RAILWAY RESERVATION SYSTEM

UCS2265 – Fundamentals and Practice of Software Development

A PROJECT REPORT

Submitted By

VIJAY K- 3122235001157
VIKRAMAN S- 3122235001159
BYSANI VISHAL-3122235001161

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)
Kalavakkam – 603110

June 2024

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report titled FOOD DELIVERY SYSTEM is the bonafide work of VIJAY K(3122235001057), VIKRAMAN S (3122235001059) and BYSANI VISHAL (3122235001061)" who carried out the project work in the UCS2265 – Fundamentals and Practice of Software Development during the academic year 2023-24.

Internal Examiner External Examiner

Date:

TABLE OF CONTENTS Page Number

Problem Statement:

Develop an integrated railway ticket booking system that enables users to book, cancel, and pay for tickets with a user-friendly interface akin to flight or bus booking systems. The system supports day and night trains, connecting trains, and allows seat selection for enhanced convenience. Usage of Zeller's Congruence Algorithm to find they day for any week in calendar and checks if train exists or not on that day. It also offers additional services such as food and porter bookings. Utilizing Dijkstra's algorithm, the system suggests the shortest path trains for a specific South Zone region. This project aims to improve user convenience and optimize travel planning.

Extended Exploration of Problem Statement:

1.Booking and Cancellation:

- **Ticket Booking**: Design a booking interface like flight booking systems, allowing users to select their departure and arrival stations, view available seats, and choose specific seat numbers.
- **Train Schedules**: Include options for day and night trains as well as connecting trains to give users flexible travel options. Use of Zeller's congruence algorithm to find the day of week for any date in Calendar and show if the trains are available on that day or not.

- **Cancellation**: Allow users to easily cancel their bookings with clear refund policies and procedures.

2.**Route Optimization:**

- **Shortest Path Calculation**: Integrate Dijkstra's algorithm to provide users with the shortest and most efficient routes within the South Zone, helping them make informed travel decisions.

3.**Additional Services:**

- **Food Booking**: Enable users to pre-order meals for their journey, ensuring availability and timely delivery.
- **Porter Booking**: Allow users to book porter services in advance, making their travel experience more comfortable and hassle-free.

4.Payment and PNR Status:

- **Payment:** We have a payment wallet so that user can add money and pay for tickets accordingly.
- **PNR Status**: Provide real-time PNR status checking to keep users informed about their booking status and any updates.

5.Admin Services:

- **Train Management**: Develop an admin module that allows administrators to add, update, and manage train details, including schedules, routes, and available seats.

- **Service Management**: Allow administrators to manage details of porter services and restaurant options, ensuring that the ancillary services are up-to-date and accurate.

6.User Experience:

- **Interactive Seat Selection**: Implement a seat selection interface that allows users to choose their preferred seats, like the experience provided by airline booking systems.
- **User-Friendly Interface**: Ensure the system is easy to navigate, with a clean and intuitive design that enhances the overall user experience.

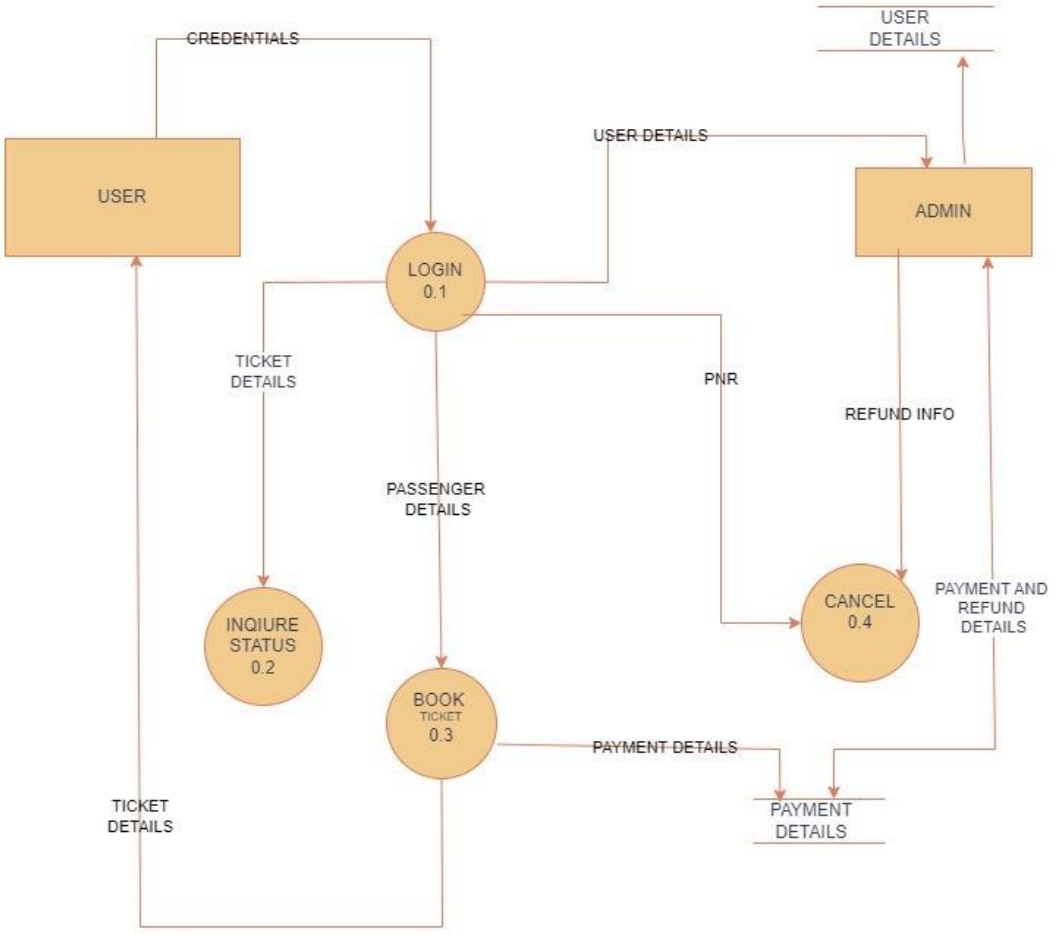
DATA FLOW DIAGRAMS:

Level 0 DFD:

The user enter the Railway reservation system by entering his/her own credentials which is already been stored as user details in the system and the system verifies the credentials and identifies the user and opens into an interface corresponding to

the user and then the user is allowed to choose what he wants to do(i.e:whether to book/cancel a ticket or check the status of the booked ticket) which is then continued by a few steps according to the choice the user makes and then the details are sent to the admin and the admin sends the data that the user requires whether be the confirmation of cancellation or booking of ticket.

