Software Engineering and Testing. BSC Year 2, 2020/2021 (Assignment 3 - 20%)

Assessment 3: Design and Draft Implementation

Submitted by: Pratham Raina (B00158273), Muhammad Muneeb Nadeem (B00158381)

Submission date:19/03/2024

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: Pratham Raina Dated:19/February/2024

Author: Muhammad Muneeb Nadeem Dated: 19/February/2024

Table of Contents

Title

EireWay Coaches: Seamless Connectivity on the Emerald Isle

The aim of this document is to present the design and draft implementation of our software engineering project (Eire-ways Coaches) that aimed comprehensive online service system which will enhance user experience. A key feature of the project is registration and login, a ticket booking system, and career opportunities and feedback submission. To structure our system into manageable components, we have used Object-Oriented Analysis and Design (OOAD) principles, visualized using Unified Modelling Language (UML). As part of our methodology, we incorporate system modelling to gain a better understanding of the functionality from multiple perspectives and to clearly communicate the architecture of the system. For the project we use class diagram to gain better understanding of structure and relationships within the system. We also used ERD diagram which help depict the data relationships crucial for database design.

1.Project Definitions

1.1 Purpose of document

The purpose of this document is to outline the design and implementation plan for our software project, explain how OOAD and UML are applied, and assess how well the plan aligns with the user requirements and functionality for the project.

1.2 What will the product do?

- This product will improve the user experience by providing them a lot of facilities online users can login as guest or register themselves for services.
- This product will also provide a comprehensive ticket booking system with options like seat choice, extras etc.
- It will include user-centric features such as feedback submission and career opportunities, help and support so users can submit feedback and apply for jobs by simply going on website.

1.3 Main components of the software system

- One of the main component of this software is users can register themselves and login as existing member so they can get the discounts after their first travel.
- Other main component is the ticket reservation system that interact with database to check the available routes, seats and generate the booking number or confirmation.

An administrator can overview the feedback, customers queries and complaints and process the career applications

2.Document Revision

The document has been updated to integrate design elements following Object-Oriented Analysis and Design (OOAD) principles, with the inclusion of Unified Modelling Language (UML) diagrams like class diagrams and ERDs for a clearer system structure and data relationships. Methodology explain the OOAD usage and system models' importance. Use case specifications have been refined to better define class structures and database design, with detailed explanations of design decisions. Changes from the initial proposal have been made to the project to meet the demands of project.

3. Methodology

3.1 System Modelling

As the name suggests, system modelling is the process of developing abstract models of a system to gain a deeper understanding of that system from a variety of perspectives. A system model helps an analyst understand the functionality of a system and models are used to communicate with customers.

3.2 UML

Software engineering's Unified Modelling Language (UML) provides a standard way to visualize a system's design and is intended for use in general-purpose, development-focused modelling roles.

3.3 Use and Necessity of OOAD

Object-Oriented Analysis and Design (OOAD) involves breaking down a system into manageable components called objects, so that it can be analysed and designed more effectively. The OOAD process uses UML as a standard modelling language for the purpose of visualizing, specifying, constructing, and documenting the artifacts of an object-oriented system.

A major benefit of OOAD is its approach to design and problem-solving, a practice that mirrors the way systems work in the real world. Instead of looking at systems as functions or sequences of commands, they are viewed as collections of objects that interrelate. The reusability of components and ease of maintenance and modification of complex systems make them easier to understand and manage.

3.3 What is a Class Diagram?

A class diagram is a type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations or methods, and the relationships among the classes.

It serves as a blueprint for the system's code structure, allowing developers to visualize the system's design and understand the interactions between its components.

The class diagram plays a very important role in software engineering lifecycle as they provide a high-level overview of a system's design, providing a means of communicating and documenting the software's structure. It is one of the most important tools in object-oriented design.

3.3 The benefits of using classes in object-oriented programming

3.3.1 Modelling Class Structure:

In a class diagram, you show the relationships between classes, their attributes, methods, and relationships between classes so that you can model the structure of a system. Using this approach, the architecture of the system becomes clearer and more organized.

3.3.2 Understanding Relationships:

The class diagram displays the relationships between classes, which include associations, aggregations, compositions, inheritance, and dependencies. This will help software developers to understand how different components of system are connected.

3.3.3 Communication:

A class diagram serves as a communication tool between members of a team and stakeholders. The use of these representations provides a visual and standardized representation that is easy to comprehend for the technical as well as the non-technical audience.

