adSemester Project Artifacts

Software Requirements:

- Frontend: Bootstrap for a dynamic easy to use user interface along with HTML5 for basic web page structure and styling
- Backend: Python/Django for rapid development and RESTful API design.
- Google Maps API: To display train locations and calculate alternate routes.
- Cloud Technology: AWS has various features to allow for an easy cloud experience along with a large community
- Payment Gateway Integration: Use Stripe API for easy integration of credit card and wallet payments

Functional Requirements:

- Login: Use an algorithm to check whether the user is a customer or staff member.
- Monitor Network Status: Use an algorithm to check the network's overall status.
- Check Seating Availability: Utilize a data-focused algorithm to check for available seats.
- Payment Processing: Use a data-processing algorithm to interpret a user customer's payment methods.
- Assign Passenger Seating: Upon selecting a seat, a customer's account will be attributed to the selected seat, where data is then stored.
- GPS Tracking: Track train locations in real-time to provide accurate updates to account for departure and arrival times.

Non-Function Requirements:

- Security: The system must ensure secure user authentication and protect sensitive user data.
- Performance: User login and seating availability checks should be completed in real-time or within milliseconds.
- Availability: The system must have high uptime, especially for network monitoring, with fast detection of network outages.
- Reliability: The system should consistently monitor the network and seating availability, detecting any failures and providing notifications for resolution.

Architecturally Significant Requirements (ASRs):

- Real-time Performance: The system must be able to perform train tracking in real time.
- Network Dependency: The system needs to be operational at all times, so we need to have fallback options when the system has issues.
- Data Security: User login and payments need to be encrypted to ensure personal data is kept private.
- Scalability: The system architecture needs to be able to handle high user demand.

Design Purpose: To improve upon existing train tracking systems along with the integration of machine learning to further enhance the user and staff experience.

Design Constraints:

- Budgetary Constraints: The project must use cost-effective technologies, prioritizing open-source tools and frameworks to minimize expenses.
- Regulatory Constraints: The system must comply with data protection laws and industry-specific standards.

- Technology stack: The system must use technologies that we are all familiar with along with using technologies that communicate efficiently with one another

Software License:

GitHub Link

https://github.com/JSimonoff-operator/CS4320Team72024

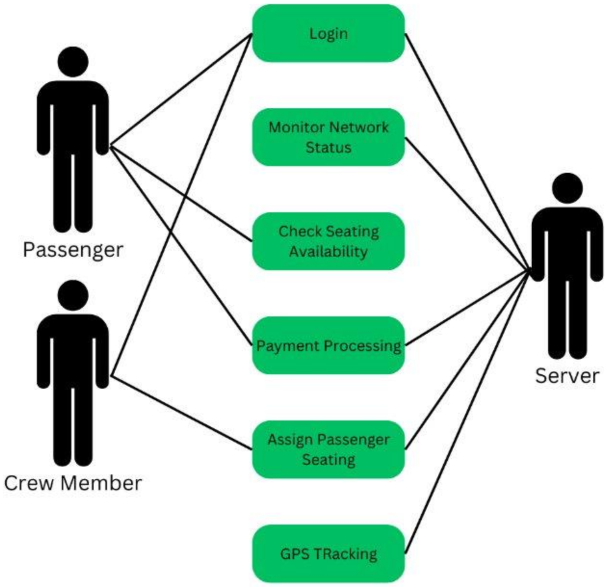
License (Open-Source)

MIT License Copyright (c) [2024] [Rayne Guinta, Jeffrey Simonoff, Jessie Meredeth] Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

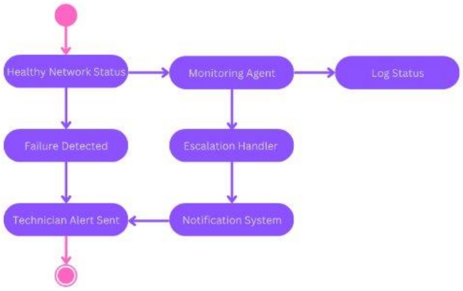
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

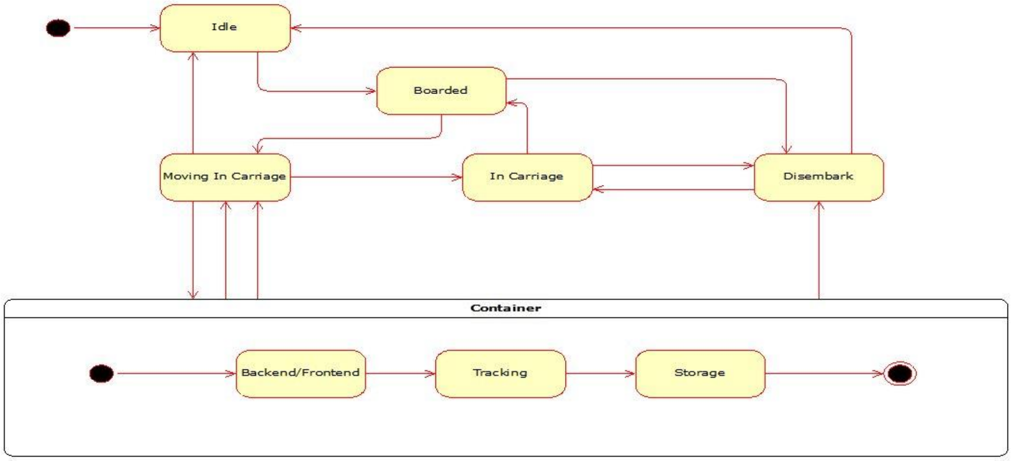
Modeling Diagrams:

Use Case Diagram:

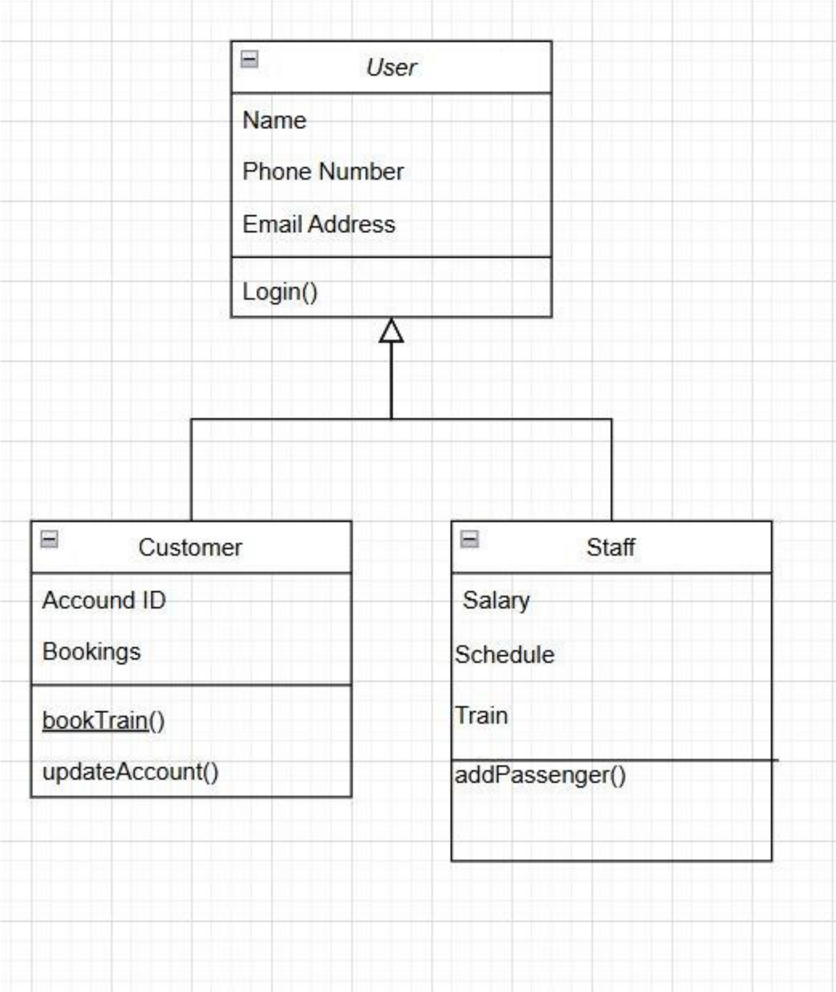


State Diagrams:

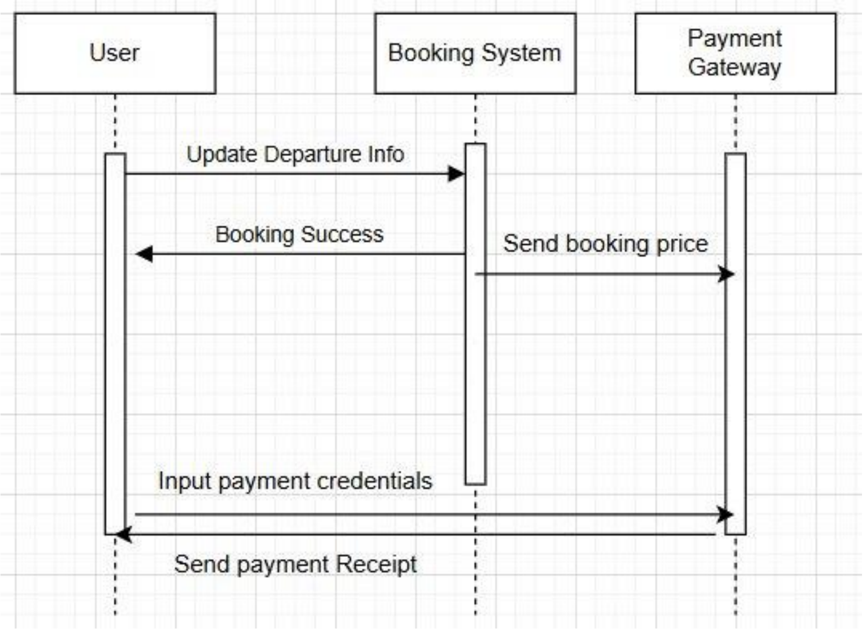


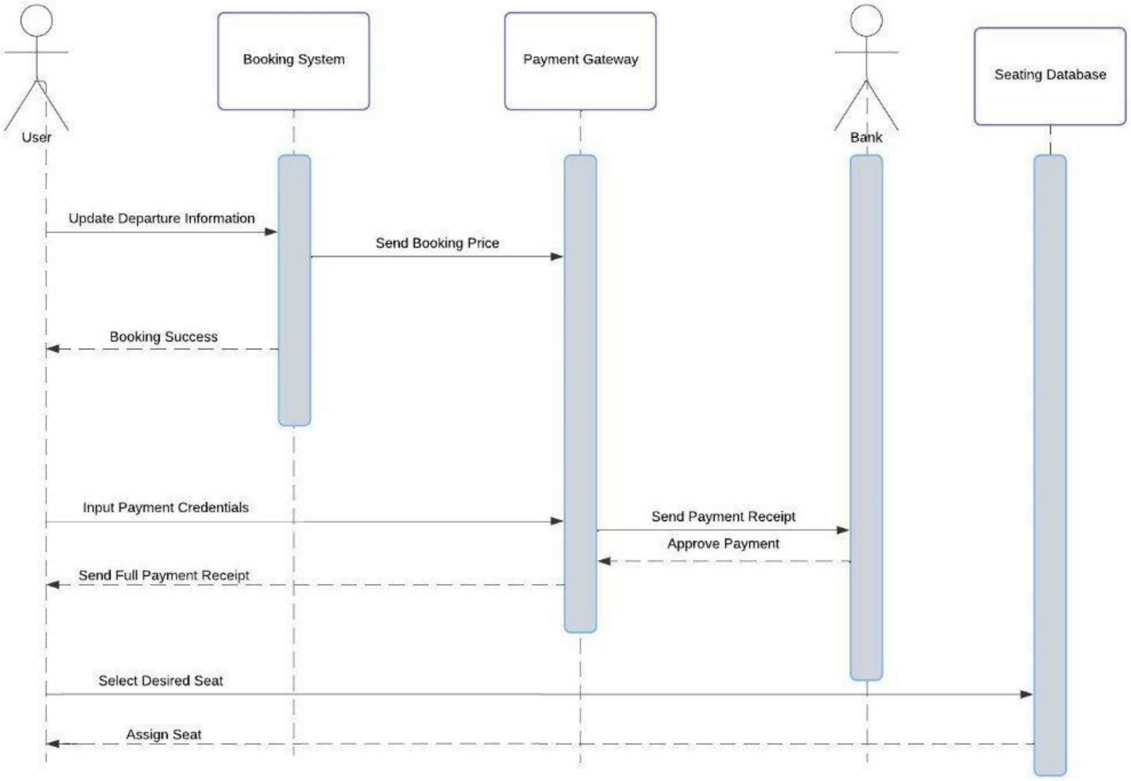


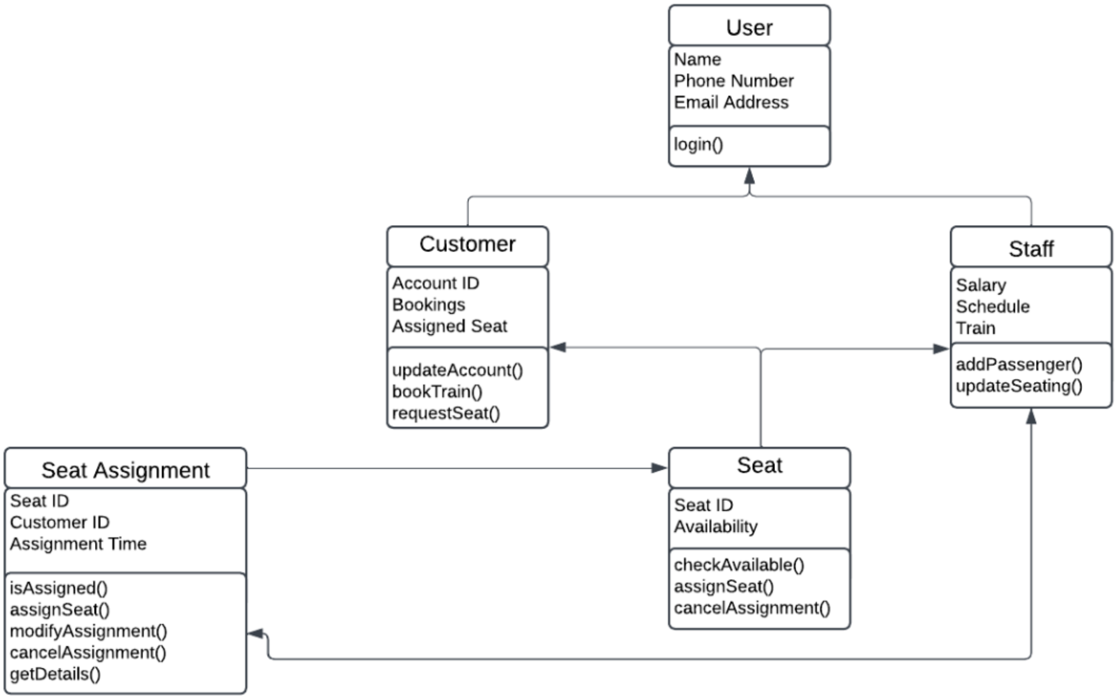
Class Diagrams:



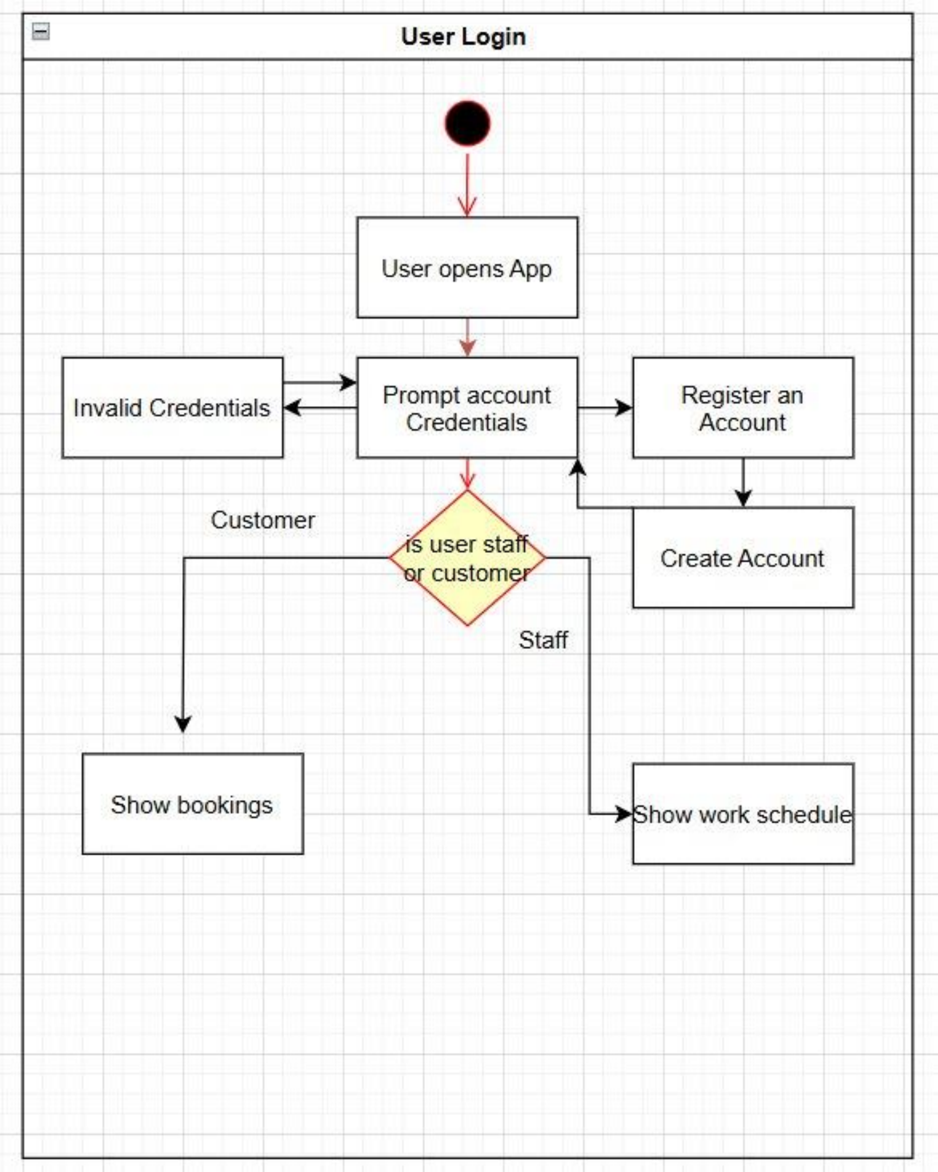
Sequence Diagrams


User
Booking System
Payment Gateway
Update Departure Info
Booking Success
Send booking price
Input payment credentials
Send payment Receipt