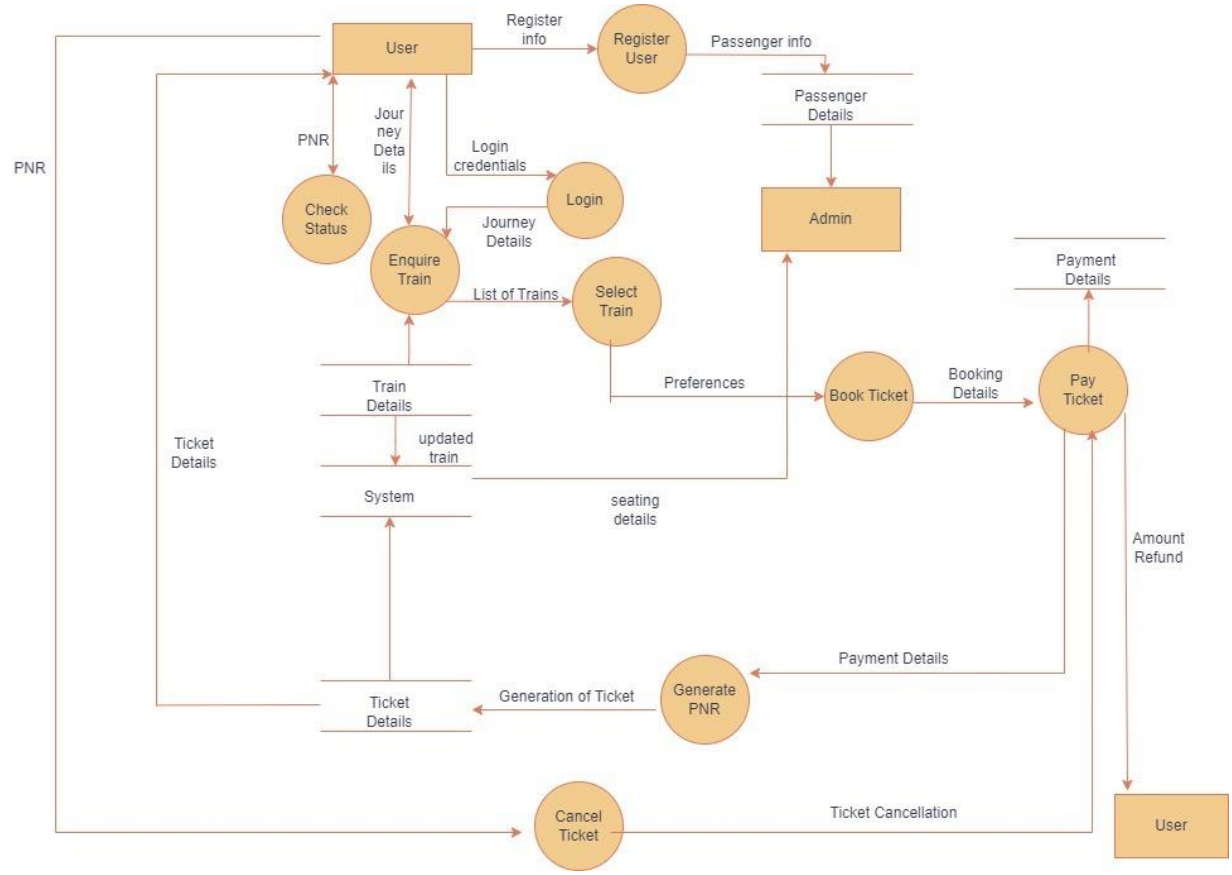
DFD Level 1:



This data flow diagram (DFD) illustrates the interaction between a user, an admin, and various processes within a ticket booking and cancellation system. The user provides credentials to the login process, which then generates user details for the admin. Users can inquire about ticket status by providing ticket details, and the system returns the ticket details to them. For booking

tickets, users submit passenger details to the booking process, which processes payment details and issues tickets. In case of cancellation, users provide the PNR to the cancel process, which handles payment and refund details. The admin manages user details and processes refund information, integrating with the cancellation process. Overall, this DFD effectively maps out the flow of information within the ticket booking and cancellation system.

DFD Level 2:



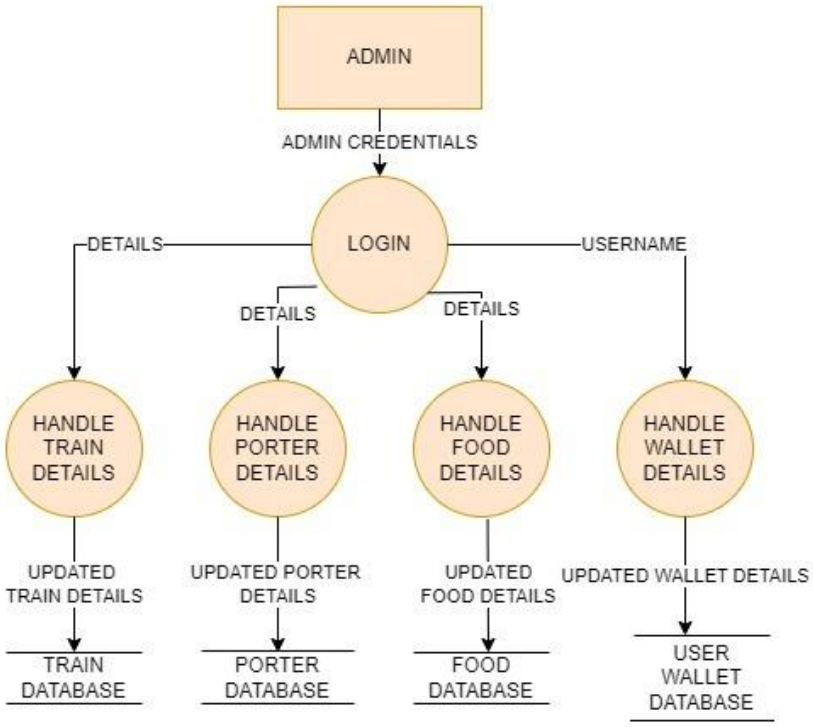
This data flow diagram (DFD) represents the flow of information in a train ticket booking and cancellation system. The process begins with the user registering their information, which includes passenger details, through the "Register User" process. Once registered, the user logs in using their credentials. After logging in, the user provides journey details to inquire about available trains through the "Enquire Train" process, which returns a list of trains based on the user's preferences. The user selects a train, and this selection is processed by the "Select

Train" function, which considers seating details and train details managed by the system.

The "Book Ticket" process then captures booking details and sends them to the admin for verification. Once verified, the user proceeds to the "Pay Ticket" process where payment details are provided. Upon successful payment, a PNR (Passenger Name Record) is generated through the "Generate PNR" process, which finalizes the ticket booking and provides ticket details to the user.

