كلية الحاسبات و الذكاء الاصطناعي
جامعة حلوان

Software Requirements Specification
of
Railway tracking and arrival time prediction application

Version 1.0

Level 2

Presented to

Prof. Dr. Amr S. Ghoneim

Academic year - 2021/2022

Software Requirements Specification
of
<<Railway tracking and arrival time prediction application >>

Version 1.0
April 29, 2022

Web Publishing System

railway tracking
9360
9360
CP
4689
4689
KCS

THIS FORM IS FOR BOTH THE GENERAL & MEDICAL INFORMATICS PROGRAMMES

SE-I COURSE PROJECT (PHASE 1 COVER SHEET)

Discussions Scheduled for Week 10 *(more details will be announced later)***.**

- Print 1 copy of this cover sheet and attach it to a printed copy of the documentation *(SRS, … etc.)*. You must also submit softcopies of all your documents *(as PDFs)*; details will be announced later.
- Please write all your names in Arabic.
- Please make sure that your students' IDs are correct.
- Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
- Please attend the discussion on time *(announced separately)*, late teams will lose 5 grades.

Project Name:A Railway Tracking and Arrival Time Prediction Application

Team Information ***(typed not handwritten, except for the attendance signature)*****:**

	ID **[Ordered by ID]**	**Full Name** **[In Arabic]**	**Attendance** **[Handwritten Signature]**	**Final Grade**
1	202000458	شيماء مدحت احمد		
2	202000664	كريم سلامه عبدالملك		
3	202000826	محمد نشأت سيد		
4	202000890	مصطفى احمد صبري		
5	202001077	يوسف اشرف مراد		
6	202001121	نورالدين جمال عبدالعزيز		
7				

Grading Criteria:

10 Items		**Grade**	**Notes**
1. Functional Requirements	**1**		
2. Non-Functional Requirements	**1**		
3. Use-Case Diagram(s) *including general use-cases for the system, and the detailed use-cases description*	**2**		
4. System Architecture – *including applied Architectural Pattern(s)*	**1**		
5. Activity Diagrams	**2**		
6. Database Specification *(ERD, Tables)*	**2**		
7. Class Diagram *(Interfaces, Classes, Relations)* – An initial version based on the requirements and Use-Case/Activity diagrams.	**2**		
8. Object Diagrams *(Including object diagrams that illustrate the preconditions and the post-conditions of selected functions)*	**1**		
9. Package Diagram(s)	**1**		
10. Sequence Diagrams *including* ***System Sequence Diagrams (SSDs)***	**2**		

Teaching-Assistant's Signature: ______________________

$$\frac{\quad}{15}$$

Contents

Subject **page**

1 Introduction

The following section of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references, and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete **Railway tracking and arrival time prediction application.**

1.1 Purpose

The purpose of to determine both functional and non-functional requirements of Counseling System also the document provide as overall description about the **Railway tracking and arrival time prediction application** with UML. The basic purpose of Software Requirement Specification (SRS) is to bridge this communication gap. SRS is the medium through which the client's and the user's needs are accurately specified; indeed, SRS forms the basis of software development. Another important purpose of developing an SRS is helping the clients understanding their own needs. An SRS establishes the basis for agreement between the client and the supplier on what the software product will do.

1.2 Product Scope

Railway tracking and arrival time prediction application is an attempt to simulate the basic concepts of an online Reservation system. The system enables to perform the following functions:

- SEARCH FOR TRAIN

- BOOKING OF A SELECTED FLIGHT
- PAYMENT
- CANCELLATION
- Freight Revenue enhancement
- Passenger Revenue enhancement
- Improved & optimized service

References

1- https://youtu.be/JTgLubxq6LI

2- https://youtube.com/playlist?list=PL4mqzqquSRgaQzbR_MiVNzVI0UHeJrS9q

3- https://fdocuments.in/document/srs-for-railway-reservation-system.html

4- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

5- https://nix-united.com/blog/10-common-software-architectural-patterns-part-1/

6- https://www.freeprojectz.com/dfd/railway-reservation-system-dataflow-diagram

2 Overall Description

This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company. It further contains a list of the stakeholders and users of the proposed solution. It also illustrates the needs and wants of the stakeholders that were identified in the brainstorming exercise as part of the requirements workshop. It further lists and briefly describes the major features and a brief description of each of the proposed system.

2.1 Product Perspective

The existing system is highly manual involving a lot of paperwork and calculation and therefore may be erroneous. This has led to inconsistency and inaccuracy in the maintenance of data. The data, which is stored on the paper only, may be lost, stolen, or destroyed due to natural calamity like fire and water. The existing system is sluggish and consumes a lot of time causing inconvenience to customers and the airlines staff. Due to manual nature, it is difficult to update, delete, add, or view the data.

2.2 Product Functions

Booking agents with varying levels of familiarity with computers will mostly use this system. With this, in mind, an important feature of this software is that it be relatively simple to use. The scope of this project encompasses:

Search: This function allows the booking agent to search for train that are available between the two travel cities, namely the "Departure city" and "Arrival city" as desired by the traveler. The system initially prompts the agent for the departure and arrival city, the date of departure, preferred time slot and the number of passengers. It then displays a list of train available with different airlines between the designated cities on the specified date and time.

Selection: This function allows a particular train to be selected from the displayed list. All the details of the train are shown:

1. train Number
2. Date, time, and place of departure
3. Date, time, and place of arrival
4. TRAIN Duration
5. Fare per head

Review: If the seats are available, then the software prompts for the booking of train. The train information is shown. The total fare including taxes is shown and flight details are reviewed.

Traveler Information: It asks for the details of all the passengers supposed to travel including name, address, telephone number and e-mail id.

Payment: It asks the agent to enter the various credit card details of the person making the reservation.

1. Credit card type
2. Credit card number
3. CVC number of the card
4. Expiration date of the card
5. The name on the *card*

Cancellation: The system also allows the passenger to cancel an existing reservation. This function registers the information regarding a passenger who has requested for a cancellation of his/her ticket. It includes entries pertaining to the train No., Confirmation No., Name, Date of Journey, Fare deducted.

2.3 User Characteristics

EDUCATIONAL LEVEL: At least users of the system should be comfortable with the English language.

TECHNICAL EXPERTISE: User should be comfortable using general-purpose applications on the computer system.

2.4 Constraints

The system will run under windows98 or higher platforms of operating system.

2.5 Assumptions and Dependencies

- Booking Agents will be having a valid username and Password to access the software.
- The software needs the booking agent to have complete Knowledge of railways reservation system.
- Software is dependent on access to the internet.

3 System Features

This system is basically concerned with the reservation and cancellation of railway tickets to the passengers. The need of this system arose because the largest railway network handle it manually is quite a tough job. By computerizing it, we will be able to overcome many of its limitations and will be able to make it more efficient. The handling of data and records for such a vast system is a very complex task if done manually but it can be made much easier if the system is computerized. The basic functions being performed by our system are reservation and cancellation. These functions will be handled with the help of following sub functions:

- It reserves and cancels seats for the passenger.
- It contains information about the trains.
- It contains information about the passenger.
- It contains the details of reservation fees, any concessions etc.
- It makes entries for reservation, waiting, cancelled tickets.
- It will update for uptime and downtime trains.

4 System Requirements Specification

4.1 Functional Requirements

User Functional Requirements

- Can create a new personal account with their details including login ID and password.
- Can login to their personal account using the ID and password.
- Can view their own profile.
- Can edit/ update their details like phone, email, and age.
- Can book a ticket.
- Can view the ticket.
- Can select the start day.
- Can select the time of start.
- Can select the end time.
- Can cancel the book
- Can select the degree
- Can select the train type
- Can check if the ticket is available or not
- Can determine if there's an empty place in the train.

Function Name	Registration
Description	Can create a new personal account with their details including login ID and password.
Input	user’s data (Name, DOB, Phone, E-mail, Address, Password).
Output	Send email which prompts that your account created successfully and view their ID and password.

Function Name	login
Description	Can login to their personal account using the ID and password.
Input	ID and password
Output	Open home page.

Function Name	view
Description	Can view their own profile.
Input	user ID.
Output	View information about profile.

Function Name	update

Description	Can edit/ update their details like phone, email, and age.
Input	user ID, password and new data to change.
Output	Show message which prompts that [data] updated successfully

Function Name	book
Description	Can book a ticket.
Input	user name, user id, day, time and place
Output	Show Message Reserved Successfully

Function Name	available trip in day
Description	Can select the start day.
Input	the date, day and time.
Output	Show a message whether or not a trip was available that day.

Function Name	available train at time
Description	Can select the time of start.
Input	time
Output	Show message if a train is available at the scheduled time of travel for that day or not.

Function Name	Degree
Description	Can select the degree
Input	First degree, second degree, third degree, business degree, VIP degree.
Output	Show message confirmation booking in case of empty Spaces.

Function Name	Empty place
Description	can determine if there's an empty place in the train.
Input	date and day,time
Output	Show a message whether or not there's an empty place on the train.

Admin Functional Requirements

- Can login to his personal account using the id and password
- Can login to the system
- Can view their own profile
- Can change their current password with new one whenever required
- Can update their own profile (username, password, phone, email)
- Explore system
- Manage timetable (add, remove, update)
- Manage ticket fare (update, remove)
- Manage train
- Manage route
- Manage station
- Manage garage
- Manage staff
- Manage user

Function Name	Login
Situation	Show error message in case of wrong ID /Password
Input	ID – Password
Output	View all the system

Function Name	View Profile
Situation	Won't have to see any information if you not have account.
Input	ID
Output	view profile

Function Name	Update Profile
Situation	• Show error message if the new password is the same of current password. So, data will not be changed. • Show error message if the new password pattern is wrong. So, data will not be changed.

	• Show error message if the current password not matching with their password. So, data will not be changed. • Show error message if the new data’s pattern not matching the current pattern. So, data will not be changed. • Show error message if the current password not matching with their password. So, data will not be changed.
Input	ID – Password
Output	Show message which prompts that [data] updated successfully.

Function Name	Show Ticket Details
Situation	• Show error message if the user forget to fill in a field of data. • Show message if there are no empty ticket.
Input	All the information about ticket.
Output	show available ticket.

Function Name	Mange Timetable
Situation	Show error message if there are appointments that are inconsistent with the timetable
Input	ID
Output	Show message which prompts that data of timetable (update or remove or add) successfully

Function Name	Mange Trains
Situation	Show error message if the train is in trouble.
Input	ID
Output	Show message which prompts that data of the train managed successfully.

Function Name	Mange Stations
Situation	Show error message if there is an error in data management.
Input	ID

Output	Show message which prompts that data of the (route, garage, station) managed successfully.

4.2 Non-Functional Requirements

Performance Requirements	User Satisfaction: The system is such that it stands up to the user expectations. Response Time: The response of all the operation is good. This has been made possible by careful programming. User friendliness: The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.
Reliability	The reliability of the overall project depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. Also, the system will be functioning inside a container. Thus, the overall stability of the system depends on the stability of container and its underlying operating system.
Security Requirements.	The system uses SSL (secured socket layer) in all transactions that include any confidential customer information. The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's password. The system's back-end servers shall only be accessible to authenticated management.
Supportability	The code and supporting modules of the system will be well documented and easy to understand. Online User Documentation and Help System Requirements.
Availability	The system should be always available, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. A customer friendly system which is in access of people around the world should work 24 hours. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also, in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the Organizer. Then the service will be restarted.

Maintainability	A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the project will be done. Also, the software design is being done with modularity in mind so that maintainability can be done efficiently.
Usability and Humanity requirements	- Centralized maintain all information. - Easy to search the information about college or university. - Facilitate communication between students and counselors. - Easy to manage whole counseling procedure. - Easy to use on the first attempt by someone without training. - The system will save significant amount of time and effort invested by the university every year.

Use Case

Use Case Name	Login
Actors	Admin,System,User
Pre-conditions	Home page is opened.
Description	The actor will open home page,and login if the password is verified
Post-conditions	The actor can do all his operations.
Alternative flow and exceptions	Fail to login and stay in the home page.
Non-functional requirements	Login process must be secure and fast

Use Case Name	logout
Actors	Admin, counselor, and user

Pre-conditions	Actor’s dashboard is opened
Description	The actor will click the logoutbutton, and the home page will be openedinstead of current one.
Post-conditions	The actor cannot do any operation until (s)he login again
Alternative flow and exceptions	Fail to logout as an operation isworking, still in current page
Non-functional requirements	Logout process must be secure and data are safe.

Use Case Name	View profile
Actors	Admin, counselor, and user
Pre-conditions	Actor’s dashboard is opened.
Description	The actor will click the profilebutton, the profile data page will be opened and canedit or update his/her data and password.
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	Fail to logout as an operation isworking, still in current page
Non-functional requirements	Data must be secure

Use Case Name	Add, update, and delete question
Actors	Admin
Pre-conditions	Admin logged in successfully
Description	After logging in, the admin dashboard will be shown and (s)he can select theoperation (s)he wants from the buttons which arealready exist in the dashboard and do the required operation
Post-conditions	The changes must be saved in thedatabase and the admin will return to the dashboard to be able to select another operation to do or logout.
Alternative flow and exceptions	Operation failed, show error message.
Non-functional requirements	All operations' processes must be secure and fast.