3.3.4 Blueprint for Implementation:

A class diagram serves as a communication tool between members of a team and stakeholders. The use of these representations provides a visual and standardized representation that is easy to comprehend for the technical as well as the non-technical audience.

3.4 Static and Dynamic Case Diagrams

Use case diagrams are a set of UML diagrams that provide a graphical representation of the interactions between the users (actors) and the system, to achieve a goal. They show the various use cases, which are the functionalities or services provided by the system, and the actors that interact with those use cases.

3.4.1 Static Diagram

As the name suggests, the static or structural view emphasizes the static structure of the system using objects, attributes, operations, and relationships. Classes, composite structures, and relationships are all included in this view.

3.4.2 Dynamic Case Diagrams?

In dynamic views, the system's behaviour is shown as a dynamic process that involves collaboration between objects and that takes place as internal states are changed. This includes sequence diagrams, activity diagrams, and uses case diagrams.

3.4 What is an ERD?

Entity Relationship Diagrams (ERDs) are a type of flowchart that shows how parts of a system, such as people, objects, or concepts, relate to one another via the relationships between them. In software engineering, business information systems, education and research, ER Diagrams (also known as ERD's or ERD models) are usually used to design or debug relational databases. To show the interconnectedness between entities, relationships, and their attributes, they use symbols such as rectangles, diamonds, ovals, and connecting lines.

3.5 Storage: Volatile and Persistent

In the context of software systems, **volatile** storage refers to temporary data storage that is typically lost when the system is powered down, like RAM.In simpler terms, volatile storage is like a system's short-term memory that clears when it's turned off, similar to our own memory forgetting things when we sleep. **Persistent** storage, on the other hand, ensures data is saved even when the system is turned off such as in databases like MySQL,persistent storage is more like writing something down on paper.For bus website, volatile and persistent storage play critical roles. Volatile storage is important because it quickly handles all the active information during user sessions, like when customers are choosing their seats. It ensures the website responds fast but doesn't need to save that information forever. Persistent storage for the website is like long-term memory. It's essential for saving important data that can't be afforded to lose, such as bus routes.

3.6 User Interface Template

For the Eireway Coaches website, the user interface (UI) has been thoughtfully designed to align with the service's functional specifications, enhancing the user experience and meeting project objectives.

3.6.1 Easy Navigation:

The UI presents modern look with intuitive navigation. The homepage clearly offers options such as 'Book Ticket,' 'Login,' and 'Register,' guiding users straightforwardly to their next step. The use of vibrant images and clear, inviting users into the experience of traveling through Ireland.

3.6.2 Functional Implementation Support:

The 'Book Ticket' page directly supports the booking system functionality. Users are presented with simple, easy-to-use drop-down menus for selecting departure and destination cities, date and time, and seat preferences, which directly corresponds with the functional requirements for booking a journey.

3.6.3 User Friendly Experience:

The UI templates for both registration and login are minimalist, ensuring that users are not overwhelmed by unnecessary information. This approach makes the process of creating an account or logging in quick and straightforward.