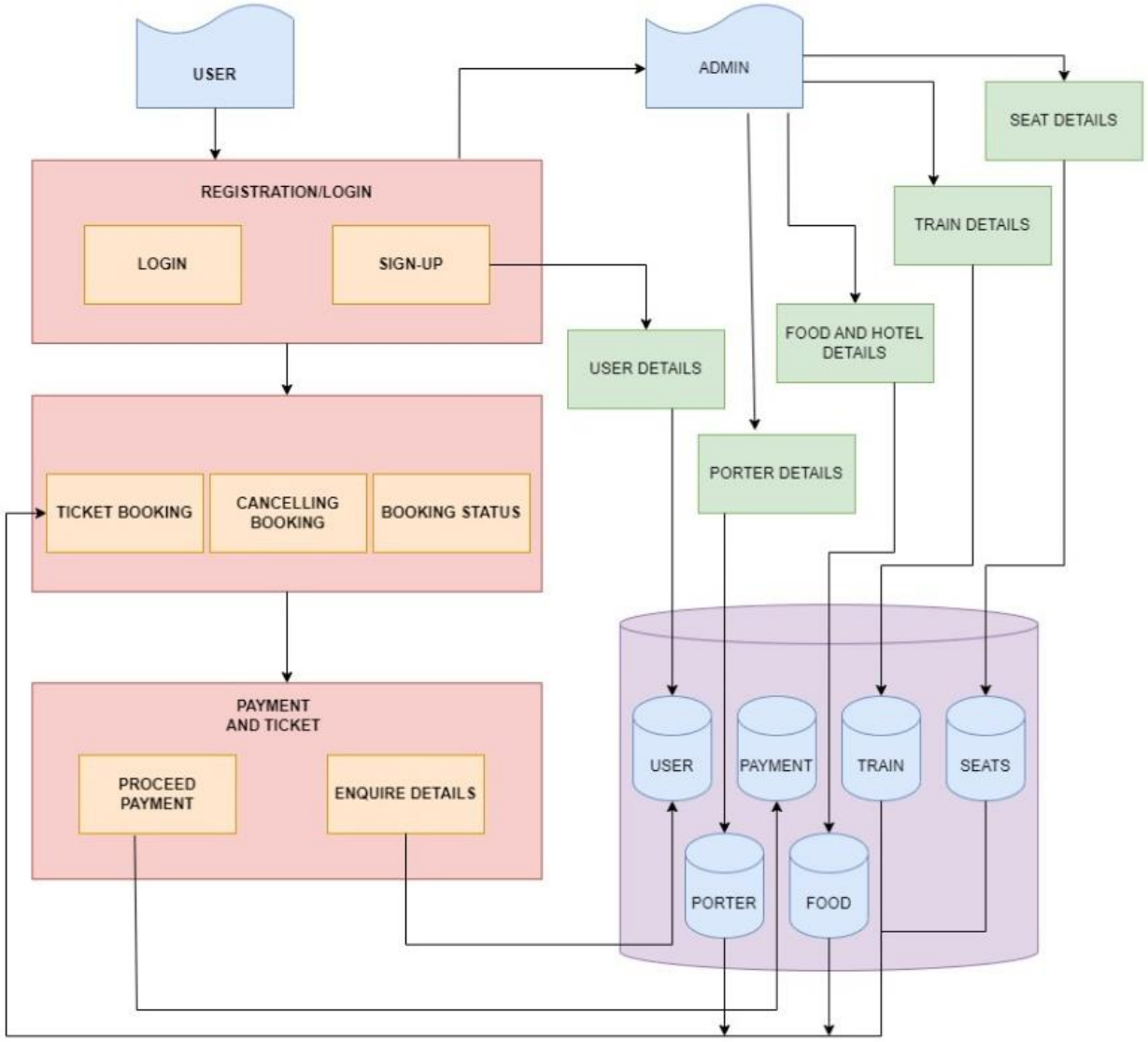
If the user needs to check the status of their booked train, they can do so through the "Check Status" process using their PNR. In the event of a cancellation, the user initiates the "Cancel Ticket" process, which handles the ticket cancellation and processes the refund. The system manages all updates to train details and seating arrangements, ensuring that the latest information is always available. Overall, this DFD provides a comprehensive view of the interactions and data flows within the train ticket booking and cancellation system, ensuring a seamless experience for both the users and the admin.

LEVEL 2 ADMIN DFD



This DFD describes what the admin can do, i.e. the admin can handle the addition of train details, any new porters or restaurants around town with their menus, the admin can handle the details of the wallet of the user accordingly and later accordingly these updated details are stored in the corresponding databases which are specific for all the 4 tasks.

ARCHITECTURE DIAGRAM:



The diagram is a representation of a train reservation system, showing the interaction between users, admin, and various system components. At the top, there are two entities: the user and the admin. Users interact with the system primarily through the Registration/Login module,

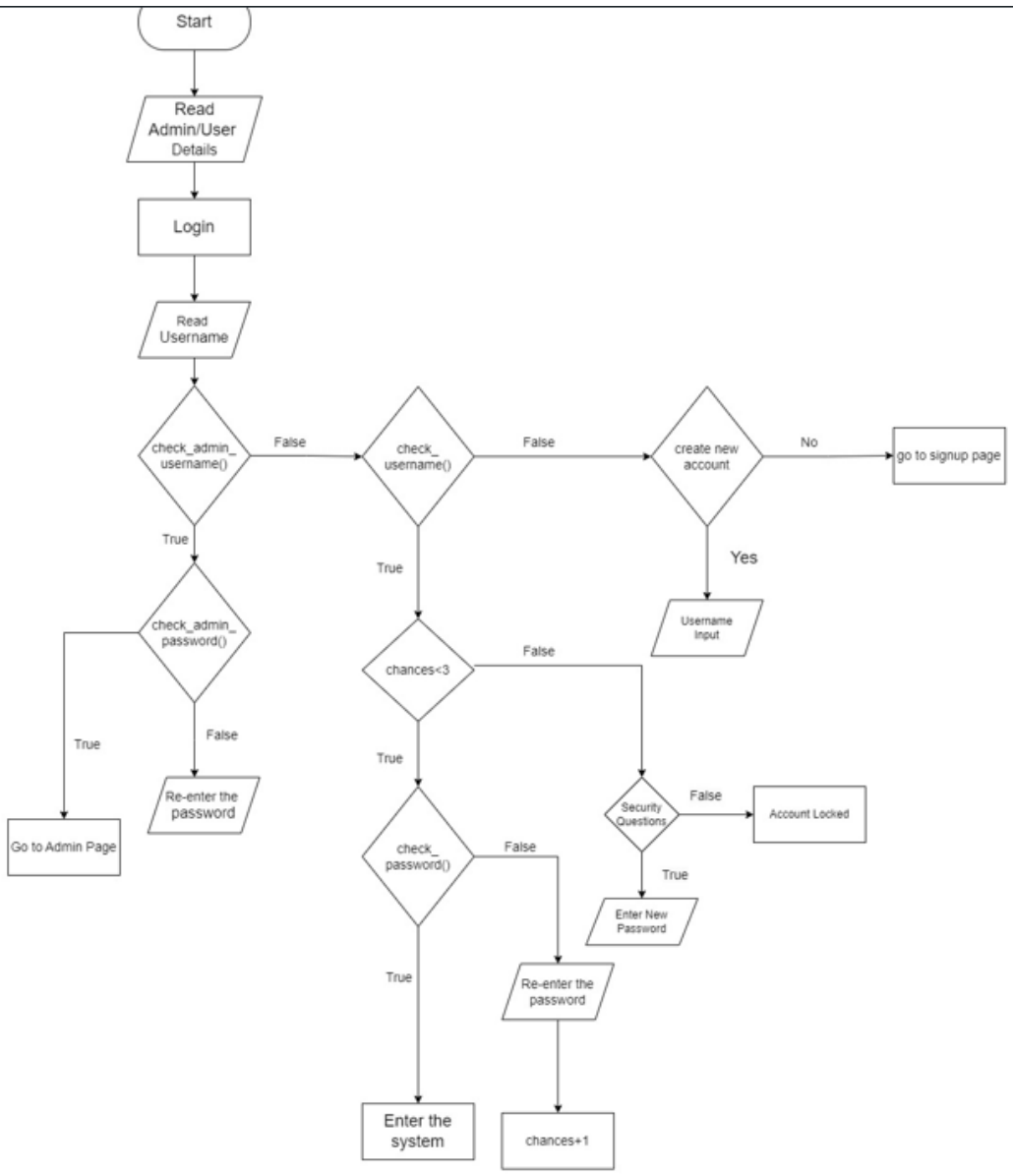
where they can log in or sign up. Once authenticated, users can engage in ticket booking, cancel booking, and check booking status. These functions are grouped under the Ticket Booking module.

Upon booking a ticket, users proceed to the Payment and Ticket section to complete their payment and can also inquire about payment details. The admin has broader access to manage the system, including viewing and updating user details, and managing seat, train, porter, and food and hotel details. This information is stored in the system's database, which comprises multiple arrays for users, payments, trains, seats, porters, and food services.

