Booking.com

A Travel Booking Solution

Introduction and Summary of Project

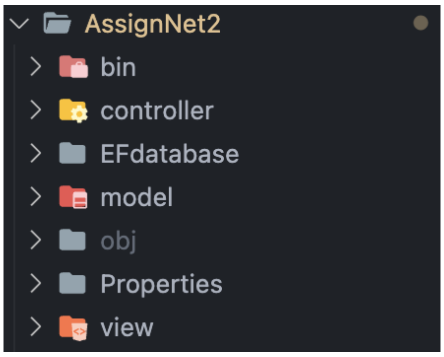
In the digital age, the way people plan and organise travel has changed dramatically, pivoting from physical agencies to online platforms. This project, "Booking.com," is an innovative application designed to simplify the often-tedious process of planning a trip. The application serves as a comprehensive travel assistant, allowing users to book hotels and airfare without the need to navigate through multiple websites and platforms. The primary objective is to provide a user-friendly, efficient, and all-encompassing platform that serves travellers' key needs during trip planning and booking.

Development Approach

We adopted the Agile methodology for this project, owing to its iterative approach and the necessity for flexibility in development, especially when integrating various external APIs for flight and hotel data.

Technical Stack

Front-end

- The GUI was designed with a focus on usability, ensuring all forms, navigation, and information display followed best practices for user experience design.
- **Windows Forms:** For the front-end, our choice was the tried-and-true Windows Forms due to its reliability and our team's familiarity with its operation.
- **Model-View-Controller (MVC):** To keep our UI clean and manageable, we adopted the MVC architectural pattern.
- This separation of concerns allowed us to isolate the business logic from the UI, making the codebase easier to navigate, test, and debug, thereby improving scalability and maintainability.
- As shown in the image below, we separated the different components of our application into their respective folders - model, view and controller
- 

```
AssignNet2
  bin
  controller
  EFdatabase
  model
  obj
  Properties
  view
```

Back-end

- We employed C# for backend logic, utilising the .NET framework's robust library for various functionalities.
- We used external libraries such as Newtonsoft.Json to parse and write JSON to and from files. We also used this library to parse the JSON response retrieved from the application's interactions with external APIs.

Database and ORM

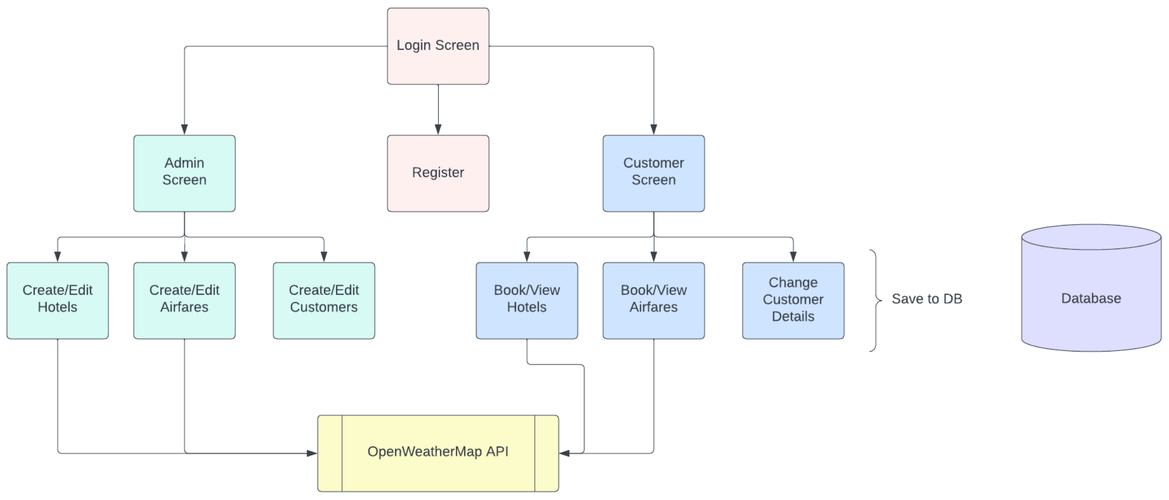
- Entity Framework was used for database interactions, allowing for more straightforward and more secure data handling.

