DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

TICKETWISE

by

2021510061 Ezgi Tan

2021510062 İrem Tekin

2021510072 Kaan Yılmaz

CHAPTER ONE

INTRODUCTION

- **Project Description:**

In our ticket sales application, there will be 2 types of users: customer and company. Initially, there will be different login options for both users. Company: can provide services by entering the agent's information and making it accessible to many people; can make fare, route and date changes; can see ticket fields; Customer: can see matching tickets by selecting vehicle, date and route; can buy tickets for future dates;can also see vehicle occupancy status.

- **Project Purpose:**

The aim of the project is to provide a seamless ticketing system where the customer receives travel services and the company provides travel services. Company: will be able to reach more people and offer more services; at the same time, with customers' route information, aircraft occupancy information, etc. will be able to create more efficient plans for the future. The customer: will be able to get more travel options and at the same time make a more economical decision by comparing prices.

- **Project Scope:**

The scope of the project encompasses the development of a ticket sales application catering to two primary user types: customers and companies.

Features for Company Users:

Login Options: Company users will have dedicated login interfaces tailored to their needs.

Service Provision: Companies can input agent information, facilitating access to a wide audience.

Flexibility in Services: Companies can make adjustments to fare, route, and date parameters as needed.

Ticket Management: Access to view ticket fields and manage ticketing operations.

Features for Customer Users:

Login Options: Customers will have distinct login functionalities tailored to their requirements.

Ticket Search: Customers can search for matching tickets based on vehicle, date, and route preferences.

Ticket Purchase: Customers can buy tickets for future travel dates.

Vehicle Occupancy Status: Ability to view the occupancy status of vehicles.

CHAPTER TWO

REQUIREMENTS

Specific Requirements

2.1 External Interfaces

- **User Interface**
- **Company Interface**
- **Vehicle Interface**

2.2 Functions

Customer:

- addCustomer(String name, String surname, String email, String phoneNumber, String gender, int identity, String birthday, String password, boolean policy): To add a new customer. (void)
- displayCustomerInfo(int identity): To display customer information. (Customer)
- getter/setter(): For accessing and modifying customer attributes. (void)

Company:

- addCompany(String companyName, String companyPhoneNumber): To add a new company. (void)
- displayCompanyInfo(String companyName): To display company information. (Company)
- getter/setter(): For accessing and modifying company attributes. (void)
- addVehicle(Vehicle vehicle, Company company, int capacity, int numberPlate): To add a new vehicle to the company. (void)
- removeVehicle(int numberPlate): To remove a vehicle from the company. (void)

Bus:

- checkAvailableBuses(String originStation, String destinationStation, String departureTime): To check available buses. (boolean)
- addBusTrip(String originStation, String destinationStation, String departureTime, int travelTime, String date, boolean wirelessConnection, boolean catering, Vehicle vehicle, int journeyNumber): To add a new bus trip. (void)
- removeBusTrip(String originStation, String destinationStation, String departureTime, Vehicle vehicle): To remove a bus trip. (void)
- getter/setter(): For accessing and modifying bus attributes. (void)
- displayBusInfo(String originStation, String destinationStation, String departureTime): To display bus trip information. (Bus)

Train:

- checkAvailableTrains(String originStation, String destinationStation, String departureTime): To check available trains. (boolean)
- addTrainTrip(String originStation, String destinationStation, String departureTime, int travelTime, String date, boolean wirelessConnection, boolean catering, Vehicle vehicle, int journeyNumber): To add a new train trip. (void)
- removeTrainTrip(String originStation, String destinationStation, String departureTime, Vehicle vehicle): To remove a train trip. (void)
- getter/setter(): For accessing and modifying train attributes. (void)
- displayTrainInfo(String originStation, String destinationStation, String departureTime): To display train trip information. (Train)

Plane:

- checkAvailablePlanes(String originStation, String destinationStation, String departureTime, Vehicle vehicle): To check available planes. (boolean)
- addPlaneTrip(String originStation, String destinationStation, String departureTime, int travelTime, String date, boolean wirelessConnection, boolean catering, Vehicle vehicle, int journeyNumber): To add a new plane trip. (void)
- removePlaneTrip(String originStation, String destinationStation, String departureTime, Vehicle vehicle): To remove a plane trip. (void)
- getter/setter(): For accessing and modifying plane attributes. (void)
- displayPlaneInfo(String originStation, String destinationStation, String departureTime): To display plane trip information. (Plane)