Use Case Name	book a ticket
Actors	user
Pre-conditions	Choose the time and date
Description	After login the user can select the time and date to book a ticket

Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If the user keep empty field, show error massage
Non-functional requirements	All operations’ processes must be secure and fast.

Use Case Name	select the start day
Actors	user
Pre-conditions	Choos the day to start the Journey
Description	The user will choos the date to his journey to book the ticket
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If the user keep empty field or entered wrong date , show error massage
Non-functional requirements	All operations’ processes must be secure and fast.

Use Case Name	select the degree
Actors	user
Pre-conditions	User logged in successfully

Description	The user can choose his degree in the trian like
	(first,Second,Second,business and vip)
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If the degree in the carriage Filledshow massage that it is completed
Non-functional requirements	All operations’ processes must be secure and fast.

Use Case Name	safty
Actors	system
Pre-conditions	Admin logged successfully
Description	If the system confirms that there is a tow station and that there is atrain bound to the same destination send to the master of the station
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If the station empty donot send to the masrter
Non-functional requirements	All operations’ processes must be secure and fast.

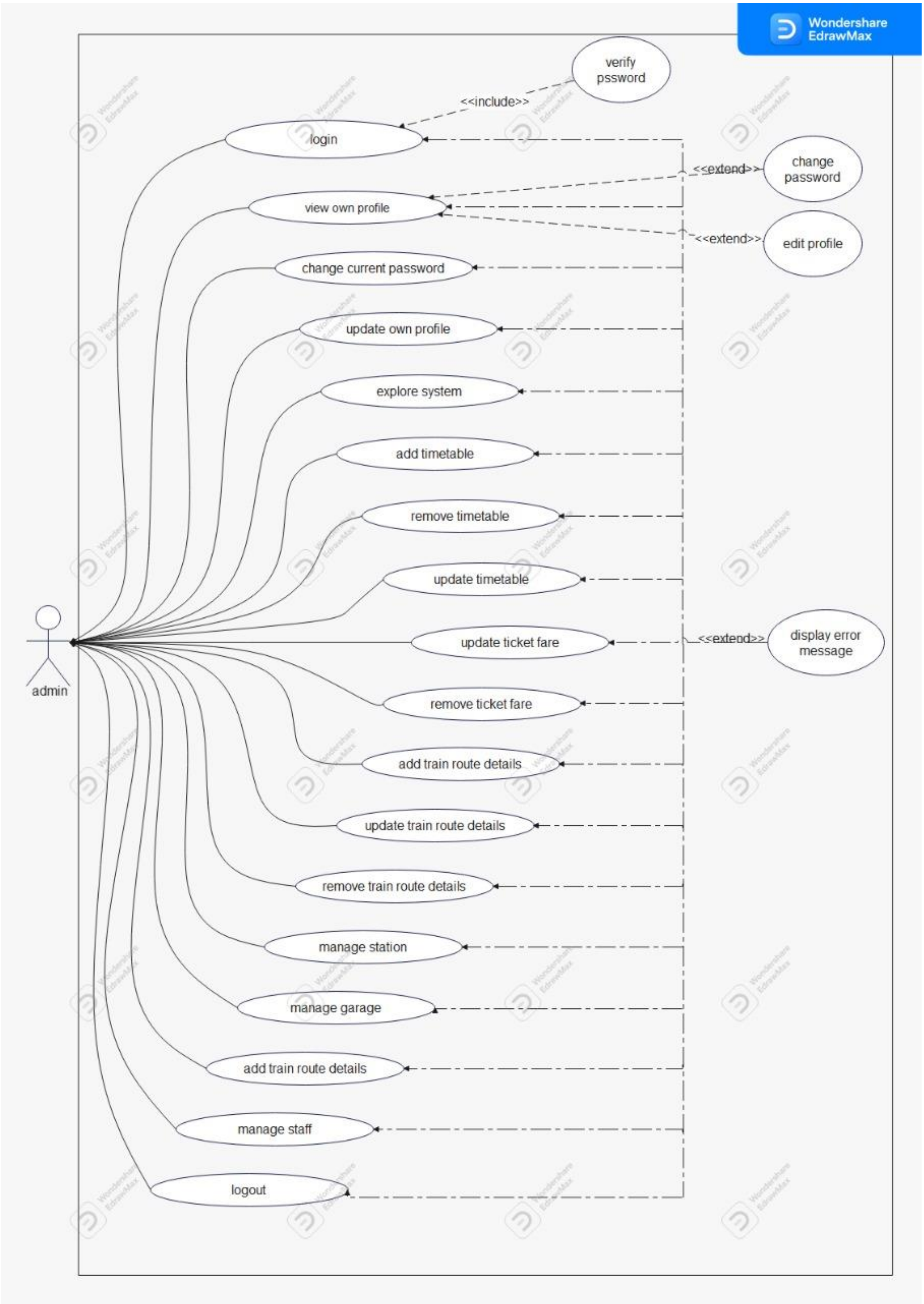
Use Case Name	offers
Actors	admin
Pre-conditions	Admin logged successfully
Description	Admin can Adds trip offerings
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If there are no offers to showmessage, there are no offers available.
Non-functional requirements	All operations' processes must be secure and fast.

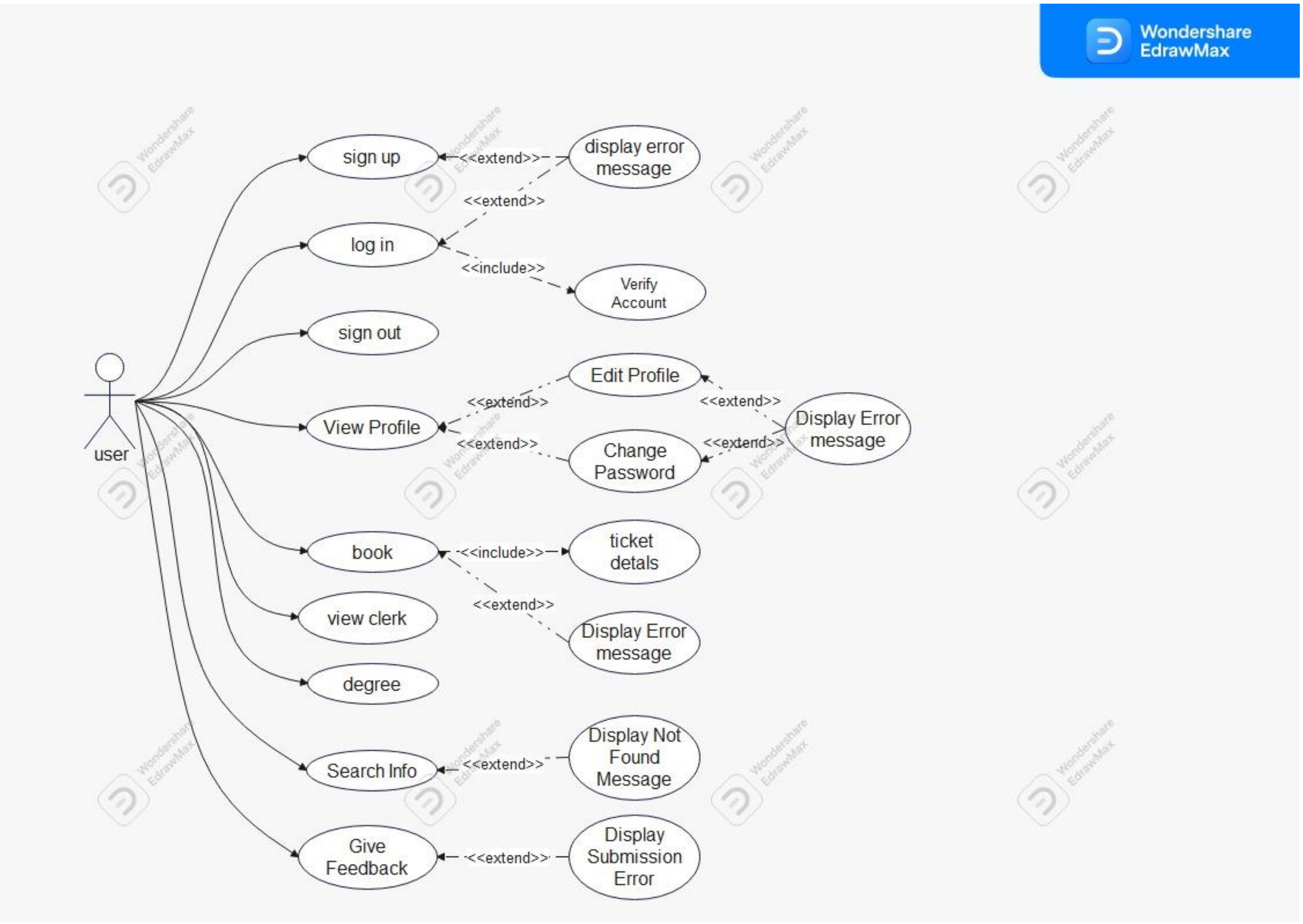
Use Case Name	Train shcedule
Actors	admin
Pre-conditions	Admin logged successfully
Description	Admin can add ,remove and Adjusted the shcedule
Post-conditions	Data must be saved in the database if changed
Alternative flow and exceptions	If the user tries to edit remove shcedule , show error massage
Non-functional requirements	All operations' processes must be secure and fast.

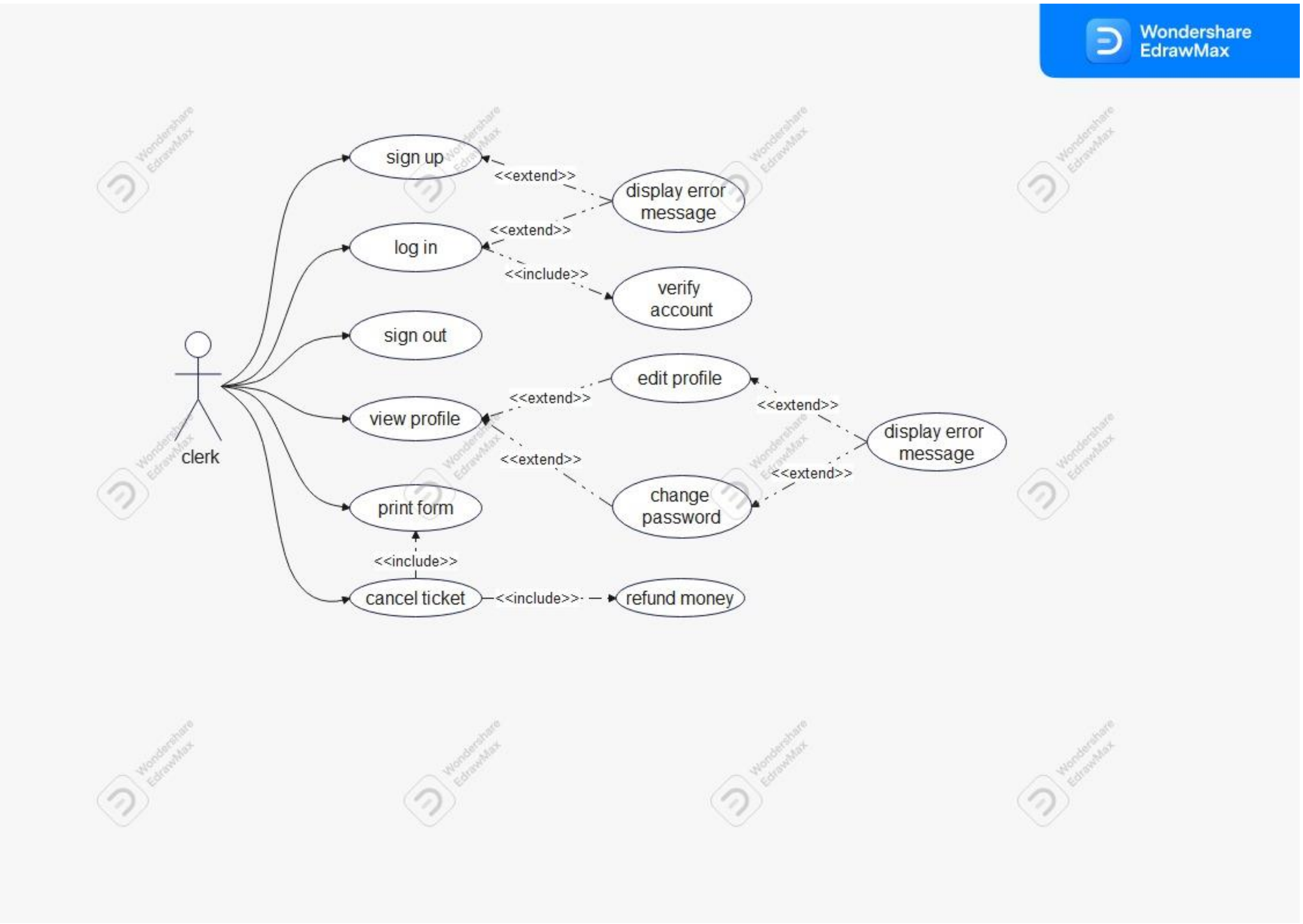
Use Case Name	Contact us
Actors	User,system,admin
Pre-conditions	Actor’s dashboard is opened
Description	User can use to contact in case want help , question, wrong in theJourney equ...... Admin and system can answer the user to help
Post-conditions	Data must be saved in the database if changed and secure
Alternative flow and exceptions	If the user or admin wrote a wrong Id, show error massage
Non-functional requirements	All operations’ processes must be secure and fast