External APIs

- We integrated a weather forecast API to show users a 5-day weather forecast in their chosen destination via hotel or air.
- The API functionality captures the destination city of the location, and dynamically retrieves weather information whenever the user checks the details of a specific hotel or airfare location.
- The weather API we used:
  Weather API - OpenWeatherMap

Flowcharts

Flowchart 1: Process Flow



Flowchart 1 provides a structured representation of the functionalities and interactions within the "Booking.com" travel application.

1. Login Screen: The primary gateway to the application. It segregates user roles, directing them to appropriate sections based on their credentials.
2. Register: An avenue for new users to sign up for the application.

3. Admin Screen: Accessed by administrators post-login, this section empowers them with various management tools:
- Create/Edit Hotels: Enables the admin to add new hotel listings or edit existing ones.
- Create/Edit Airfares: Allows the admin to add or modify flight details.
- Create/Edit Customers: Admins can manage customer profiles, for example when users request manual assistance or to resolve discrepancies.

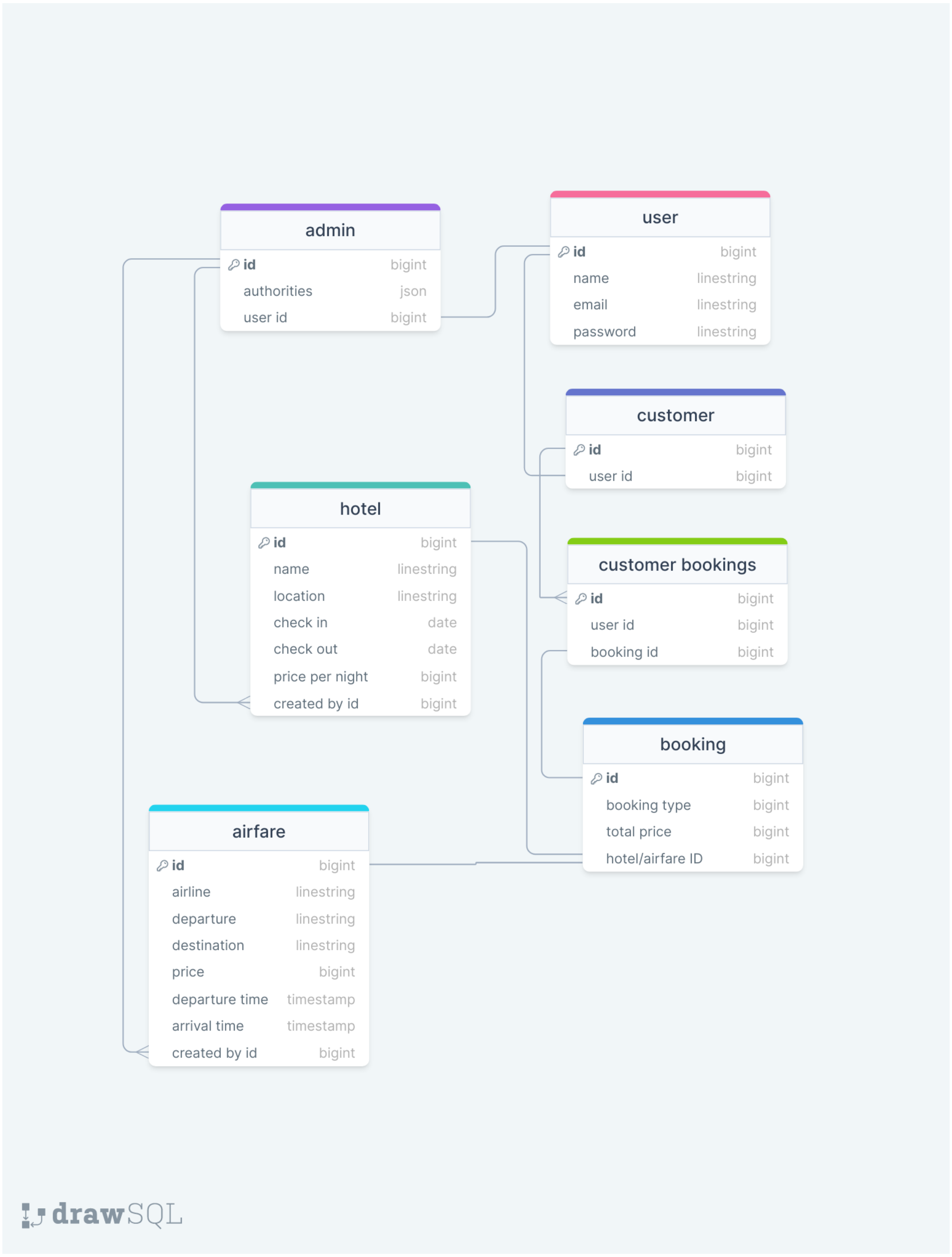
4. Customer Screen: After a customer logs in, they're presented with this interface which offers several functionalities:
- Book/View Hotels: Customers can browse available hotels, check details, and make bookings.
- Book/View Airfares: Similar to hotels, customers can view available flights, get pertinent details, and book tickets.
- Change Customer Details: Enables users to update or modify their personal information.

