

Candidate Number 18

ACIT4070 Programming and API for Interaction

DISCUSSION ABOUT THE CASE

A railways ticketing software system is a complex business application to handle multiple transaction requests instantly with large system loads. Considering the variety of functionality and error checks that a rail ticketing system provides, such application is indeed a complex software process.

“Norwegian Rails” is a railway ticketing company. The software of their application is in operation since many years and is written in old fashioned Java, before the new architecture and interactive things came. Also, the interface functionality is not behaving as expected. There were faults in the existing software due to which the entire network had to be taken down multiple times. There are many challenges to be able to continue working with the existing software as it is now showing signs of deterioration.

Managing Director (MD) of Norwegian Rails company needs a proper architectural design with brand new system written from scratch in a high level language. The requirements from the MD is to present the system architecture and design for the new system giving the details about possible styles of architecture, design and universal design.

OBJECTIVE:

The aim of this project is to produce a prototype and a technical report for the MD containing the information about the requirements, architecture, interface design and universal design of Norwegian Rails ticketing software. A database and logic for handling database requests from the prototype UI would also be implemented.

SCOPE

The scope of the prototype implementation for Norwegian Rails would include implementing the features as listed below.

1. *Purchase a ticket from Norwegian Rails* – The customer or user should be able to purchase the ticket for train journey from the Norwegian Rails.
2. *Reserve seat for journey* – The customer should be able to reserve seats for the journey during the purchase or later.
3. *Change or cancel ticket* – The customer should be able to modify the train booking or cancel the train ticket.
4. *Show ticket (for control / validation)* – The customer should be able to show the purchased ticket details for control/validation when requested.

IDENTIFIED REQUIREMENTS

Railway Reservation System as an application domain in general when implementing is a very large scale and complex project with several transaction requests to be handled at a quick response time.

Below are the identified requirements and assumptions in consideration when designing and implementing prototype for Norwegian Rails.

Main Functional Modules

User Login Functionality

User Registration Page - Create New User

Search Train Information

Book Ticket with passenger details

Purchase ticket with payment method

Reserve Seats for journey

Cancel or Change reservation

Show ticket for control/validation

Functional requirements

- User login functionality would be implemented where user would provide email address and password to login
- User should be able to search train information without logging in.
- Visitors or unregistered users should be able to complete the user registration.
- After logging, the user would be provided with the options of Search Trains, Purchase ticket, Cancel Change Reservation, Reserve Seats, Show Ticket for Control/Validation
- User should be able to search dates on given date providing source and destination value
- User should be able to proceed to book the ticket and provide passenger details such as name, age, gender, category (Adult, Child, Infant, Senior Citizen, Student)
- User should be able to reserve seats for his journey with the available choices of carriage and seat selection.
- User should be able to make payment for the ticket purchased via debit/credit card or Vipps payment.
- User should be able to cancel or change ticket reservation and user should get the refund amount after cancellation.
- User should be able to show the ticket information for control and validation after successful login if the user has purchased any tickets.
- Information messages should be shown whenever user successfully purchases a ticket, or cancels/ changes the reservation or reserves a seat for the journey.
- User should be able to change his personal information.
- User should be able to reset his password in case of forgotten.

Assumptions

- The database contains dummy data for demo of prototype implementation and not exact real data.
- When the user registration page is created and user is created, the database would be updated.
- The train information would be visible only for future 7 days in calendar.
- The train information would not be displayed for the dates in the past.
- The amount of the ticket would remain the same for all types of categories of people (Adult, Child, Student) but it can be changed later in future.
- The payment information provided would not be validated with the banking application as banking systems currently won't be integrated for the prototype. But in future, this would be done to ensure the proper validation of payment methods.
- If the ticket is booked or seats are reserved or ticket is cancelled or changed, the database would be updated.
- If the ticket purchase is not made, the ticket would not be booked and confirmed
- If the user clicks on Back button it would take to the previous screen
- If the user closes the application window, the application would be terminated and any ongoing transaction would not be committed to database.

Error Validation Checks

- If user provides incorrect login details, an error message would be displayed.
- For user registration, the mandatory fields need to be provided such as email address, password and mobile number.
- If the user searches train journey in the past, an error message would be displayed.
- If the user does not have any tickets and tries to reserve seats, an error message would be displayed
- If the user does not have any tickets and tries to cancel or change the reservation, an error message would be displayed.

Technical Requirements

- The GUI application would be developed using Python tkinter and SQL database would be created with dummy information for Norwegian Rails

User Interface Design Requirements

- The system should be intuitive to be able to perform next action logically.
- There should be error and validation checks
- There should be sufficient messages, information and error messages displayed to the user wherever needed.
- There should not be any abrupt termination of application system
- The application should be interactive and have a user appealing look and feel.

WORK PROCESS / PROJECT PLANNING

The work process methodology used for this prototype implementation is agile scrum methodology, also shown in figure 1. The requirements are defined in a list as user stories which are added to the product backlog. Product backlog grooming is done to prioritize the work list and work is broken into user stories. The user stories are divided into tasks. The user stories are then estimated according to the complexity and time required to complete it and assigned the user story points accordingly. In the sprint planning, user stories are selected to be completed in a sprint of 2 weeks and are added to the sprint backlog. Once the definition of done for the user story is met, the user story is marked as completed and new functionality is added to the prototype implementation. A retrospective is done on what could have been improved in terms of overall sprint work for future sprints after each sprint is completed.

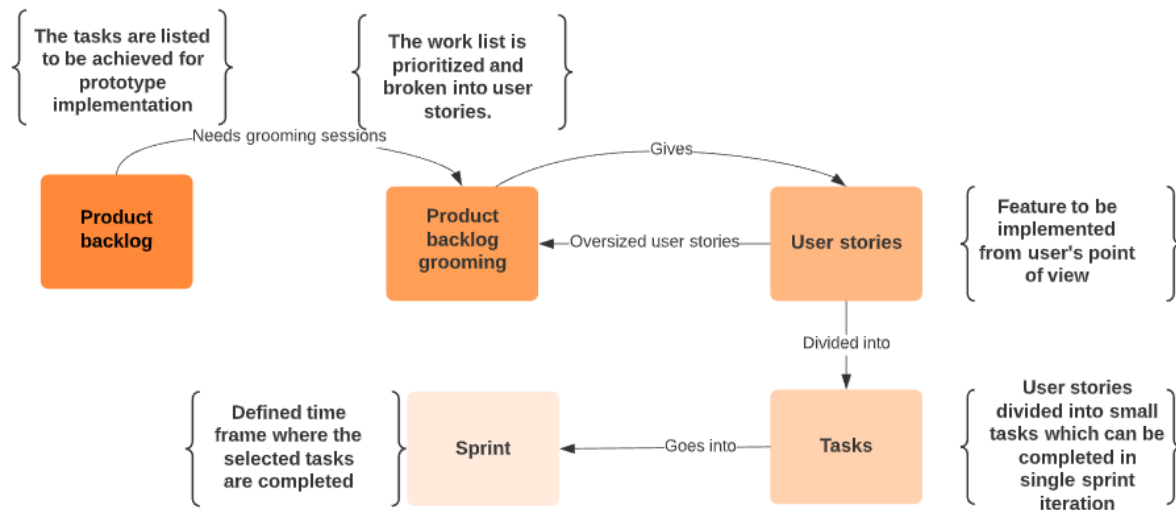


Figure 1 Agile Scrum Methodology for prototype implementation

The figure 2 presents the project planning with timelines carried out to complete the prototype implementation.

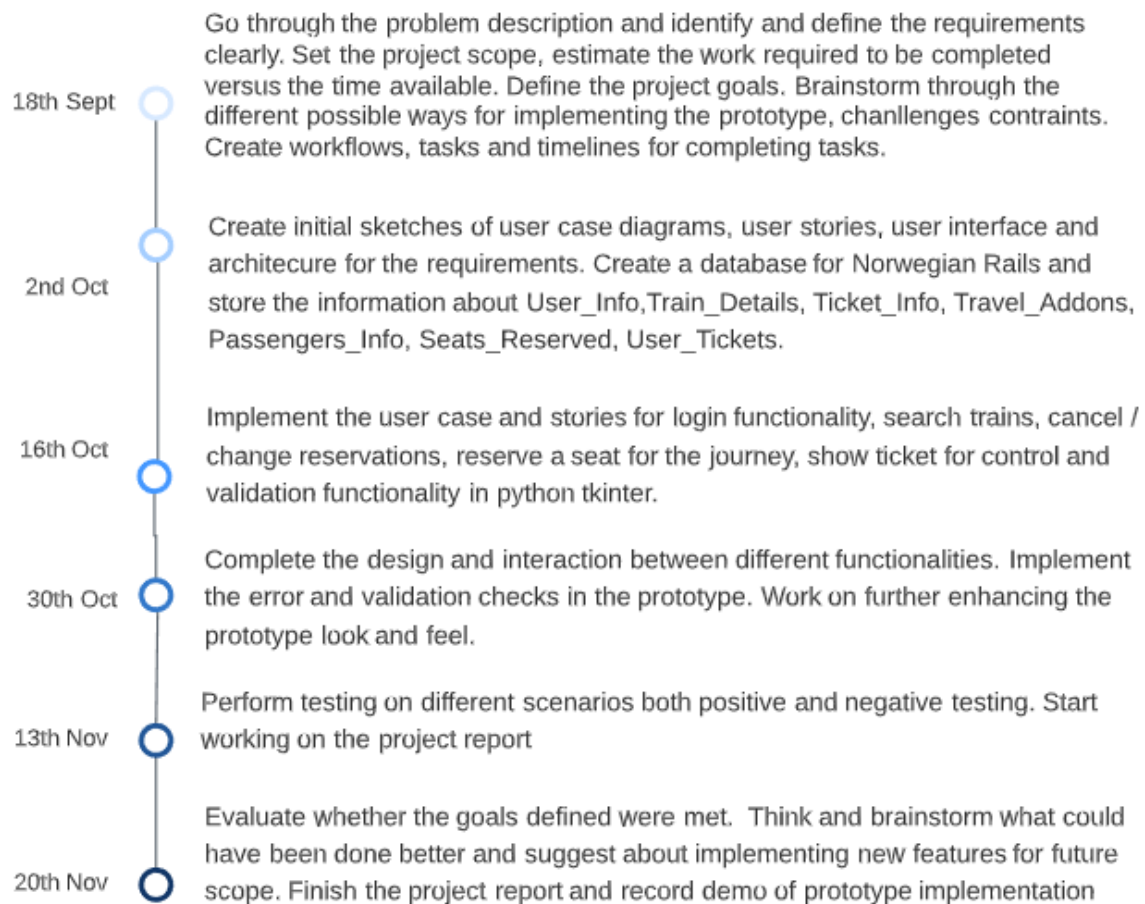


Figure 2 Project Planning Timeline

The figure 2 represents the project planning timeline carried out. It also included the software development life cycle process of requirement understanding, analysis, design, development and testing for each of the tasks completed.

SYSTEM ARCHITECTURE AND DESIGN

The figure 3 represents a high level view of the system design where the GUI would be available on PC/ Laptop and the accessible to various users such as registered users, visitors and admin users. The submission of transaction requests by the users would be handled by the business logic implemented in Python. The application code would send and retrieve the requests for updating and collecting the information from the database application in SQL.