3.6.4 Technical Advantage:

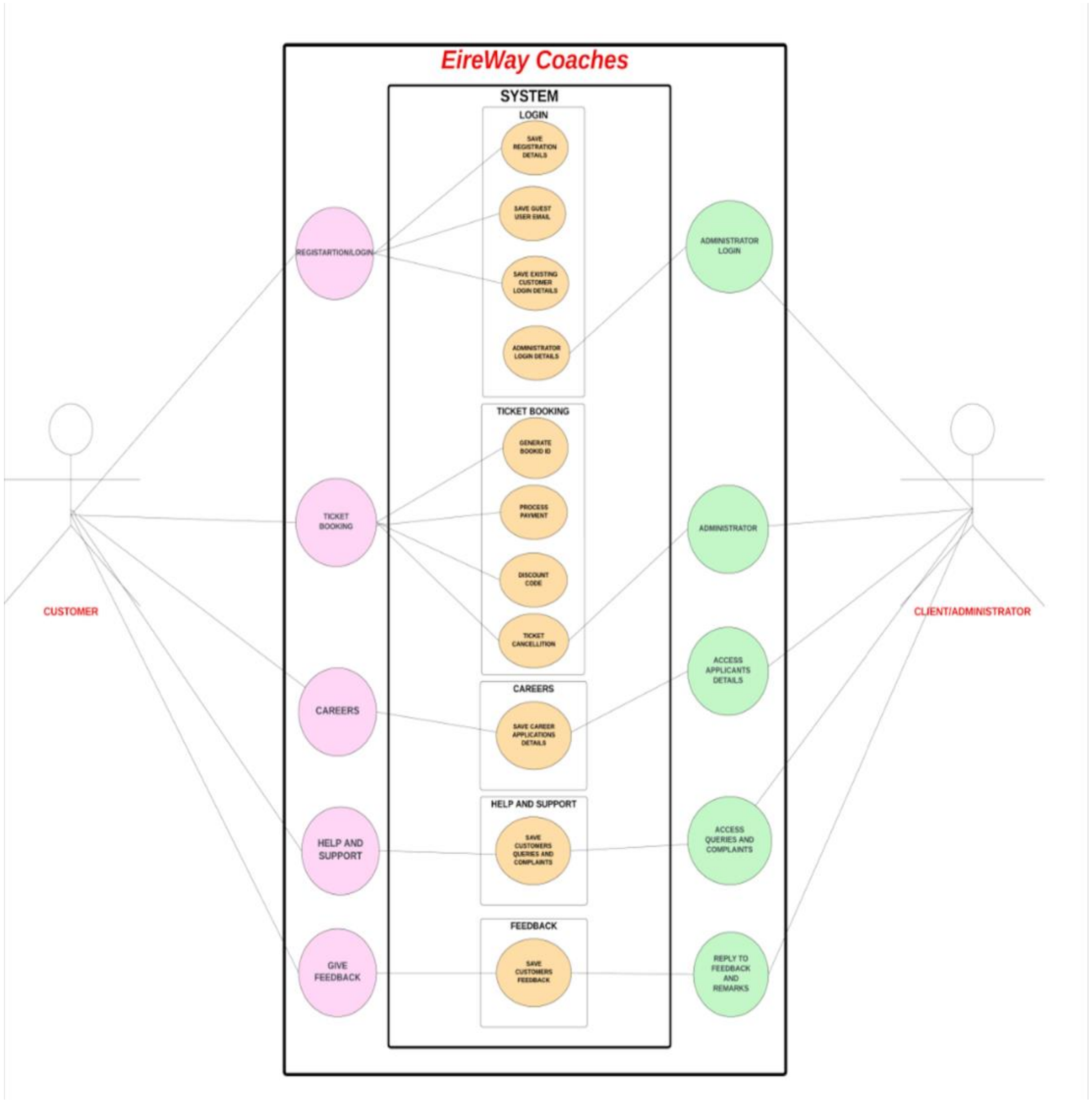
The UI's responsive design ensures that the website will display well on a variety of devices and screen sizes. Quick load times and compatibility across different browsers contribute to a seamless user experience.

This UI design template is carefully selected to ensure the successful execution of the project's functional specifications and to support the overall goal of providing a smooth and enjoyable booking experience for Eireway Coaches' customers.