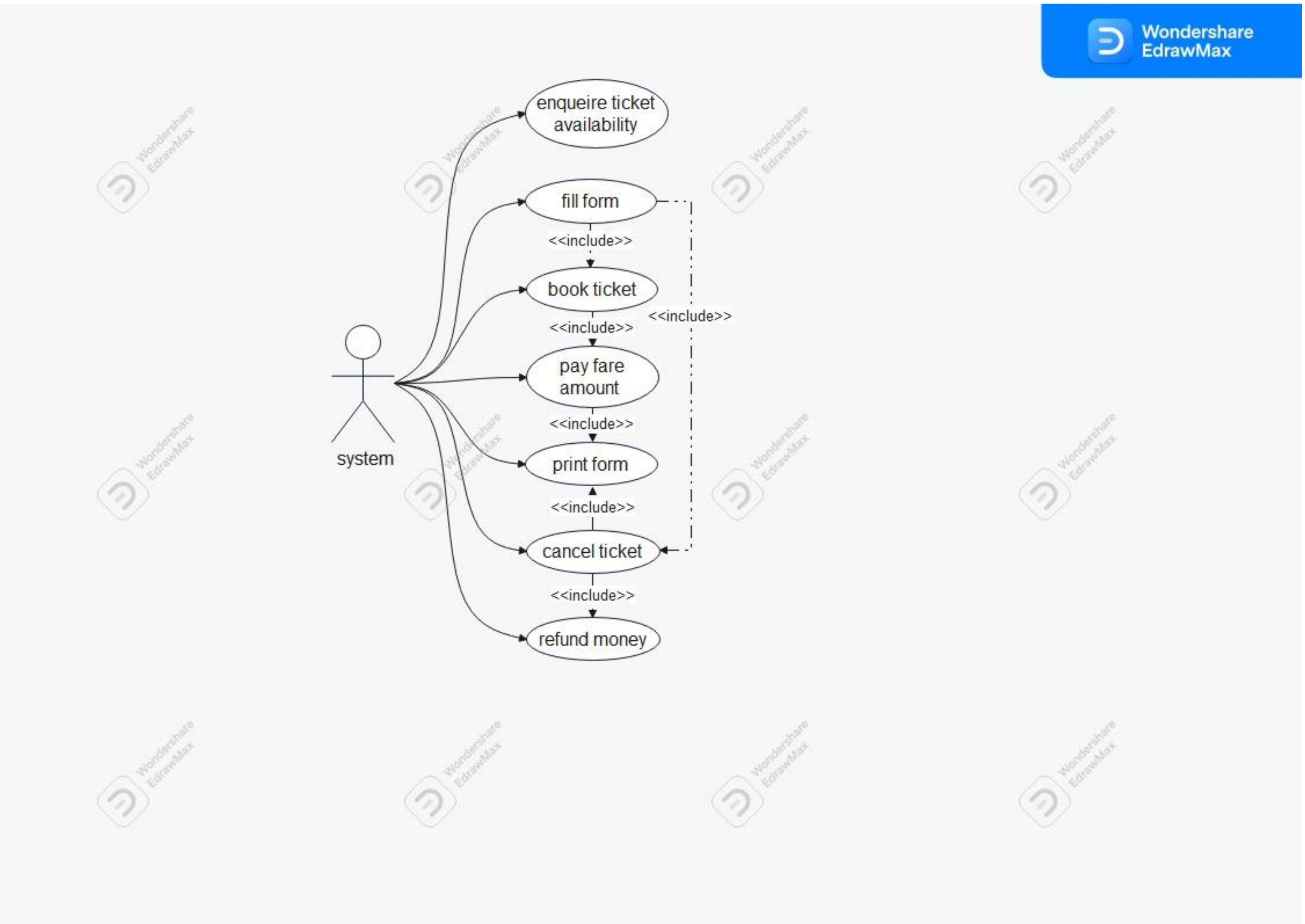
Use Case Name	Number of trains at the station
Actors	system
Pre-conditions	Master of the station logged in
Description	System told to the master of the station how many trains at the
	station to know if station can afford another train.
Post-conditions	Data must shared quickly and aved in the database if changedand secure
Alternative flow and exceptions	In case there's no room at the station for another train, send a message to the station master

Non-functional requirements	All operations’ processes must be secure andfast


verify pssword
<<include>>
login
<<extend>>
change password
view own profile
<<extend>>
edit profile
change current password
update own profile
explore system
add timetable
remove timetable
update timetable
admin
update ticket fare
<<extend>>
display error message
remove ticket fare
add train route details
update train route details
remove train route details
manage station
manage garage
add train route details
manage staff
logout

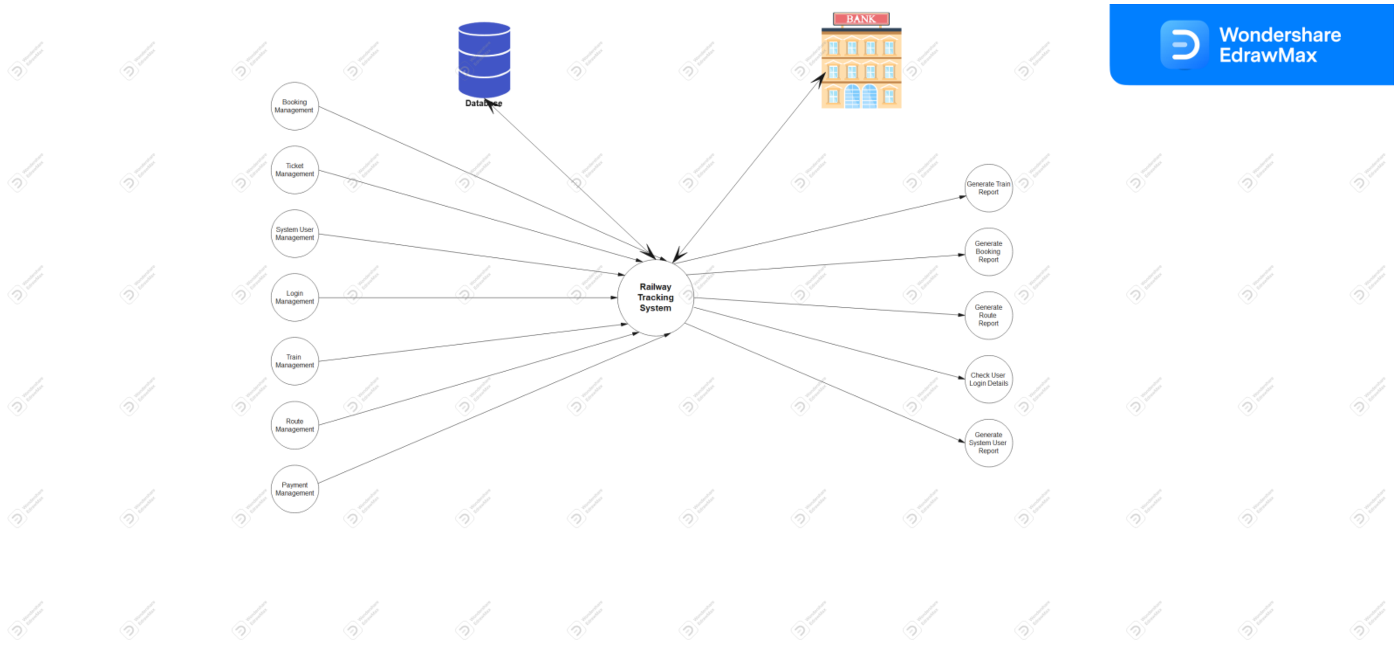

Wondershare EdrawMax
user
sign up
<<extend>>
display error message
<<extend>>
log in
<<include>>
Verify Account
sign out
Edit Profile
<<extend>>
<<extend>>
Display Error message
View Profile
<<extend>>
<<extend>>
Change Password
<<extend>>
book
<<include>>
ticket detals
<<extend>>
Display Error message
view clerk
degree
Search Info
<<extend>>
Display Not Found Message
Give Feedback
<<extend>>
Display Submission Error


Wondershare EdrawMax
clerk
sign up
log in
sign out
view profile
print form
cancel ticket
<<extend>>
<<extend>>
<<include>>
display error message
verify account
edit profile
change password
<<extend>>
<<extend>>
<<extend>>
<<extend>>
display error message
<<include>>
<<include>>
refund money

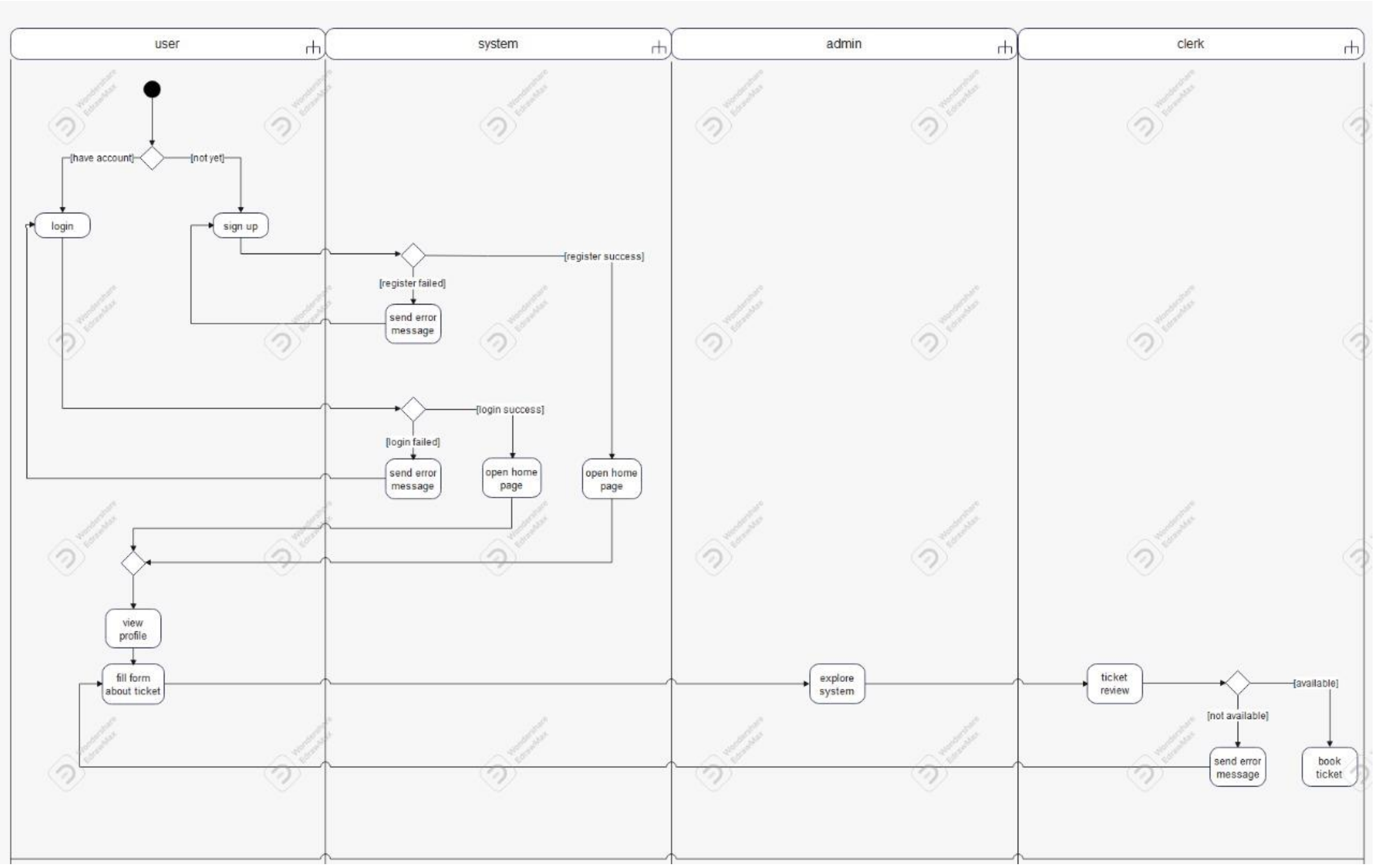


System Architecture

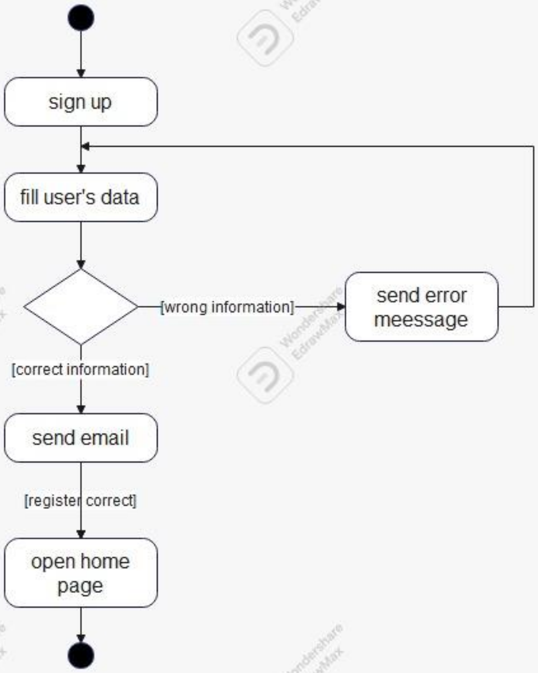
Data-Centred System Archeteture:

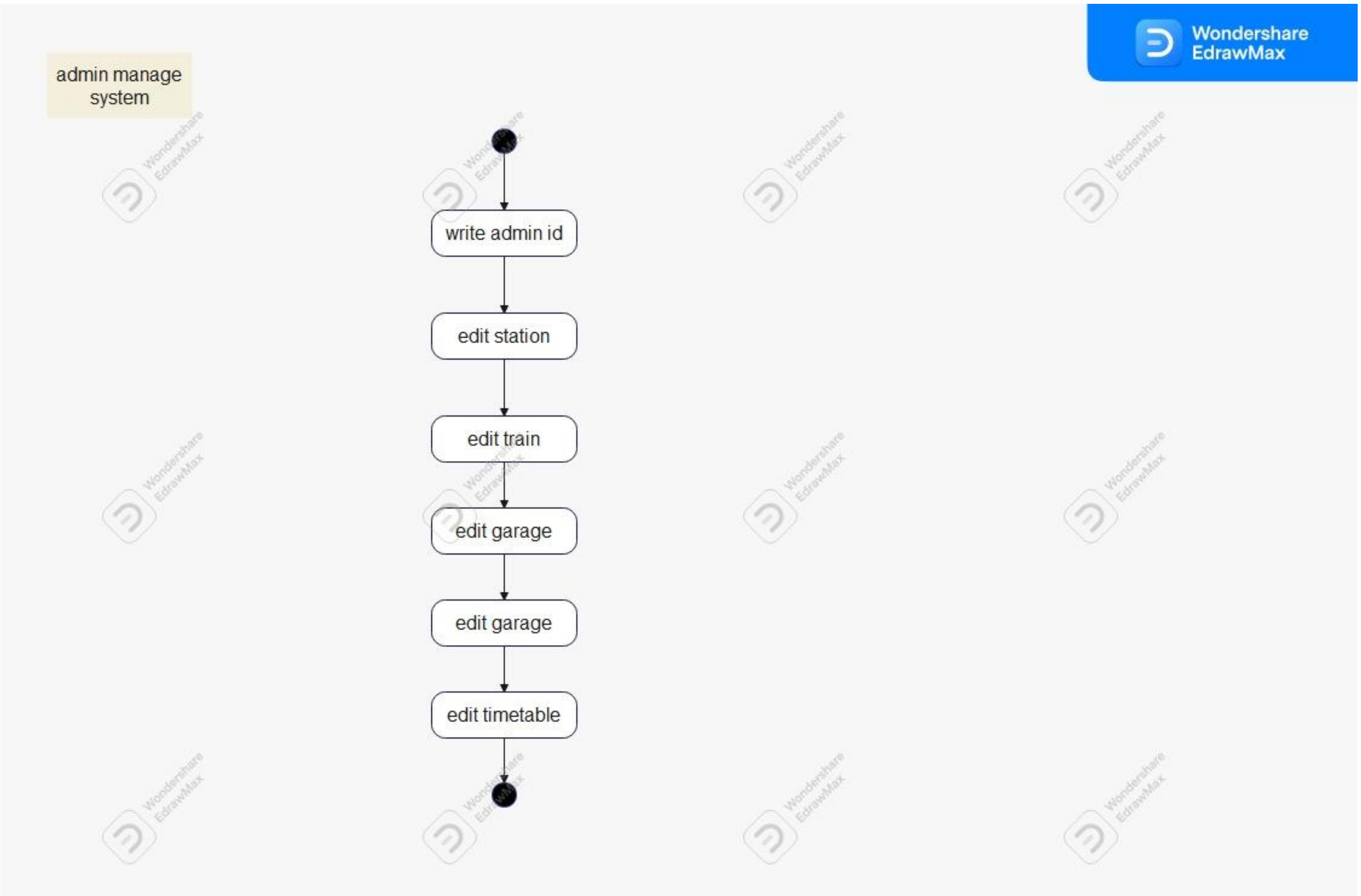


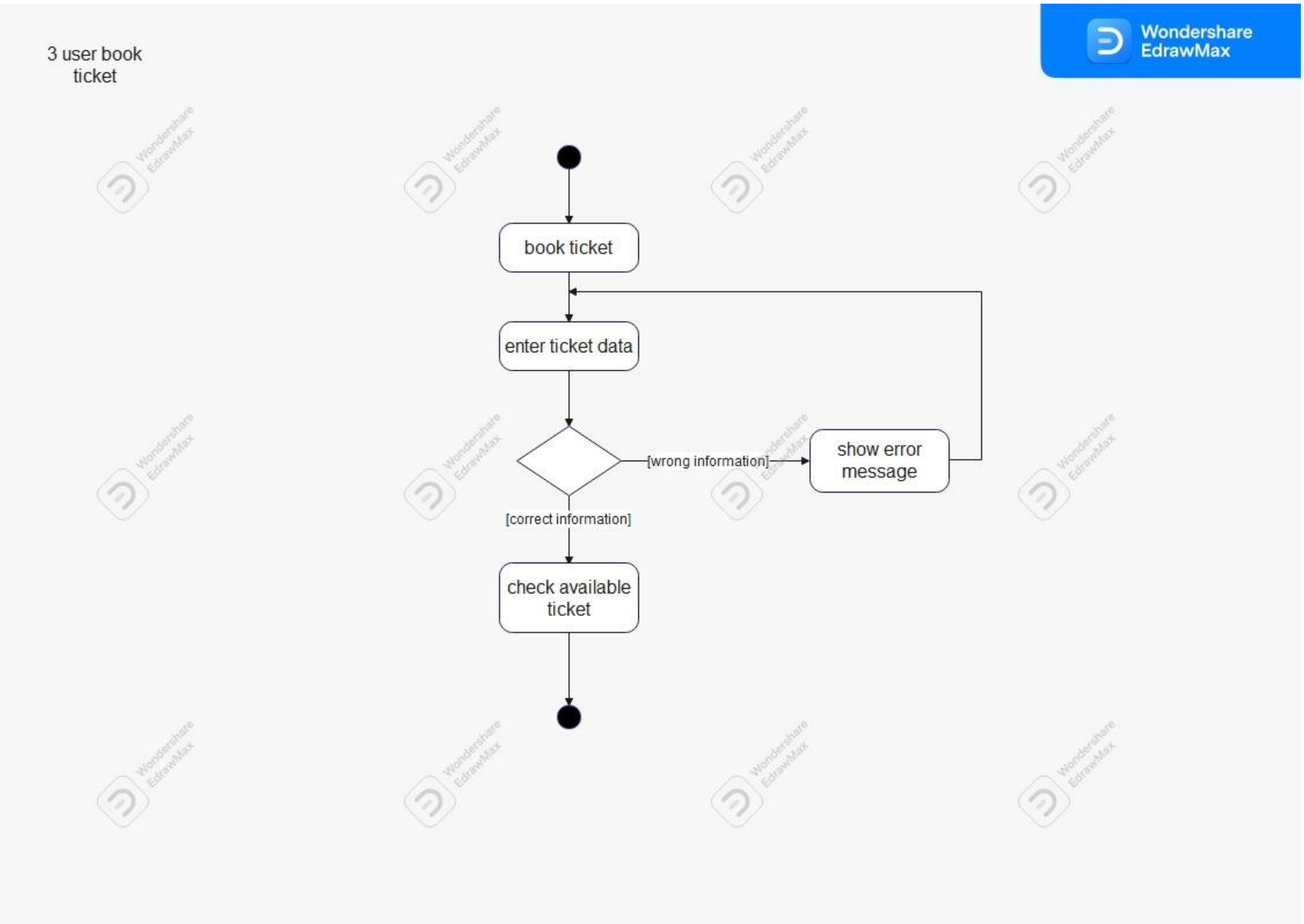
Activity Diagrams

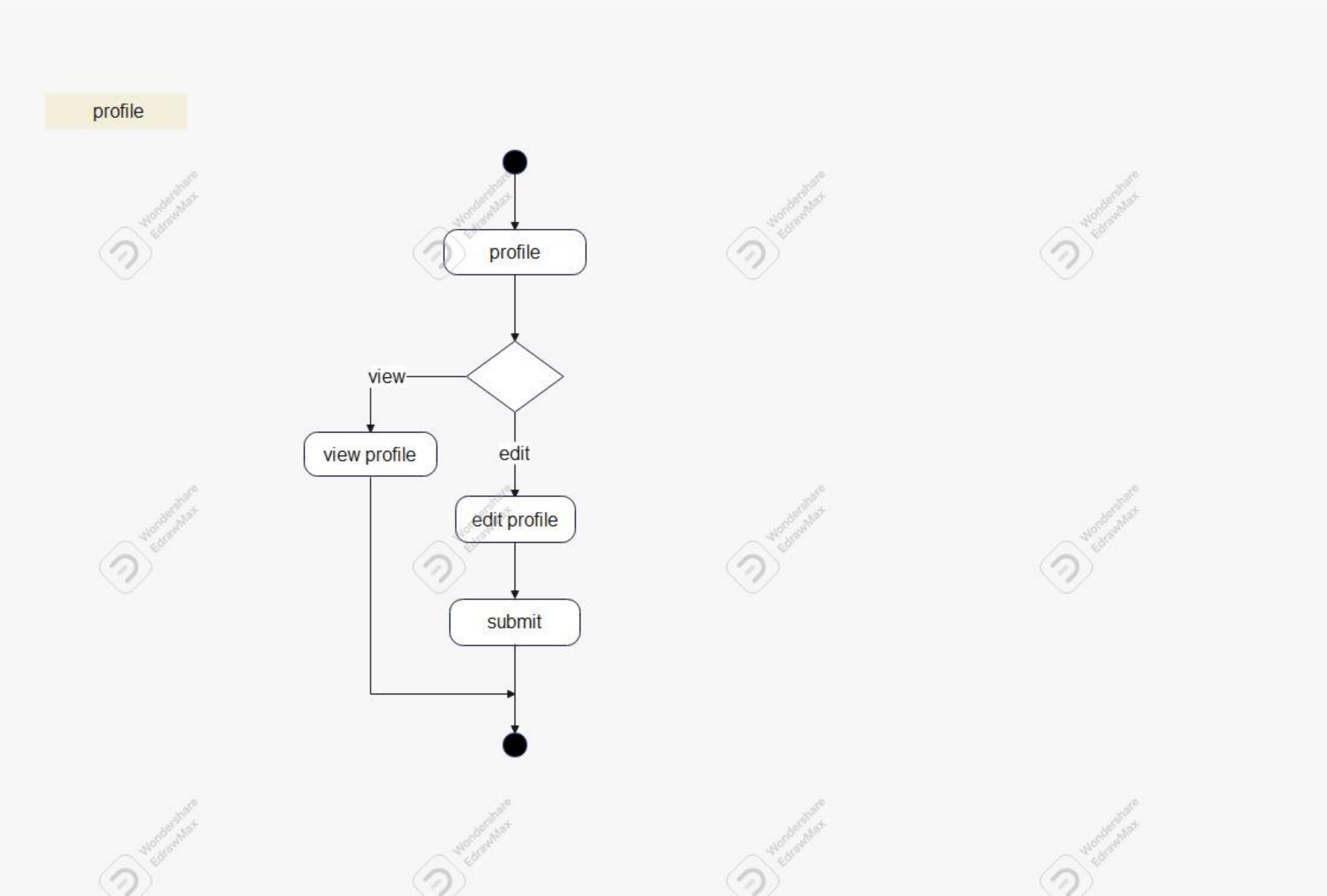


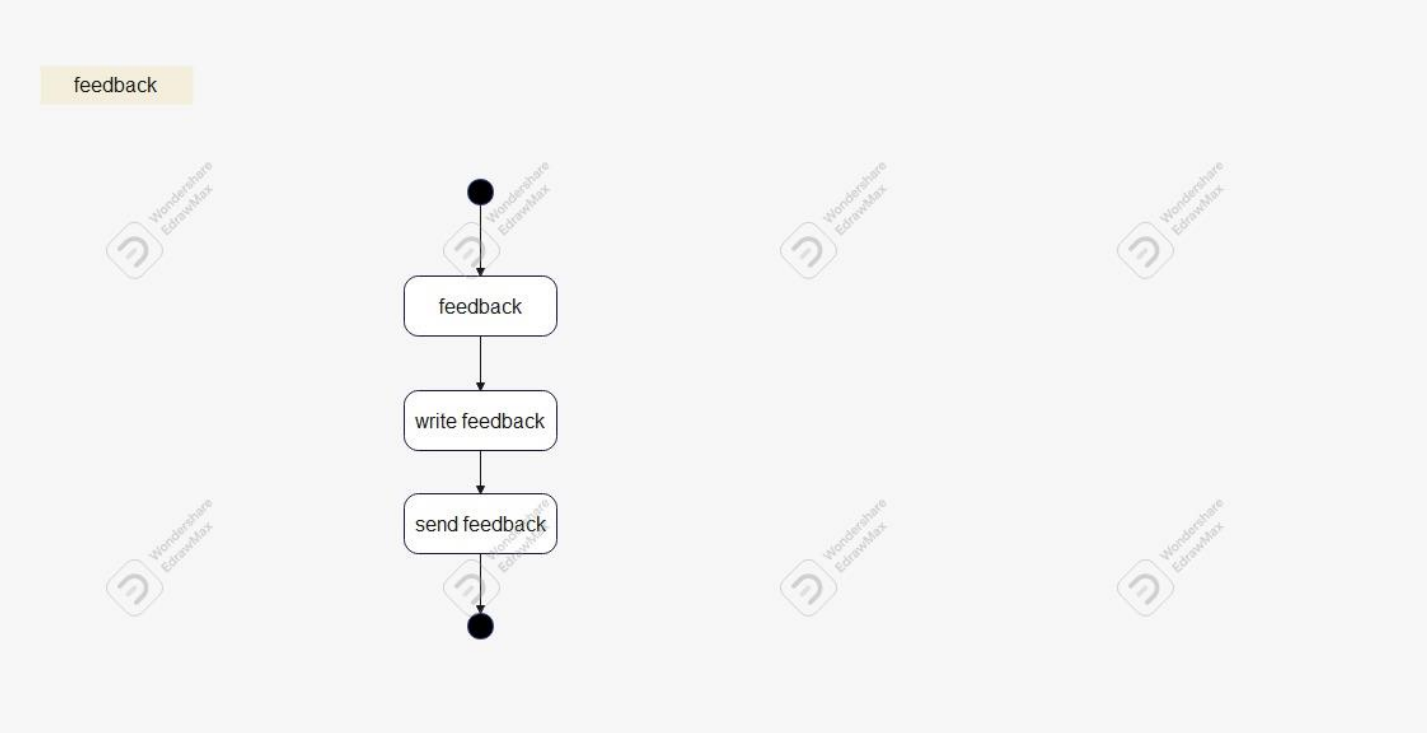
2 sign up user

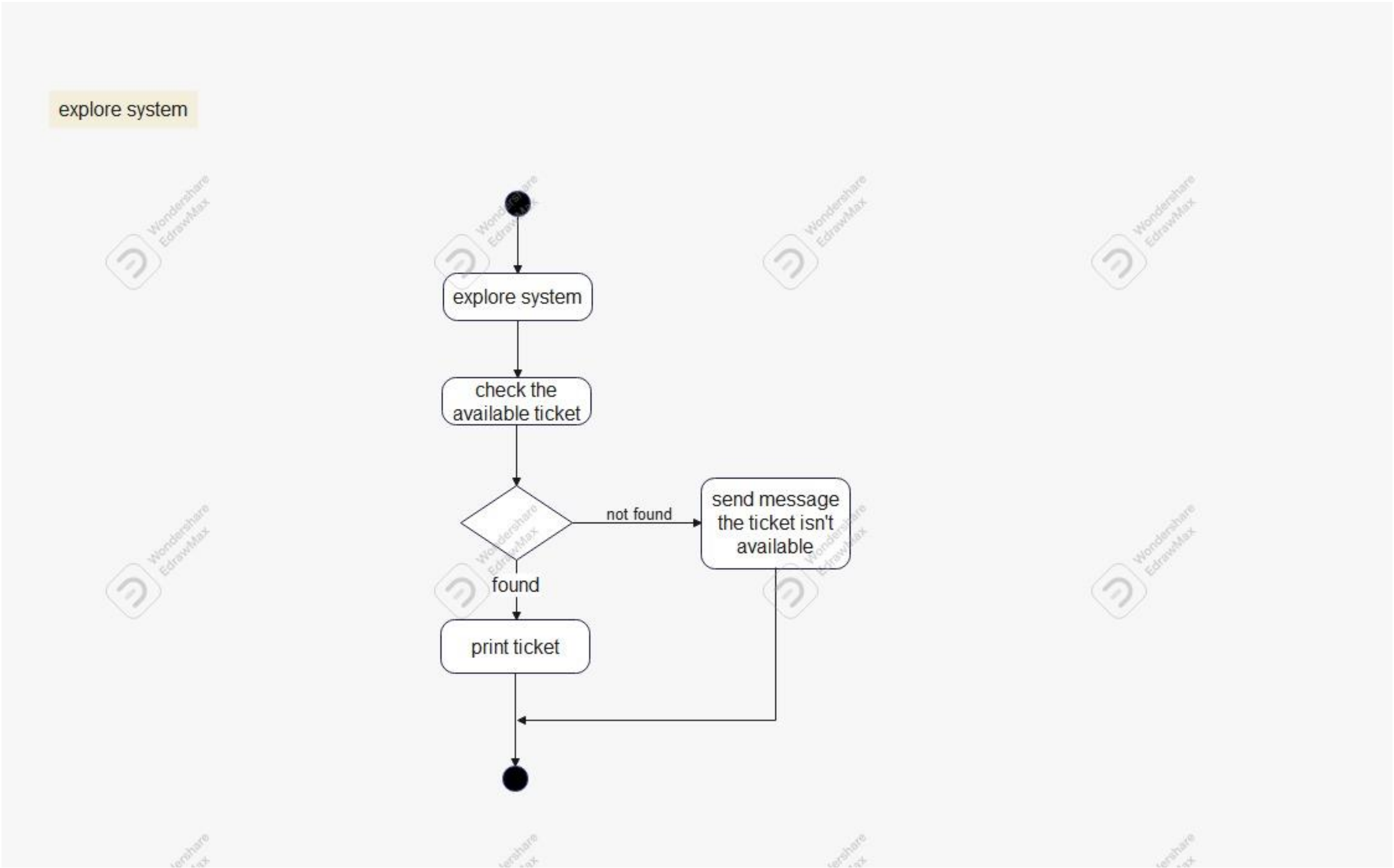