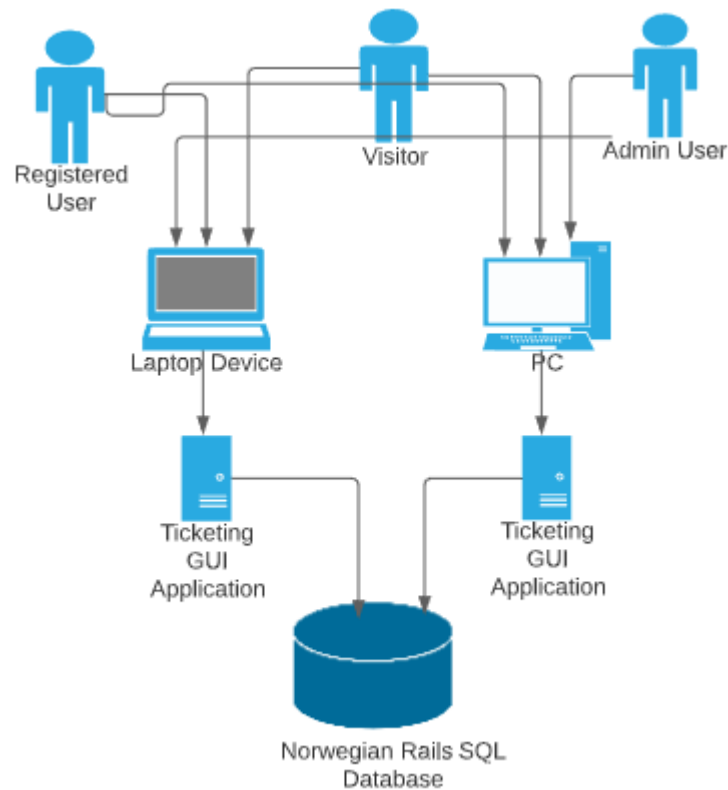


Figure 3 Architectural Design Sketch - High Level Overview

In order to visualize the system design, system interaction and structural model for Norwegian Rails system, the UML diagrams such as use-case diagrams, sequential diagrams are suitable to represent different interactive functionality in the prototype design and implementation.

Use-Case / User Story Diagrams for main features of the implementation

The Figure 4 represents the Use-Case diagram depicting the features from the requirements for Norwegian Rails. The Customers / Users, Ticket Inspectors, Ticketing System and Banking System are the actors.

Login/New Registration is use-case where registered users can sign in and unregistered users would need to sign up for registration. The use-case search trains provides the user with the option to search trains for the journey and proceed with the booking ticket if conditions are met.

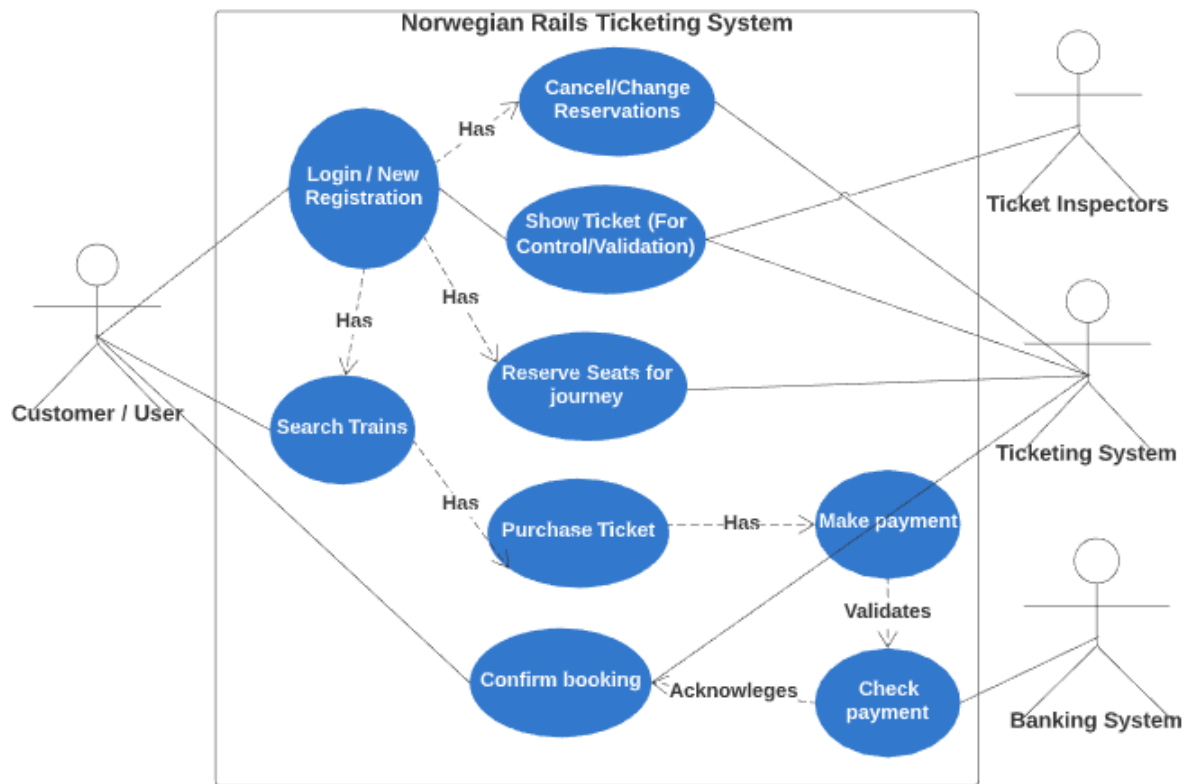


Figure 4 Use-Case Diagram for Norwegian Rails Prototype Implementation

The Purchase ticket is a use-case where in the passenger requests for purchasing ticket. The Purchase ticket use-case has make payment use-case where the payment is made through bank debit/credit cards and the payment is validated through Check Payment use-case with the Banking system. If the payment is successful, the user is notified with the confirmation of the booking and the ticket is confirmed.

In the Reserve Seats for journey use-case, the user requests for reserving for his seats for the journey. Similarly, Change or Cancel ticket is where the customer or traveller needs to change or cancel the ticket through the Rails ticketing system. The show ticket (for control and validation) is where the ticket inspectors ask the travellers to show a valid ticket for the journey. The ticket is validated and verified through the Norwegian Rails application.

Sequence Diagrams

1. Purchase a ticket from Norwegian Rails

The figure 5 shows the sequence diagram with series of actions performed for purchasing the ticket. The user after successful login searches for train, this request goes to the database and the out if fetched and displayed to the user. If the user is interested to buy the ticket, he proceeds with the booking providing the passenger information. Then user proceeds further to make the payment, the payment is validated with the banking application and if the payment is successful, the ticket is

booked and confirmed. The booked ticket information is displayed to the user and the database is updated with the details.

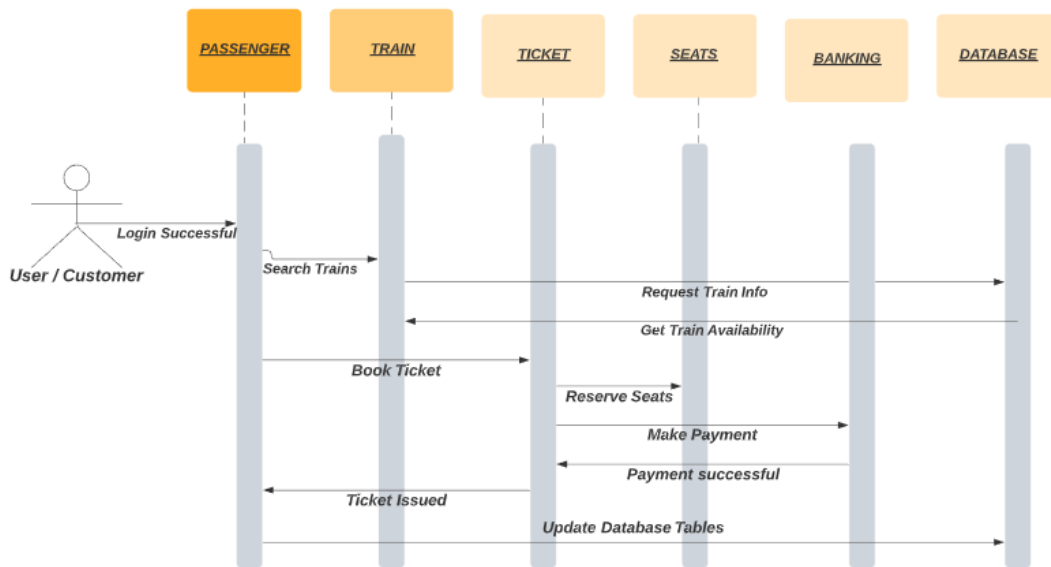


Figure 5 Purchase Ticket Use-case: Sequence Diagram

2. Cancel/Change Reservation

The figure 6 shows the sequence diagram for cancel or change reservation. After user is logged in successfully, from the available tickets, the user selects the ticket to cancel or change and submits the request for cancellation. The ticket is cancelled and message is displayed to the user for ticket cancellation. The amount is refunded back to the user and the database is updated with the deletion of the ticket booked information for the user.

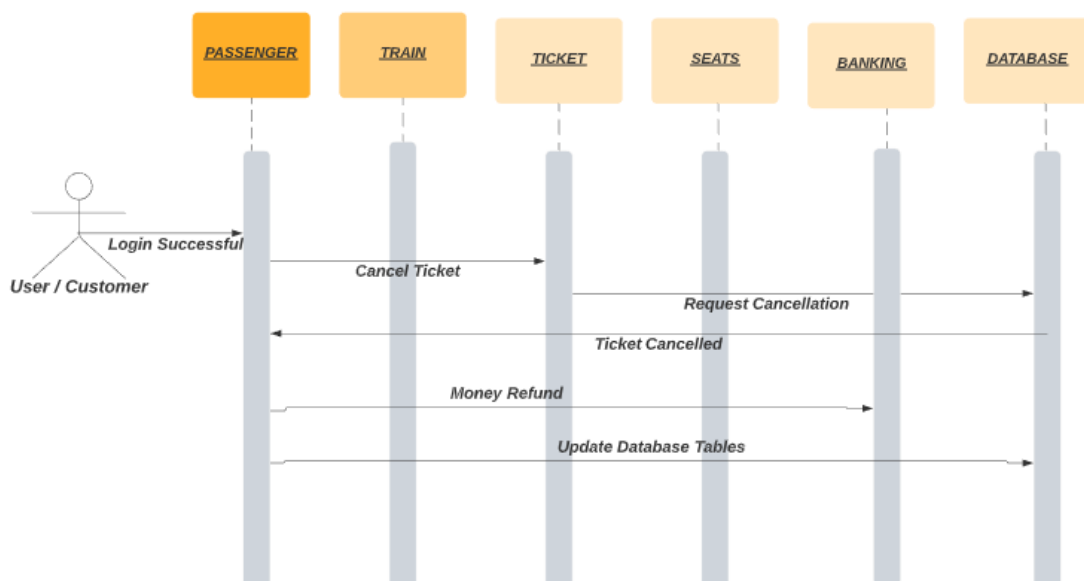


Figure 6 Cancel/Change Reservation Use-case: Sequence Diagram

3. Reserve Seats

The figure 7 shows the sequence diagram for the Reserve seats functionality, the user after login, checks the ticket and submits a request to reserve the seats. The available seats are displayed and user selects the seats of his choice and proceeds to confirm the seat reservation. The seat reservation is confirmed and information is displayed to the user and the database is updated with the seat reservation information.