4. Requirements

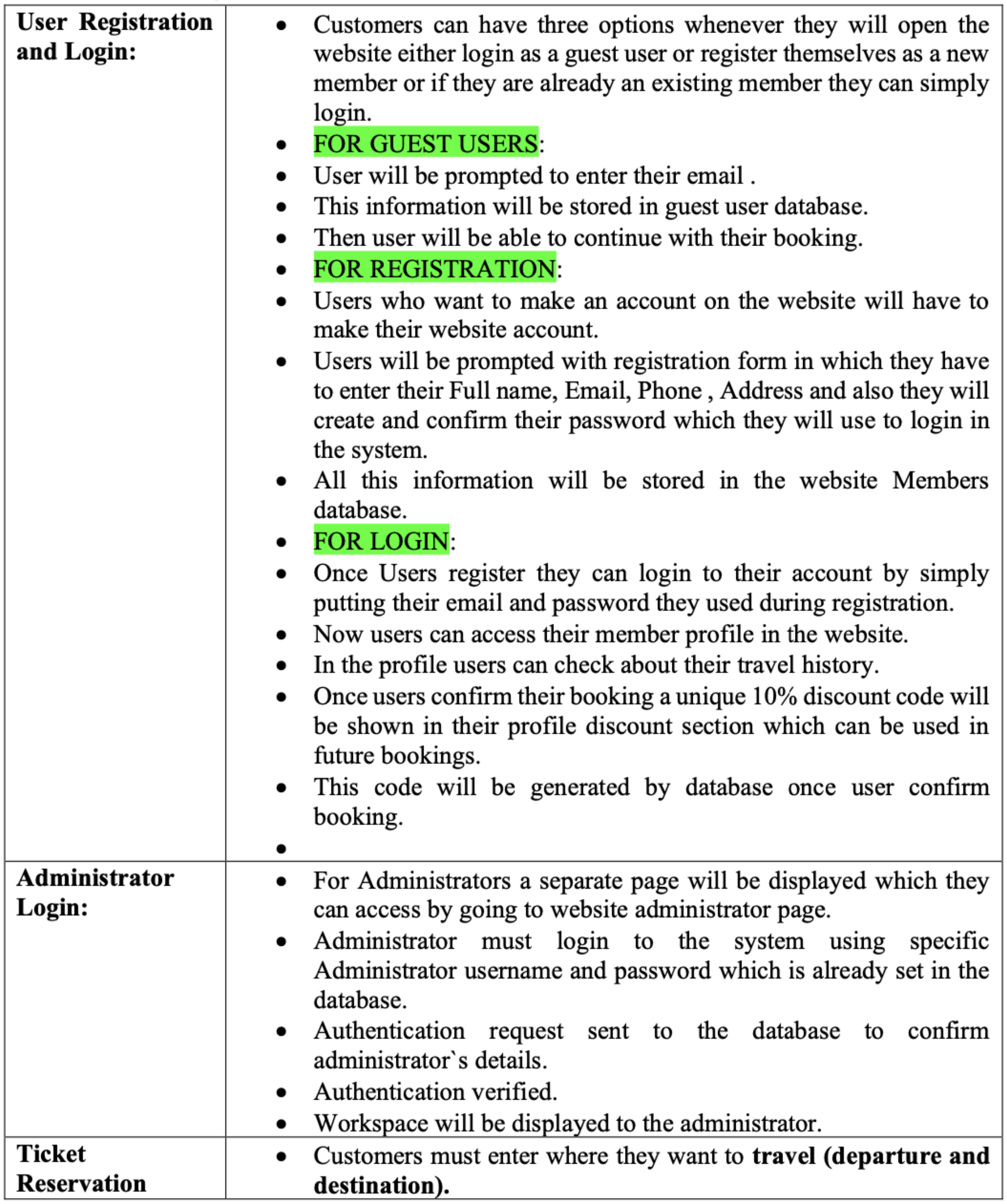
4.1 Use Cases

4.1.1 Use Case Diagram:



4.2 Use Case Specifications

Use Case



User Registration and Login:	• Customers can have three options whenever they will open the website either login as a guest user or register themselves as a new member or if they are already an existing member they can simply login. • FOR GUEST USERS: • User will be prompted to enter their email . • This information will be stored in guest user database. • Then user will be able to continue with their booking. • FOR REGISTRATION: • Users who want to make an account on the website will have to make their website account. • Users will be prompted with registration form in which they have to enter their Full name, Email, Phone , Address and also they will create and confirm their password which they will use to login in the system. • All this information will be stored in the website Members database. • FOR LOGIN: • Once Users register they can login to their account by simply putting their email and password they used during registration. • Now users can access their member profile in the website. • In the profile users can check about their travel history. • Once users confirm their booking a unique 10% discount code will be shown in their profile discount section which can be used in future bookings. • This code will be generated by database once user confirm booking. •
Administrator Login:	• For Administrators a separate page will be displayed which they can access by going to website administrator page. • Administrator must login to the system using specific Administrator username and password which is already set in the database. • Authentication request sent to the database to confirm administrator`s details. • Authentication verified. • Workspace will be displayed to the administrator.
Ticket Reservation	• Customers must enter where they want to **travel (departure and destination).**

