

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**ADVENCED OBJECT ORIENTED PROGRAMMING**

**2024–2025 SPRING SEMESTER**

**PROJECT REPORT**

**(Online Reservation System)**

**DELIVERY DATE**

20/06/2025

**PREPARED BY**

05220000056, Arda Ferad

05230000339, Ecesu Kocaerler

05220000370, Berkay Can Benli

05220000991, Ömer Faruk Çakmakcı

Contents

[1) UML Class Diagram 4](#_Toc201229504)

[2) Design Patterns 4](#_Toc201229505)

[2.a) Factory Pattern 4](#_Toc201229506)

[2.b) Builder Pattern 5](#_Toc201229507)

[2.c) Facade Pattern 5](#_Toc201229508)

[2.d) Singleton Pattern 5](#_Toc201229509)

[3) Source Codes 6](#_Toc201229510)

[3.a) Model Package 6](#_Toc201229511)

[3.a.1) User Class 6](#_Toc201229512)

[3.a.2) UserType Enum 6](#_Toc201229513)

[3.a.3) Admin Class 7](#_Toc201229514)

[3.a.4) Passenger Class 7](#_Toc201229515)

[3.a.5) Seat Class 8](#_Toc201229516)

[3.a.6) SeatLayoutType Enum 8](#_Toc201229517)

[3.a.7) VoyageType Enum 9](#_Toc201229518)

[3.a.8) Voyage Class 9](#_Toc201229519)

[3.a.9) Reservation Class 11](#_Toc201229520)

[3.b) Factory Package 11](#_Toc201229521)

[3.b.1) UserFactory Class 11](#_Toc201229522)

[3.b.2) VoyageBuilder Class 12](#_Toc201229523)

[3.c) Util Package 12](#_Toc201229524)

[3.c.1) Database Class 12](#_Toc201229525)

[3.c.2) PDFGenerator Class 13](#_Toc201229526)

[3.d) ControllerPackage 14](#_Toc201229527)

[3.d.1) SystemController Class 14](#_Toc201229528)

[3.e) View Package 16](#_Toc201229529)

[3.e.1) LoginPanel Class 16](#_Toc201229530)

[3.e.2) AdminPanel Class 17](#_Toc201229531)

[3.e.3) PassengerPanel Class 18](#_Toc201229532)

[3.e.4) MyReservationsPanel Class 19](#_Toc201229533)

[3.e.5) SeatSelectionPanel Class 20](#_Toc201229534)

[3.e.6) MainFrame Class 21](#_Toc201229535)

[3.e.7) RoundedList, RoundedPanel, RoundedPasswordField, RoundedScrollPane, RoundedTextField, SeatButton, HintTextField, IconCustomizer Classes 22](#_Toc201229536)

[3.f) Main Package 23](#_Toc201229537)

[3.f.1) FinalProject Class 23](#_Toc201229538)

[4) Unit Tests 23](#_Toc201229539)

[5) User’s Manual 26](#_Toc201229540)

[5.a) Register 26](#_Toc201229541)

[5.b) Log in 26](#_Toc201229542)

[5.c) Add voyage 27](#_Toc201229543)

[5.d) Delete voyage 28](#_Toc201229544)

[5.e) Search and select voyage 28](#_Toc201229545)

[5.f) Seat reservation 29](#_Toc201229546)

[5.g) Generate reservation PDF 30](#_Toc201229547)

[5.h) Cancel reservation 30](#_Toc201229548)

# 1) UML Class Diagram

metin, diyagram, doküman, belge, paralel içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

# 2) Design Patterns

## 2.a) Factory Pattern

In this project, the **Factory Design Pattern** is used to create user objects (Admin or Passenger) without exposing the instantiation logic to the client. The Factory pattern is a **creational design pattern** that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. It helps in managing and encapsulating object creation, making the code more modular and scalable.

The UserFactory class encapsulates the object creation process and allows for centralized and controlled creation of different User types based on the provided user role. It includes two overloaded static methods named createUser, one for Passenger and one for Admin. Based on the given type parameter, the appropriate subclass is instantiated and returned. This approach improves **code maintainability**, **readability**, and **flexibility**, especially when the instantiation logic becomes more complex. By applying the Factory pattern, we avoid the need for multiple new calls scattered throughout the code, and we isolate the decision logic for which class to instantiate, resulting in cleaner and more maintainable code.

