|  |  |  |
| --- | --- | --- |
| A logo of a university  AI-generated content may be incorrect. | **School of Engineering & Technology** | |
| **Department: CSE** | **Session:** Odd |
| **Programmed**: B .Tech CSE (Specialization in AI & ML) | **Semester:** 1 |
| **Course Code:** ETCCCP105 | **Number of students:1**  **Name of student-**  **PAWAN KUMAR SINGH (2501730252)** |
| **Course Name:** Computer Science Fundamentals & Career Pathways | **Faculty: Dr. Ravinder Beniwal** |

**PAWAN KUMAR SINGH**

**2501730252**

**Railway Digitization Project**

***Using Computational Thinking to Design a Smarter Railway Ticket Reservation System***

**1. Introduction – Setting the Context**

In today’s digital world, convenience and efficiency have become necessities. Yet, railway ticket booking still often involves long queues, manual errors, and time-consuming procedures. To solve this, we can design an **automated railway ticket reservation system** that makes the process faster, more reliable, and user-friendly.

This project applies the core principles of **computational thinking** — namely **abstraction**, **decomposition**, and **pattern recognition** — to model and simplify a real-world system. The result is a structured approach to problem-solving that transforms a complex human task into a clear, logical, and efficient process.

By walking through these stages, we can see how computational thinking not only aids programming but also enhances critical reasoning for real-world problem-solving.

**2. Problem Description**

**2.1 Step 1 – Problem Selection**

**Chosen Scenario:** *Railway Ticket Reservation System*

**Objective:**  
To create a system that allows passengers to easily search for trains, check availability, calculate fares, make payments, and receive a digital ticket — all within a few steps.

**Inputs:**

* Passenger details (Name, Age, Gender)
* Journey details (Source, Destination, Date, Class, Train Number)
* Payment details

**Processes:**

1. Search for available trains based on journey details.
2. Check seat availability for the selected train and class.
3. Calculate the fare according to class and distance.
4. Process payment securely.
5. Generate and confirm ticket with PNR.

**Outputs:**  
A confirmed e-ticket containing:

* Passenger and journey details
* Train information
* Fare and payment confirmation
* Unique PNR number

**3. Problem Analysis**

**a) Abstraction – Focusing on What Really Matters**

When approaching this system, it’s crucial to identify what’s essential to the booking process and what can be ignored.

**Essential Elements:**

* Passenger details
* Train information (routes, schedules, seats)
* Fare calculation
* Payment and ticket generation

**Ignored Details:**

* Train model or color
* Onboard services (e.g., meals, entertainment)
* Station architecture or background operations

**Core Focus:**  
Matching a passenger’s travel request to an available seat on a suitable train, processing payment, and generating a valid ticket.

Abstraction ensures that we only deal with the key elements required to achieve the system’s main purpose.

**b) Decomposition – Breaking Down the Problem**

Large systems can be overwhelming to handle all at once. **Decomposition** helps by breaking the project into smaller, logical parts that are easier to design, develop, and test individually.

| **Module** | **Purpose** |
| --- | --- |
| **User Interface** | Collects passenger and journey details, allows selection and confirmation of booking. |
| **Train Search Module** | Finds trains based on source, destination, and date. |
| **Availability Check Module** | Verifies if seats are available in the chosen train and class. |
| **Fare Calculation Module** | Determines the ticket price based on class, distance, and passenger type. |
| **Payment Module** | Handles secure online transactions and payment verification. |
| **Ticket Generation Module** | Creates a unique PNR, generates an e-ticket, and sends confirmation. |
| **Database Module** | Stores all essential data — train schedules, bookings, and passenger details. |

This structured breakdown makes it easier to manage complexity, as each component focuses on one specific task within the larger system.

**c) Pattern Recognition – Learning from Other Systems**

**Pattern recognition** involves spotting similarities between our problem and other systems we already understand. This allows us to apply existing knowledge to new situations.

**Similar Patterns Identified:**

* **Search and Filter:** Similar to how users search for products on e-commerce websites.
* **Availability Check:** Works like booking a seat at a movie theatre or a flight ticket.
* **Transactions and Payments:** Follows the same structure as online checkout systems.
* **Data Validation:** Ensures that user inputs are accurate, similar to most online forms.
* **Unique Identifier Generation:** PNR creation works like order or invoice IDs in e-commerce.

Recognizing these patterns helps reduce development effort and increases reliability, since we’re using well-understood approaches to common digital problems.