Policy:

- displayTermsAndConditions(): To display terms and conditions. (void)
- fileOperations(): To read the KVKK (Personal Data Protection Law) text. (void)
- checkPolicy(boolean userConfirmation): To check acceptance of KVKK. (boolean)
- getter/setter(): For accessing and modifying policy attributes. (void)

Payment:

- processPayment(int debitCardNumber, String debitCardName, String debitCardDate, int cvv): To process a payment. (boolean)
- getter/setter(): For accessing and modifying payment attributes. (void)
- displayPayment(): To display the payment screen. (void)

Search:

- searchForTicket(Customer customer): To search for a ticket. (boolean)
- getter/setter(): For accessing and modifying search attributes. (void)
- displaySearch(): To display the search screen. (void)

2.3 Performance Requirements

The website needs to respond quickly and users need to complete ticket search and purchase.

It must be resistant to heavy traffic and be able to meet high user demands.

2.4 Logical Database/File System Requirements

To store user profile information, vehicle information, ticket information, policy information, and payment information, the usage of .txt files.

2.5 Design Constraints

- A desktop interface will be created using Java Swing.
- Necessary measures should be taken for the security of user information. (for example, password hashing and secure session management)

2.6 Software System Quality Attributes

- **Reliability:** The system must process user information correctly and make ticket reservations without errors.
- **Usability**: A user-friendly desktop interface should be designed, and users should be able to easily search for trips and purchase tickets.
- **Portability:** The application to be written in Java must be able to run on different operating systems.
- **Flexibility:** The system should be able to adapt to changing requirements and ensure that new features are easily added and existing features can be updated.
- **Testability:** Different components and functions of the software should be testable separately. This simplifies the debugging and quality control processes.
- **Functionality:**
  Ensuring that the system delivers all intended features and functions accurately and effectively, meeting user requirements and expectations.

2.7 Object Oriented Models

- 2.7.1 Analysis Class Model (Static Model)

- **Customer Class:**

  The Customer class encompasses information related to the customer, including personal details (name, surname, email, phone number, etc.), and manages their interactions within the system. This class acts as a bridge between the user and the application functionality.

- **Company Class:**

  The Company class represents companies involved in ticket sales, storing details such as name, address, and contact information. It acts as a container to organize and manage companies within the system, facilitating interactions with different ticket-selling organizations and providing necessary information for transactions.

- **Bus Class:**

  The Bus class handles specific bus-related properties such as seating capacity, routes, departure times, and ticket pricing. It manages tasks like bus ticket availability checks and fare calculations. This class serves as a model for managing bus-related functions within the application and allows for the display of ticket information.

- **Plane Class:**

  The Plane class controls airplane ticket transactions and includes details such as flight schedules and ticket pricing. It manages aspects like flight availability, seat availability, and ticket structures. This class enables users to reserve airplane tickets, view flight information, and efficiently plan their travel.

- **Train Class:**

  The Train class manages train ticket functions and includes train specifications and ticket pricing. It handles tasks such as train schedules and fare management for train journeys. This class allows users to view train tickets, check availability, and plan their travels efficiently.

- **Policy Class:**

  The Policy class manages the business policies of the application, including other operational guidelines and rules. It contains rules such as data protection regulations and provides necessary information to customers about these policies, ensuring compliance with established regulations. This class serves as a repository for storing and managing business rules and personal data protection law guidelines within the system.

- **Payment Class:**

  The Payment class manages payment transactions and processes ticket purchases made by customers. It securely manages payment information, facilitates payment processes through various methods, and ensures the integrity and confidentiality of financial transactions. This class plays a crucial role in facilitating smooth and secure payment transactions within the application.

- **Menu Class:**

  The Menu class manages the application's main menu and handles user interactions. It facilitates access to different functionalities and interfaces within the system, serving as an access point for users to various features offered by the application.

- 2.7.2 Analysis Collaborations (Dynamic Model)

- **Search Class:**

  The Search class performs ticket searches based on criteria specified by the user and efficiently finds suitable travel options. It uses search algorithms to fetch available tickets that match user preferences. This class optimizes the process of discovering and selecting suitable travel options within the system, enhancing the user experience.