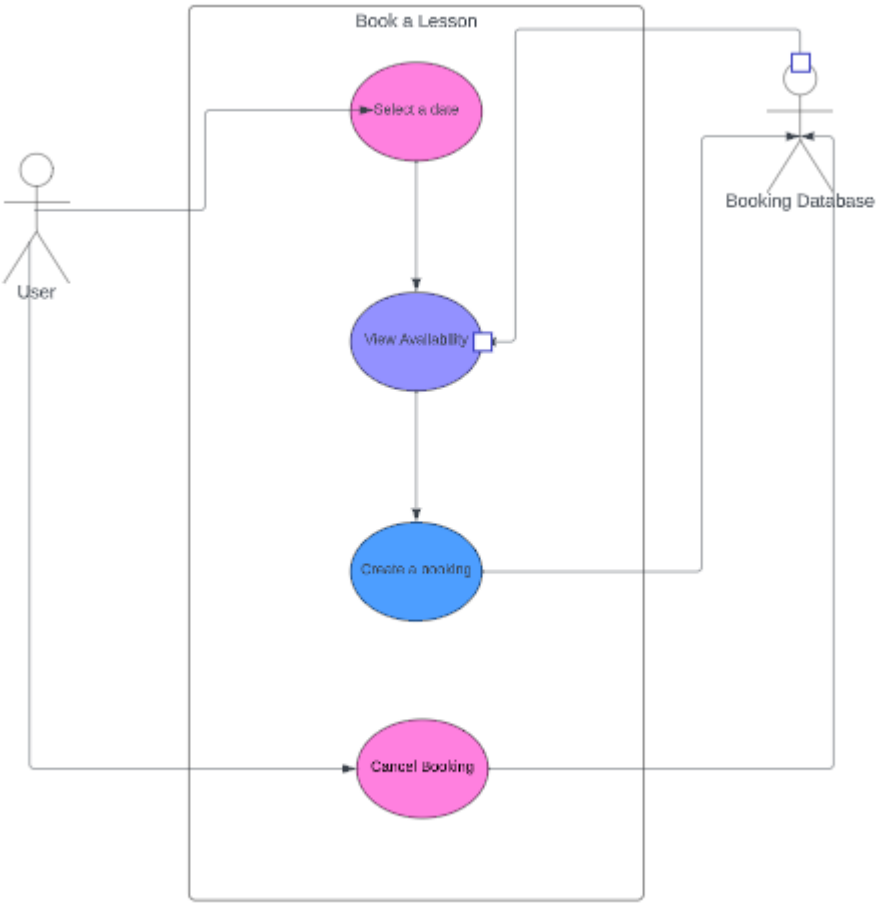
The Driving School App simplifies lesson management with a user-friendly interface. Users log in to seamlessly book, modify, or cancel lessons, securely saving data to a database. A clear list displays current booking slots, and the map feature identifies the nearest instructor based on the user's location. Instructors and their cars are listed for easy selection, fostering personalized choices. Users enjoy flexibility with the ability to add or remove bookings as needed. The app ensures a smooth experience with its clean and interactive interface, facilitating efficient communication with both instructors and the map.

2. Use Case Diagrams

The primary actor in the application is the user. The application relies on a database to store information regarding current bookings, which in turn dictates the available options for the user when initiating a booking.

Upon opening the app, users are presented with a choice of actions. They can either promptly create a booking or opt to peruse the instructor map, which displays the geographical locations of available instructors.

When a user decides to create a new booking, the application performs a database check to ensure there are no duplicate entries for lessons and to prevent overbooking scenarios.

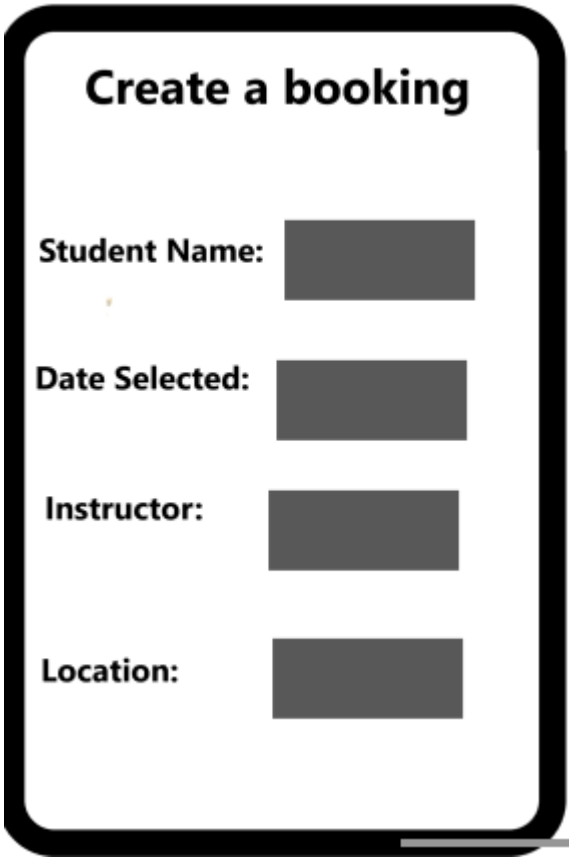


3. Screens Flow

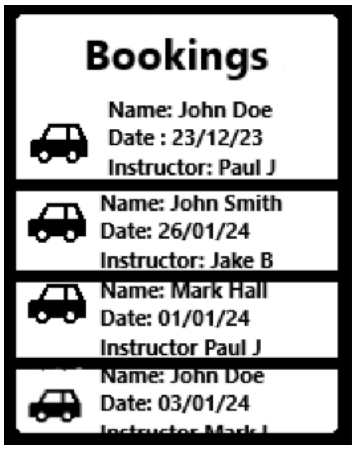
Home Page:

This home page allows you to press 3 options. One to take you to the instructor details page, one to view the map of the instructors based on locations and another to create a booking for a lesson.