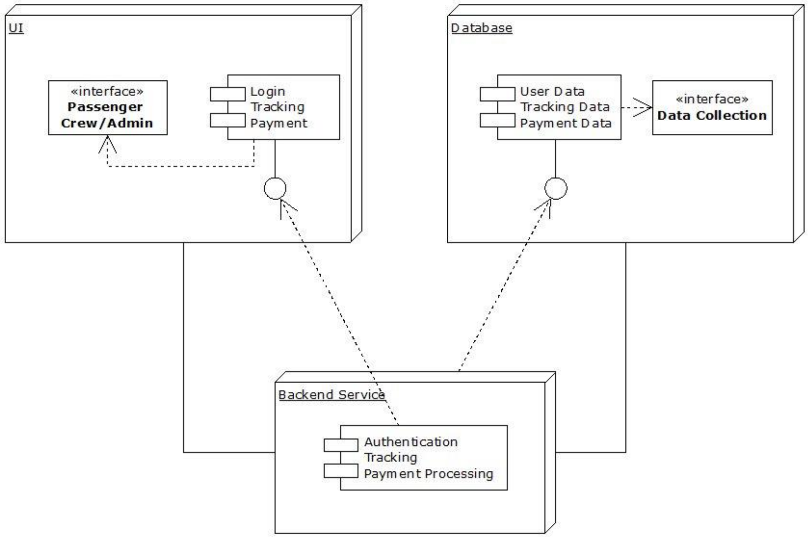

User
Booking System
Payment Gateway
Bank
Seating Database
Update Departure Information
Send Booking Price
Booking Success
Input Payment Credentials
Send Payment Receipt
Approve Payment
Send Full Payment Receipt
Select Desired Seat
Assign Seat



Activity Diagram



Component Diagram

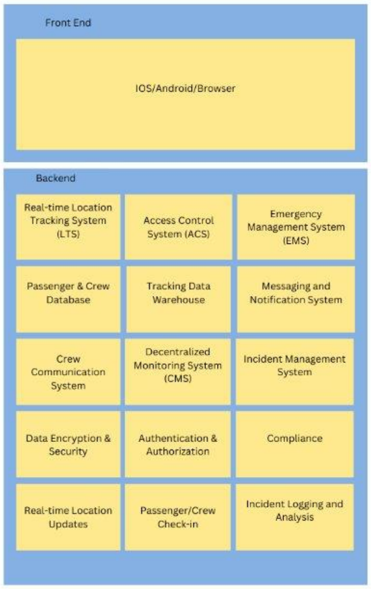


ADD:

How we used ADD – We utilized ADD's first step by reviewing our design purpose, where we decided upon a train tracking and passenger seating system that would be utilized by customers and staff alike. The drivers we used in step two consisted of network performance and dependability, data security, and scalability. For step three, we decided to first refine security and network dependency, as evidenced by iterations two and three, where step four showed its use through reference architecture, in which we used event-driven architecture in step five due to the various scenarios involved in train departure and seating. After allocating specific responsibilities in step five for each event, we began to create our UML diagrams and documented our decisions in a table for step six. As for step seven, we decided to use previous iterations to review our current progress, before moving onto the next iteration to make more progress. We created a total of four iterations to satisfy all of our necessary requirements in the forms of use cases, quality attributes, concerns, and constraints; therefore, we believe that by using ADD, we have accomplished our goal of making a plan for a train tracking and seat assignment system for high-end railway companies.

Iteration 1

Reference Architecture


Front End
IOS/Android/Browser
Backend
Real-time Location Tracking System (LTS)
Access Control System (ACS)
Emergency Management System (EMS)
Passenger & Crew Database
Tracking Data Warehouse
Messaging and Notification System
Crew Communication System
Decentralized Monitoring System (CMS)
Incident Management System
Data Encryption & Security
Authentication & Authorization
Compliance
Real-time Location Updates
Passenger/Crew Check-in
Incident Logging and Analysis

- Design Decision 1: GPS and Bluetooth Beacons
  - Rationale: GPS is highly effective for tracking objects outdoors, but GPS by itself is not effective inside the train due to poor signal. To fix this each train company should install Bluetooth beacons in the carriages to have precise positioning.
- Design Decision 2: Decentralized Cloud-Based Data Storage and Processing

    - Rationale: Allows real-time data from GPS and Bluetooth to be processed and stored in multiple accessible locations, consisting of passenger, platform, and train-focused servers.
- Design Decision 3: Data Privacy Protocols
    - Rationale: Minimizing data access, by implementing a Role-Based Access Control.

Main contributing factors: Decentralized Server Coordination; Data collection of passenger and staff attendance; Communications between passenger, platform operative, and train staff servers.

Element	**Responsibility**
Presentation Client Side	This layer contains modules that control user interaction and use case control flow.
API Gateway	Manages requests from clients, routing them to appropriate microservices while providing a single-entry point.
User Authentication Module	Handles user registration, login, and token management to ensure secure access to the system.
Train Attendance Microservice	Manages attendance records, including check-ins and check-outs, providing CRUD operations for users.
Decentralized Data Storage	Uses a distributed ledger or decentralized database to store