	• System makes request to search in the company database whether the route is **available** or **not**. • Customers will **select the route** from displayed list. • Customers must enter the **travel date** and **time** to view available tickets • System makes request to the database to search all available tickets. • Customer must select the particular ticket to book. • After selecting ticket, customer will select the seat from given options(Basic Economy, Recliner Economy). • Database will display the list of available seats for both seating options(Basic Economy, Recliner Economy). • Customer will select the seat number from given list. • Now new page will appear and customers can select the **extras** if they want to. • Booking summary will be displayed and also the edit option if customer want to add or delete anything. • The payment form is displayed to the customer. • The customer must fill in payment details and confirm the booking. • Database store the details of the booking. • Database return a unique booking number to the system. • System display booking number to the customer with confirmation message.
Ticket cancellation:	• Users must enter the Booking number. • System makes request to the database to search this exact booking number. • Database return the booking details to the system. • System display Booking details to the guest. • Users must confirm the cancellation for their booking. • Database delete the booking . • Database send the refund to the system. • System displays refund information to the User.
Feedback	• Customers can give feedback about the user experience in website and their travel experience in Feedback Page. • This Information will be stored in the database. • Administrator can access this information in particular feedback section of database and can make necessary improvements.
Careers	• Users can access the Careers page from home page of the website. • Whoever chooses to go to careers page will be displayed two option – Drivers and Staff. • Users will now fill up the forms with desired information. • This information will be stored in database.

	• Admins can view this information from database and can contact them by email.
Help And Support	• Users can access the Help And support page from home page of the website. • Users will enter their email and can mention their problem in comment section in help and support page. • Their entered information will then be stored in the database. • Administrators can see this information in the database can accordingly address the problem.

4.3 Use Case Specifications in Development of System Structure

In this section we described how each use case has led to the creation of specific classes, attributes, and methods within the system, as well as the design of the database tables.

- **User Registration and Login**: The use cases for user registration and login have led to the creation of a 'User' class, which includes attributes like name, password, email. Methods within this class include AddUser() for adding a user to database . This also translates into a User's table in the database with columns for each attribute. This use case also led to creation of Customer class which is inherited from User class with relevant methods and attributes.

- **Ticket Booking**: This use case has defined the Ticket class, Booking Class, Route Class and Payment Class. The database design reflects this with a Ticket table, Booking table, route table and payment table that records all necessary details. Various attributes and methods have been declared in these classes discussed further in class diagram section.

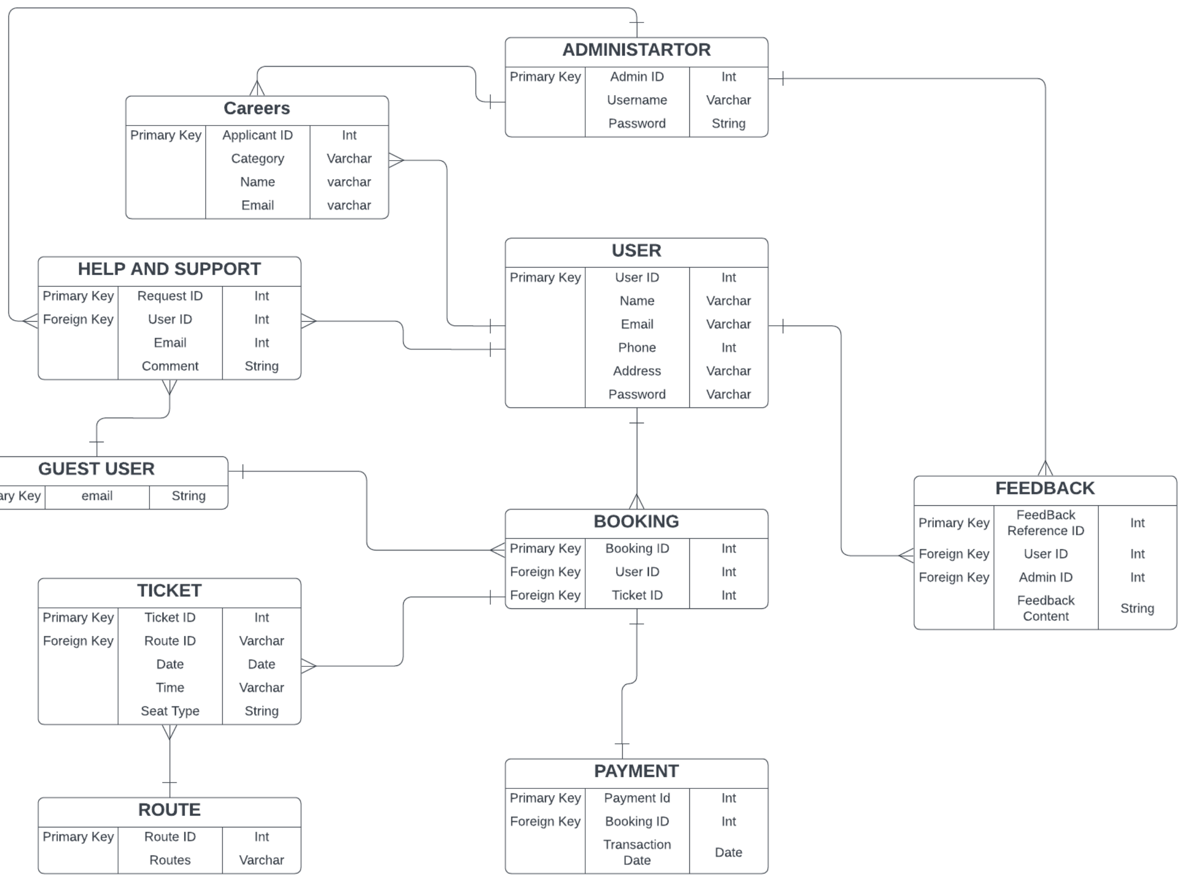
- **Administrator Login**: From this use case, an Admin class is identified with methods to authenticate login and manage bookings. The database contains an Admins table with login credentials and a link to actions they can perform within the system**.(This Class is yet to be implemented in the system)**