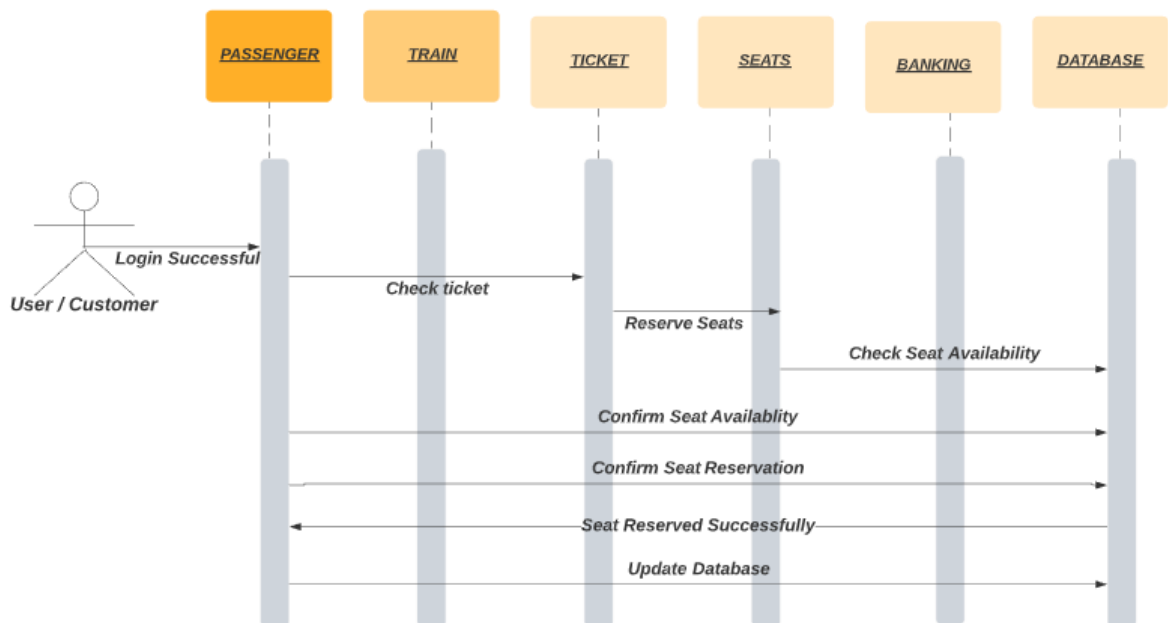


Figure 7 Reserve Seats Use-case: Sequence Diagram

PROTOTYPE IMPLEMENTATION

The railway ticketing software system should be able to do customer login, search trains, book and make purchase for the ticket, allow the user to reserve the seats, cancel or change the tickets at later point of time.

When deciding the architecture style and patterns, the software quality attributes need to be considered. Software quality attributes are properties of the system which can be measured and tested to determine how the system fulfils the expectations from the stakeholders [2]

The figure 8 shows the software quality attributes specified by ISO25010 standard.

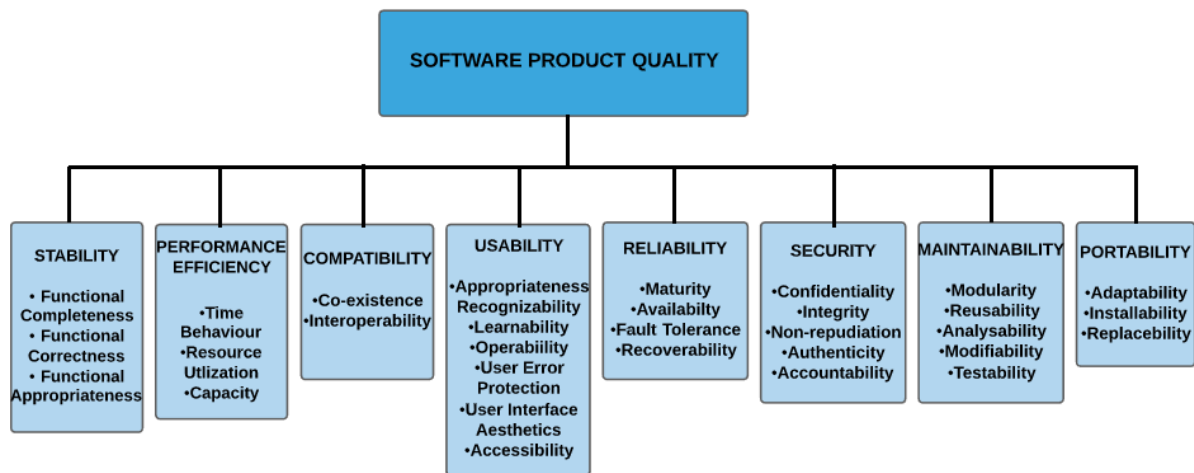


Figure 8 ISO25010 Software Quality Attributes Ref [1]

The software quality attributes can be mainly categorized into two parts:

1. *Attributes describing system's runtime property*
 - These include availability of the system and throughput, usability of the system, performance and quick response time to the submitted transaction or process, time taken to come back to normal state in case of any problems encountered and so on.[2]
2. *Attributes describing development of system*

These include attributes with ease of modification or changes to the system, testability of the system and so on.[2]

ARCHITECTURE STYLE

An n-tier architecture or 3-tier architecture style is used for this prototype implementation. N-tier application architecture gives the model through which reusable and flexible applications could be created. The division of application into tiers gives the developers freedom to modify or add changes to only specific layer rather than working with the whole application. [3]

The figure 9 shows the 3-tier architecture for Norwegian Rails. The 3-tier architecture comprises of view or presentation tier, business logic tier and data storage tier. In the case of Norwegian Rails ticketing system, the top-most view layer would be the user interface, it would display the information and results in an understandable format to the user. In case of business logic tier, it would be the programming code which contains the different logical steps and calculations to execute the processes. In data storage tier, the information is stored in SQL database in this case and it is given to the logical tier for processing and finally passed to the view tier where it is displayed to the user.[3]

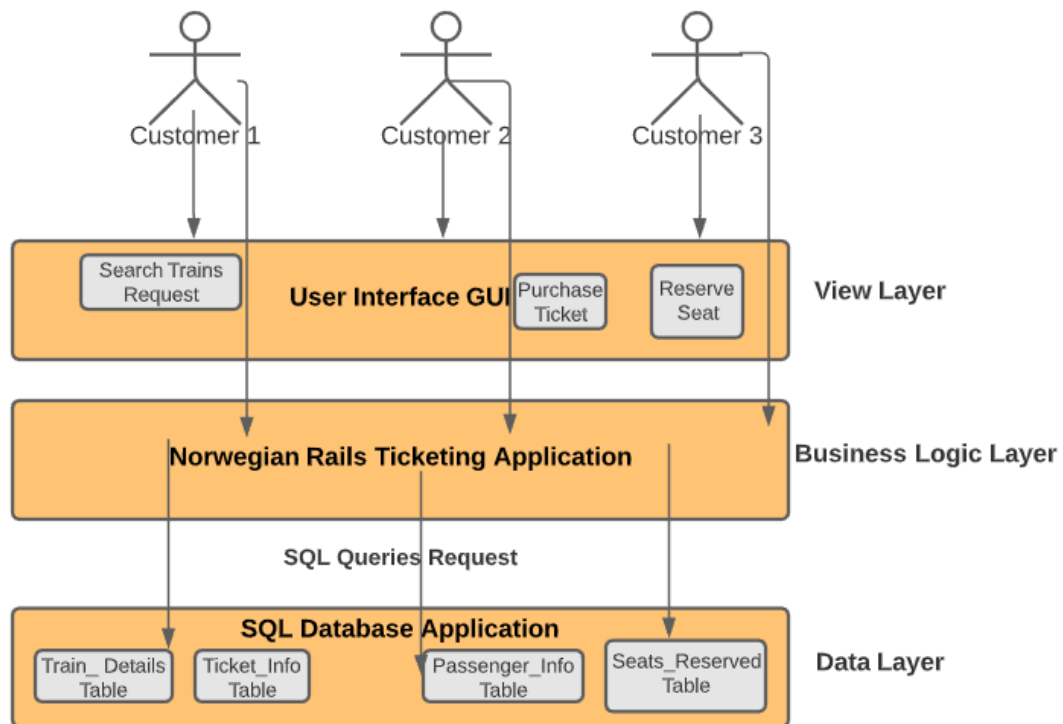


Figure 9 Architecture Style: 3-Tier Architecture

The UML diagrams for architecture modelling are: Package Diagrams, Component Diagrams and Deployment Diagrams. The figures show the various UML diagrams for architecture modelling for Norwegian Rails prototype implementation.

Architecture Modelling: Deployment Diagram

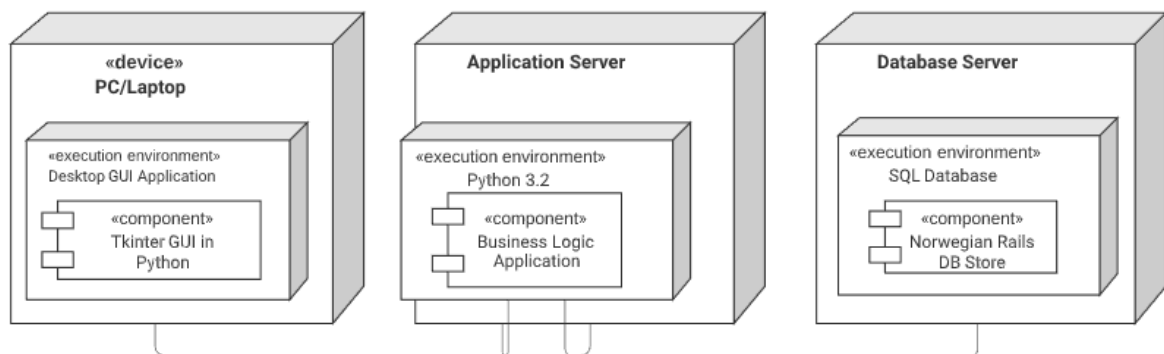


Figure 10 Architecture Modelling: Deployment Diagram for prototype implementation

Architecture Modelling : Package Diagram

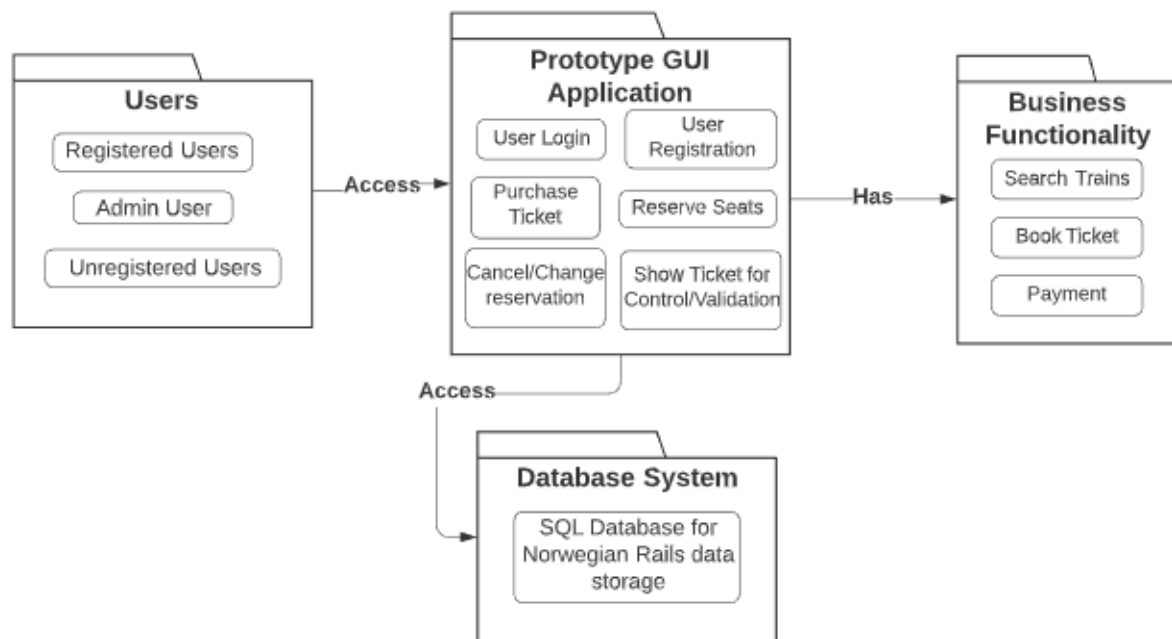


Figure 11 Architecture Modelling: Package Diagram

The figure 10 shows the deployment diagram and figure 11 shows the package diagram for the prototype implementation for Norwegian Rails Ticketing system.

