Train management system

**A Local Train Ticketing & Railway Tracking and Arrival Time Prediction Application**

**1) Introduction** :

**Purpose** :

The purpose of this document is to present a detailed description of the local train and railway tracking and arrival time prediction system . It is explain what is the system will do .

**Scope** :

A local train ticketing application is a system allows users to book local train tickets . The ticketing process consists of a ticket booking form .The form allows users to choose thier source & destination and choose the ticket is single or return and first or second class . The system’s database should be filled with all available station which the admin is manage it . Station master manage the schedule of train and if change in the time of train the system display on the station screen.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | A passenger who book ticket train |
| Admin | A person will be responsible for the management station and trip info |
| Stasion admin | A person who manage timing and delay train |
| System’s database | A field include system of details |
| Stackeholder | Any person with an interest in the project is not a developer |

**2) Functional Requirements**

1. Check seat availability: It enables the user to check the availability of the seats for a train before booking the ticket.
2. Quick reservation: It gives the user the facility to do the reservation on the urgent basis with extra charges.

1. Reservation of the ticket: It enables the user to do the reservation.

1. Cancellation: It provides the user with the facility to cancel the ticket at any instant of time.

1. Improved & optimized service: Provides a well versed and optimized service.

1. select class :Allows the user to select the class he wants (ex:standard , economy , first class …etc).

1. select ticket type :Allows the user to select the ticket type he wants (ex:vip , normal … etc).

1. The user shall see train timing

1. The user shall search for the train by train number

1. The user shall see train departure details

1. The user shall see train arrival details
2. The user shall search for the train by source station
3. The user shall search for the train by arrival station

1. The user shall search for the train by departure station
2. The user shall print ticket

1. The Admin shall login

1. The Admin shall logout

1. The Admin shall enter train details ( train number ,source , destination …)

1. Calculate the total duration

1. The Admin shall enter halts information (name ,time ..)

1. The Admin shall enter station master information (name ,phone ,Email ..)

1. The Admin shall see and update train details

1. The Admin shall search for the train by train number

1. The station master shall login

1. The station master shall logout

1. The station master shall see train arrival details

1. The station master shall enter delays

1. The Admin shall update train details after delay

1. The system shall pass on these timing details to another station’s system.

1. The system shall display that information on a screen.

**3) Non functional requirements**

#### 1) Performance Requirements

· This attribute pertains to the ability of a software-driven system to conform to timing requirements. it implies that Software Testing engineers must check whether the system responds to various events within defined time limits. These events may occur in the form of ​process interruptions, messages, and requests from different stations, and others.

#### 2) Security Requirements

· the system shall be able to protect data and defend information from unauthorized access.

· the system shall provides Security includes authorization and authentication technologies

· the system shall provide Network attack protection

· the system shall provides Data encryption

· It is essential for software testing professionals to perform regularly

updated security checks on systems.

#### 3) Software Quality Attributes

Things to consider are:

· Respecting work time

· Clarity in data

· Flexibility at work

· Provide enough time to work

#### 4) Constrains

· One of the assumptions about the software is that the user must be familiar with the English language because the site is written in English.

· The program is limited by shared internet connection, as well as the capacity of the database, so it must have a lot of free space.

Goals:

-To provide a user-friendly interface for booking local train tickets and getting real-time updates on train schedules.

-To provide users with the option to select single journey or return ticket and first-class or second-class carriage

-To provide admins with the ability to recharge user account balance and check journey tickets processed within the system

-To track train timing and provide real-time updates on train schedules and arrival times