- **Feedback**: This leads to a 'Feedback' class with a method for users to submit feedback, and for admins to review it. The corresponding database table stores feedback entries linked to user IDs. **.(This Class is yet to be implemented in the system)**

- **Help And Support:** Similar to Feedback use case this leads to a Help and Support class with a method for users to submit their queries, and for admins to review and reply user queries. The corresponding database table stores feedback entries linked to user IDs. **.(This Class is yet to be implemented in the system)**

5. Case Diagrams

5.1 Entity Relation Diagram:



The Entity Relationship Diagram (ERD) for Eireway Coaches is designed to show the essential components of the system and their interconnections. Below are the attributes, relationships, and multiplicities of each entity within the ERD:

Entities and Attributes:

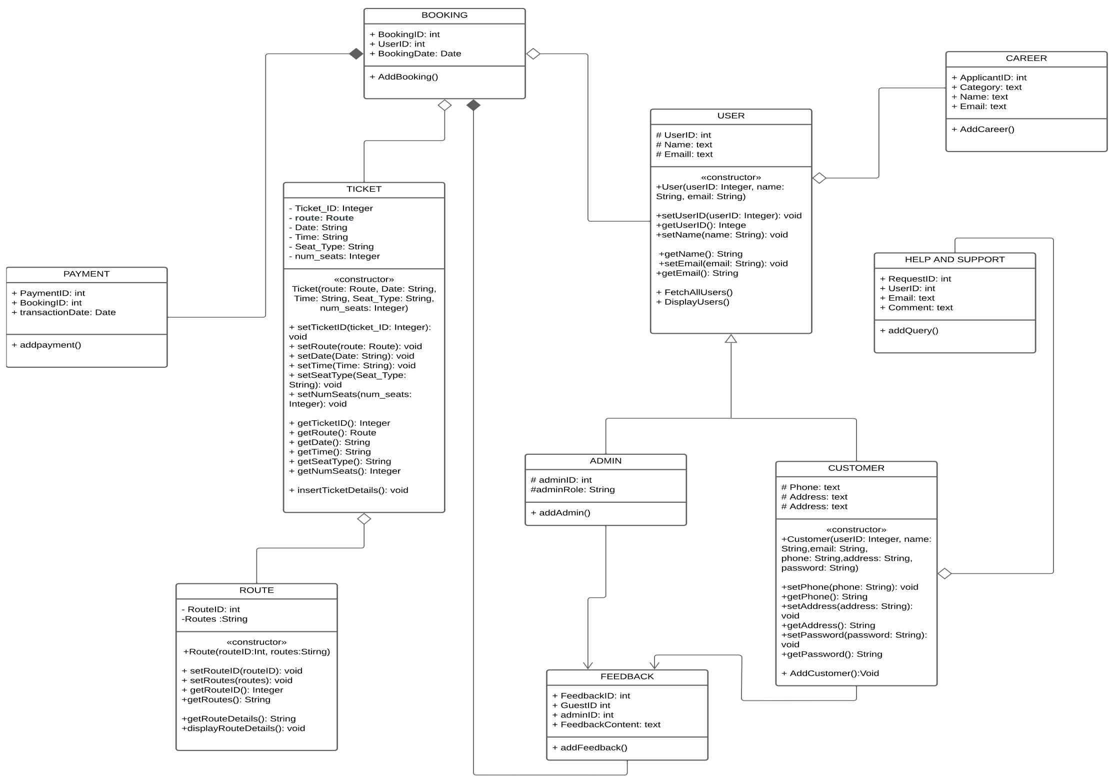
- **User**: Attributes include User ID (primary key), Name, Email, Phone, Address, and Password, capturing information for registered members.
- **Guest User**: Has a single attribute, Email, recognizing that guest users are identified uniquely by their email without creating a full profile.
- **Administrator**: Attributes include Admin ID (primary key), Username, and Password, ensuring administrators can securely access their admin functionalities.
- **Booking**: Composed of Booking ID (primary key), User ID (foreign key), and Ticket ID (foreign key), representing the link between a user and their ticket purchase.
- **Ticket**: Includes Ticket ID (primary key), Route ID (foreign key), Date, Time, and Seat Type, detailing the specifics of the travel.
- **Route**: Contains Route ID (primary key) and Routes (detailing the path of the route), essential for ticket booking.
- **Payment**: With Payment ID (primary key) and Booking ID (foreign key), it stores transaction data linked to bookings.
- **Careers**: Holds Applicant ID (primary key), Category, Name, and Email, enabling job application submissions.
- **Help and Support**: Consists of Request ID (primary key), User ID (foreign key), Email, and Comment, tracking user inquiries and issues.
- **Feedback**: Includes Feedback ID (primary key), Reference ID, User ID (foreign key), Admin ID (foreign key), and Feedback Content, for user feedback on services.