ARCHITECTURAL PATTERNS

There are several architectural patterns available to choose from. The below architecture patterns are discussed and reviewed according to the requirement.

Pipe and Filter Architecture: This architectural pattern is suitable for sequential model. The railway ticketing system follows a sequential process of processing transactions at each step based on several conditions, therefore the pipe and filter architecture is suitable for implementing this prototype.

For instance, when the user wants to purchase the train ticket, he would search for the available train journey. If a suitable match is found for his train search with seat availability, for booking the train ticket, the passenger should provide his passenger details in order to book. After providing the required travel information, the user is then prompted for making the payment. Only after the payment is successful the train ticket is booked for the user.

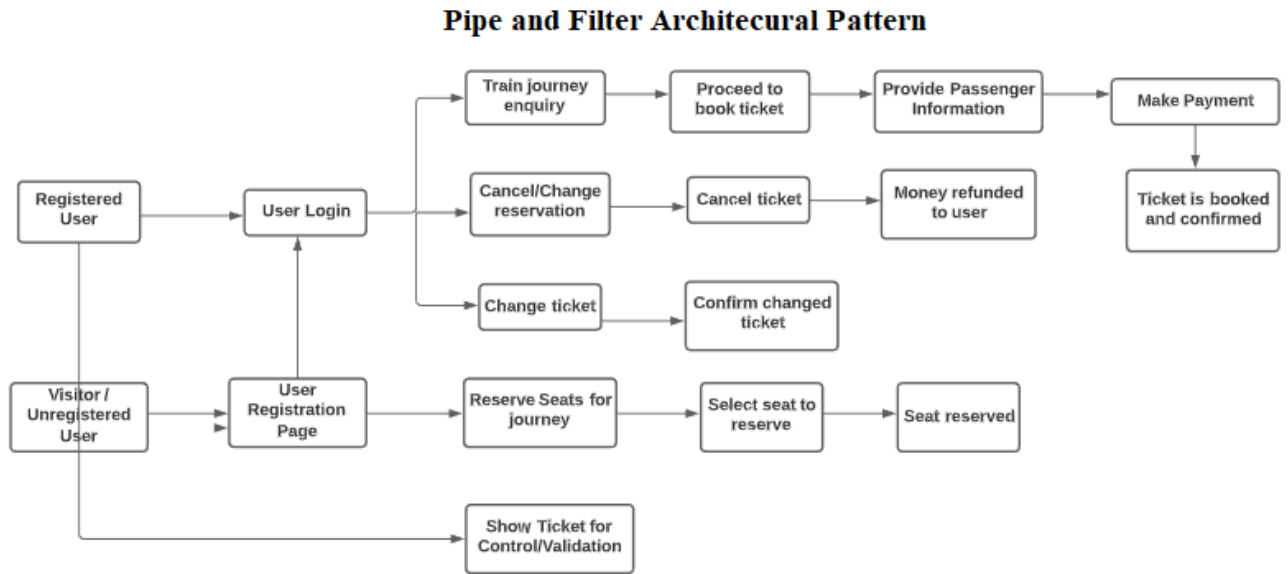


Figure 12 Pipe and Filter Architectural Pattern for Norwegian Rails

Advantages: The different transactions related to railway reservation such as customer login, search trains, book and purchase ticket being processed one after the other can be interpreted easily. The shifting of the different transaction executions from one to another can be easily monitored. The input and output data workflow could be easily understood and analyzed.

Disadvantages: When displaying the information on the GUI which is retrieved from the Database, it requires some formatting and different widgets and tools to show the details. This sometimes may increase the development workload.

The figure 12 shows Pipe and Filter Architectural Pattern in the case of implementing Norwegian Rails ticketing system.

Layered Architecture: The layered architecture also suits the system requirements. This architecture consists of several horizontal layers in which the components are organized in each layer. The figure 13 shows layered architecture diagram for Norwegian Rails system.

Layered Architectural Pattern

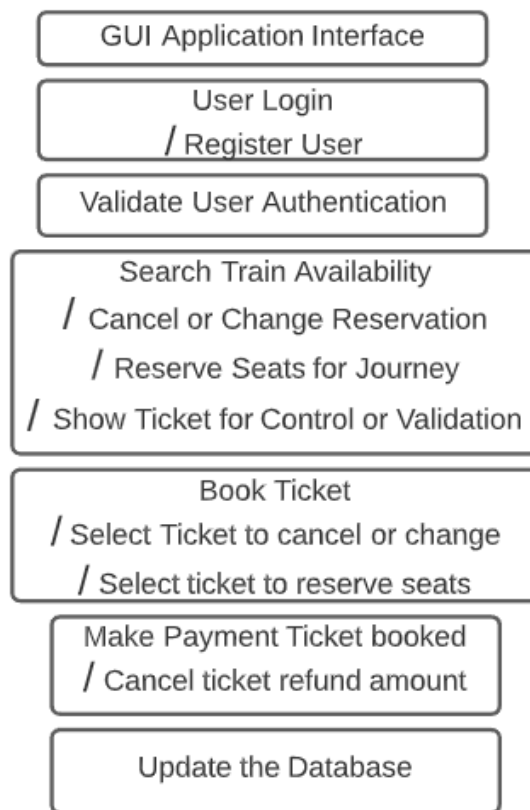


Figure 12 Layered Architectural Pattern for Norwegian Rails

Advantages: This architecture is easy to implement as most of the software applications work in layered approach. In case of additional requirements, a new layer can be added to support the incremental development approach. The hierarchical structure and isolation of layers from each other helps to group similar components in the same layer and separate the different types of components in different layers making them independent of each other. As components are part of particular layers, the testing can be carried out independently.

Disadvantages: It could affect the performance of the system as the transaction request and response need to go and process through multiple layers. Therefore, in some cases the time taken to execute the transaction may increase.

As each of the layers interact and communicate with each other. If there is any failure in any of the later then it would result in system crash and failure.

Client - Server Architectural Pattern: In this architectural pattern, the server handles all the requests received by the clients. This could create additional load on the server in case of too many requests. This architectural pattern currently does not suit to prototype as it is a standalone application currently and not a web based application which could be accessible to all the multiple users or clients at the same time on the internet.

But for future scope, this architectural pattern is also a best option which could be considered for the rails ticketing application. The figure 14 shows the client-server architecture for future scope.

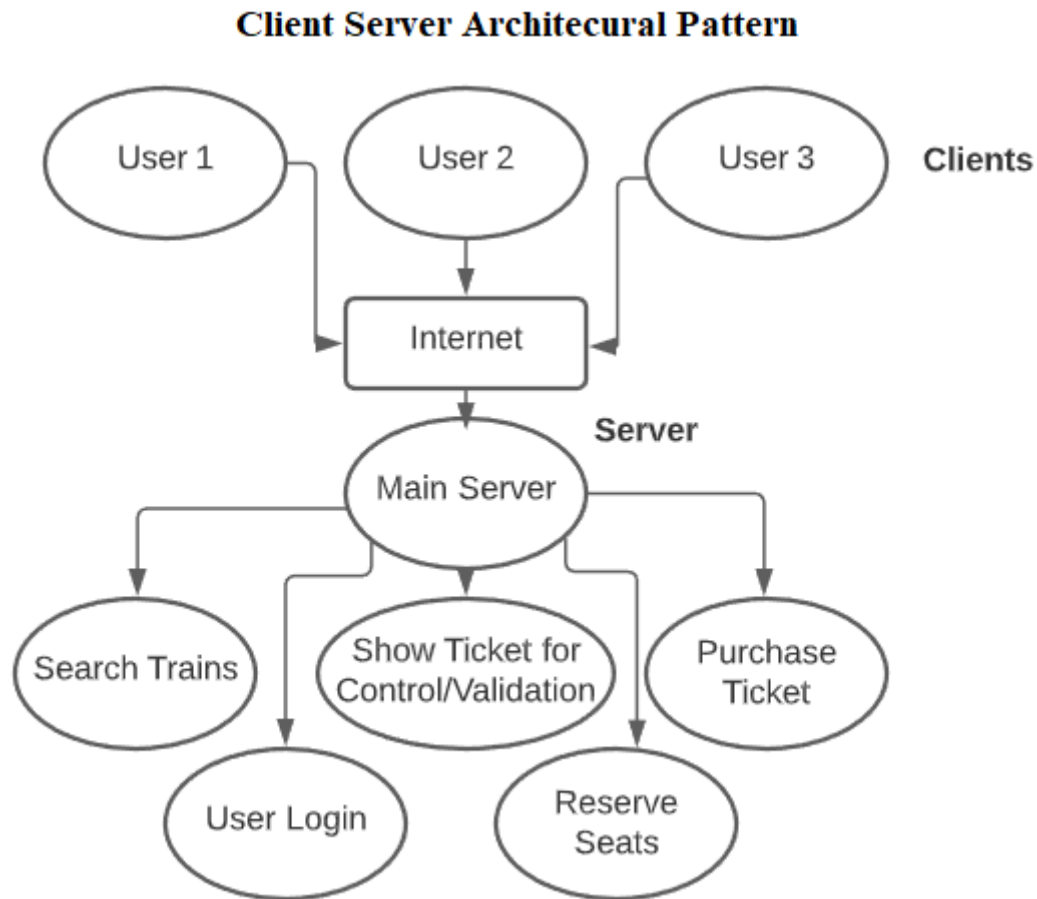


Figure 14 Client Server Architectural Pattern for future scope

System Repository architecture: This architecture that enables multiple interfacing components to share the same data. The same dataset is accessed by every component. Changes in the data from one component would make universal change in the dataset and the changed dataset would be available for other components. This type of architecture does not fit for the current prototype design as the application would be managed only on one system. [6]

But in future, when the application is deployed on web and through internet service multiple users could access the rails ticketing system to book the train tickets and so on, this type of architecture would be one of the suitable options.

Model-View-Controller : This model divides the design into logical components: Model, View and Controller. This pattern has the benefit that the information displayed is segregated and stored internally from the way in which the information is displayed to the user and the input is accepted from the user. This pattern has gained quite popularity especially for web applications and desktop GUI applications. This architectural pattern could also be useful for designing the railway ticket system. The figure 15 shows the model-view-controller pattern for implementing Norwegian Rails.