**4. Solution Design**

Designing the solution involves visually representing and logically outlining how the system will function before writing any code. This step focuses on **flowcharting** and **pseudocode** — tools that translate human logic into computational steps.

**a) Flowchart – The Visual Overview**

Below is a textual representation of the system’s flowchart. It shows the main decision points and the flow of operations from start to finish:

[Start]

↓

[Input Source, Destination, Date, Class]

↓

[Search for Trains]

↓

⧫ Are Trains Available?

↳ No → [Display "No Trains Found"] → [End]

↳ Yes → [Select Train]

↓

[Check Seat Availability]

↓

⧫ Are Seats Available?

↳ No → [Display "No Seats Available"] → [End]

↳ Yes → [Enter Passenger Details]

↓

[Calculate Fare]

↓

⧫ Confirm Booking?

↳ No → [End]

↳ Yes → [Process Payment]

↓

⧫ Payment Successful?

↳ No → [Display "Payment Failed"] → [End]

↳ Yes → [Generate Ticket and PNR]

↓

[Display or Send Ticket Confirmation]

↓

[End]

This flowchart maps the user’s journey — from searching for a train to receiving a confirmed ticket — and captures the logical flow of decisions within the system.

**b) Pseudocode – The Logical Blueprint**

START

FUNCTION Railway Ticket Reservation System:

DISPLAY "Welcome to the Railway Ticket Reservation System"

INPUT Source Station

INPUT Destination Station

INPUT Travel Date

INPUT Preferred Class

CALL Search For Trains(Source Station, Destination Station, Travel Date)

IF No Trains Found THEN

DISPLAY "No trains found for your search criteria."

RETURN

END IF

DISPLAY "Available Trains:"

FOR EACH Train IN Available Trains:

DISPLAY Train. Name, Train. Number, Train. Departure Time, Train. Arrival Time

END FOR

INPUT Selected Train Number

CALL Check Seat Availability(Selected Train Number, Preferred Class, Travel Date)

IF No Seats Available THEN

DISPLAY "No seats available for the selected train and class."

RETURN

END IF

INPUT Passenger Name

INPUT Passenger Age

INPUT Passenger Gender

SET Fare = CALL Calculate Fare(Selected Train Number, Preferred Class, Passenger Age)

DISPLAY "Total Fare: ", Fare

INPUT Confirmation Choice

IF Confirmation Choice IS NOT "Yes" THEN

DISPLAY "Booking cancelled."

RETURN

END IF

CALL Process Payment(Fare)

IF Payment Successful THEN

SET PNR = CALL Generate PNR()

CALL Generate Ticket(PNR, Passenger Name, Selected Train Number, Source Station, Destination Station, Travel Date, Preferred Class, Fare)

DISPLAY "Ticket successfully booked!"

DISPLAY "Your PNR is: ", PNR

DISPLAY "A confirmation email/SMS has been sent."

ELSE

DISPLAY "Payment failed. Please try again."

END IF

END FUNCTION

END

This pseudocode outlines the main workflow of the booking system, describing inputs, decision points, and expected outputs. It acts as the foundation upon which code can later be written.

**5. Reflection and Discussion**

**Challenges Faced:**

* Simplifying a real-world process that involves multiple variables and dependencies.
* Organizing large amounts of data such as schedules, seat availability, and user information.
* Handling user input validation and ensuring accurate search results.
* Structuring the process to make it both logical and user-friendly.

**Key Learnings:**

* **Abstraction** taught us how to focus only on essential components and remove unnecessary details.
* **Decomposition** showed the importance of breaking large problems into smaller, manageable units.
* **Pattern Recognition** helped us identify familiar digital structures, making it easier to design an efficient system.

**Potential Improvements:**

* Adding features such as ticket cancellation, modification, and refunds.
* Integrating a secure online payment gateway.
* Allowing real-time seat updates and notifications.
* Implementing user accounts for booking history and preferences.
* Using AI to suggest optimal train routes and schedules.

**6. Conclusion**

Through this project, we’ve seen how **computational thinking** serves as a practical framework for solving complex, real-world problems. By applying **abstraction**, **decomposition**, and **pattern recognition**, we successfully designed a conceptual model for a **Railway Ticket Reservation System** that enhances efficiency and user experience.

This exercise highlights that computational thinking is not just about coding — it’s about structured problem-solving that can be applied to any domain, transforming real-world challenges into clear, logical, and actionable solutions.