Relationships and Multiplicities:

- **User to Booking**: One-to-Many, as one user can have multiple bookings.
- **Booking to Payment**: One-to-One, where each booking corresponds to a single payment.
- **Booking to Ticket**: Many-to-One, multiple bookings may relate to one ticket for cases like round trips or group bookings.
- **User to Feedback, Careers, Help and Support**: One-to-Many, one user can submit multiple entries in each of these areas.
- **Ticket to Route**: Many-to-One, as various tickets will be available for a single route.

Design Decisions:
In deciding how to set up the database for the Eireway Coaches booking service, we picked out the most important bits of information each part of the system needs while keeping everything straightforward and easy to handle. The entities are selected keeping mind the need of a bus booking system. The 'Booking' and 'Ticket' entities are at the heart of the service, detailing the reservations and specific journey information that customers need. 'Routes' organize the travel options, while 'Payment' tracks the financial aspects linked to each booking, critical for business accounting. Entities like 'Careers', 'Help and Support', and 'Feedback' directly engage with users, aiding in recruitment, customer service, and quality enhancement. This sets a solid foundation for managing bookings, user interactions, and administrative tasks within Eireway Coaches bus booking system. We acknowledge that the current ERD has certain limitations and issues that have been identified, which will be addressed in upcoming assignments There are discrepancies between the ERD and the class diagram that reflect the evolving understanding of the system's requirements. These differences are recognized and will be gradually rectified as the project progresses.

5.2 Class Diagram:



The class diagram for the Eireway Coaches booking system has been carefully designed to encapsulate the functionality and relationships between different components of the system.

Classes, Attributes, and Methods:

1. **User**
   - Attributes: UserID, Name, Email, Phone, Address, Password
   - Methods: User(), setUserDetails(), getName(), setName(), getEmail(), setEmail(),FetchAllUsers(),DisplayUsers()
     etc.
   - Inherits: Customer, Admin
2. **Customer** (inherits from User)
   - Additional Attributes: Phone, Address
   - Methods: Customer(), setPhone(), getPhone(), setAddress(), getAddress(), etc.
3. **Admin** (inherits from User)
   - Additional Attributes: adminID, adminRole
   - Methods: addAdmin(), etc.
4. **Ticket**
   - Attributes: TicketID, route (Route), Date, Time, SeatType, seat_num (int)
   - Methods: Ticket(), setRoute(), setDate(), setTime(), setSeatType(), setSeatNum(), insertTicketDetails() etc.
5. **Route**
   - Attributes: RouteID, Routes
   - Methods: Route(), setRouteID(), setRoutes(), getRouteID(), getRoutes(),getRouteDetails() ,DisplayRouteDetails()
     etc.
6. **Booking**
   - Attributes: BookingID, User (User), Ticket (Ticket), bookingDate (Date)
   - Methods: Booking(), addBooking(), etc.
7. **Payment**
   - Attributes: PaymentID, BookingID (int), transactionDate (Date)
   - Methods: addPayment(), etc.
8. **Career**
   - Attributes: ApplicantID, Category, Name, Email
   - Methods: addCareer(), etc.
9. **Help and Support**
   - Attributes: RequestID, User (User), Email, Comment
   - Methods: addQuery(), etc.
10. **Feedback**
    - Attributes: FeedbackID, User (User), Admin (Admin), FeedbackContent
    - Methods: addFeedback(), etc.