The diagram highlights the flow of data and interactions among the different components. Admins feed into various system details which are then utilized by the user-facing modules. The centralized database supports efficient system operations. This design ensures that users can efficiently book and manage their train journeys while administrators maintain control over the operational aspects of the reservation system.

FLOWCHART OF MODULES:

Login-Module:



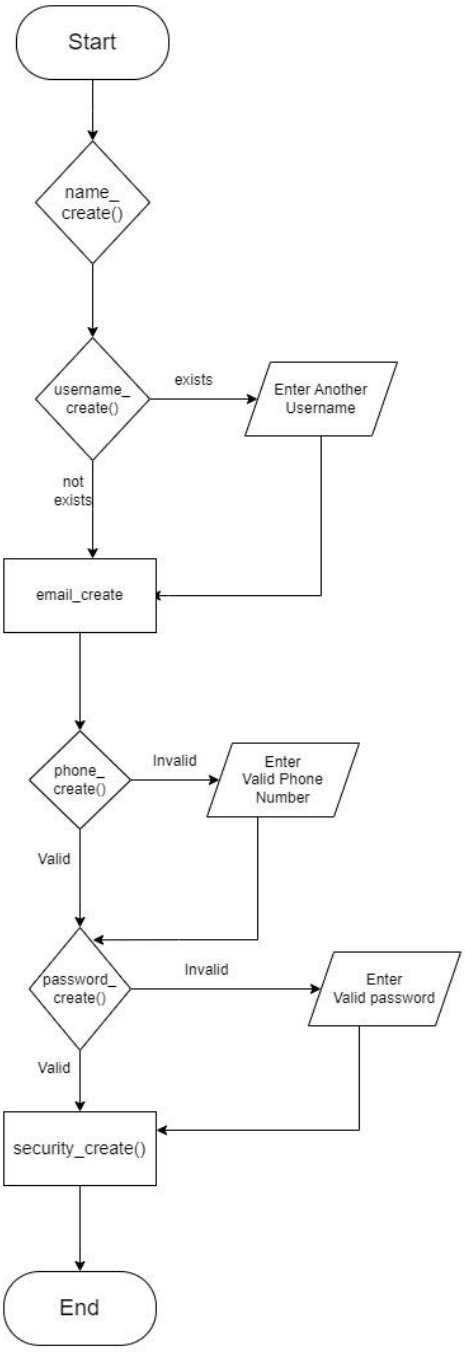
Explanation:

The function login_user() gets credentials from user.

The functions check_admin_username(), check_username(), check_password() and verify_password() check if the user is admin,if the user has signed up already, if the password is correct and if the password passes the password criteria.

The functions recovery_password() and change_password() Implement the process of recovering the password by using security questions if the user forgot the password.

Signup Module:



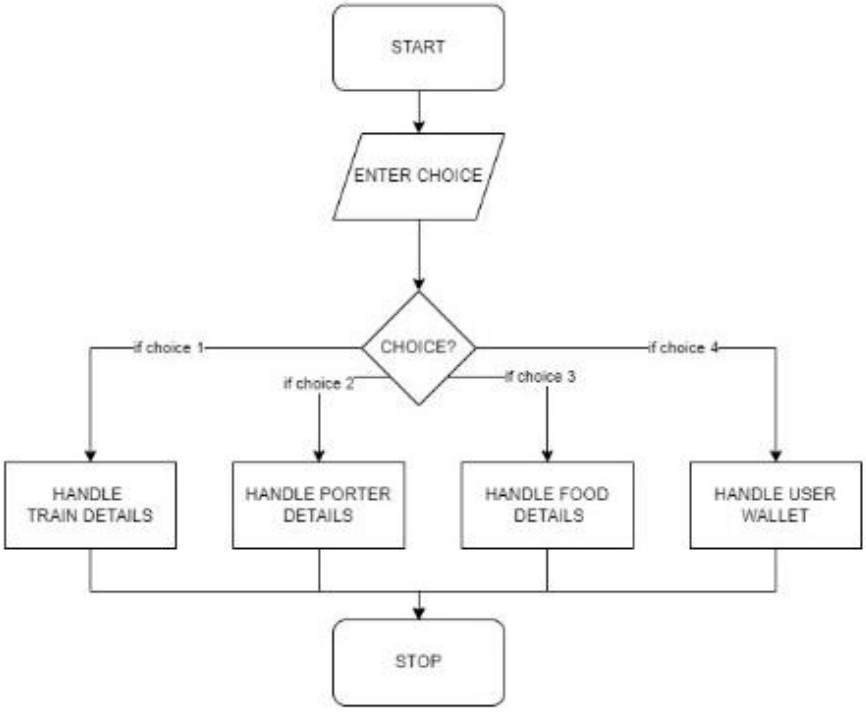
Explanation:

Register Function:

All the functions in the module implement registering the user's detail in userdb file using **signUp()** function.

And the stored data is verified so that no fault is there using the other functions.

Admin Module:



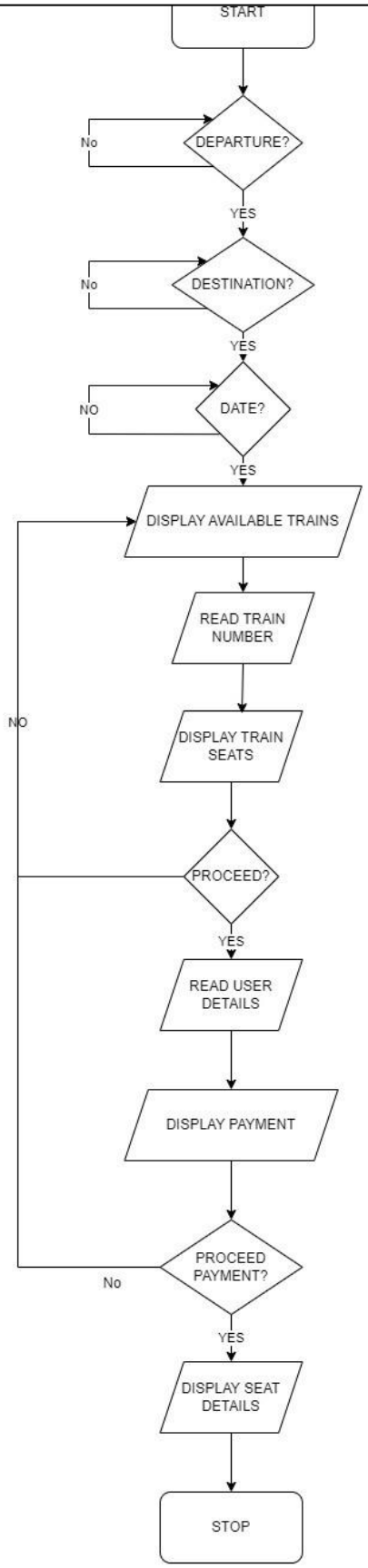
Explanation:

trains_handle(): This function is to handle the train details

pay_store() : This function is to add money to the wallet

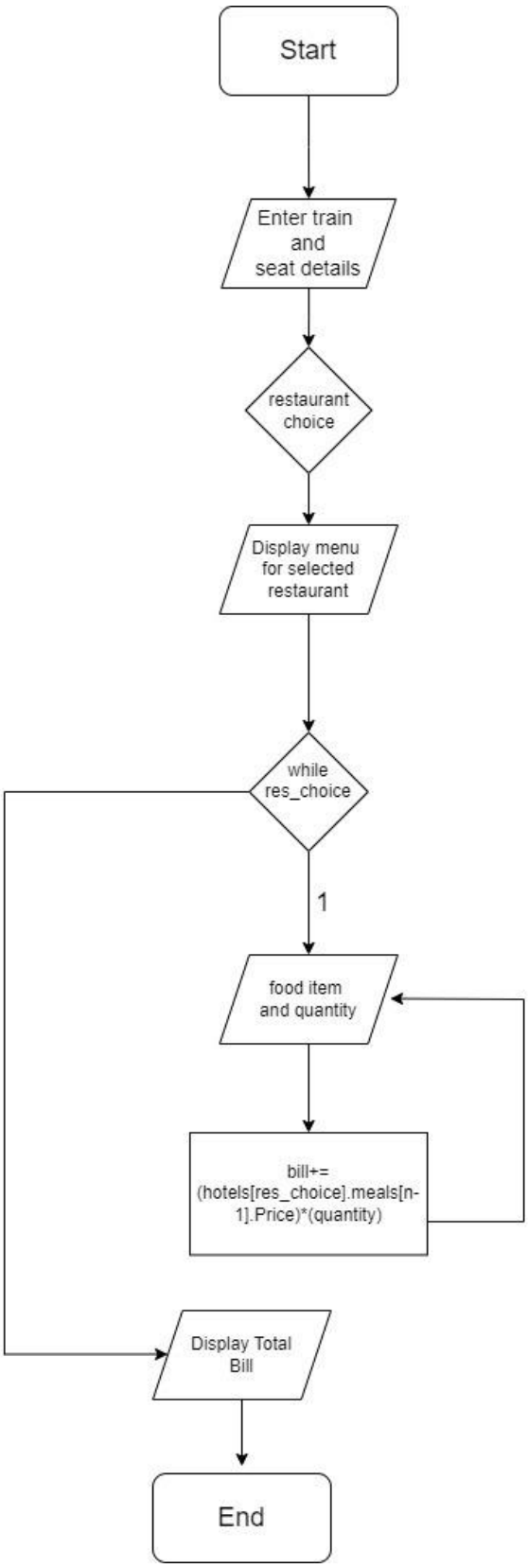
create_food(): This function is to add new restaurant and menu details.

BOOKING MODULE:



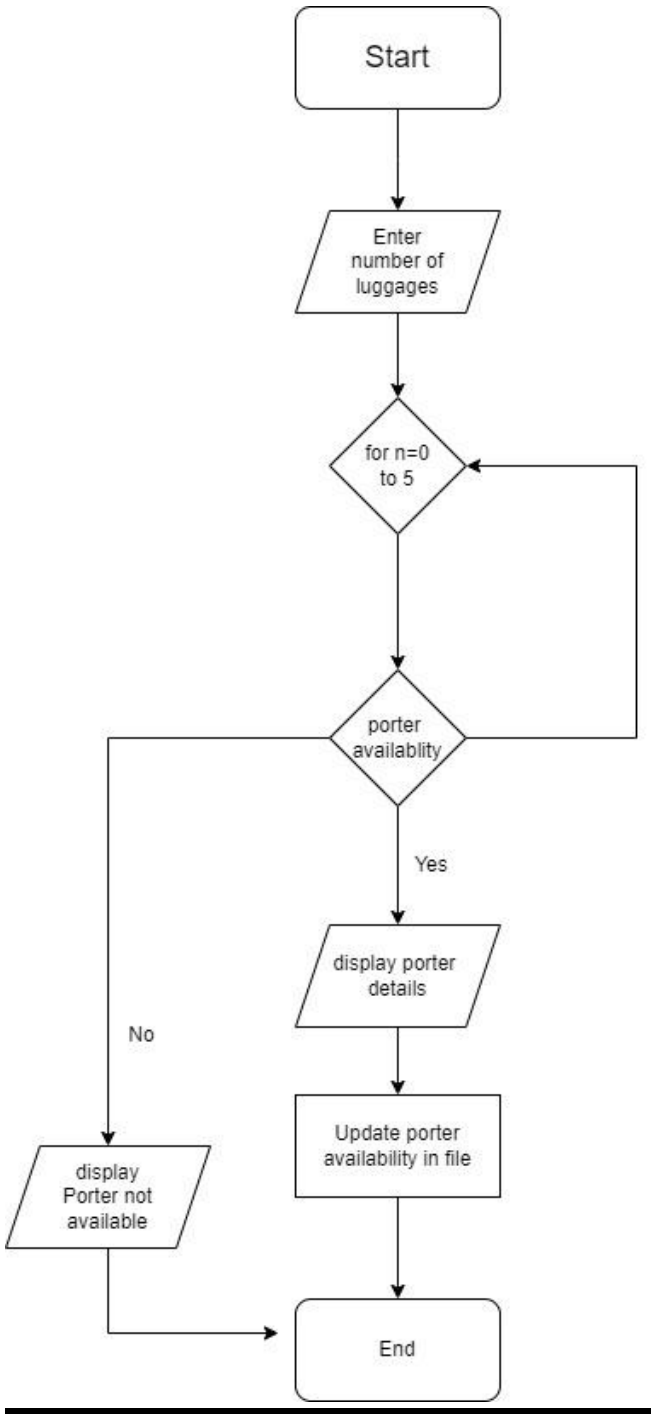
The flowchart details the steps for booking a train ticket, starting with the user initiating the process. First, the system checks if the departure location is entered; if not, it returns to the start. If yes, it then checks for the destination, following the same return logic if absent. Upon entering the destination, the system asks for the travel date. If the date is provided, the system displays available trains for the specified route and date. The user selects a train by entering its number, and the system shows the available seats on that train. The user is then asked if they wish to proceed with the booking; a negative response loops back to displaying trains, while a positive one moves to reading user details. After collecting user information, the system displays payment details and asks if the user wants to proceed with the payment. If the user decides not to proceed, the process loops back to displaying trains; if they proceed, the system finalizes the booking by displaying the seat details, marking the end of the process.

Food ordering module:



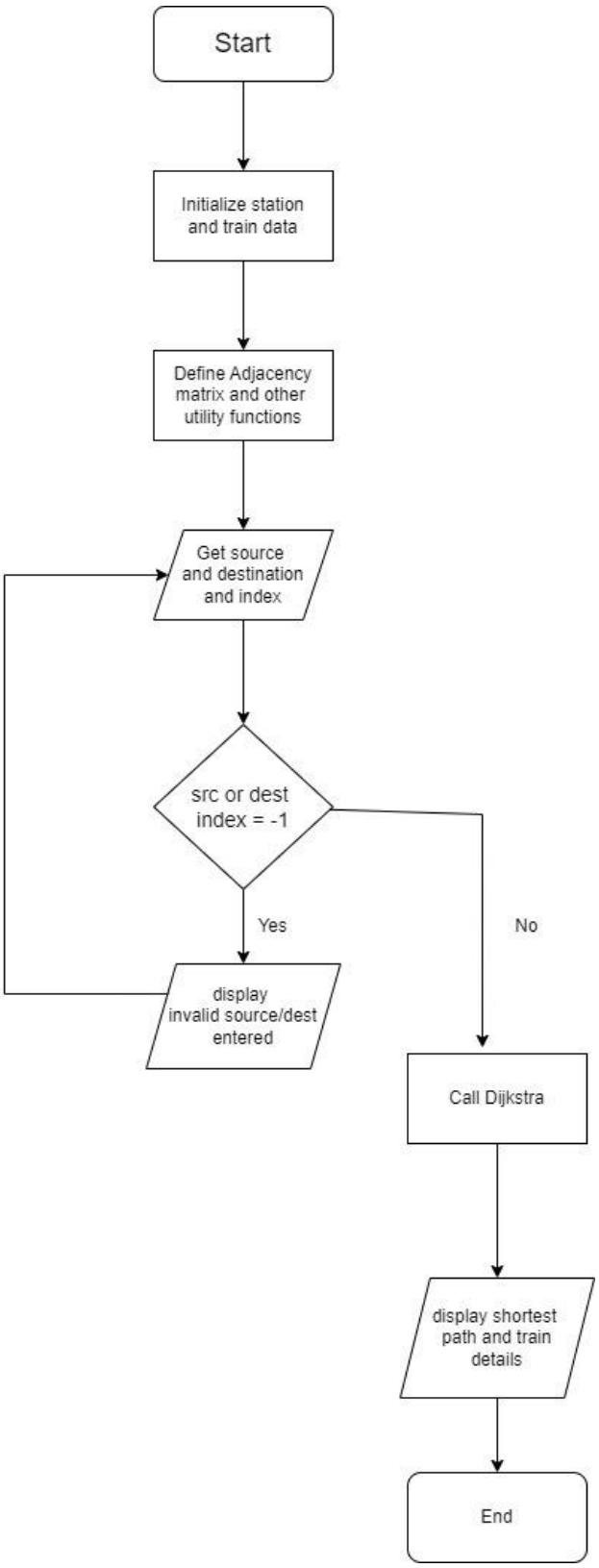
- Here the users can choose the restaurant they want along with that, they can access the menu and order how many ever items they want along with the required quantity.

Porter Booking Module:



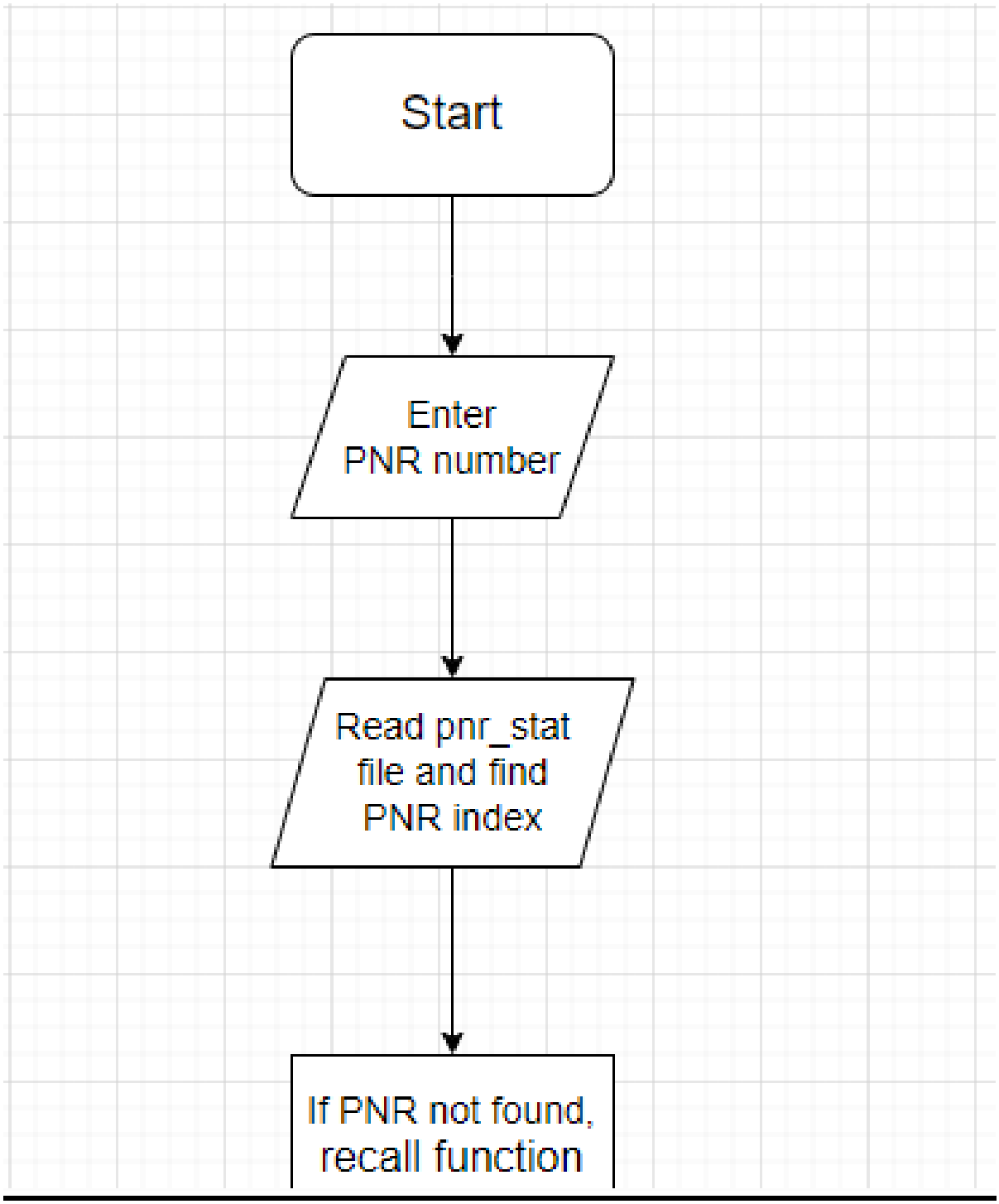
- Here the user can enter the details about the Number of luggage's and therefore the porter will get assigned accordingly based on the luggage requirement.

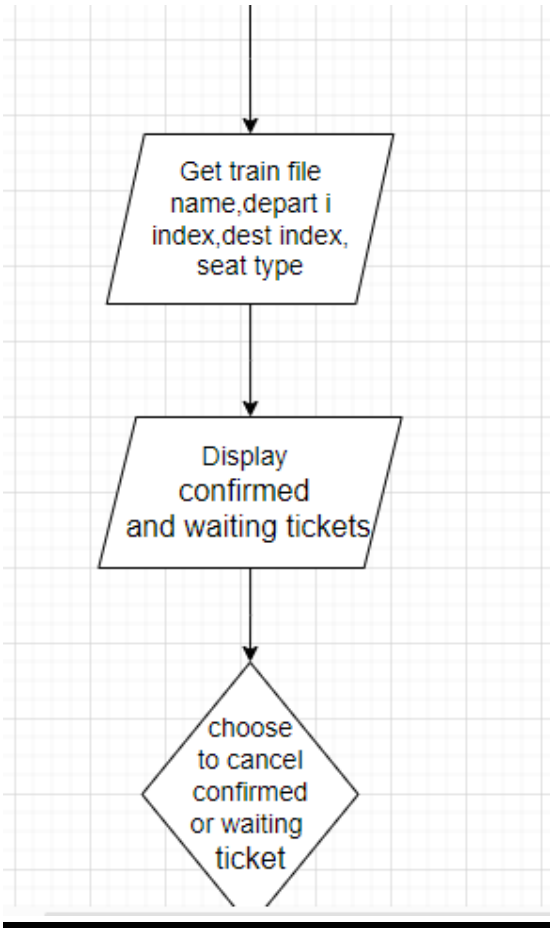
Dijkstra Module:

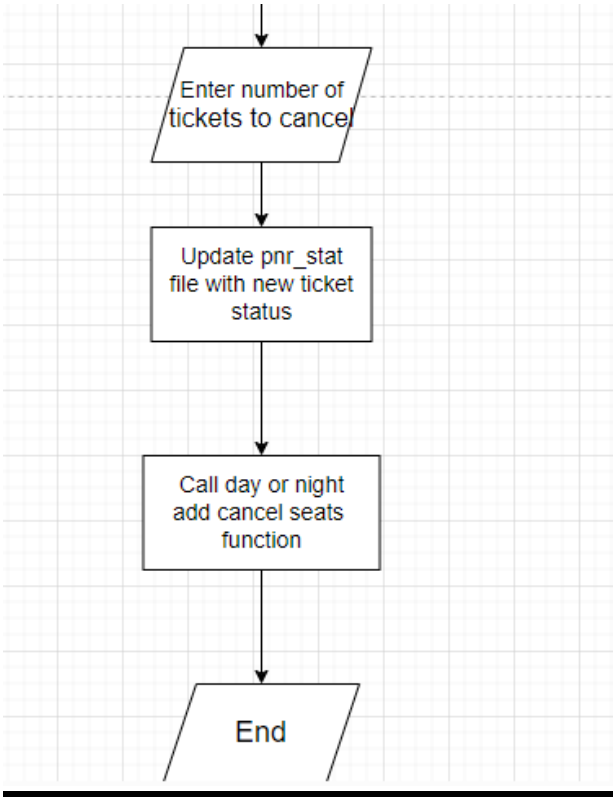


- Here the user can enter the destination and source station, Accordingly the shortest route along the trains in that path are suggested.

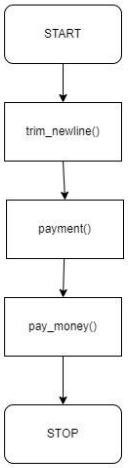
Cancel Module:




Get train file name,depart i index,dest index, seat type
Display confirmed and waiting tickets
choose to cancel confirmed or waiting ticket

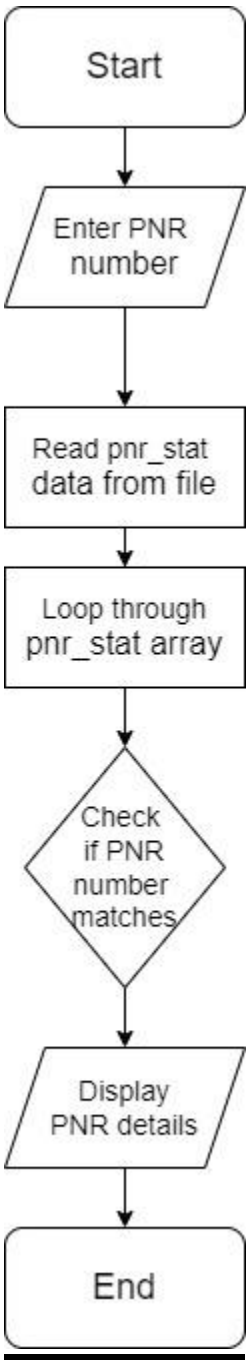

Enter number of tickets to cancel
Update pnr_stat file with new ticket status
Call day or night add cancel seats function
End

Payment Module:



In this module, the user can access the wallet and get details regarding his payment transactions and can perform the payment process smoothly

PNR_Status Module:



- In this module the user enters the PNR number, resulting in which details are shown regarding how many tickets are in waiting list, how many are confirmed are shown accordingly

ORGANISATION OF DATA:

ARRAYS:

Arrays have been primarily used for saving credentials and train lists so that it can be manipulated to be used at any part. Also arrays are used in storing porter details and food details and also an array for checkinhg the ticket's status.

IMPORTANT ARRAYS:

- User users[]:stores user information
- restaurant hotels[]:stores hotel and food information
- Porter porter_list[]:stores porter information
- train_struct train[]:stores train details
- Pnr_stats[] : stores the status the booked ticket
- Tkt_booked[] : stores the seat information
- Train_list[] : stores all train details

USE OF STRUCTS:

Here,structs are predominantly used to store user information which is then used at many process like in accessing the wallet file of the user and all and there are also other structs in different modules that are used to store data of their own purposes.
For example: Struct :restaurant - which stores hotel names and the food in those hotel for the user to order from them.

IMPORTANT STRUCTS:

- Seats
  - tot_coach:int
  - avail_coach:int
  - tot_seats:int
  - avail_seats:int
  - seats:int
- Train_struct
  - train_no:int
  - train_name[100]:char
  - train_stops[100][100]:char
  - train_day[7][10]:char
  - train_time[100]:char

- train_type:char
- day_night:char
- dist[99]:float
- no_station:int
- no_days:int

- User
  - first_name[50]:char
  - last_name[50]:char
  - username[50]:char
  - password[50]:char
  - phone[20]:char
  - email[50]:char
  - security_one[50]:char
  - security_two[50]:char
  - wrong_pass_count:int
  - wrong_security_count:int
- Porter
  - name[20]:char
  - ID:int
  - phone_number[15]:char
  - luggage_capacity:int
  - availability:int
- Admin
  - username[50]:char
  - password[50]:char
- food
  - f_name[25];:char
  - Price:int

- restaurant
  - name[25]:char
  - meals[8]:int

- Paymentdetails
  - Username[50]:char

- Train_name:char
- Payment_amount:float

- Date_type
  - Day:int
  - Month:int
  - Year:int

- Booked
  - Coach_no:int
  - Seat_no:int
  - Type:int

- Pnr_stat
  - int pnr_num;
  - int confirm_tkt;
  - int wait_tkt;
  - int cancel_tkt;
  - char train_filename[20];
  - char trn_type;
  - int depart_inx;
  - int dest_inx;
  - int seat_type;

- Day_seat_detail
  - int normal[7][120];
  - int nr_waitlist[10];
  - int nr_waitlist_count;
  - int ac_chairs[5][60];
  - int acch_waitlist[10];
  - int acch_waitlist_count;
  - int ac_excecutive[2][45];
  - int acex_waitlist[10];
  - int acex_waitlist_count;

- Day_file_train_detail
  - char day_night;
  - day_seat_detail stops[75];

- Night_seat_deatil
  - int sleeper[12][72];
  - int sl_waitlist[10];
  - int sl_waitlist_count;
  - int ac_3tier[4][64];
  - int ac3_waitlist[10];
  - int ac3_waitlist_count;
  - int ac_2tier[3][32];
  - int ac2_waitlist[10];
  - int ac2_waitlist_count;
  - int ac_first[1][16];
  - int ac1_waitlist[10];
  - int ac1_waitlist_count;

- Night_file_train_detail
  - char day_night;
  - night_seat_detail stops[75];

FILES:

DAT files are versatile and can store various types of data, both binary and text, making them flexible for different uses. They offer efficient read/write performance and compact storage, which is ideal for large datasets.

IMPORTANT FILES:

- trainlist123.dat:storing list of trains
- users_data.dat:storing user data information
- porter_list.dat:storing porter details
- Userwallet.dat files - stores the amount of money a user has in a file exclusive for user
- restaurants.dat :stores food order information
- Payment.dat :stores information about all the payments
- Pnr_stat_file.dat :stores details of pnr number

- Pnr_list.dat : stores all the available pnr numbers
- Seat _availability.dat - stores the availability of files

SCHEMAS AND INSTANCES:

1.Login Module & Sign-Up Module:

Schema:

In both Login and Sign-Up Module, the data of the user and admin is stored in the userdb file using signUp() in Sign-Up Module and this file is used to check the whether the user has already signed up or not.