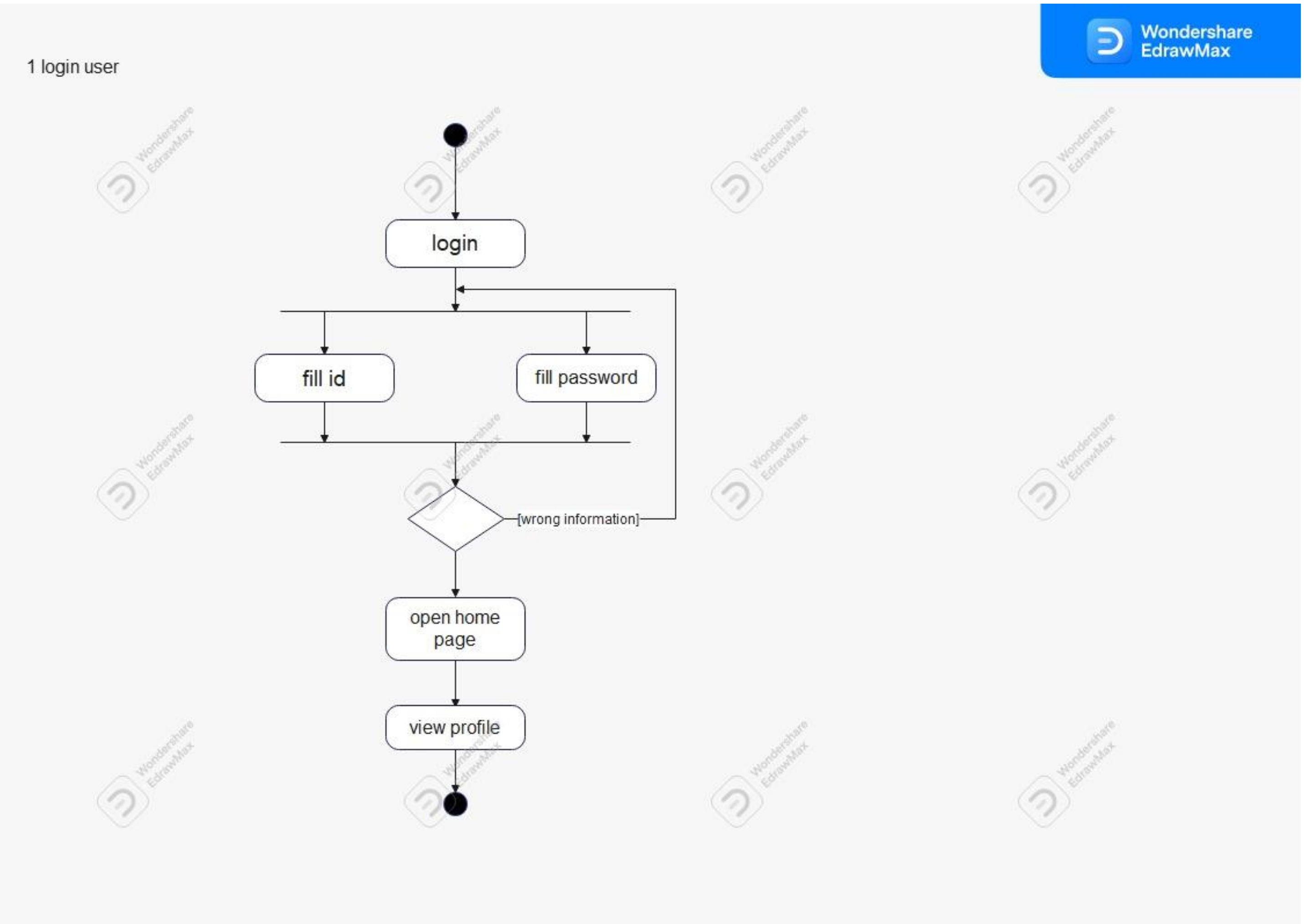

admin manage system
write admin id
edit station
edit train
edit garage
edit garage
edit timetable


3 user book ticket
book ticket
enter ticket data
[wrong information]
show error message
[correct information]
check available ticket