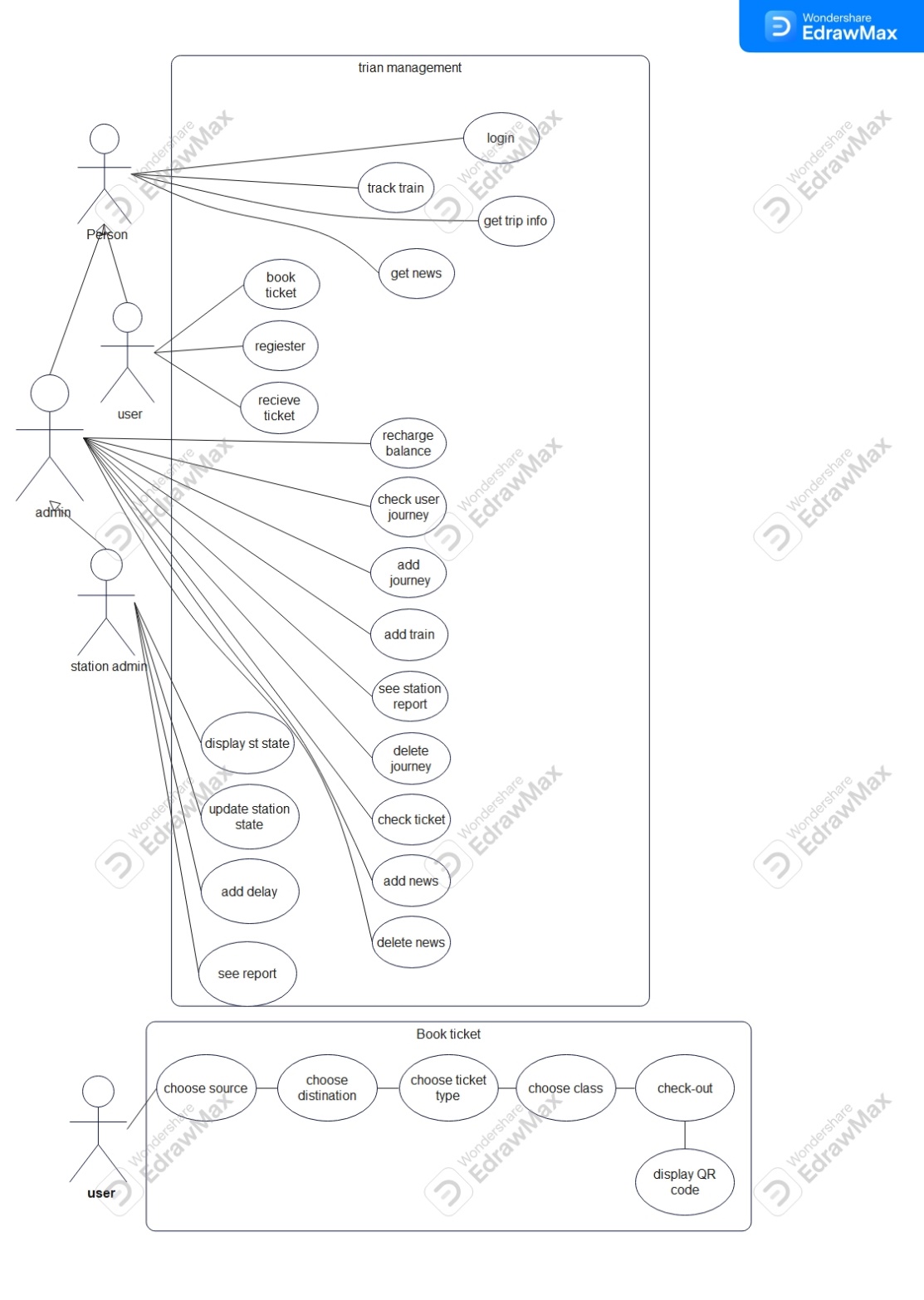
-To track train timing and update expected arrival times at each station based on actual departure times

-To provide station masters with login to update train arrival and departure times at their station

-To publish real-time train schedule events to multiple clients subscribing to the service through their applications.

-To create a scalable, reliable, and secure system designed to meet the needs of the client and users.

**Use case)**

****

**Use case description)**

|  |  |
| --- | --- |
| Name | Book ticket |
| Goal | the user be able to book a journey ticket |
| Precondition | Login, and have enough money in his balance |
| Postcondition | Book the ticket and get QR-code |
| Error state | No enough balance, the journey no longer exist |
| Actors | User |
| Trigger | Book button |
| Standard process | Choose his/her source station, destination station, class, and ticket type submit check balance and get the qr-code |
| Alternative Processes | If the inputs are not filled or there isn’t enough balance return flase |

|  |  |
| --- | --- |
| Name | Track train, or get trip info |
| Goal | To know where the train is |
| Precondition | Have train’s journey id |
| Postcondition | Return where the train is |
| Error state | The id is wrong or not exist. the journey is ended |
| Actors | User, Admin, Admin station |
| Trigger | Press find button |
| Standard process | Enters the id return the respons |
| Alternative Processes | The id is not exist |

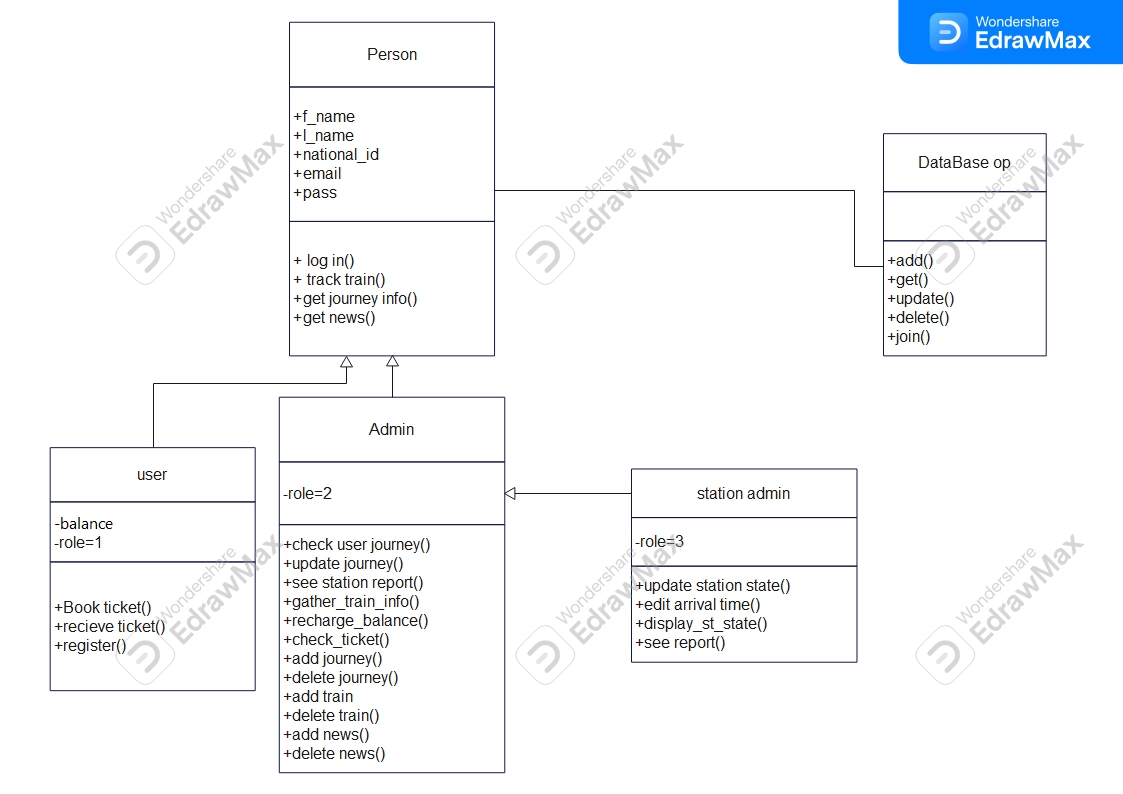
|  |  |
| --- | --- |
| Name | register |
| Goal | To add new user |
| Precondition | Enter his/her information (id, name, email, password) |
| Postcondition | Signup successfully and enter the system |
| Error state | The fields are to filled or the user as an account already |
| Actors | user |
| Trigger | Register btn |
| Standard process | Enter his/her information and get to the system |
| Alternative Processes | Login if has an account |