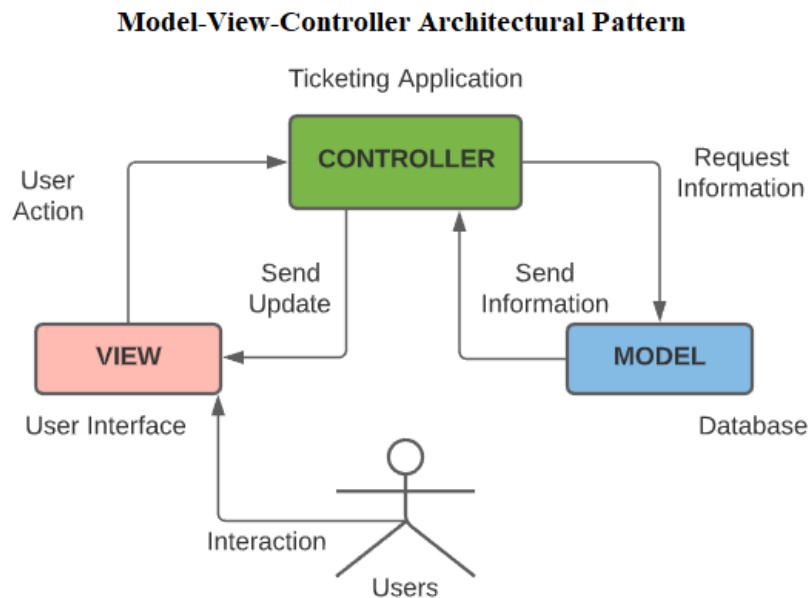


Figure 13 Architectural Pattern: Model-View-Controller

For the current prototype implementation, model-view-controller design approach is implemented. But the others such as pipe and filter, and layered design approaches are also suitable.

INTERFACE DESIGN AND INTERACTION

This section discusses the different interface and interaction design patterns while implementing the prototype for Norwegian Rails.

For the implementation of rails ticketing system, a database for storing the information needs to be designed and developed. To view the database structure and design, the figure 16 shows the E-R diagram for Railway Reservation system for Norwegian Rails.

Entity Relationship Diagram for Norwegian Rails

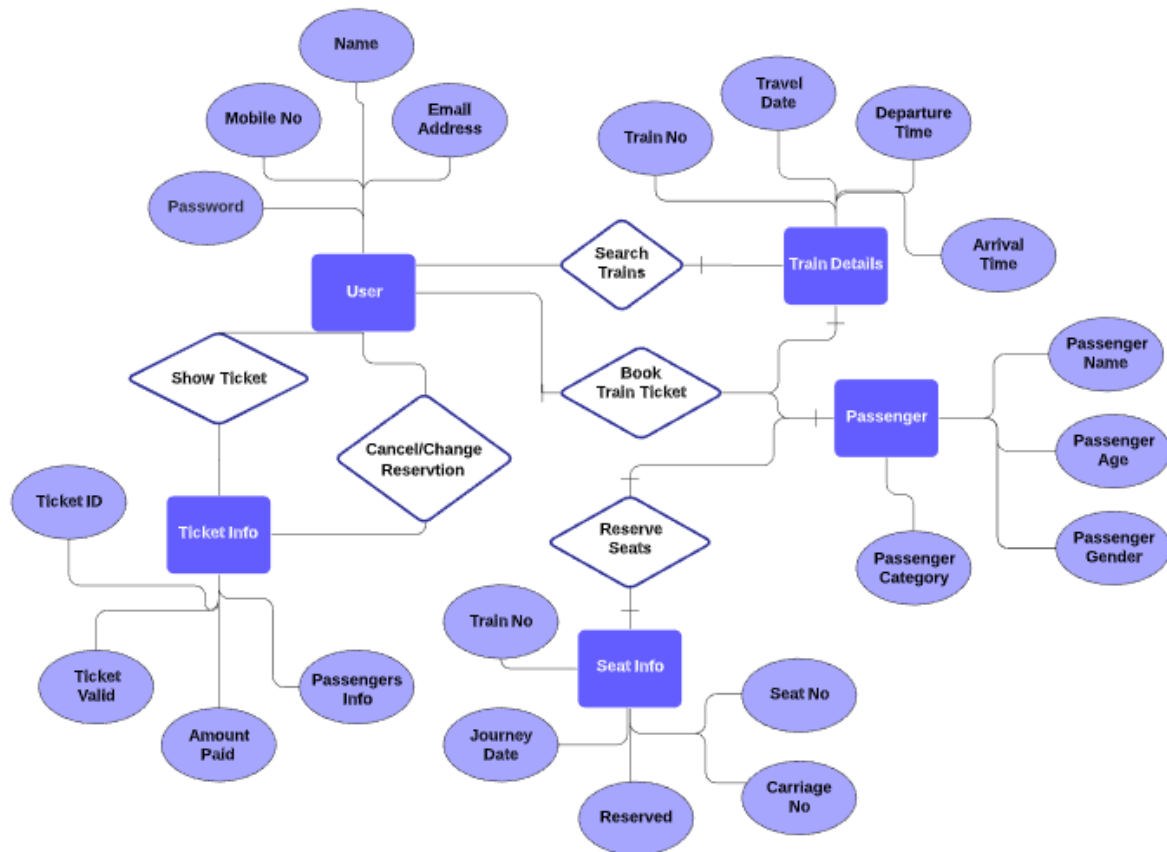


Figure 14 Entity Relation Diagram for Norwegian Rails

A class diagram is useful to describe the structure of different classes, class attributes and class operations or methods, relationships among the classes. The figure 17 shows the list of classes, their attributes and methods implemented in the prototype.

Main Classes for Rails system:

User Information Class: Manage operations on User Details.

Train Details Class: Manage operations on train details

User Tickets Class: Manage operations on user tickets

Passengers Info Class: Manage operations on passenger information

Seats Reserved Class: Manage operations on seat booking and reservation

Ticket Info Class: Manage operations on Ticket Generation

Class Diagram for Norwegian Rails

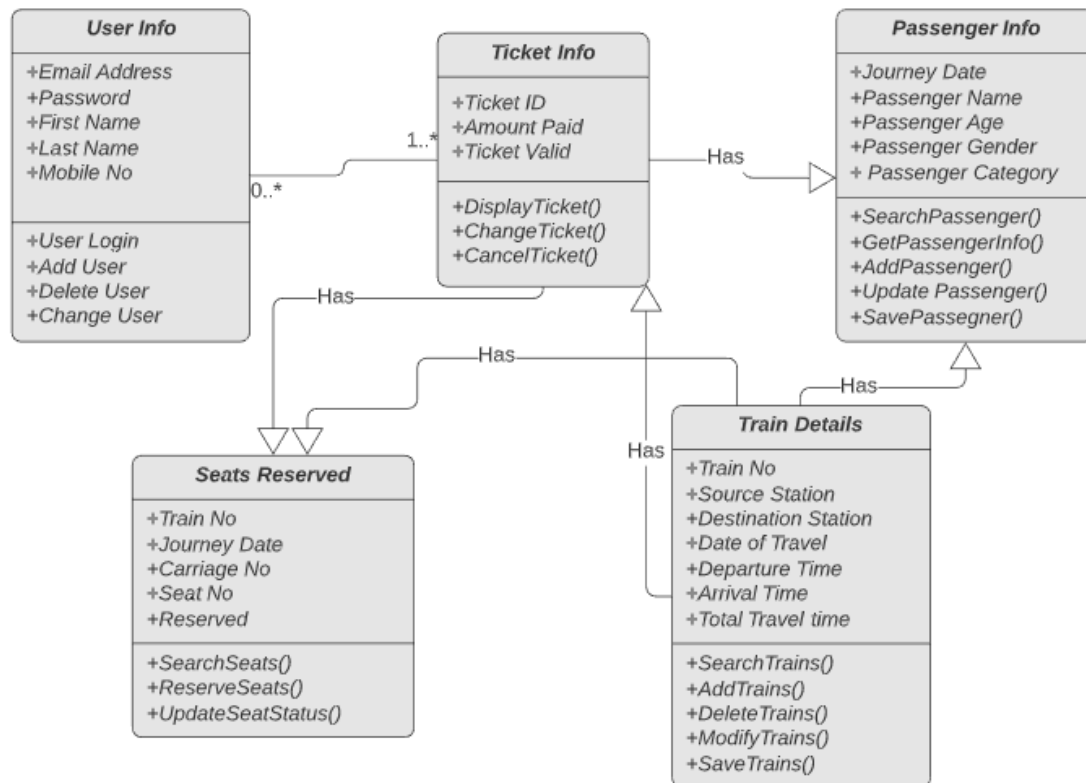


Figure 15 Class Diagram for Norwegian Rails

Classes and their attributes for Rails system:

User Information Attributes: Email Address, Password, First Name , Last Name, Mobile No

Train Details Attributes: Train No, Source Station, Destination Station, Date of Travel, Departure Time, Arrival Time, Total Travel Time

User Tickets Attributes: Email Address, Ticket ID

Passengers Info Attributes: Ticket ID, User Tickets, Train No, Source, Destination, Journey Data, Passenger Name, Passenger Age, Passenger Gender, Passenger Category

Seats Reserved Attributes: Email Address, Train No, Journey Date, Carriage No, Seat No, Reserved

Ticket Info Attributes: Ticket ID, Amount Paid, Ticket Valid

Classes and their method for Rails system:

User Information Methods: SearchUser(), LoginUser(), AddUser(), DeleteUser()

Train Details Methods: SearchTrains(), AddTrains(), DeleteTrains(), ModifyTrains(), SaveTrains()

User Tickets Methods: SearchUserTickets(), GetUserTickets(),UpdateUserTickets(), SaveUserTickets()

Passenger Info Methods: SearchPassengerInfo(),
GetPassengerInfo(),AddPasseenger(),UpdatePassenger(), SavePassengerInfo()

Seats Reserved Methods: SearchSeat(), ReserveSeat(), UpdateSeatStatus()

Ticket Info Methods: DisplayTicket(), SaveTicket(), UpdateTicket(), DeleteTicket()

There are mainly three categories of namely creational, structural and behavioural patterns. For creation of different objects, Factory method design pattern is implemented. For implementing the structure and relationship between the classes, Strategy Pattern is implemented. The prototype follows the below Interface and Interaction Design Principle such as visibility, providing the feedback, is consistent and validates the user constraints.[7]

DATABASE DESIGN:

A database in SQL is created for Norwegian Rails with the tables User_Info, Train_Details, Passenger_Info, Seats_Reserved, User_Tickets, Seats_Reserved. The CREATE sql queries are executed in Python code to create the tables with the attributes in the tables below.

Table 1: User_Info

<i>Field Name</i>	<i>Data Type</i>	<i>Width</i>	<i>Primary Key</i>	<i>Note</i>
Email_Address	Varchar	20	Yes	Email address
Password	Varchar	20	No	Password
First_Name	Varchar	20	No	First Name
Last_Name	Varchar	20	No	Last Name
Mobile_No	Integer	10	No	Mobile Number

Table 2: Train_Details

<i>Field Name</i>	<i>Data Type</i>	<i>Width</i>	<i>Primary Key</i>	<i>Note</i>
Train_No	Integer	3	UNIQUE	Train Number
Source_Station	Varchar	-	No	Source Station
Destination_Station	Varchar	-	No	Destination Station
Date_of_Travel	Date	-	No	Journey Date
Departure_Time	Time		No	Time of Departure
Arrival_Time	Time	-	No	Time of Arrival
Total_Travel_Time	Varchar	-	No	Travel Time

Table 3: Passenger_Info

<i>Field Name</i>	<i>Data Type</i>	<i>Width</i>	<i>Primary Key</i>	<i>Note</i>
Ticket_id		-		
Train_No	Integer	3	No	Train Number
Source	Varchar	-		Source station
Destination	Varchar	-	No	Destination station
Journey_Date	Date	-	No	Data of Journey
P1_Name	Varchar	20	No	Passenger Name

P1_Age	Integer	100	No	Passenger Age
P1_Gender	Varchar	-	No	Passenger Gender
P1_Category	Varchar	10	No	Passenger Category

Table 4: Seats_Reserved

<i>Field Name</i>	<i>Data Type</i>	<i>Width</i>	<i>Primary Key</i>	<i>Note</i>
Email_Address	Varchar	20	No	User email id
Train_No	Integer	3	No	Train Number
Journey_Date	Date	-	No	Journey Date
Carriage_No	Varchar	3	No	Carriage Number
Seat_No	Integer	3	No	Seat Number
Reserved	Varchar	3	No	Reserve Flag value

Table 5: User_Tickets

<i>Field Name</i>	<i>Data Type</i>	<i>Width</i>	<i>Primary Key</i>	<i>Note</i>
Ticket_id		-		
Train_No	Integer	3	No	Train Number
Source	Varchar	-		Source station
Destination	Varchar	-	No	Destination station
Journey_Date	Date	-	No	Data of Journey
P1_Name	Varchar	20	No	Passenger Name
P1_Age	Integer	100	No	Passenger Age
P1_Gender	Varchar	-	No	Passenger Gender
P1_Category	Varchar	10	No	Passenger Category

The database is inserted with dummy information relevant to Norwegian rails in all the tables in order to make the prototype work with the help of INSERT sql queries.