Associations and Relationships:

Inheritance (Generalization):

- **User>>Customer and Admin**: The "Customer and Admin(**not implemented**)" classes are inherited from the "User" class. This is indicated by the line connecting "Customer" and "Admin" to "User," with a hollow triangle near the "User" class. This shows that "Customer" is an inherited from "User" with additional attributes.

Full Aggregation (Composition):

- **Payment and Booking:** The “Payment” class have a composition relationship with "Booking," indicated by a solid diamond connecting them, which would imply Payment cannot exist without a Booking.
- **Feedback and Booking:** Similarly, “Feedback” class cannot exist without “Booking” class which also shows another composition relationship.

Partial Aggregation:

- **Ticket and Booking**: There is an association called weak aggregation between “Ticket ” and “Booking ” class denoted by a hollow diamond notation.Ticket can be associated with a "Booking," but the ticket itself is a distinct entity that can exist even if the booking is cancelled or removed.
- **Ticket and Route:** The association between "Ticket" and "Route" also represents a weak aggregation. A "Ticket" is associated with a particular "Route," but routes are independent entities that exist regardless of whether a ticket has been issued for them. This means a "Route" can be defined without any tickets, and a "Ticket" refers to a "Route" to specify the journey it represents.
- **User and Booking:** We have another aggregation between “User” and “Booking” class. A "User" can have multiple "Bookings," but the user entity doesn't cease to exist if all bookings are deleted.
- **User and Careers:** The "User" class is connected to "Careers" through weak aggregation relationship. A "User" can have multiple "Career" entries, possibly representing job applications or interests in various career opportunities within the company. The career details don't rely on the user's existence and can be managed independently.
- **Customer and Help and Support**: The "Customer" class, which inherits from "User," is similarly associated with "Help and Support" through a weak aggregation relationship. This implies customers can make multiple help and support inquiries, and the system needs to address these inquiries separately from the customer profile itself.

Multiplicities:

User to Booking: A one-to-many relationship is likely, as one user can have multiple bookings.

User to Payment: Since each booking may only correspond to one payment, and assuming a user may have multiple bookings, this could be a one-to-many relationship.

User to Feedback: A one-to-many relationship, as a user could potentially provide multiple feedback entries.

User to Help and Support: Also likely a one-to-many relationship, with a single user able to make multiple help and support requests.

User to Career: This would typically be a one-to-many relationship if users can apply for more than one job.

Booking to Ticket: This might be a one-to-one relationship if each booking corresponds to one ticket, or one-to-many if a booking can include multiple tickets (e.g., for group bookings).

Booking to Payment: Likely a one-to-one relationship, with each booking resulting in a single payment.

Ticket to Route: A many-to-one relationship, as multiple tickets can be issued for the same route.

Design Decision:

These relationships, attributes, and methods have been designated to construct a system that captures the complexity of a bus booking platform while ensuring the architecture remains logical and manageable. The attributes and methods are chosen to provide the necessary functionality for each part of the system, such as handling user information, processing bookings, managing payments, and enabling feedback and career applications. The inheritance and associations are designed to reflect the natural hierarchy and interactions within the system.

6.Conclusion

The document has successfully outlined the design and implementation strategies for the Eireway Coaches online service system, using Object-Oriented Analysis and Design (OOAD) and Unified Modelling Language (UML) to detail the project. The functional elements of the system, such as user registration, ticket booking, and administrative features, have been clearly defined, with careful attention to both user interaction and backend processes. The class diagrams and Entity Relationship Diagrams (ERDs) provide a visual and structural representation of the system, ensuring that every component is logically connected and efficiently organized for both current needs and future scalability.

There's an acknowledgment of the need for improvement, with plans to refine the system based on user feedback and technical developments. As we progress through the project, we anticipate making these enhancements to optimize the website's functionality and user experience. This ongoing commitment to improvement will ensure that Eireway Coaches keeps offering a user-friendly, reliable, and efficient online booking service.