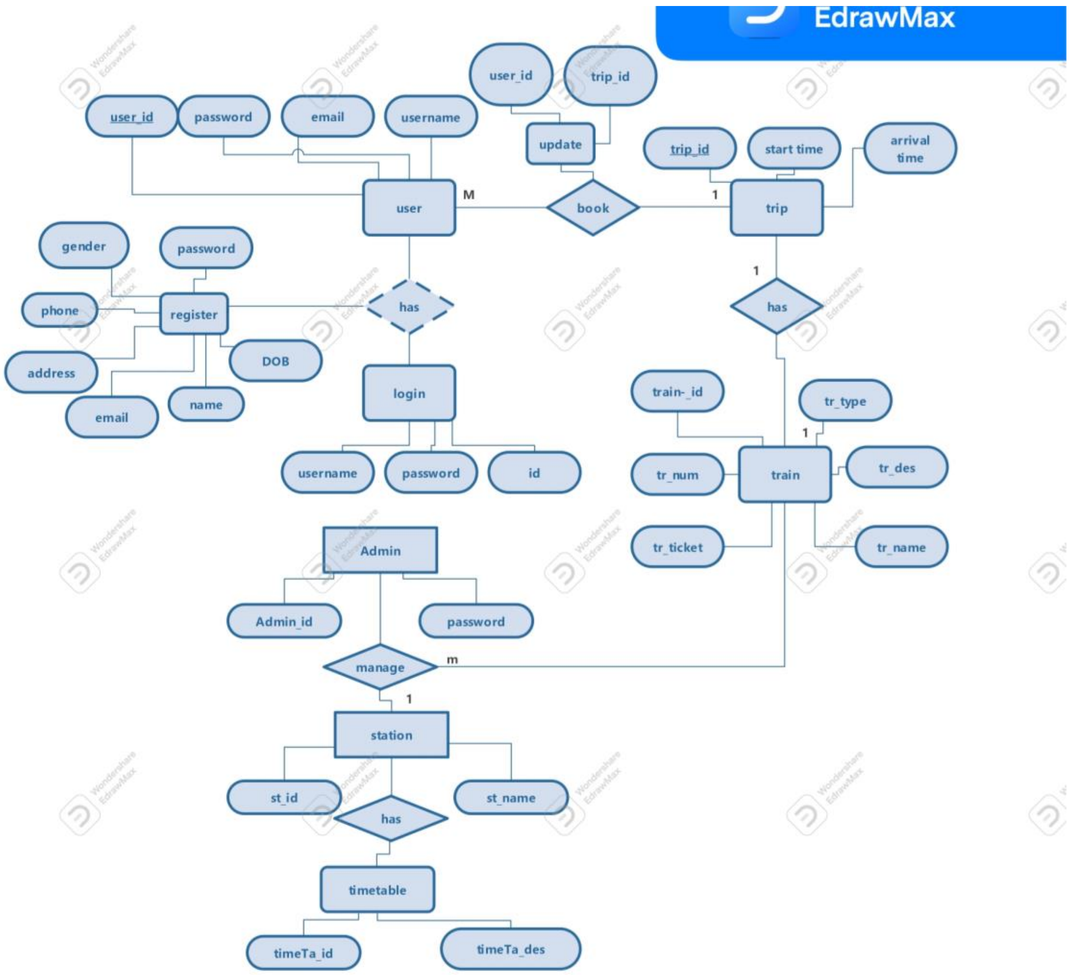

profile
profile
view
edit
view profile
edit profile
submit

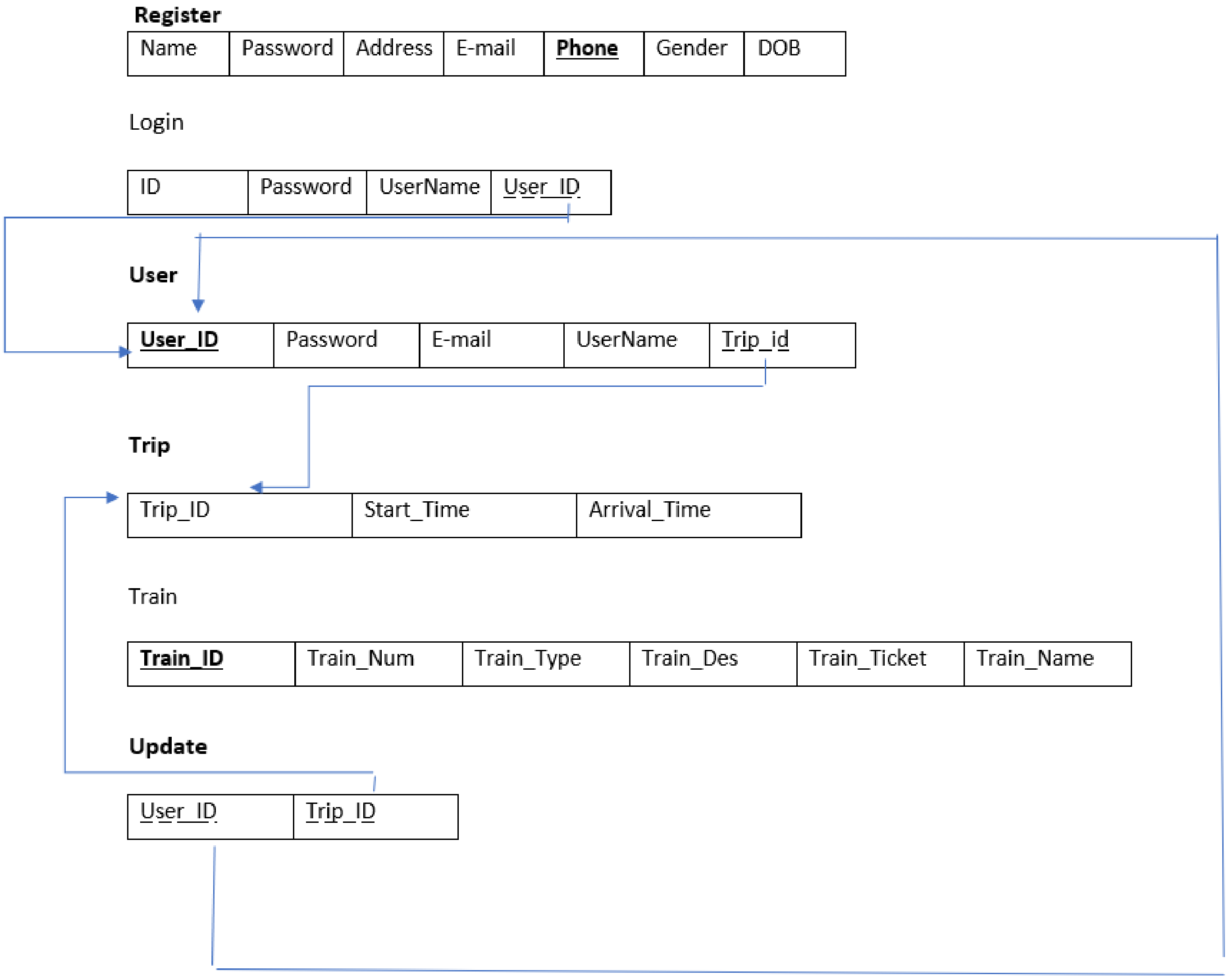

feedback
feedback
write feedback
send feedback


explore system
explore system
check the available ticket
not found
send message the ticket isn't available
found
print ticket



Database Specification


EdrawMax
user_id
trip_id
user_id
password
email
username
update
trip_id
start time
arrival time
user
M
book
1
trip
gender
password
1
phone
register
has
has
address
DOB
login
train-_id
tr_type
email
name
1
username
password
id
tr_num
train
tr_des
Admin
tr_ticket
tr_name
Admin_id
password
manage
m
1
station
st_id
has
st_name
timetable
timeTa_id
timeTa_des