UNIVERSAL DESIGN PRINCIPLES

Universal Design is a process with a goal to provide choice with the objective of providing diversity, equality and inclusiveness. It enables accessibility and use the system for all kinds of people. It is a continuous design process on the current system to make it more better in terms of designing the system universally for everyone. The choice of the system design should be such that it is flexible, adaptable, simple to use. The system design and prototype implementation should be suitable to all kinds of people include people from various human diversity regardless of age, ability, gender and financial status.[4]

The figure 18 displays the seven principles of Universal Design as follows: Equitable, Flexibility, Simple and Intuitive, Perception Information, Tolerance for error, Low physical effort, Size and space [4]

7 PRINCIPLES OF UNIVERSAL DESIGN

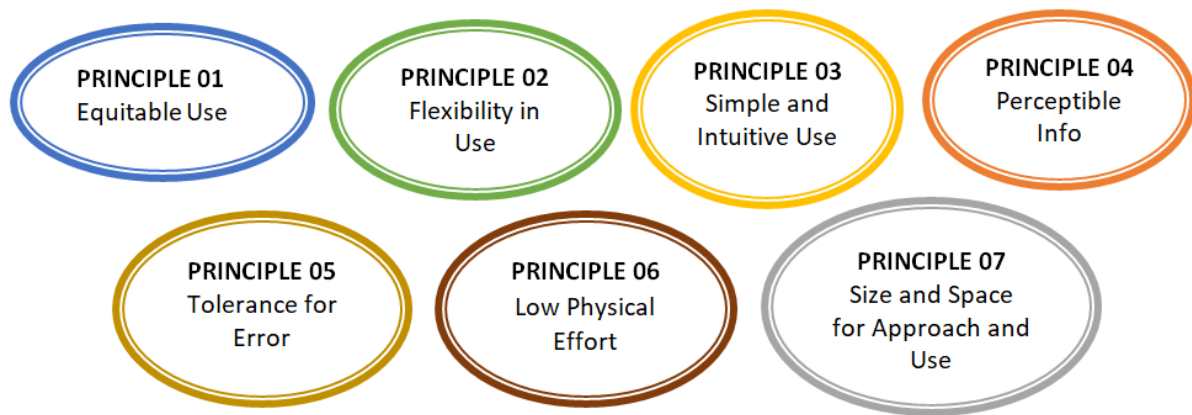


Figure 16 Seven Principles of Universal Design [4]

The universal design principles used for Norwegian Rails prototype implementation are described as follows:

PRINCIPLE ONE: Equitable Use

The idea behind this principle is that the system design can be provided and used for people with diverse abilities. The system design provides same ways for use, exact whenever feasible and equivalent when not possible. The design should avoid separating or stigmatizing any user groups. The security, privacy and safety features are available to the kinds of users where there the personal information is safeguarded. [5]

The system design could be accessed by all kinds of users except visually impaired people.

PRINCIPLE TWO: Flexibility in Use

The principle is that the design accommodates a wide range of individual preferences and abilities. [5]

The prototype gives options to search trains for journey, reserve seats for journey. It also allows to cancel or change the reservation at later point of time.

PRINCIPLE THREE: Simple and Intuitive Use

According to this principle, the system design should be simple to use and should be designed irrespective of user experience, technical or language skills. [5]

The prototype implementation is not complex in nature and easy to use. The prototype is designed according to user expectation. Currently it supports English language but for future a new feature could be implemented for having the language option in Norwegian as well. The information screens are loaded logically in a serial way according to the nature of importance. The user is always provided with confirmation or feedback for his actions.

PRINCIPLE FOUR: Perceptible Information

This principle states that the design should be able to communicate required details to the user efficiently and effectively in a way it could be easily perceived without any troubles to all kinds of users irrespective of their sensory abilities. [5]

The information displayed using various graphical interactive style and design to display essential information to the user. For instance, once the user reserves a seat for the journey, a confirmation message is displayed to the user about it.

PRINCIPLE FIVE: Tolerance for Error

This principle means that the system design is very minimally affected in case of accidental or unintentional transactions by the user.[5]

The prototype implementation for Norwegian Rails is error tolerant. Also, there is logic implemented for doing error checks and the user is prompted with the error and information wherever there is an unintended action by mistake.

PRINCIPLE SIX: Low Physical Effort

This principle means that the system design can be used with comfort and ease with minimum amount of stress [5].

The prototype implementation can be used comfortably and with less fatigue and has interactive functionality. This is helpful even for the users who are not experienced with using software applications.

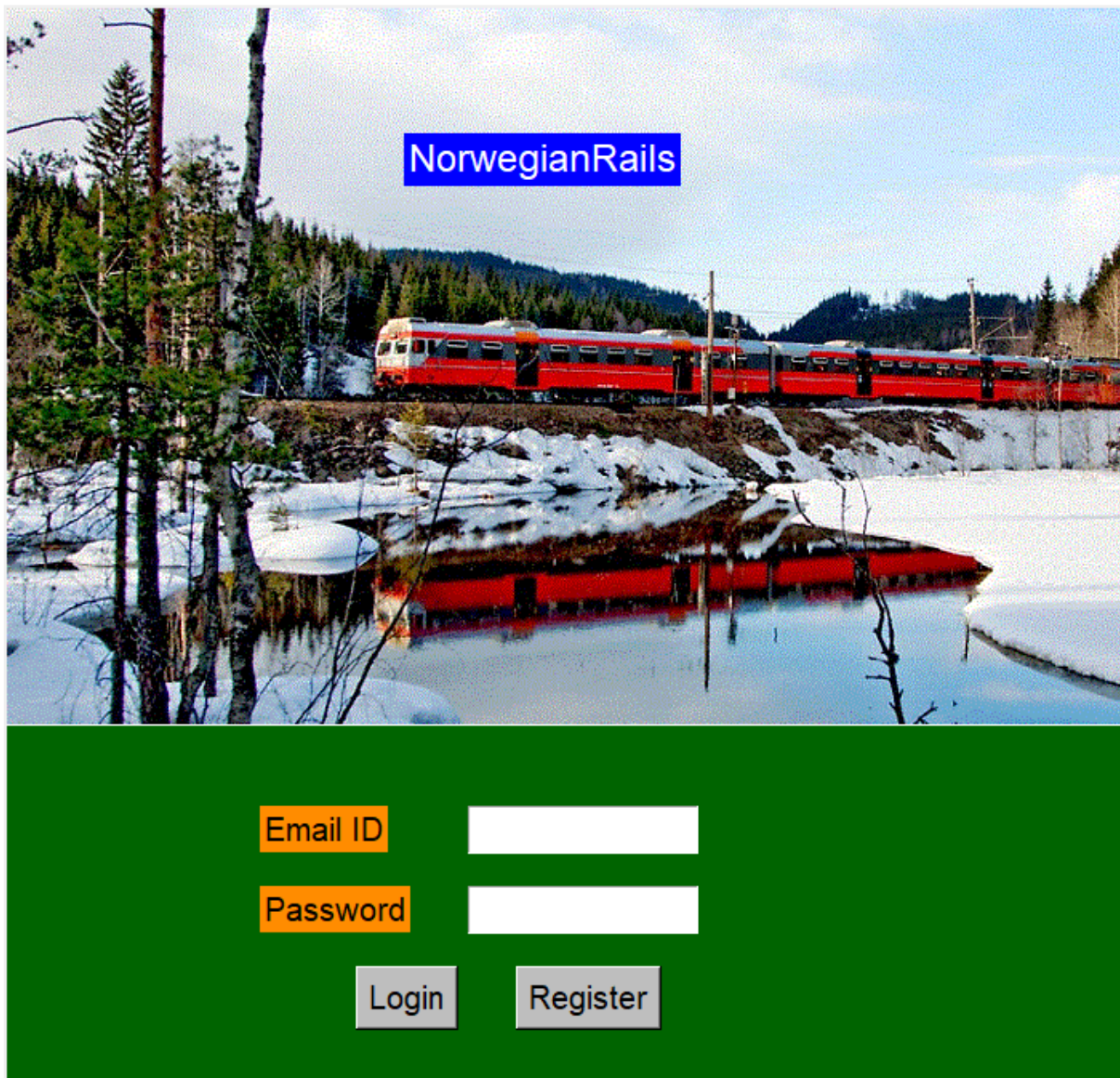
PRINCIPLE SEVEN: Size and Space for Approach and Use

There is adequate size and space provided for approach, use, manipulation irrespective of physical body size, posture or mobility of the user. [5]

The current design supports sufficient size and space to show the demo prototype. But in future, when the prototype solution needs to be implemented on a large scale, there would be a need for large size and space to accommodate these changes.

SCREENSHOTS OF RUNNING PROTOTYPE

User Login Screen



The image shows a web application window titled "76 Railway reservation". The main visual is a scenic photograph of a red and white Norwegian Rail train traveling through a snowy, forested landscape. A blue rectangular box with the text "NorwegianRails" is overlaid on the upper part of the image. Below the image, there is a green rectangular area containing the login and registration form. The form consists of two rows of labels and input fields: "Email ID" and "Password", each followed by a white text input field. At the bottom of the green area are two grey buttons labeled "Login" and "Register".

NorwegianRails

Email ID

Password

Login Register

Provide the Email ID and Password input and click on Login button for login.