	attendance data securely and transparently.
Notification System	Sends alerts and notifications to users regarding attendance updates or reminders via email or push notifications.
Analytics Engine	Gathers data from attendance records to generate insights and reports for users and administrators.
Admin Dashboard	Provides administrative tools for managing users, viewing attendance metrics, and system health monitoring.
Integrations Module	Facilitates connections to third-party services, such as calendar apps or transport systems for enhanced user experience.
Security Layer	Ensures data encryption, secure communication, and protection against common vulnerabilities.
Passenger Server	Handles passenger application module to operate client(passenger)-server communications, including ticket storage.
Train Server	Handles train staff client-server communications, including

	passenger records. Communicates to the passenger and operative servers.
Operative Server	Handles platform operative client-server communications, often communicating with the train server on passenger capacity.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		Separate user roles between passenger and staff have been identified. Login functionality is utilized in selected related work.
	UC-2		Selected technologies through related work and backend systems address this use case.
UC-3			No relevant decisions made as it is vital to examine what systems will be utilized.
UC-4			No relevant decisions made as it is important to address how communications and technologies will be used with payment processors.
	UC-5		Selected reference architecture supports modules that can utilize this functionality.
UC-6			No relevant decisions made.

	QA-1		Separate user roles in login have been addressed, but payment processing has yet to be implemented.
	QA-2		Login has been addressed, but not payment processing or seating availability
	QA-3		Selected backend system technologies acknowledge this attribute.
	QA-4		Network status monitoring has been partially addressed with this attribute, while payment processing still requires a decision.
CRN-1			No relevant decisions made.
	CRN-2		A decentralized server algorithm ensures the security of some data that passengers and staff may have had access to while online. Additionally, there will be dedicated staff for each server.
	CON-1		Concern is partially addressed through the idea of a proper UI, which will allow passenger and staff users to utilize the system in a simple but efficient manner. The technology will be open source to address the issue of a user interface. Training will allow for smoother processes between all users.
CON-2			No relevant decisions made.

Iteration 2

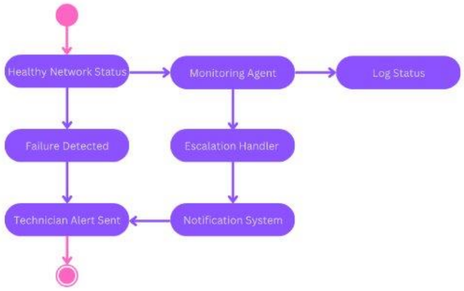
Requirement Analysis: UC-2:

- Functional Requirements: The system needs to perform health checks every 5 seconds and a system technician needs to be notified by email or SMS within 2 minutes of a failure.
- Non-Functional Requirements: Response time for system health check needs less than 1 second. And system uptime target at 99%

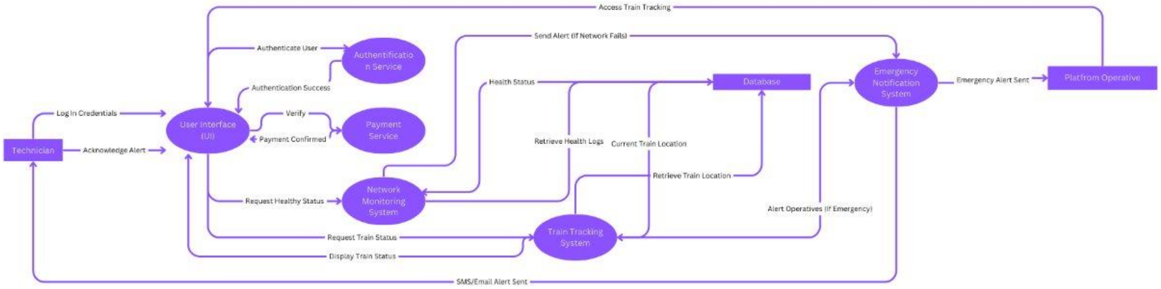
UC-6:

- Functional Requirements: Track the train's position and update it at intervals of 1 second or less and notify emergency services within 30 seconds of emergency detection.
- Non-Functional Requirements: The location accurately depicts the train within 10 meters and the success rate of delivery is greater than 95%.

State Diagram



System Architecture



Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		Separate user roles between passenger and staff have been identified. Login functionality is utilized in selected related work.
		UC-2	Defined requirements for network status and explained with detailed system architecture.
UC-3			No relevant decisions made as it is vital to examine what systems will be utilized.
UC-4			No relevant decisions made as it is important to address how communications and technologies will be used with payment processors.
	UC-5		Selected reference architecture supports modules that can utilize this functionality.
		UC-6	Defined Requirements with and created a system architecture to address train tracking.
	QA-1		Separate user roles in login have been addressed, but payment processing is still missing key aspects.
	QA-2		Login has been addressed along with payment processing, but seating availability has not been.
	QA-3		Selected backend system technologies acknowledge this attribute.

	QA-4		Network status monitoring has been partially addressed with this attribute, while payment processing still requires a decision.
CRN-1			No relevant decisions made.
		CRN-2	Addressed the issue of network status dependency affecting the rest of the networks and systems.
	CON-1		Concern is partially addressed through the idea of a proper UI, which will allow passenger and staff users to utilize the system in a simple but efficient manner. The technology will be open source to address the issue of a user interface. Training will allow for smoother processes between all users.
CON-2			No relevant decisions made.