5. OpenWeatherMap API: When viewing a hotel or airfare with a location that actually exists, the application retrieves a 5-day weather forecast for the specified location.

6. Database: This symbolises the storage component of the application. Actions such as "Create/Edit" or "Book/View" will invariably interact with this database. The "Save to DB" arrow indicates that any changes or additions made in the application are stored in this central database for retrieval and modification.

In summary:
- The flowchart encapsulates the different user roles, their respective interfaces, and actions, along with illustrating how data flows and is stored within the system.
- The application is designed to be user-centric, ensuring ease of navigation and efficiency in tasks, whether it's an admin managing listings or a customer planning their next journey.

Flowchart 2: ERD



Flowchart 2 is a simple ERD representing the relationship between different entities within the Booking.com application.

Entities

1. **user**: General users who register on the platform.
2. **admin**: Special users with elevated privileges, linked to the user entity.

3. **customer**: Individuals using the service, associated with a user.
4. **hotel**: Represents accommodations available for booking.
5. **airfare**: Denotes flight details and pricing for travel.
6. **booking**: A record of either hotel or airfare reservations made by customers.
7. **customer bookings**: A linking table that associates customers with their respective bookings.

Relationships:

1. Admins are directly related to users.
2. Customers are linked to users and have multiple bookings through the customer bookings table.
3. Bookings can be associated with either a hotel or airfare.
4. The customer bookings table links customers and their respective bookings.

Role of Team Members

Our team strategically divided the work to play to each member's strengths, ensuring efficient progress and high-quality output.

Mostafa

- Worked on the Airfare portion of the project, integrated external API & wrote the report.
- Github Repo creation, keeping files organised, gitignore
- Documentation

Features:

- 1. Airfare. Book/View/Cancel Airfare as Customer
- 2. Airfare. Create/View/Edit View Airfare as Admin
- 3. Weather API: Display weather forecast for Hotel And Airfare

Leonna

- Worked on the majority of the codebase and project setup.
- Windows Forms app creation
- User registration and login
- Hotel. Book/Cancel/View Hotel as Customer
- Hotel. Create/Edit Hotel/View as Admin
- Create and manage (edit) customer as admin or customer
- Delete customer as admin
- LINQ & external database (via Entity Framework) setup
- Saving objects to file & reading from file

Combined

- Code review
- Manual testing

Acknowledgments

- We would like to express our gratitude to Dr. Avinash Singh for his insightful lectures and our tutor Jack Donaldson for his support and interactive teaching methods.
- We also give credit to the authors of the OpenWeatherMap API used, which were integral to our application's functionality.
- Additionally, we appreciate the valuable resources provided by our institution (UTS), contributing significantly to this project's success.