Create a booking:
This page allows the user to create a booking and insert their details which will be saved to the database.



View Bookings:
This is a list of the bookings which uses the details saved in the database and displays them to the user.



Map of instructors:

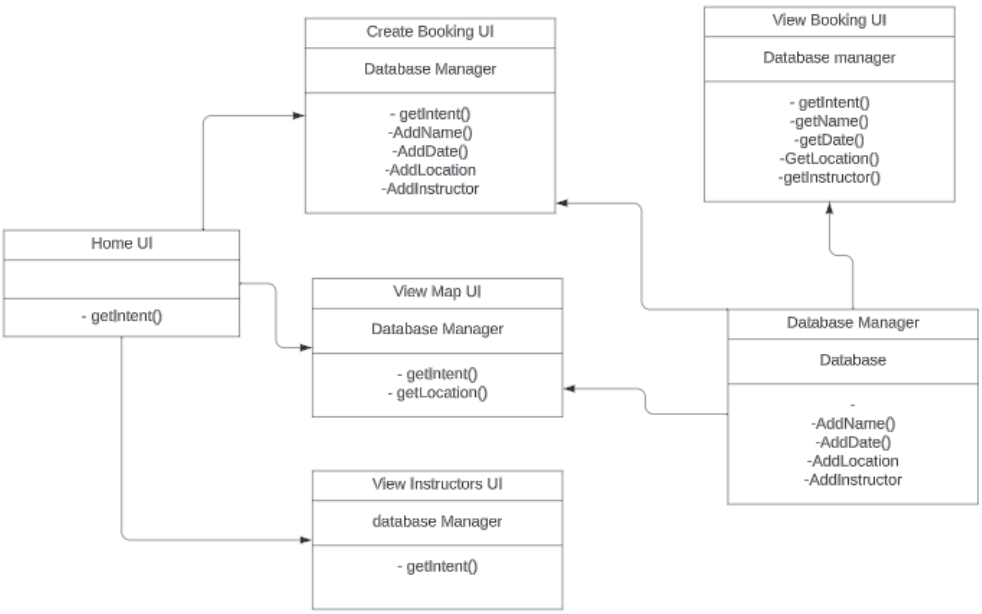
This is the map page which shows the locations of each instructor on the map allowing the user to decide which instructor may be the best suited to them.

Instructor Information:

This page contains images and relevant information about each driving instructor.

4. Class diagrams

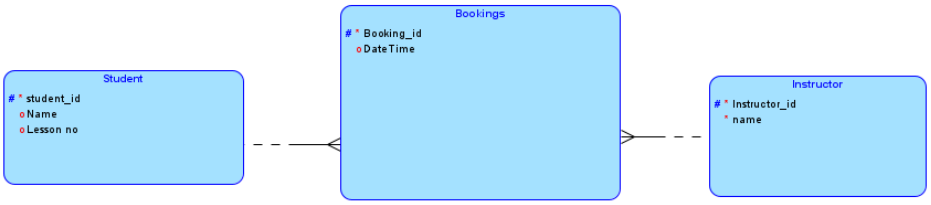
The diagram displays the classes and methods required for my app. I'll use getIntent() to switch between screens and navigate back. The database manager will handle all methods, pulling information from the database.



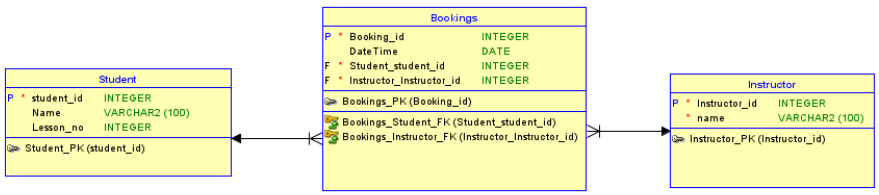
5. Database design

The provided Logical Entity-Relationship Diagram (ERD) illustrates the structure of the booking database. It comprises three key tables: one for student information, another for instructor details, and a bookings table that comprehensively stores data related to scheduled lessons.

The relational links are established between the bookings table and both the instructor and student tables. Foreign keys are employed to solidify these relationships. Each booking is uniquely identified by a dedicated booking ID, ensuring that each reservation is distinctly represented in the system.



The next diagram shows the relational model of the bookings database. This shows in more detail the relationships between each table and the details of each attribute.



Documentation:
https://brightspace.tudublin.ie/d2l/le/286387/discussions/threads/54022/View

Lists

Learning about lists in Android development is captivating because they form the backbone of dynamic and scrollable user interfaces, profoundly impacting the overall user experience. Lists act as architects, orchestrating organized and interactive presentations, providing a structured way to present information. This functionality not only ensures seamless content navigation but also significantly elevates user engagement.

Mastering the nuances of lists empowers developers to craft applications adept at efficiently managing diverse datasets. Picture effortlessly scrolling through contacts, dynamic updates in a social media feed, or a news app presenting articles in a user-friendly format—all made possible by the versatile nature of lists.

In essence, lists transcend mere technicalities; they represent a fundamental building block for creating user-friendly and engaging mobile applications. Their simplicity and adaptability underscore their significance in Android development. Developers can transform static interfaces into vibrant, dynamic, and user-centric experiences with lists. For those eager to delve deeper, resources like the Android Developer Guide and CodePath Android Guides offer valuable insights into mastering the art of lists. This not only enhances app functionality but ensures users find the interface intuitive, making their interaction both efficient and enjoyable.