In case of new users, click on Register button to Sign Up

New User Registration Screen

Welcome to NorwegianRails

Kindly provide the below details to Sign Up

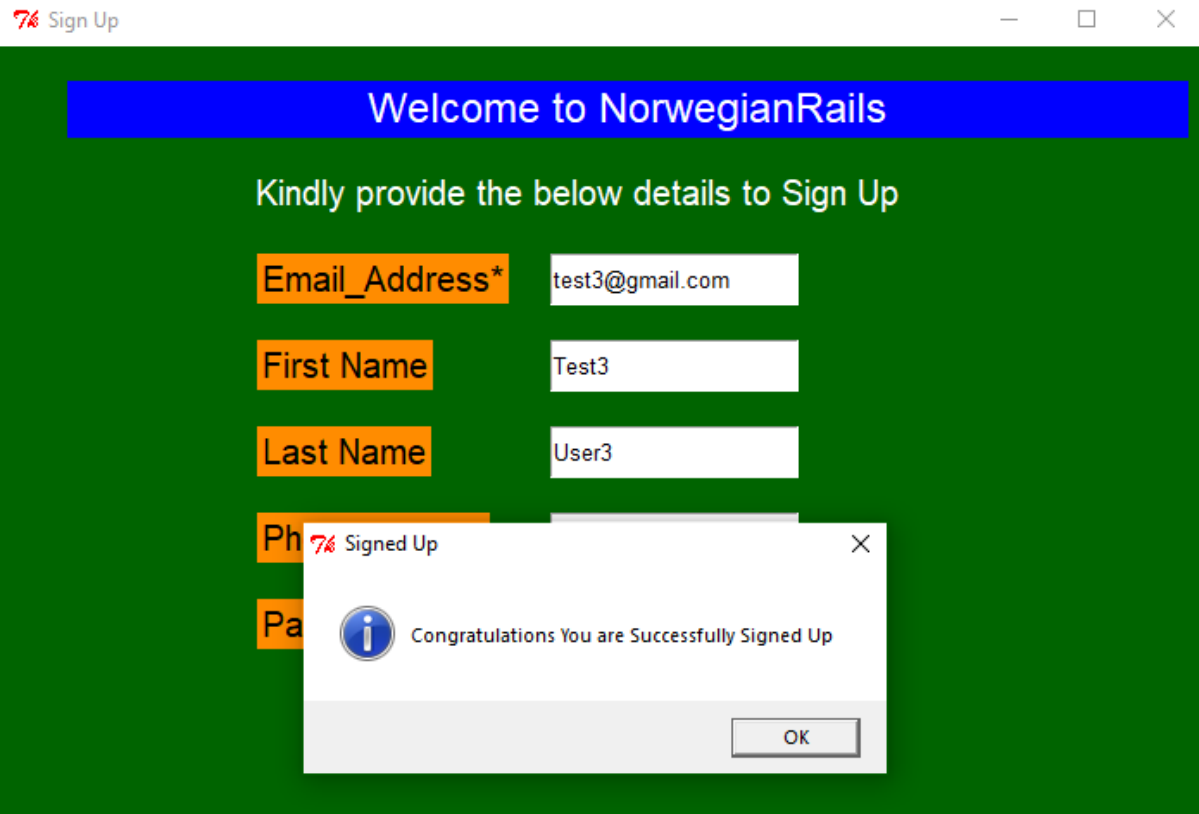
Email_Address*	<input type="text"/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Phone Number	<input type="text"/>
Password	<input type="password"/>

Enter the fields Email_Address, First Name, Last Name, Phone Number and Password. Click on SignUp button to create New User. Click on Back button to go back to the Main screen

Welcome to NorwegianRails

Kindly provide the below details to Sign Up

Email_Address*	test3@gmail.com
First Name	Test3
Last Name	User3
Phone Number	78623112
Password	*****



Choose Option Screen



Click on Purchase Ticket to Search train journeys

Search Trains Screen

74 Search Trains

Where do you want to travel?

Departure Station

Arrival Station

Date of Travel

74 Search Trains

Where do you want to travel?

Departure Station

Arrival Station

Date of Travel

Click on Search Trains

74 RAILWAY SCHEDULE

Train number	Source	Destination	Date of Travel	Departure Time	Arrival Time	Travel Time	
124	Oslo	Bergen	2020/11/20	11:30:00	21:30:00	10 Hrs	<input type="button" value="Book"/>

Click on Book

76 Passenger Details

S.No	Name	Age	Gender	Category
1	Tom	23	Male	Student
2				
3				
4				

Proceed

Back

Seat Reservation

76 Seat Reservation

Do you wish to reserve a seat for journey

Yes

No

76 Seat Reservation

Do you wish to reserve a seat for journey

Yes

No

Choose a seat to reserve

Select the Train Carriage

C1

Select Seat Number

2

Confirm Reserve Seat

Cancel

Where do you want to travel?

Departure Station

Oslo

Arrival Station

Bergen

Date of Travel

2020/11/20

7% Seat Reserved



Seat is Reserved

OK


Payment screen

How do you want to pay?

Select Payment Method

OK

How do you want to pay?