|  |  |
| --- | --- |
| Name | Recharge balance |
| Goal | Recharge user’s balance |
| Precondition | Enter user id and amount of money |
| Postcondition | Update user’s balance |
| Error state | Wrong id |
| Actors | Admin, admin station |
| Trigger | Recharge btn |
| Standard process | Enters user id and amount of money system returns balance updated successfully |
| Alternative Processes | Nothing changed in user account |

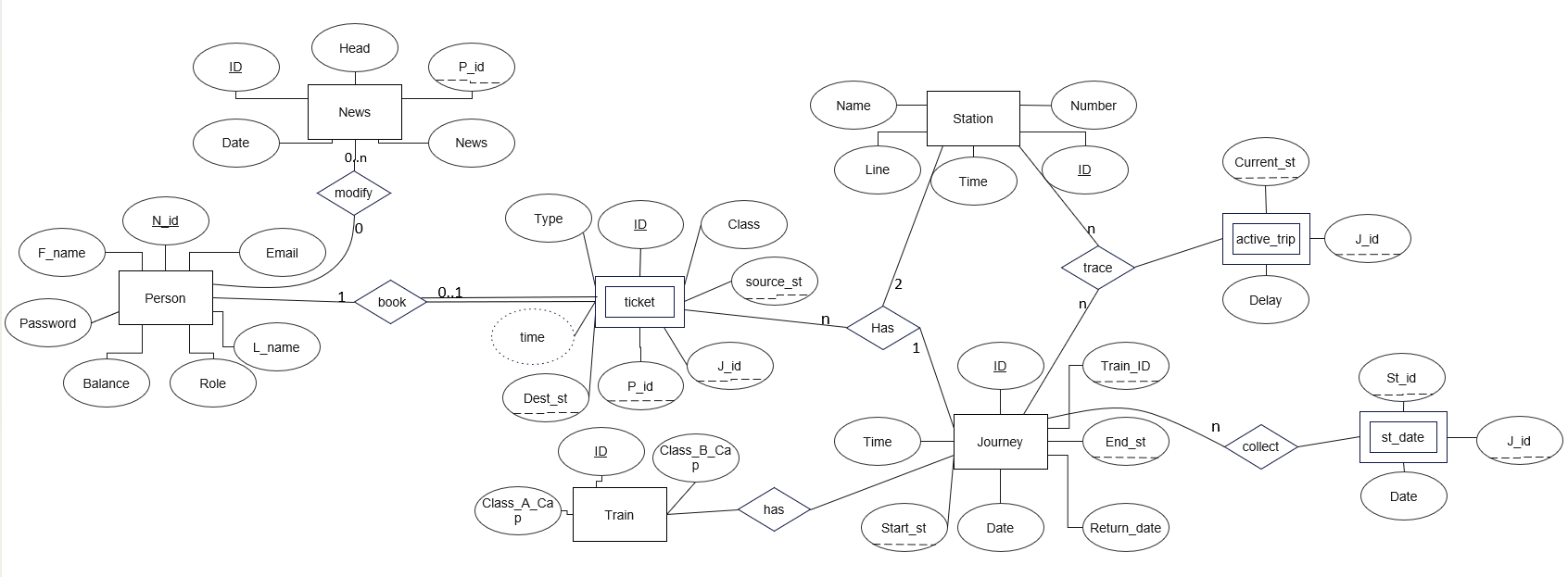
|  |  |
| --- | --- |
| Name | Add journey, add train, add news |
| Goal | To add values to the system |
| Precondition | Fill the field with the data you want to insert |
| Postcondition | Data added to the system |
| Error state | The database is full or have no access |
| Actors | Admin, admin station |
| Trigger | Add button |
| Standard process | Add values press the button |
| Alternative Processes | Returns error is the database |

|  |  |
| --- | --- |
| Name | Delete news, journey |
| Goal | Delete data from the system |
| Precondition | Press the delete button |
| Postcondition | The data deleted |
| Error state | No error state |
| Actors | Admin, admin station |
| Trigger | Delete button |
| Standard process | Press delete button then the data deleted |
| Alternative Processes | No error state |