Instances:

COLUMNS	DATATYPE	DESCRIPTION	EXAMPLE
Name	string	Name of User	Vikraman S
username	string	Username of user	Vikram1234
password	string	Password of user	Vikram1234@
Email	string	Email address of user	Vikraman1234@gmail.com
Phone number	string	Phone number of user	9966485875
Sec1	string	Security question 1's answer	RV

2. Admin Module:

Schema:

In this module , the admin creates trains db through create() which is then used in Booking Module to book tickets

The train_struct structure represents the train details which includes train name,train number, starting

point,destination,…etc which are stored in array inside trainsdb file.
And also it has create_food(),create_porter() and pay_store() functions for adding new menu for food delivery,adding new porters and adding money in wallet respectively.

COLUMNS	DATATYPE	DESCRIPTION	EXAMPLE
Train_no	integer	Train number	12386
Train_name	string	Train name	Vaigai Express
Train_stops	integer	Name of stops	TAMBARAM
Train_day	string	Days of working	Monday
Train_time	string	Approx Time of reaching a station	22.00
Day_night	string	Day Train or Night Train	D
dist	float	Distance from final point	215.00
No_station	integer	Number of stops	25
No_days	integer	Number of days of working	3

3.Booking Module:

Schema:

In this Booking module ,
The date_type structure represents date.
The seats structure represents the seats array used int fill_train_detail structure.
The fill_train_detail structure contains type of seatings based on day or night.

The train_struct structure represents the train details which includes train name,train number, starting point,destination,…etc which are in array in trainsdb file.

And this module uses file like trainsdb and seatsdb which are updated by admin in admin module.

PNR DATABASE

COLUMNS	DATA TYPE	DESCRIPTION	EXAMPLE
Pnr_num	integer	PNR Number	130699
Confirm_tkt	integer	Number of confirmed tickets	68
Wait_tkt	integer	Number of waiting list tickets	8
Cancel_tkt	integer	Number of cancelled tickets	2
Train_filename	file	Name of train file	266202412693.dat
Trn_type	string	Day or Night Train	D
Depart_inx	integer	Index of Departure station	12
Dest_inx	integer	Index of Destination station	16
Seat_type	integer	Seat type represented as numbers	2

4.PAYMENT MODULE:

Schema:

In this module,the function payment() takes the distance to travel as argument and then computes the fare for the travel and displays it to the user and asks if the user wants to proceed or not if yes it proceed to pay_money() function and completes the payment if he has enough money in wallet which is checked in the user exclusive wallet file stored in our database.

COLUMNS	DATATYPE	DESCRIPTION	EXAMPLE
username	string	Username of user	vijay123
train_name	string	Name of train	Vaigai Express
payment	float	Payment	250.00

5.FOOD MODULE:

Schema:

In this module,the user can order his food based on his taste through the function food_order() which shows the menu from restaurants.dat file and the payment is offline mode.

The user can skip this module if he doesn’t want to order food.

COLUMNS	DATATYPE	DESCRIPTION	EXAMPLE
hotel	string	Name of hotel	A2B
price	integer	Price of meal	50
meal	string	Name of meal	Dosa

6.PORTER MODULE:

Schema:

In this module, the user an book a porter so it will be convenient for him to move his luggage and the booking is done by porter_allot() function which allots the porter based on the details in the file porter_list.dat file and the payment here is again offline.

The user can skip this module if he doesn't want to book porter.

COLUMNS	DATATYPE	DESCRIPTION	EXAMPLE
name	string	Name of porter	Srini
ID	integer	Porter ID	123
phone number	integer	Porter phone number	9865423642
luggage_ capacity	integer	Porter's luggage capacity	3
availability	integer	Porter's availability	1

USER INTERFACE:

TERMINAL:

The terminal provides powerful tools and utilities, offering precise control over system functions. Additionally, its flexibility and customization options make it adaptable to various environments, promoting a deeper understanding of the underlying system.

TERMINAL INTERACTION:

We have used Visual Studio Code to integrate our modules and display the output of the programs along with taking inputs.

LIMITATIONS OF THE SOLUTION PROVIDED:

I. Our system doesn't use GUI (Graphical User Interface), so it may not be very attractive to work with.

II. Our system may not ensure 100% security for the users as we do not have any particular method for verifying OTP with a particular e-mail or phone number.

III. Our system doesn't take into consideration about all the trains running in India.The databases contain data about only particular set of trains running only for some places.
IV. In our system, even though there is a food delivery system but once you order it beforehand you can not cancel the food ordered even though you are not travelling.
V. Our system works only for long-distance trains,It does not keep in ind any local rain connections with other stations as of now.
VI. If the user enters wrong password or fails to pass the security questions,the admin is not acquainted with the knowledge regarding this.
VII. More than one user can use a similar email As our system doesn't have any verification pass-code sent to the email as of now.
VIII. The porter's payment should be negotiated by the user personally and the system will not involve in that part.
IX. The Dijkstra algorithm is restricted to 15 places as of now.
X. The refund process in our system is a bit lacking which needs to be re-established according to IRCTC policy later.
XI. Our system also takes the dates that have already passed into consideration.

CONCERNS WITH ETHICAL , LEGAL AND SOCIAL PERSPECTIVE:

I. **Food Waste:** Food delivery services may contribute to increased food waste through packaging, cancelled orders, or excess inventory. Implementing strategies to minimise food waste, such as portion control, sustainable packaging, and donation programs, can help mitigate this issue.

II. **Data Privacy:** Collecting and storing user data, including personal information raises privacy concerns. The application must comply with data protection laws and regulations, ensuring that user data is collected and used responsibly, with explicit consent from users.

III. **Accessibility and Inclusivity**: The application should be accessible to people of all abilities, including those with disabilities. This includes features such as screen readers, text-to-speech options, and alternative input methods.

IV. **Urban Development and Mobility**:Railways play a crucial role in urban development and mobility. They can reduce traffic congestion and promote sustainable urban growth.However, careful planning is required to integrate railway systems with other modes of transportation and urban infrastructure.

V. **Contracts and Agreements**:Legal issues can arise from contracts with suppliers, service providers, and public-private partnerships. Ensuring that contracts are clear, fair, and enforceable is crucial to avoid disputes

VI. **Environmental Impact**:Railways can have significant environmental impacts, including noise pollution, habitat disruption, and greenhouse gas emissions.Ethical considerations involve implementing sustainable practices, such as using cleaner energy sources, minimizing environmental footprint, and protecting wildlife habitats.

VII. **Safety and Responsibility**:Ensuring the safety of passengers and workers is a paramount ethical concern. Railway operators must prioritize maintenance, employee training, and safety protocols to prevent accidents

Addressing these concerns requires a comprehensive approach that considers the interests of all stakeholders, including consumers, workers, restaurants, and the wider community. By prioritising ethical practices and legal compliance, food delivery applications can contribute to a more sustainable and equitable food system.

LEARNING OUTCOMES:

● To get a better hold on the basic constructs and execution of C programmes which involves complex programming structures and logic handling
●To learn how to have fun while coding various kind of modules
● To develop a self paced programme to address real world problems.
● To identify and work on the strength of each team member to provide a efficient and plausible solution.
● To pool the ideas and flow of thoughts of each team member and come up with the most accessible solution with respect to the requirement of the problem statement.
● To teach and both cover up for our own team mate when someone doesn't know something
● To address the most common errors that arise while executing the programme.

REFERENCES:

https://github.com/shashirajraja/Train-Ticket-Reservation-System/
https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/
https://www.geeksforgeeks.org/structures-c/