## 2.b) Builder Pattern

The **Builder Design Pattern** is a **creational design pattern** used to construct complex objects step by step. Unlike constructors with many parameters, the Builder pattern provides a clear and readable way to create objects by chaining method calls. It is especially useful when an object has many optional fields or requires a specific construction process.

In this project, the VoyageBuilder class is used to create instances of the Voyage class. Instead of calling a constructor with many parameters directly, we use a fluent interface where each setter method (like setOrigin, setTime, etc.) returns the VoyageBuilder itself. Once all necessary fields are set, the build() method creates the Voyage object.

This approach improves **readability**, **maintainability**, and **flexibility**, and avoids the need for multiple constructors or unclear parameter lists. By using the Builder pattern, the object creation becomes more readable and easier to manage, especially as the number of fields increases.

## 2.c) Facade Pattern

The **Facade Design Pattern** is a **structural design pattern** that provides a simplified interface to a complex subsystem. It hides the internal complexity of the system and exposes a higher-level interface that makes the subsystem easier to use.

In this project, the SystemController class implements the **Facade pattern** by acting as the **central interface** between the user interface (UI), the model classes (such as Voyage, User, Reservation), and the file-based database (Database class). Instead of letting UI components interact directly with many different parts of the system, all interactions are handled through the SystemController.

This pattern improves **modularity**, **separation of concerns**, and **maintainability**, since the logic for user registration, login, voyage management, and reservation handling is grouped in a single, cohesive class. Using this facade, the UI layer can perform all essential operations without worrying about how objects are created or data is stored internally.

## 2.d) Singleton Pattern

The **Singleton Design Pattern** is a **creational design pattern** that ensures a class has **only one instance** and provides a global access point to that instance. It is commonly used when exactly one object is needed to coordinate actions across a system — for example, for managing a centralized database or configuration manager.

In this project, the Database class implements the **Singleton pattern** to manage file-based data storage (users.txt, voyages.txt, reservations.txt). Instead of creating multiple instances of the class, the application uses a single shared instance, obtained via the static getInstance() method. This ensures that all operations like reading, writing, and updating data are performed through the same instance, avoiding data inconsistency or redundancy. The private static Database instance field holds the only instance of the class, and the constructor is private to prevent external instantiation. This pattern ensures **controlled access**, **consistent data handling**, and **efficient resource use** across the application.

# 3) Source Codes

## 3.a) Model Package

### 3.a.1) User Class

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.2) UserType Enum

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.3) Admin Class

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.4) Passenger Class

metin, ekran görüntüsü, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.5) Seat Class

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.6) SeatLayoutType Enum

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.7) VoyageType Enum

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.8) Voyage Class

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.a.9) Reservation Class

metin, ekran görüntüsü, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

## 3.b) Factory Package

### 3.b.1) UserFactory Class

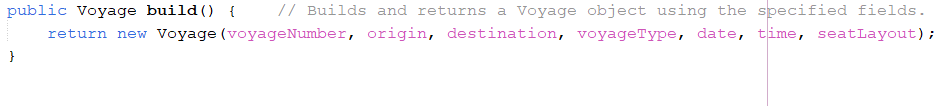
metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.b.2) VoyageBuilder Class

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.



## 3.c) Util Package

### 3.c.1) Database Class

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. The Database class is a singleton that handles all file-based data operations for the application. It reads and writes users, voyages, and reservations to corresponding .txt files. It supports adding/removing users, voyages, and reservations, checking for duplicates, loading data into model objects, and linking reservations to seat and passenger data.

### 3.c.2) PDFGenerator Class

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

## 3.d) ControllerPackage

### 3.d.1) SystemController Class

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. The SystemController class manages the core logic of the reservation system. It handles user registration, login/logout, voyage creation/removal (for admins), seat reservations/cancellations (for passengers), and reads/writes data using a file-based Database. It also manages session tracking and generates reservation IDs.

## 3.e) View Package

### 3.e.1) LoginPanel Class

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. **LoginPanel** is the initial screen of the reservation system, designed using Java Swing. It allows users to either **log in** or **register** as a **Passenger** or **Admin**. In registration mode, name and user type selections become visible. If the user selects Admin, an additional dialog prompts for an admin code (3510) to authorize the registration. Upon successful login or registration, the panel redirects the user to the appropriate interface (PassengerPanel or AdminPanel). The panel also includes responsive resizing of UI components and displays decorative background images (plane and bus).