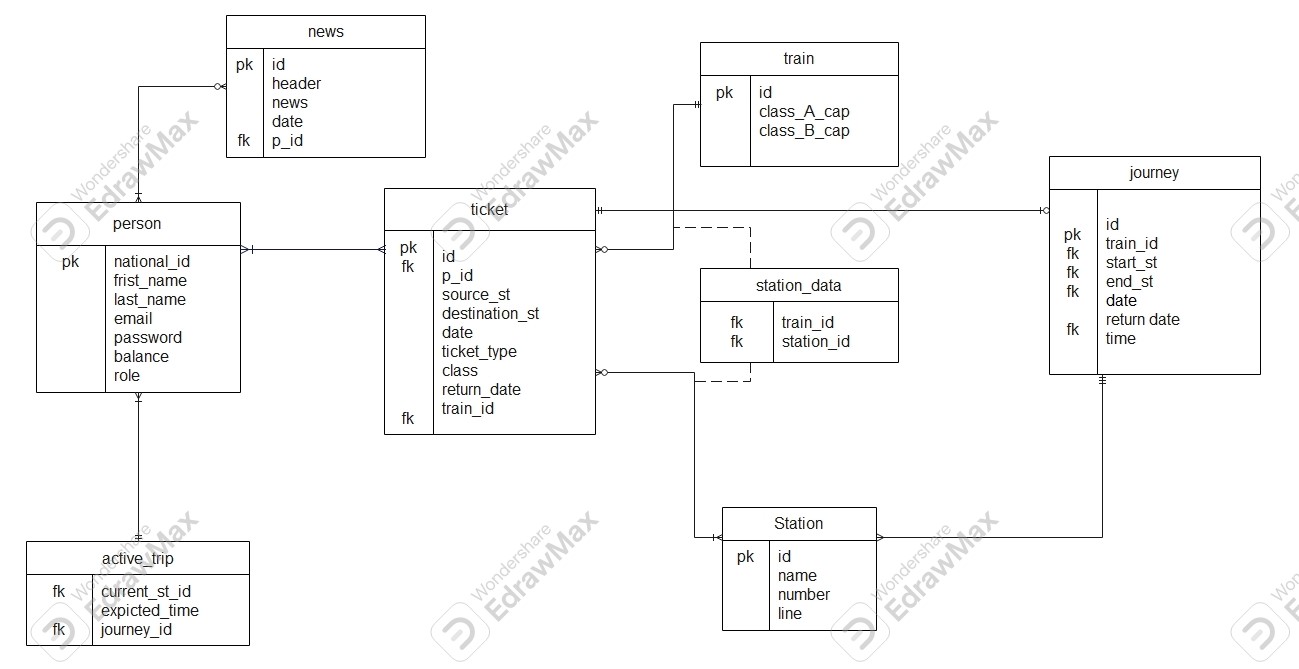
|  |  |
| --- | --- |
| Name | Update station statues |
| Goal | Check when the train leave the station or add delay time if exist |
| Precondition | Enters the delay if exist, check when the train leaves |
| Postcondition | Update the station place |
| Error state | no train in the station |
| Actors | admin station |
| Trigger | Leave button |
| Standard process | Train enters the station add see if there is delay then press leave when the train leaves the station |
| Alternative Processes | No train at the moment to leave |

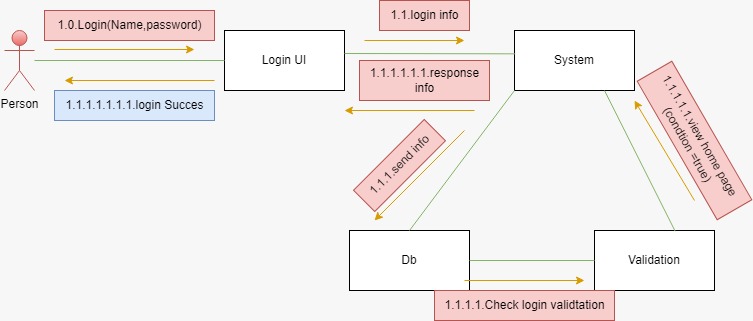
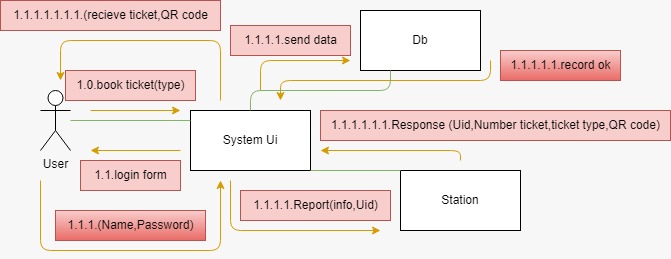
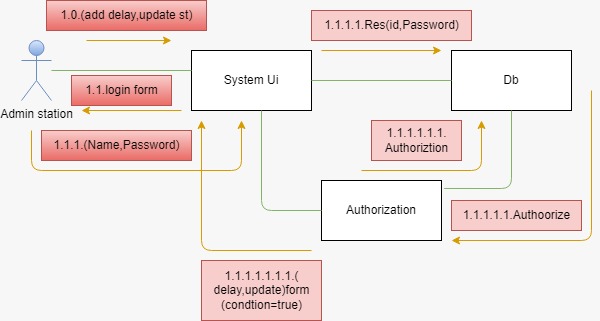
**Class diagram)**