Register

Name	Password	Address	E-mail	**Phone**	Gender	DOB

Login

ID	Password	UserName	User_ID

User

User_ID	Password	E-mail	UserName	Trip_id

Trip

Trip_ID	Start_Time	Arrival_Time

Train

Train_ID	Train_Num	Train_Type	Train_Des	Train_Ticket	Train_Name

Update

User_ID	Trip_ID

TimeTable

TimeTable_ID	TimeTable_Des

Admin

Admin_ID	Password

Station

Station_ID	Station_Name

Train_Trip

Trip_ID	Train_ID

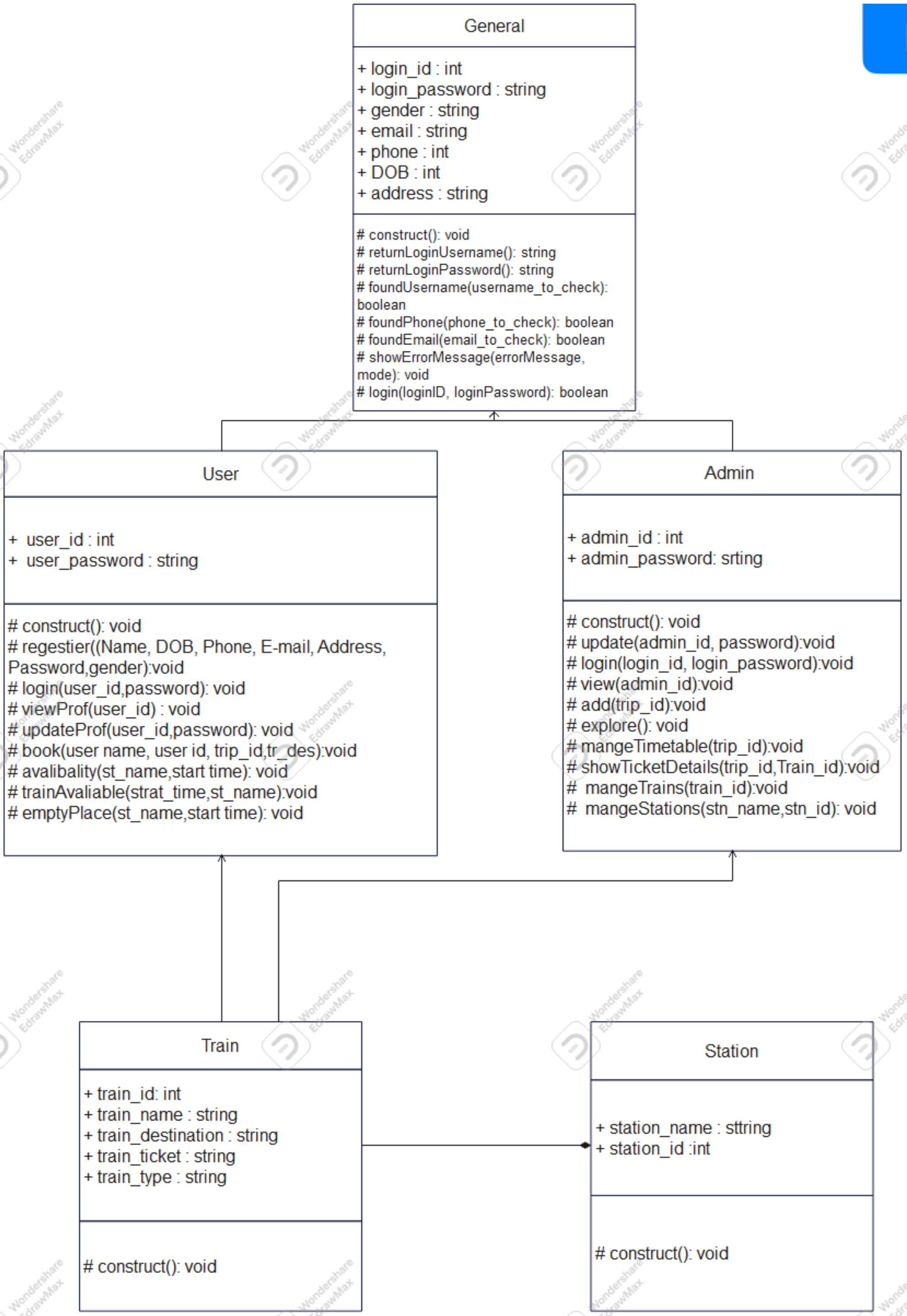
Register_User

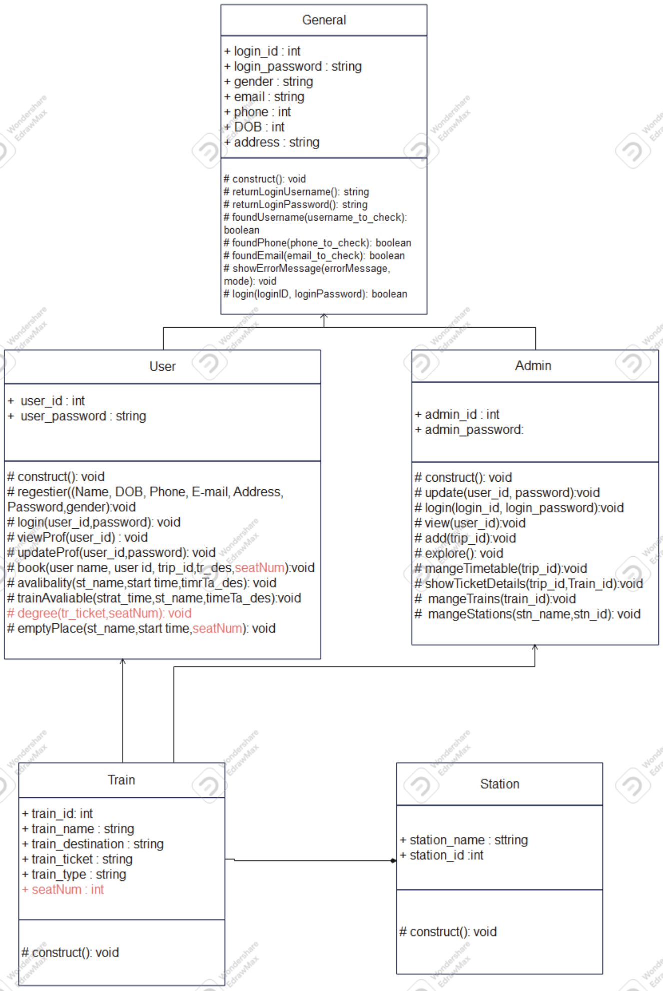
User_ID	Phone

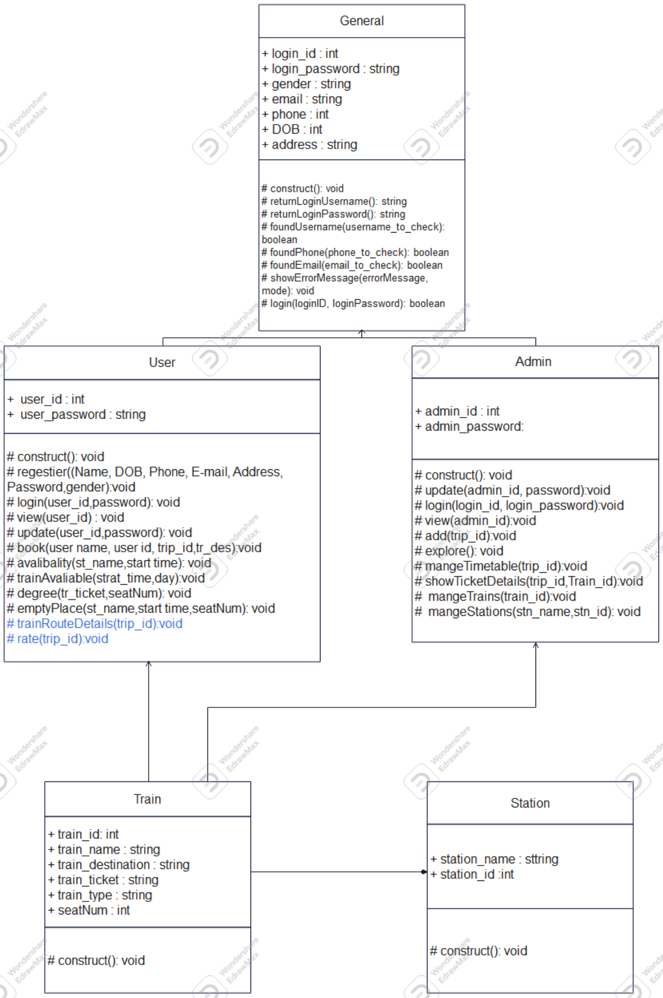
Station_Admin

Admin_ID	Station_id

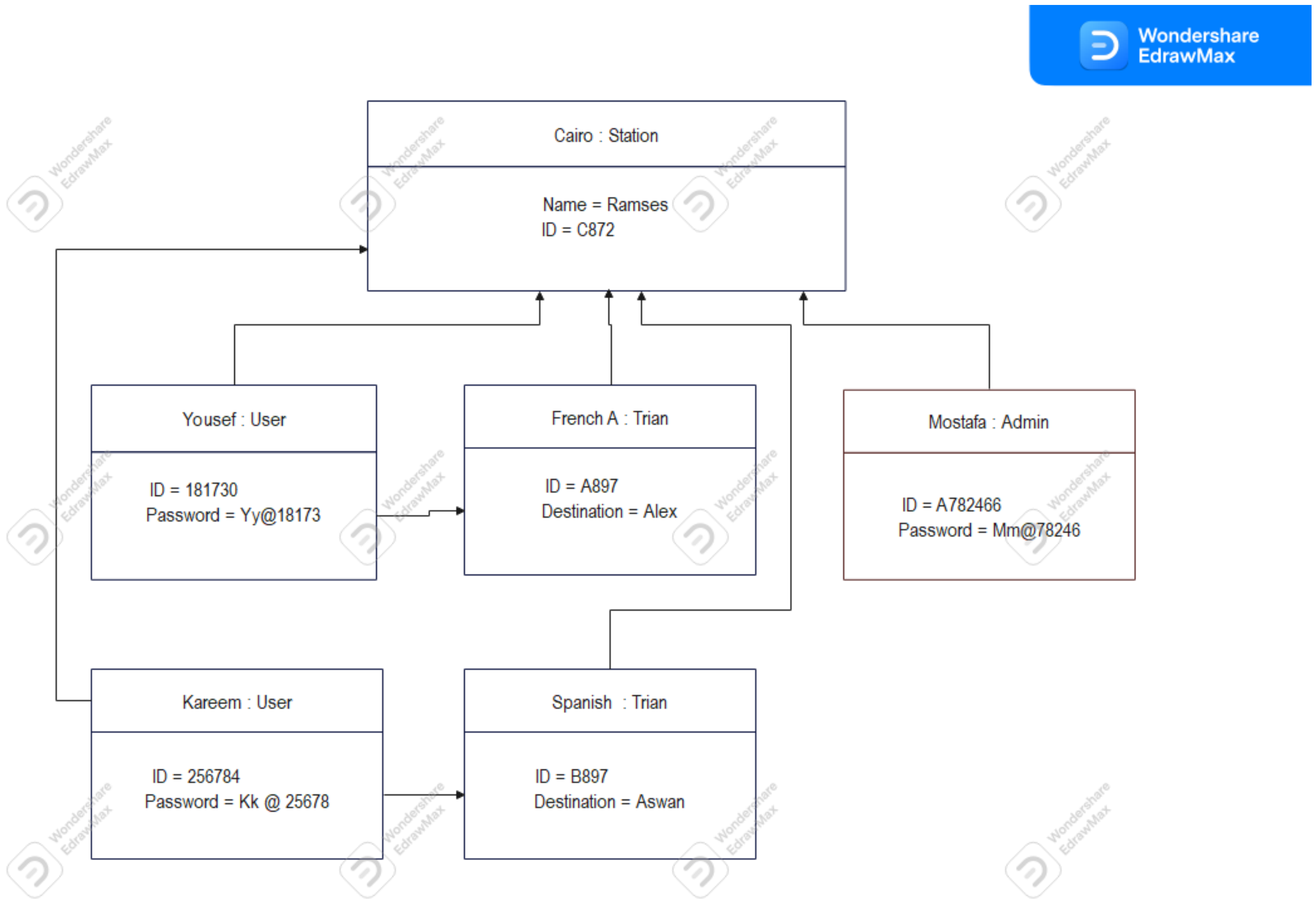
Class Diagram


General
+ login_id : int
+ login_password : string
+ gender : string
+ email : string
+ phone : int
+ DOB : int
+ address : string
# construct(): void
# returnLoginUsername(): string
# returnLoginPassword(): string
# foundUsername(username_to_check): boolean
# foundPhone(phone_to_check): boolean
# foundEmail(email_to_check): boolean
# showErrorMessage(errorMessage, mode): void
# login(loginID, loginPassword): boolean
User
+ user_id : int
+ user_password : string
# construct(): void
# regestier((Name, DOB, Phone, E-mail, Address, Password,gender):void
# login(user_id,password): void
# viewProf(user_id) : void
# updateProf(user_id,password): void
# book(user name, user id, trip_id,tr_des):void
# avalibality(st_name,start time): void
# trainAvaliable(strat_time,st_name):void
# emptyPlace(st_name,start time): void
Admin
+ admin_id : int
+ admin_password: srting
# construct(): void
# update(admin_id, password):void
# login(login_id, login_password):void
# view(admin_id):void
# add(trip_id):void
# explore(): void
# mangeTimetable(trip_id):void
# showTicketDetails(trip_id,Train_id):void
# mangeTrains(train_id):void
# mangeStations(stn_name,stn_id): void
Train
+ train_id: int
+ train_name : string
+ train_destination : string
+ train_ticket : string
+ train_type : string
# construct(): void
Station
+ station_name : sttring
+ station_id :int
# construct(): void

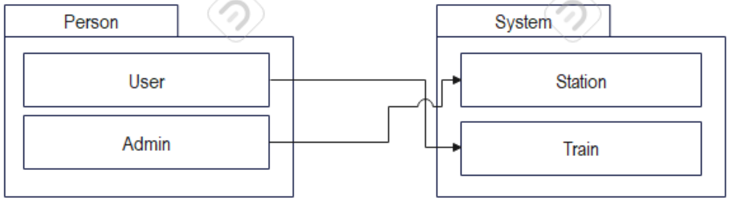

General
+ login_id : int
+ login_password : string
+ gender : string
+ email : string
+ phone : int
+ DOB : int
+ address : string
# construct(): void
# returnLoginUsername(): string
# returnLoginPassword(): string
# foundUsername(username_to_check): boolean
# foundPhone(phone_to_check): boolean
# foundEmail(email_to_check): boolean
# showErrorMessage(errorMessage, mode): void
# login(loginID, loginPassword): boolean
User
+ user_id : int
+ user_password : string
# construct(): void
# regestier((Name, DOB, Phone, E-mail, Address, Password,gender):void
# login(user_id,password): void
# viewProf(user_id) : void
# updateProf(user_id,password): void
# book(user name, user id, trip_id,tr_des,seatNum):void
# avalibality(st_name,start time,timrTa_des): void
# trainAvaliable(strat_time,st_name,timeTa_des):void
# degree(tr_ticket,seatNum): void
# emptyPlace(st_name,start time,seatNum): void
Admin
+ admin_id : int
+ admin_password:
# construct(): void
# update(user_id, password):void
# login(login_id, login_password):void
# view(user_id):void
# add(trip_id):void
# explore(): void
# mangeTimetable(trip_id):void
# showTicketDetails(trip_id,Train_id):void
# mangeTrains(train_id):void
# mangeStations(stn_name,stn_id): void
Train
+ train_id: int
+ train_name : string
+ train_destination : string
+ train_ticket : string
+ train_type : string
+ seatNum : int
# construct(): void
Station
+ station_name : sttring
+ station_id :int
# construct(): void


General
+ login_id : int
+ login_password : string
+ gender : string
+ email : string
+ phone : int
+ DOB : int
+ address : string
# construct(): void
# returnLoginUsername(): string
# returnLoginPassword(): string
# foundUsername(username_to_check): boolean
# foundPhone(phone_to_check): boolean
# foundEmail(email_to_check): boolean
# showErrorMessage(errorMessage, mode): void
# login(loginID, loginPassword): boolean
User
+ user_id : int
+ user_password : string
# construct(): void
# regestier((Name, DOB, Phone, E-mail, Address, Password,gender):void
# login(user_id,password): void
# view(user_id) : void
# update(user_id,password): void
# book(user name, user id, trip_id,tr_des):void
# avalibality(st_name,start time): void
# trainAvaliable(strat_time,day):void
# degree(tr_ticket,seatNum): void
# emptyPlace(st_name,start time,seatNum): void
# trainRouteDetails(trip_id):void
# rate(trip_id):void
Admin
+ admin_id : int
+ admin_password:
# construct(): void
# update(admin_id, password):void
# login(login_id, login_password):void
# view(admin_id):void
# add(trip_id):void
# explore(): void
# mangeTimetable(trip_id):void
# showTicketDetails(trip_id,Train_id):void
# mangeTrains(train_id):void
# mangeStations(stn_name,stn_id): void
Train
+ train_id: int
+ train_name : string
+ train_destination : string
+ train_ticket : string
+ train_type : string
+ seatNum : int
# construct(): void
Station
+ station_name : sttring
+ station_id :int
# construct(): void