Iteration 3

UC-1

- Functional Requirements: The system must verify the existence of a user's account upon login attempt. Must differentiate between staff and user roles
- Non-Functional Requirements: Must be responsive and redirect user within 2 seconds under normal conditions. System must handle multiple simultaneous login attempts

UC-4

- Functional Requirements: The system must accept multiple payment methods, including credit cards, debit cards, and digital wallets. The system must verify the validity of the provided payment method with the user’s bank.

- Non-Functional Requirements: The system must ensure 99.9% uptime for the payment processing service. The system must encrypt all sensitive payment data using industry-standard encryption methods

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	Machine learning-based security enhancements detect and flag suspicious login patterns to reduce unauthorized access. Distinction between staff and user roles are clear and easy to distinguish.
		UC-2	Defined requirements for network status and explained with detailed system architecture.
UC-3			No relevant decisions made as it is vital to examine what systems will be utilized.
		UC-4	Introduced ML-driven fraud detection to identify irregular transaction patterns and enhanced encryption for secure bank communications.
	UC-5		Selected reference architecture supports modules that can utilize this functionality.
		UC-6	Defined Requirements with and created a system architecture to address train tracking.
	QA-1		Separate user roles in login have been addressed, but payment processing is still missing key aspects.

	QA-2		Login has been addressed along with payment processing, but seating availability has not been.
	QA-3		Selected backend system technologies acknowledge this attribute.
	QA-4		Network status monitoring has been partially addressed with this attribute, while payment processing still requires a decision.
		CRN-1	Login authorization between staff and customer has been addressed.
		CRN-2	Addressed the issue of network status dependency affecting the rest of the networks and systems.
	CON-1		Concern is partially addressed through the idea of a proper UI, which will allow passenger and staff users to utilize the system in a simple but efficient manner. The technology will be open source to address the issue of a user interface. Training will allow for smoother processes between all users.
CON-2			No relevant decisions made.

Iteration 4

UC-3

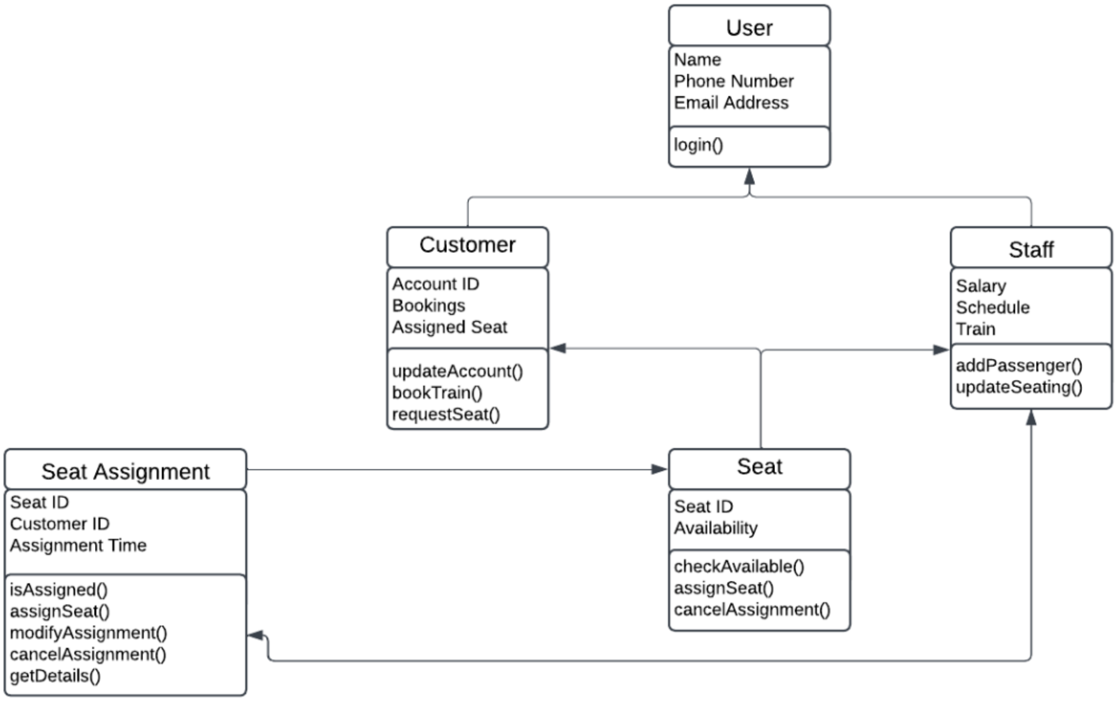
- Functional Requirements: The system must examine seating availability in regular 15 minute intervals, and validate the status of each individual seat.

- Non-Functional Requirements: The chart accurately depicts seating availability, where the system can perform these tasks with low latency and can utilize its checks during peak usage times of ~1000 users.