**Erd diagram)**

****

**Table diagram)**

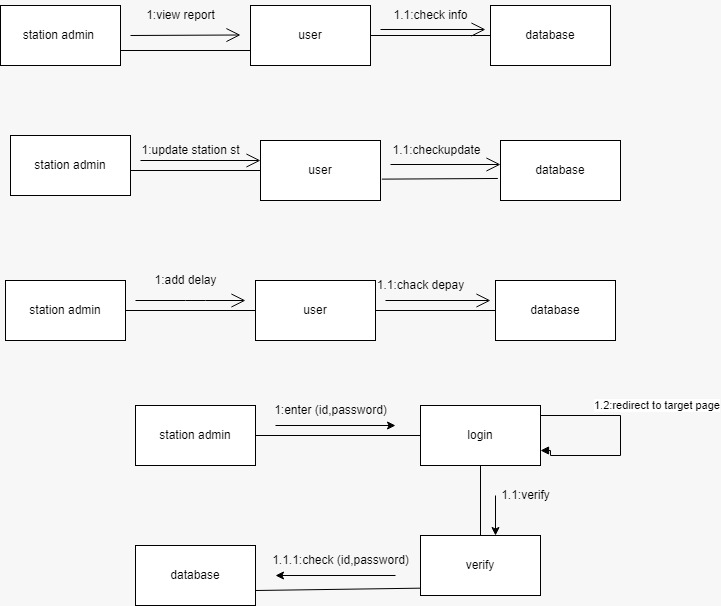
****

**Collaboration diagram)**

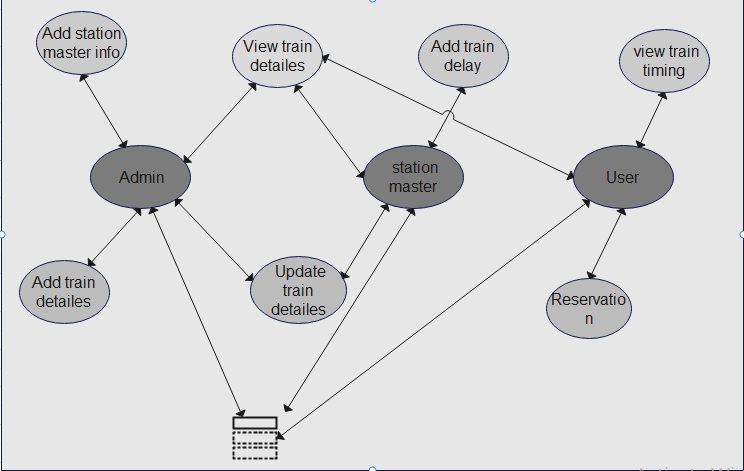
**Communication diagram) A diagram of a company

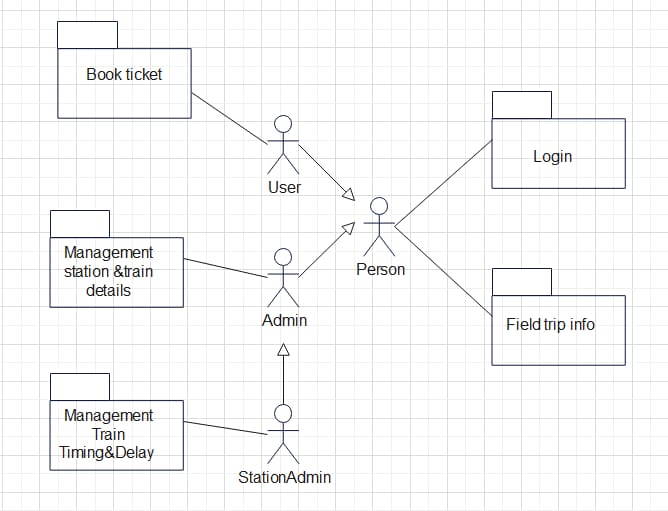
Description automatically generated with medium confidenceA diagram of a login