### 3.e.2) AdminPanel Class



We couldn't put them all because the code was too long. The entire code is in the source code. AdminPanel is a user interface panel for administrators. It lets admins view all voyages, add new ones using a detailed form (with fields like date, time, layout), and delete selected voyages. The panel updates seat layout options based on transport type (bus or plane), adapts to screen size, and features a smooth gradient background.

### 3.e.3) PassengerPanel Class

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. PassengerPanel is the main screen for passengers. It lets users search and browse available voyages, make reservations, view their own reservations, and log out. The panel adapts to screen size and uses a clean, two-column layout with a gradient background.

### 3.e.4) MyReservationsPanel Class

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. MyReservationsPanel displays the logged-in passenger’s reservations. Users can view their reservations in a styled list, cancel selected ones, or download them as PDF. The panel adjusts layout dynamically to different screen sizes.

### 3.e.5) SeatSelectionPanel Class

metin, ekran görüntüsü, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

We couldn't put them all because the code was too long. The entire code is in the source code. SeatSelectionPanel allows the user to visually select an available seat for a chosen voyage. It displays a dynamic seat layout based on the transport type and lets the user confirm or cancel the selection.

### 3.e.6) MainFrame Class

metin, ekran görüntüsü, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

### 3.e.7) RoundedList, RoundedPanel, RoundedPasswordField, RoundedScrollPane, RoundedTextField, SeatButton, HintTextField, IconCustomizer Classes

These classes are written to improve the visual appearance. All of them are available in the source code. The SeatButton class is provided as an example implementation.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

## 3.f) Main Package

### 3.f.1) FinalProject Class

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

# 4) Unit Tests

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, yazı tipi, çizgi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

# 5) User’s Manual

## 5.a) Register

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

The user selects **"Kayıt Ol"** from the top, enters the required information, and chooses either **Yolcu** or **Admin**. Then, the **"Devam Et"** button is clicked.

metin, ekran görüntüsü, oyuncak, taşıt, araç içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Admin code is required to complete the operation.

## 5.b) Log in

metin, taşıt, araç, hava taşıtı, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

The user selects **"Giriş"** from the top, enters the required information, and clicks the **"Devam Et"** button.

## 5.c) Add voyage

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

After the admin registers or logs in, the **Admin Panel** opens. The required voyage details are entered, and the **"Sefer Ekle"** button is clicked.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

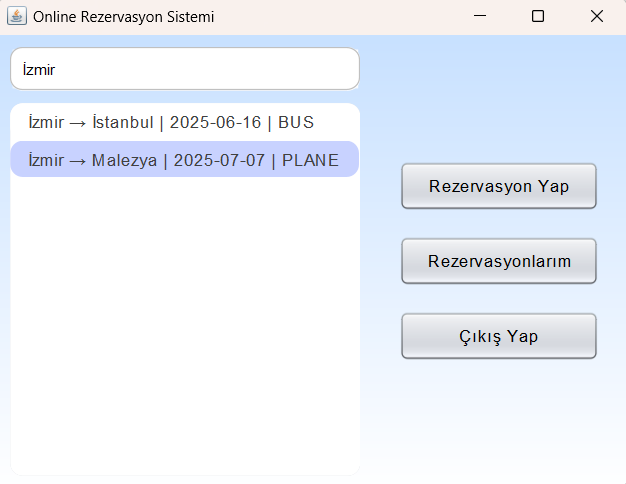
## 5.d) Delete voyage

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

After clicking the **"Seferi Sil"** button, the voyage number is entered, and the voyage is deleted.

## 5.e) Search and select voyage



After the passenger registers or logs in, the **Passenger Panel** opens. A search can be performed using the **search bar**, and after selecting the desired voyage, the **"Rezervasyon Yap"** button is clicked.

## 5.f) Seat reservation

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

After clicking the **"Rezervasyon Yap"** button, the **Seat Selection Panel** opens. Once a seat is selected, the **"Onayla"** button is clicked.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

## 5.g) Generate reservation PDF

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

When the **"Rezervasyonlarım"** button is clicked in the **Passenger Panel**, the reservations panel opens. After selecting a reservation from the list, the **"PDF"** button is clicked.

## 5.h) Cancel reservation

To cancel a reservation, a reservation is selected from the list and the **"Rezervasyonu İptal Et"** button is clicked.