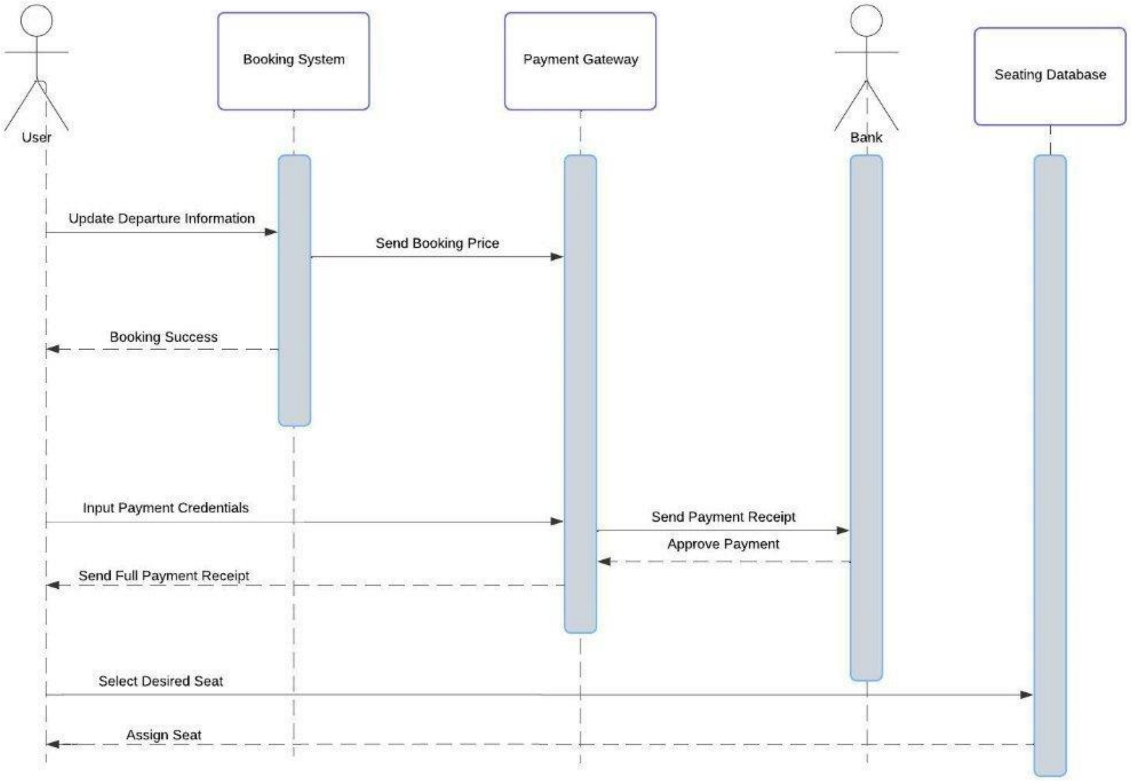
UC-5

- Functional Requirements: The system needs to assign the seat to the customer, where a notification via email or text should be sent to the user about their selected seat within 5 minutes, and if the seat is taken, the system should prevent the user from selecting it.
- Non-Functional Requirements: Seat assignment can function 99.9% of the time, especially in peak usage times of 1000 users, and the system is able to assign a seat and update seating availability within 2 seconds in real time

Class Diagram



Sequence Diagram



Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	Machine learning-based security enhancements detect and flag suspicious login patterns to reduce unauthorized access. Distinction between staff and user roles are clear and easy to distinguish.
		UC-2	Defined requirements for network status and explained with detailed system architecture.

		UC-3	Systems utilize seating availability and allow for optimal decisions during peak assignment times. Seat selection discrepancies are handled by storing a user into a specific seat.
		UC-4	Introduced ML-driven fraud detection to identify irregular transaction patterns and enhanced encryption for secure bank communications.
		UC-5	Selected reference architecture supports modules that can utilize this functionality. Seat assignments can operate at peak times in real time, where a customer is placed into a seat, where a proper system is in place to avoid a second user from taking an already assigned seat.
		UC-6	Defined Requirements with and created a system architecture to address train tracking.
		QA-1	Separate user roles in login have been addressed, and payment processing functionality has been planned for

		QA-2	Login has been addressed along with payment processing, as well as seating availability.
		QA-3	Selected backend system technologies acknowledge this attribute. Network status is monitored for both train tracking emergencies and customer seating times.
		QA-4	Network status monitoring has been partially addressed with this attribute, where payment processing has also been addressed.
		CRN-1	Login authorization between staff and customer has been addressed.
		CRN-2	Addressed the issue of network status dependency affecting the rest of the networks and systems.
		CON-1	Concern is addressed through the idea of a proper UI, which will allow passenger and staff users to utilize the system in a simple but efficient manner. The technology will be open source to address the issue of a user interface. Training will allow for smoother

			processes between all users.
		CON-2	Budgetary constraints have been addressed through the usage of AWS and runtime functionality.