Description automatically generated**

****

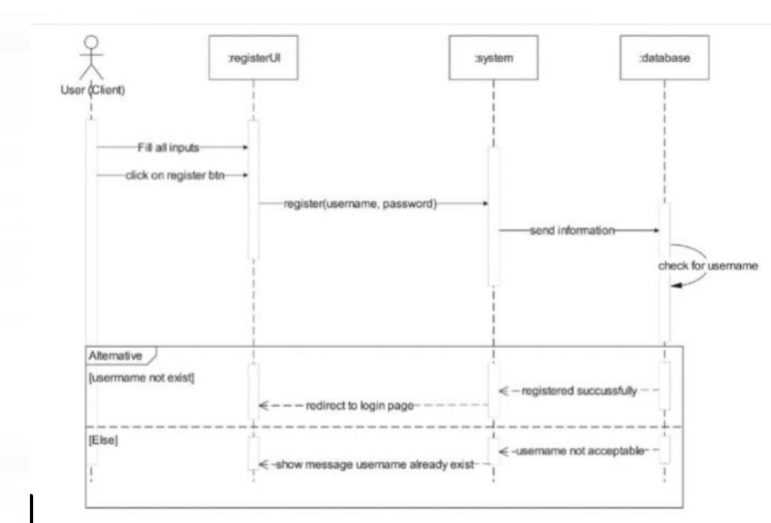
**System Architecture)**

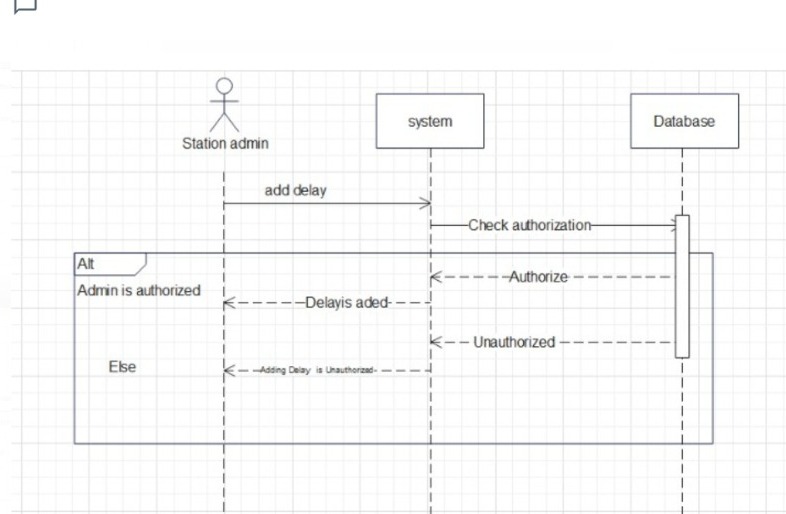
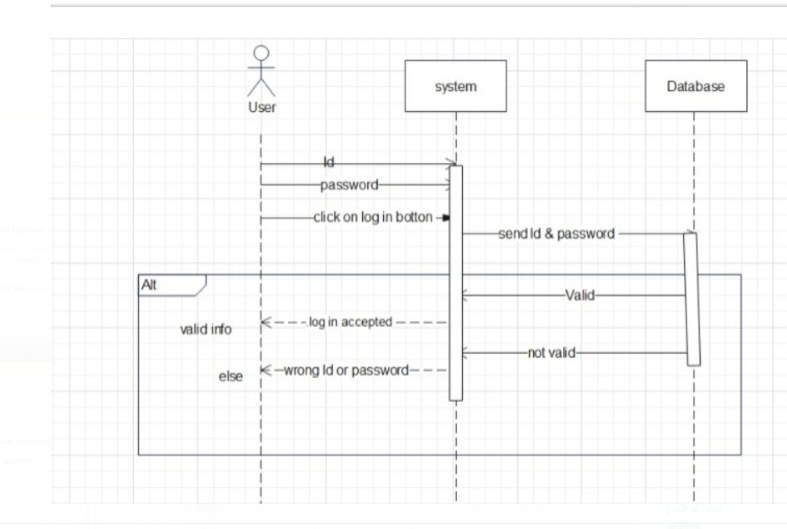