VIPPS 

OK

Provide the VIPPS details

VIPPS ID*

79134293

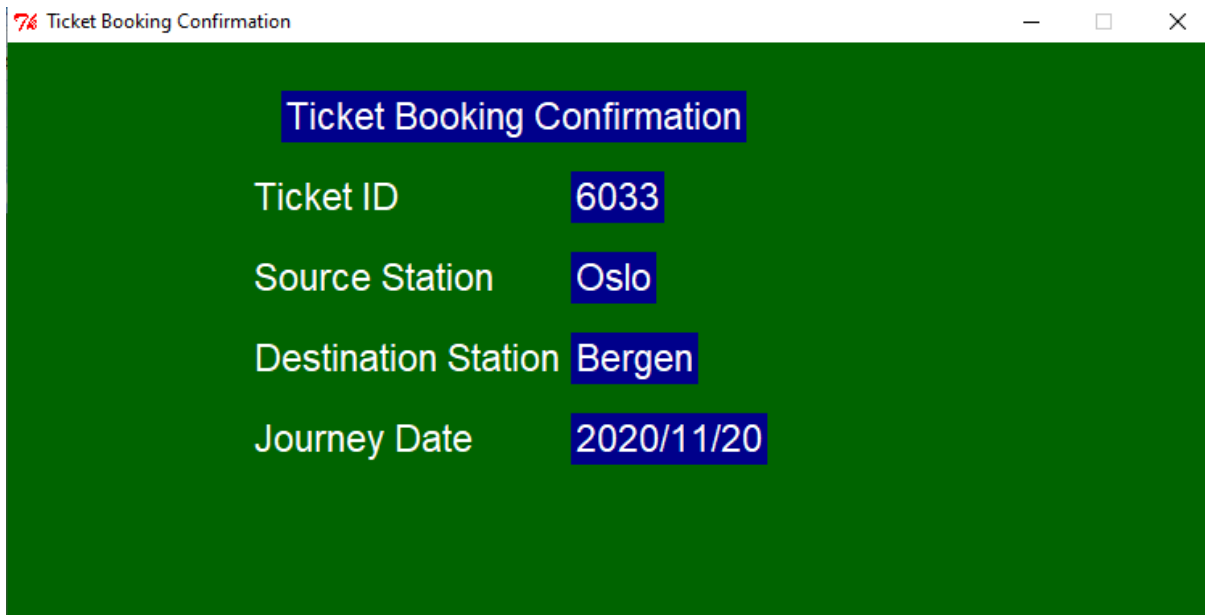
OK



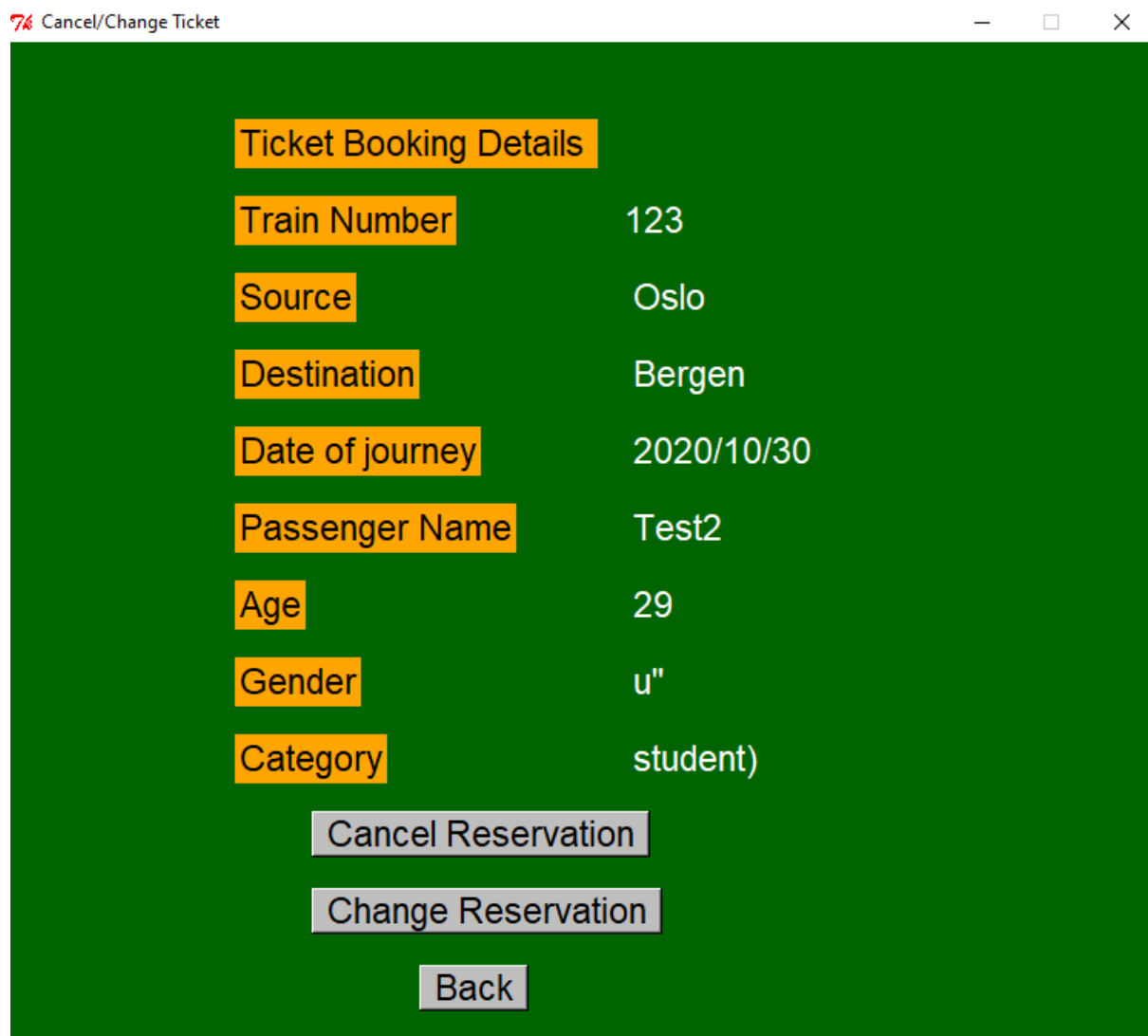
Ticket is booked

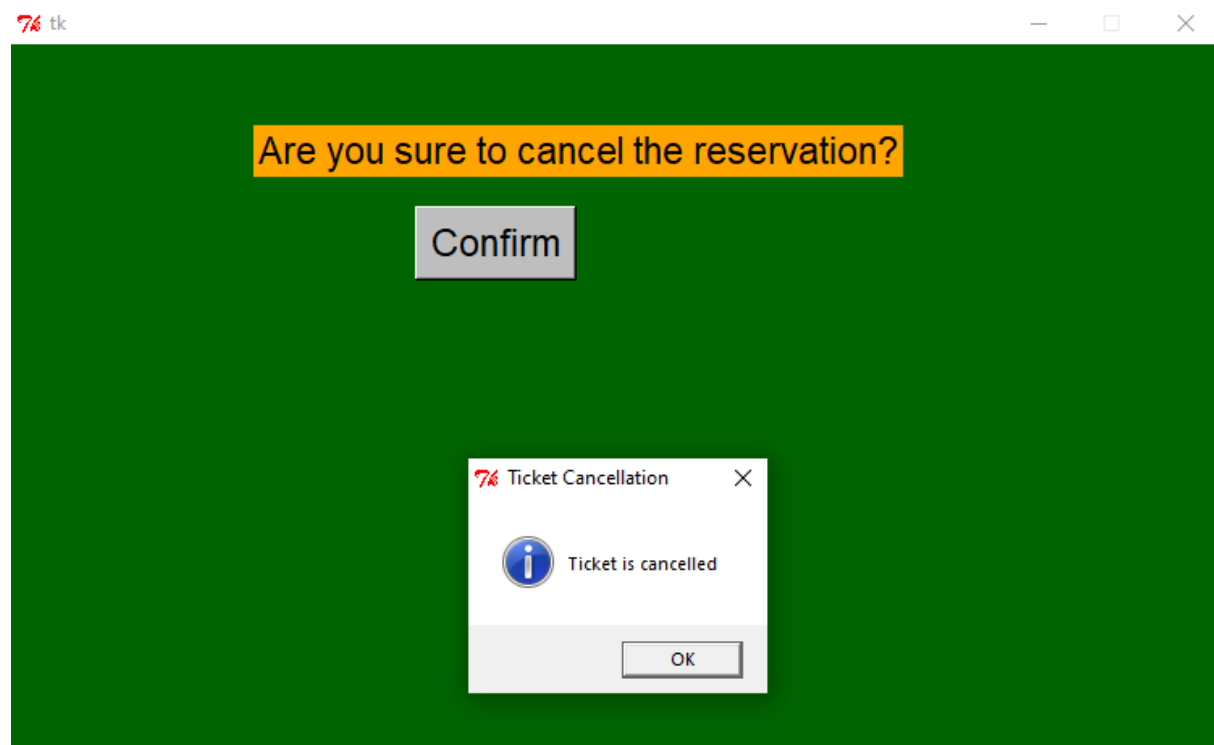
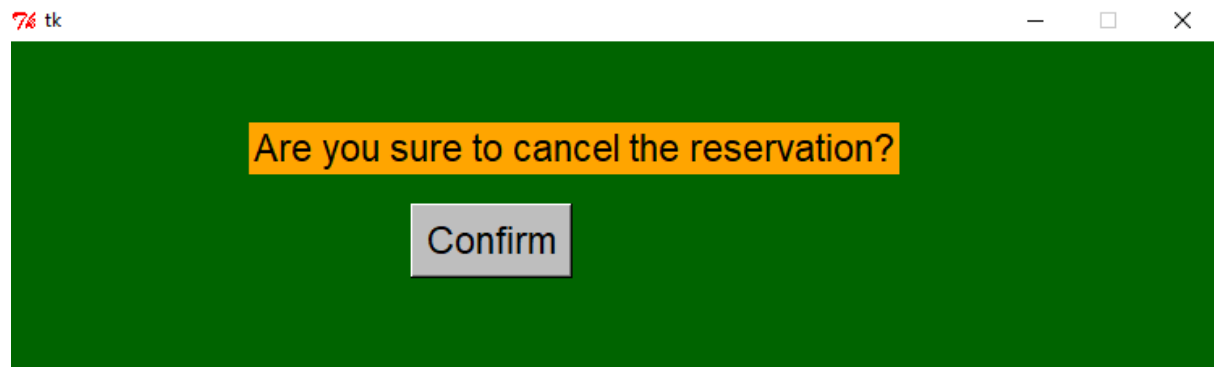
OK

Ticket booking Confirmation



Cancel or Change ticket

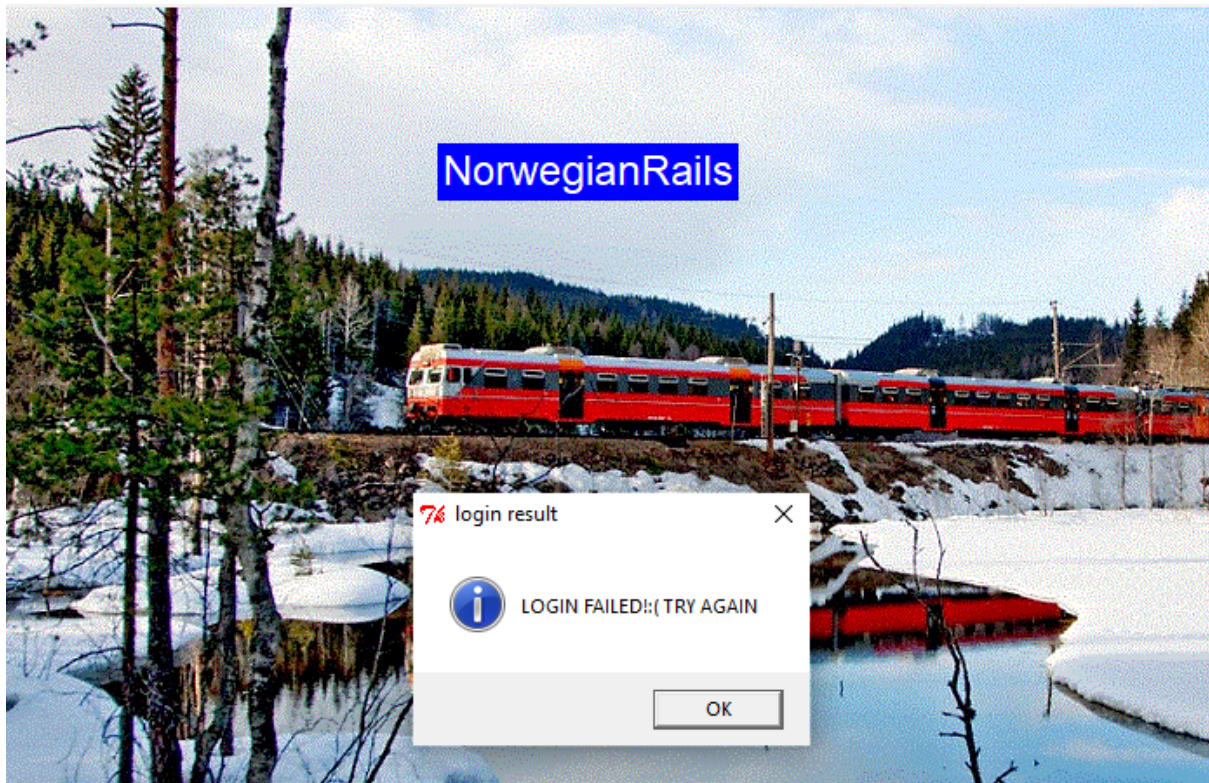




Show Ticket for Control/Validation

Ticket Booking Details	
Ticket Number	2621
Train Number	123
Source	Oslo
Destination	Bergen
Date of journey	2020/10/30
Passenger Name	Test2
Age	29
Gender	u"
Category	student
<input type="button" value="OK"/> <input type="button" value="Back"/>	

Login Failed



NorwegianRails

7% login result



LOGIN FAILED: (TRY AGAIN

OK

Email ID

test1@gmail.com

Password

Login

Register

Welcome to NorwegianRails

Kindly provide the below details to Sign Up

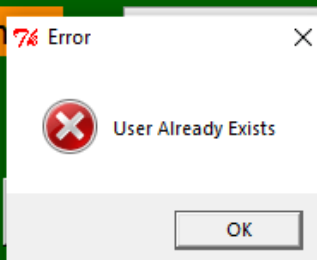
Email_Address* test1@gmail.com

First Name tom

Last Name test

Phone Number

Password

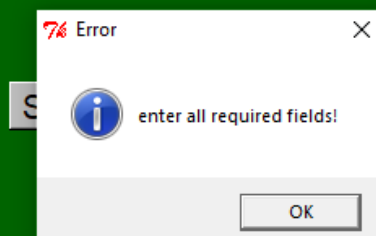


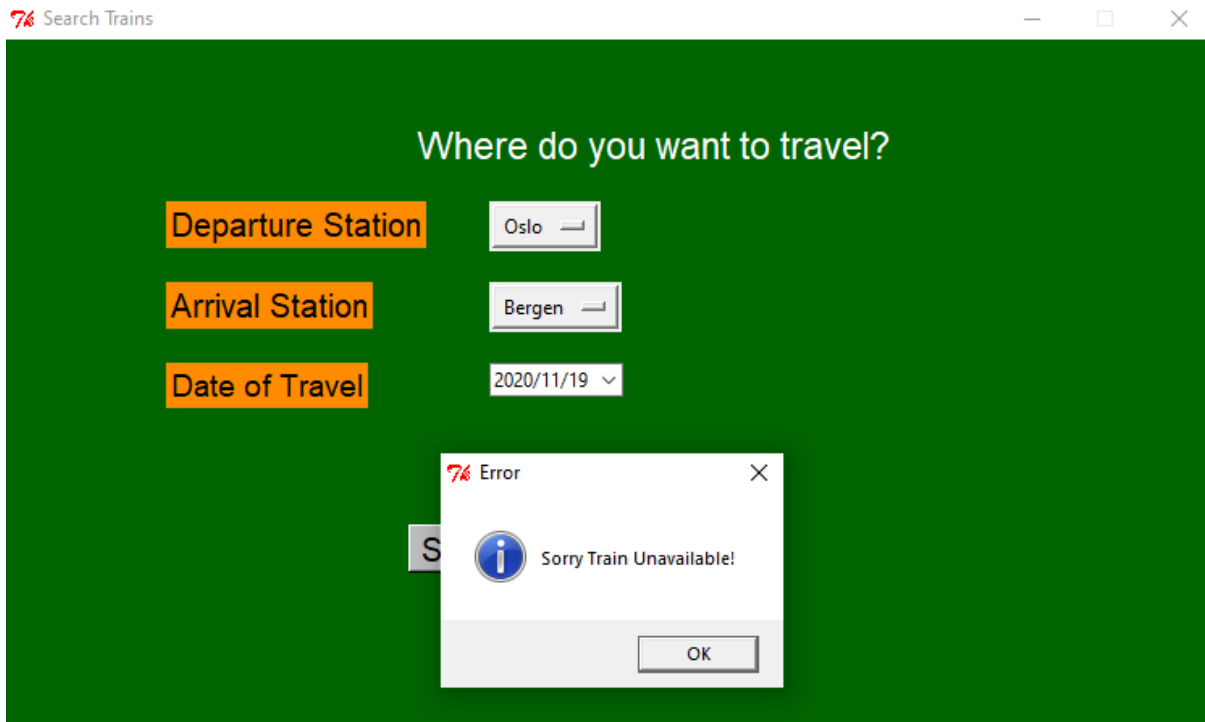
Where do you want to travel?

Departure Station Select Source

Arrival Station Select Destination

Date of Travel 2020/11/19





DISCUSSION AND FUTURE SCOPE

Due to time constraints, not all of the features could be implemented and there is a lot of scope for the prototype to be enhanced with more features in future. Following are some of the features

- Forget Password functionality
- User changes his details
- Admin User functionality
- App language selection in both English and norsk
- Add-ons to the journey
- Discounts applicable to the user groups such as Student, children and senior citizen
- Captcha when signing into the account to avoid the logging of robot

APPENDIX

GitHub Repository Link : <https://github.com/AMITAKASHIKAR/ACIT4070-Programming-API-and-Interaction>

REFERENCES

- [1] ISO25000 software and data quality

<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

[2] Priyal Walpita, *Software Architecture Patterns*, July 2019

<https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057>

[3] *From Wikipedia, Multitier architecture*

https://en.wikipedia.org/wiki/Multitier_architecture

[4] *7 Principles of Universal Design*

<http://bijoumind.com/what-is-universal-design/>

[5] *The Seven Principles of Universal Design*

<https://www.udll.com/media-room/articles/the-seven-principles-of-universal-design/>

[6] Missak Boyajian, *Question on Repository and Client Server Architecture*, 14, November 2013

<https://www.codeproject.com/Questions/682701/repository-and-client-server-architecture#:~:text=A%20repository%20architecture%20is%20a,to%20share%20the%20same%20data.&text=Components%20can%20be%20interchanged%20and,be%20a%20database%20management%20system.>

[7] *Gangs of Four Design Patterns*

<http://www.blackwasp.co.uk/gofpatterns.aspx>