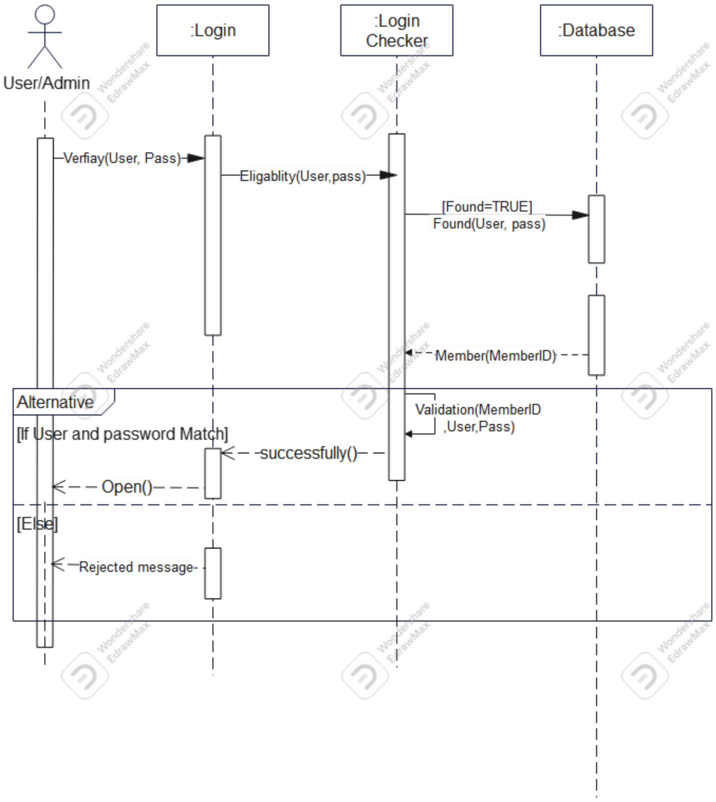
Object Diagrams

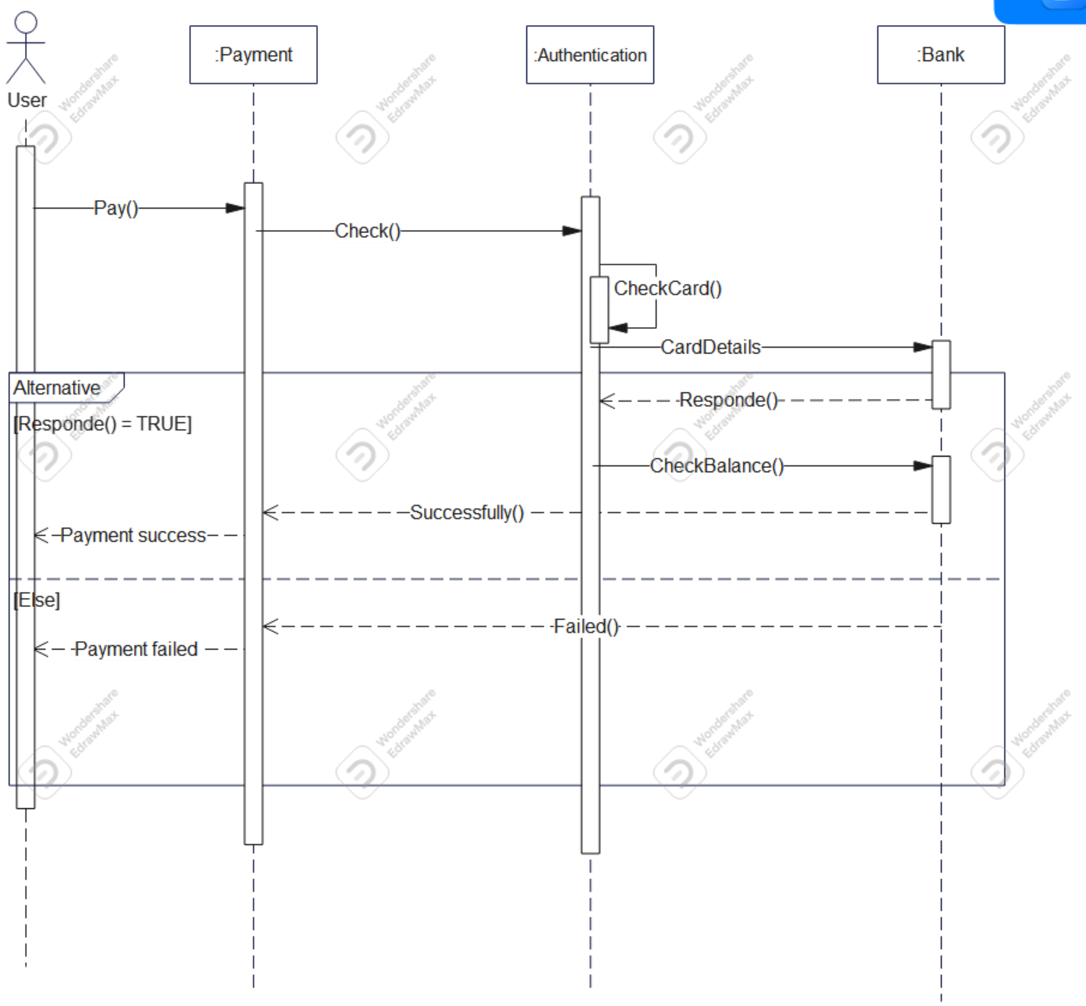


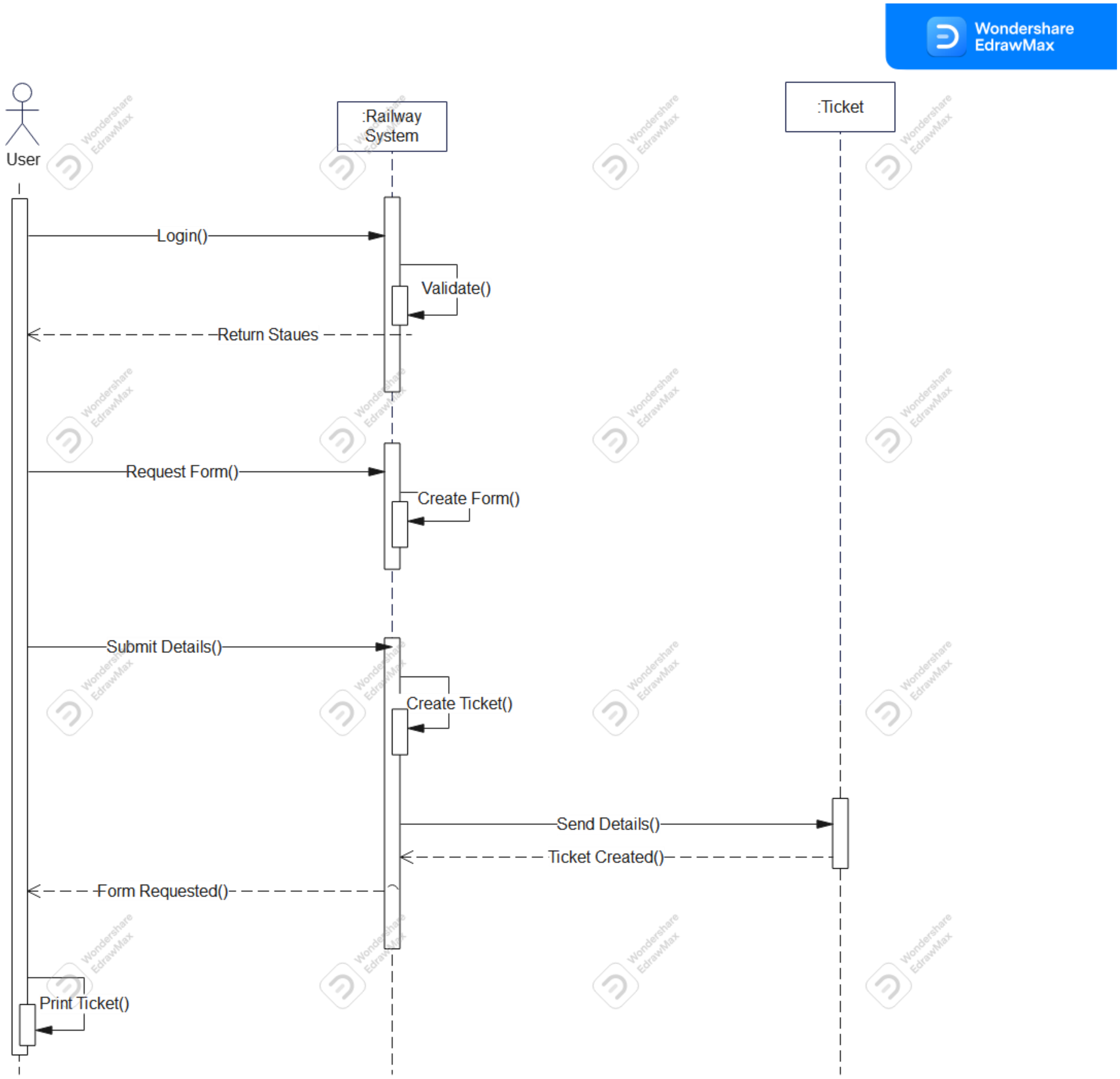
Package Diagrams



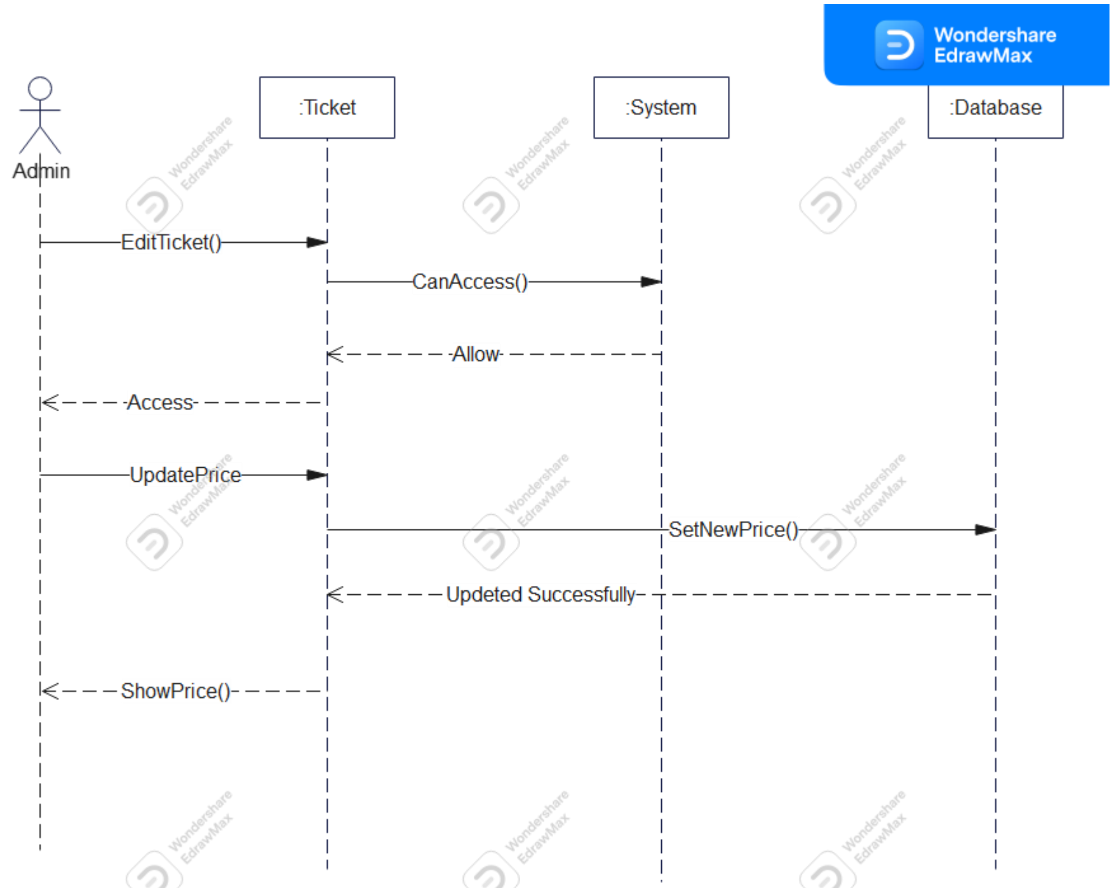
Sequence Diagrams

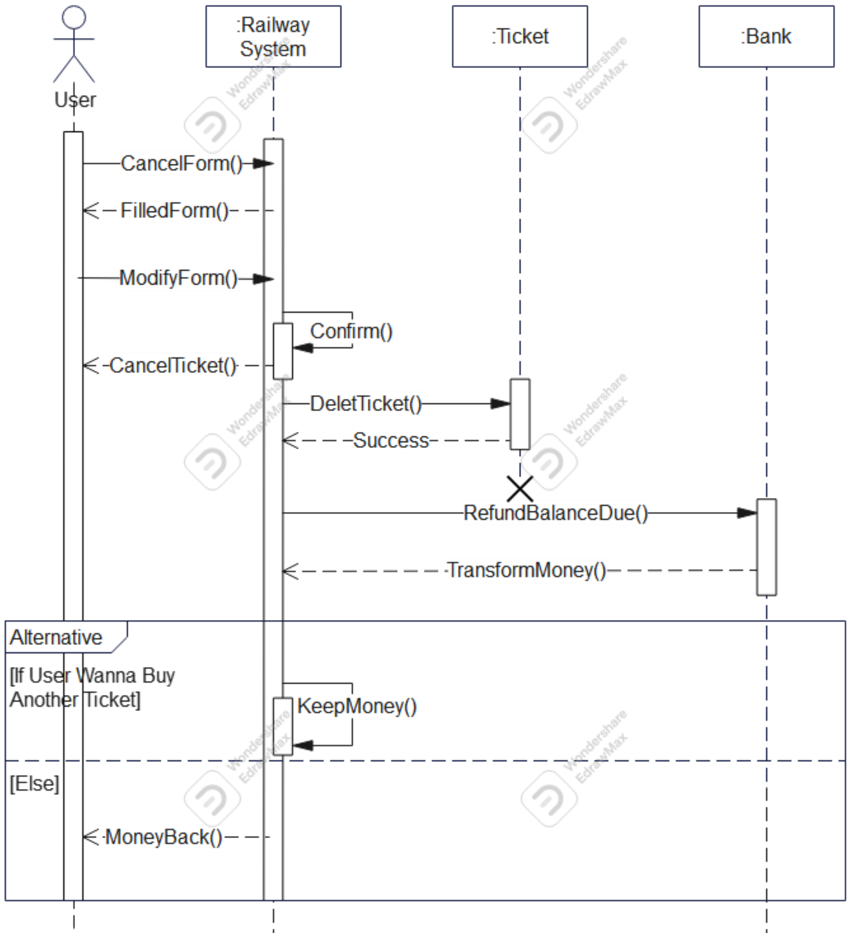

User/Admin
:Login
:Login
Checker
:Database
Verfiay(User, Pass)
Eligablity(User,pass)
[Found=TRUE]
Found(User, pass)
Member(MemberID)
Alternative
Validation(MemberID
,User,Pass)
[If User and password Match]
successfully()
Open()
[Else]
Rejected message


User
:Payment
:Authentication
:Bank
Pay()
Check()
CheckCard()
CardDetails
Alternative
[Responde() = TRUE]
Responde()
CheckBalance()
Successfully()
Payment success
[Else]
Failed()
Payment failed



Wondershare EdrawMax
User
:Railway System
:Ticket
Login()
Validate()
Return Staues
Request Form()
Create Form()
Submit Details()
Create Ticket()
Send Details()
Ticket Created()
Form Requested()
Print Ticket()


Wondershare EdrawMax
Admin
:Ticket
:System
:Database
EditTicket()
CanAccess()
Allow
Access
UpdatePrice
SetNewPrice()
Updeted Successfully
ShowPrice()


User
:Railway System
:Ticket
:Bank
CancelForm()
FilledForm()
ModifyForm()
Confirm()
CancelTicket()
DeletTicket()
Success
RefundBalanceDue()
TransformMoney()
Alternative
[If User Wanna Buy Another Ticket]
KeepMoney()
[Else]
MoneyBack()