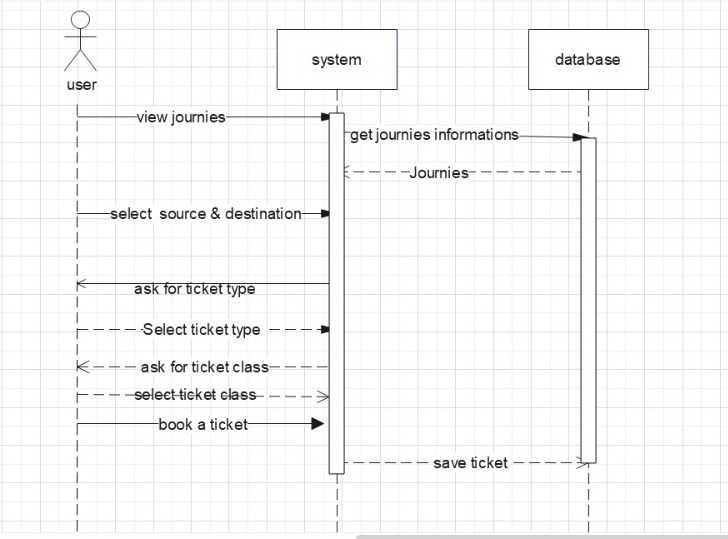
**Package diagram)**

**Sequence diagram)**

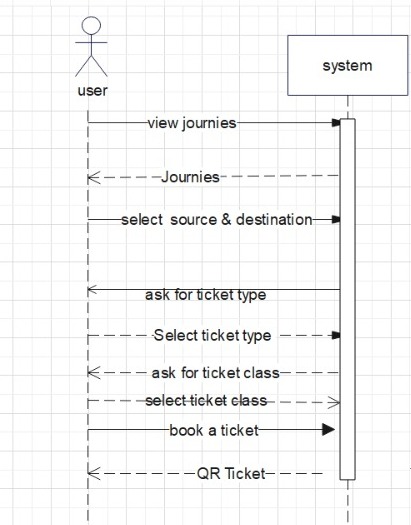
**A diagram of a system

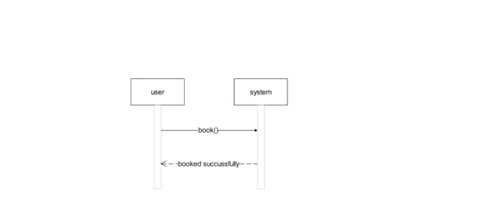
Description automatically generated**

****

****

**SSD)**

****



A diagram of a diagram

Description automatically generatedA diagram of a computer

Description automatically generated

A diagram of a diagram

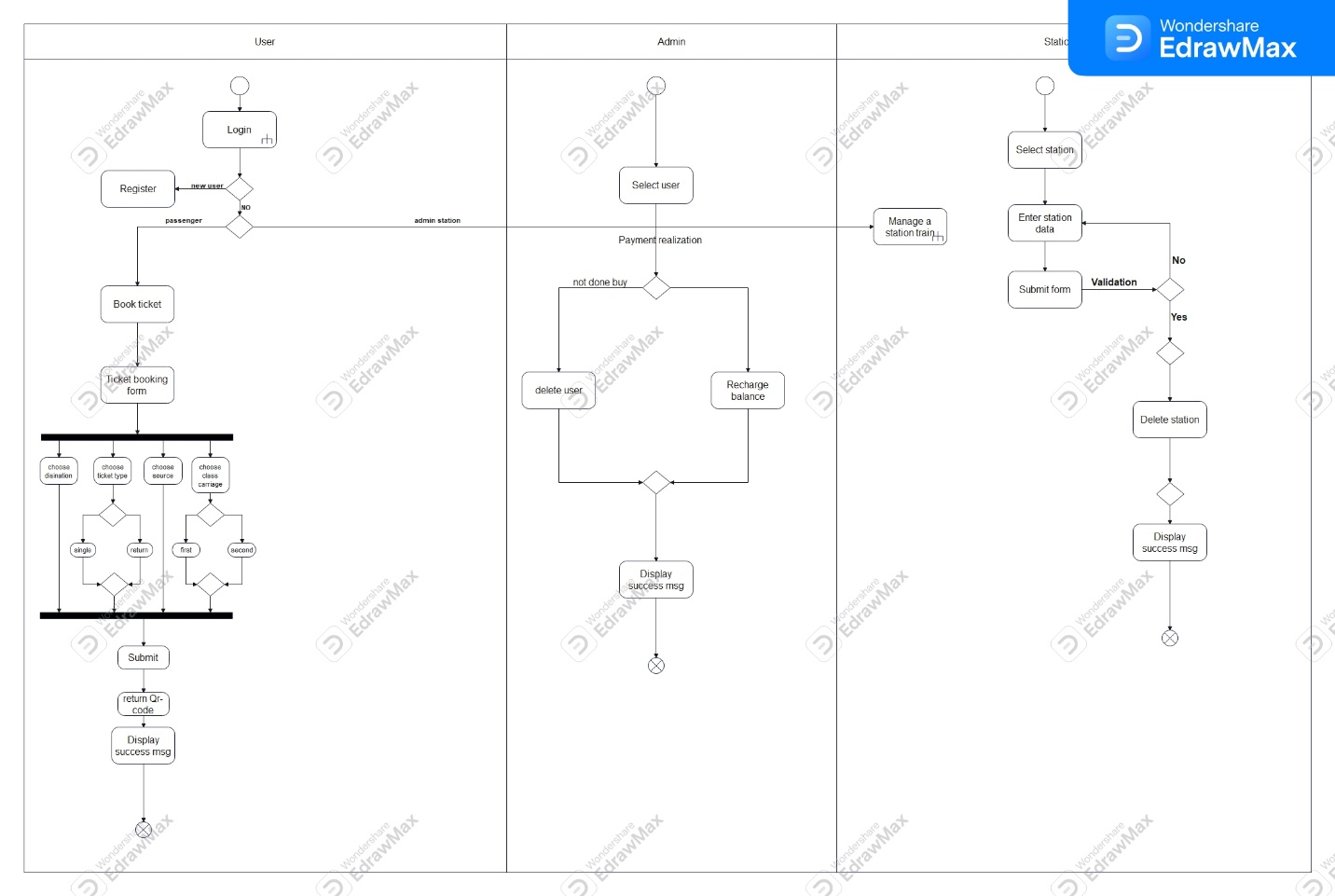
Description automatically generated

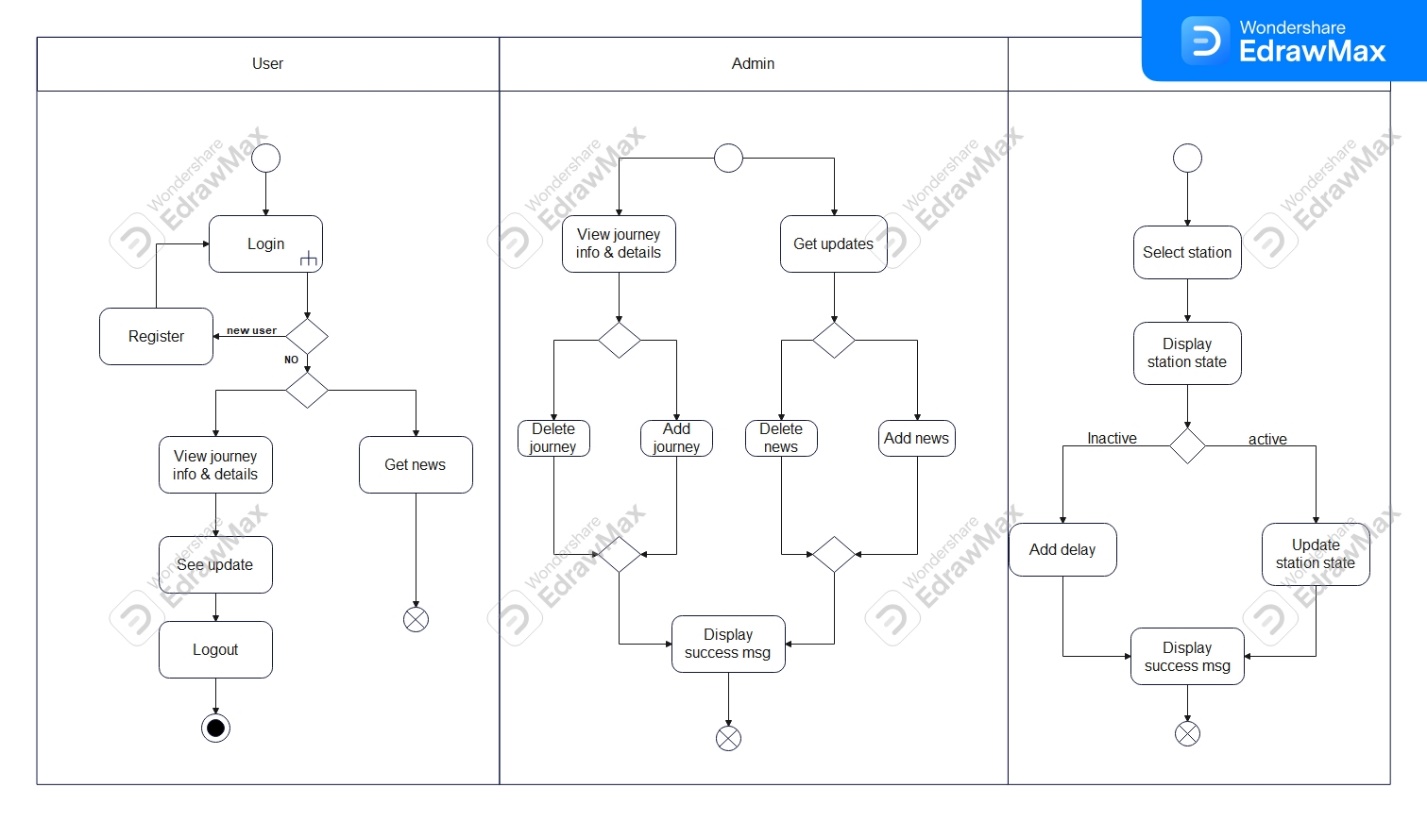
A diagram of a computer program

Description automatically generated with medium confidence

A diagram of a diagram

Description automatically generated

**Activity diagram) **

****

# **Design Pattern**

#### (SINGLETON PATTERN)

**Type: Creational Design Pattern**

**Name: Singleton Pattern**

**Context:**

• only one of login instance should exist .

#### **Problem**:

• ensure that it is never possible to create more than one instance of login class.

### **Forces**:

• The use of a public constructor cannot guarantee that no more than one instance will be created .

• The login instance must also be accessible to all classes that require it, therefore it must often be

public .

##### **Solution**:

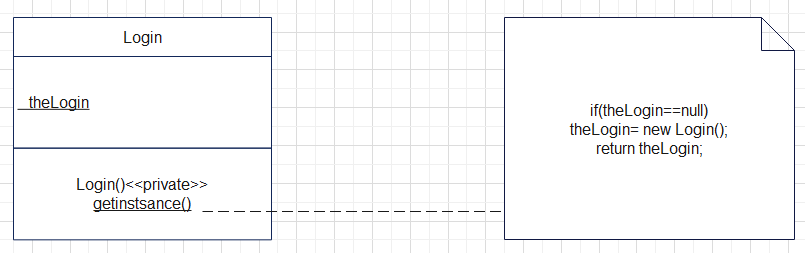
• Have the constructor private to ensure that no other class will be able to create an instance of the class

Login .

• Define a public static method, The first time this method is called , it creates the single instance of the

class “login ” and stores a reference to that object in a static private variable .

Diagram:

****

# **Type:** **Creational Design Pattern**

**Name: Singleton Pattern**

**Context**

• only one of Registration instance should exist .

#### **Problem**:

• ensure that it is never possible to create more than one instance of Registration class. And provide a global

point of access to it .

**Forces**:

• The use of a public constructor cannot guarantee that no more than one instance will be created .

• The Registration instance must also be accessible to all classes that require it, therefore it must often be

public .

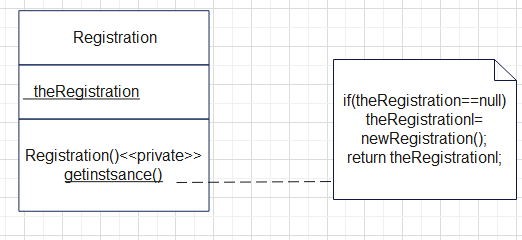
##### **Solution**:

• Have the constructor private to ensure that no other class will be able to create an instance of the class Registration.

• Define a public static method, The first time this method is called , it creates the single instance of the

class “Registration” and stores a reference to that object in a static private variable .

Diagram:

****

**Type : Creational design patterns**

**Name : Immutable pattern**

**Context** :

Login class don't change after creation

**Problem** :

* ensure that once a login class is created, its properties can't change

**Forces :**

* Take into account that if the class use inheritance property for this use final

**Solution**:

* Make all properties of login class private and read only , meaning they can only be set during object creation
* Don't provide any setter method for the properties . once the object is created its properties can't be modified
* Provide getter method for all properties to allow access to their values.

A diagram of a computer program

Description automatically generated with medium confidenceclass diagram: pattern

Object diagram

A diagram of a computer

Description automatically generated

A diagram of a train

Description automatically generatedClass diagram package