CS731: Final Report

MyTravel.com – Blockchain-Based Ticket Management System

Group 2
Deepanshu (210310), Narendra Singh (210649)

April 28, 2025

1 Overview

MyTravel.com is a blockchain-based ticket management system designed to enable customers to easily book, update, and cancel transport tickets. The system is built using **Hyperledger Fabric** for managing transactions and maintaining a decentralized ledger, ensuring secure, transparent, and verfiable ticketing processes.

This system allows customers to:

- Book tickets for transport services.
- View available servics and their details.
- Update or cancel bookings with dynamic pricing logic and penalties for cancellations.

For service providers, it offers functionality to:

- Register their transport services.
- Manage their services and bookings.

You can find code for our project at GitHub Link.

2 High-Level Hyperledgr Architecture

- **Organizations:** Two organizations
  - **Customers and Providers:** Interact with the system for booking, updating, canceling tickets, and registering transport services.
  - **Administration:** Manages user and provider registration, and oversees payment procedures.

- **Peers:** One peer per organization, hosting ledgers and executing chaincode for transaction validation and updates.
- **Orderer:** Single orderer node for transaction ordering across the network.

3 Technologies Used

3.1 Frontend

The frontend is developed using **React.js**. The following functionalities have been implemented:

- Signup Page
- Login Page
- Book Ticket Page
- View Available Tickets Page
- Money Addition Option

3.2 Backend

The backend is developed using **Go (Golang)** for the API services. **MongoDB** is used as the database for storing user metadata, transport services, and bookings.

3.3 Hyperledger Fabric

Hyperledger Fabric is used to maintain the blockchain ledger. It operates primarily on the unique user identifier, which in our system is the user's unique **email ID**. Blockchain transactions are executed based on this identifier for verifying ticket bookings and service provider authenticity.

4 Progress Overview

- **Frontend Development:** Basic UI for registration, login with the jwt token, transport service registration, and ticket booking, transport addition, viewing and cancelling booked tickets are completed using React.js.
- **Backend APIs:** The backend APIs follow RESTful principles to facilitate interaction between the frontend and the blockchain network. Following are implemented and tested endpoints:
  - /ledger/createuser
  - /ledger/login

  - /ledger/addMoney
  - /Addtransport
  - /tickets (GET)
  - /tickets (Post)
  - /tickets/:id (Get)
  - /tickets/:id (Put)
  - /tickets/:id (DELETE)
  - /GetTransport

- **Chaincode:** Following functionalities Implemented and tested:
  - New User/Provider Registration
  - New Transport Addition
  - Ticket Booking Logic with Dynamic Pricing
  - Virtual Payment Logic
  - dyanmic Pricing of tickets
  - Ticket Update/Delete with Penalty
  - Payment Verification
  - Making User Anonymous/Public

  Chaincode tested via the command-line interface and some via frontend/backend.

5 Functional and Non-Functional Features

5.1 Functional Features

- Onboarding of users and service providers.
- Users can view available option for each transport option
- Booking, updating, and canceling tickets (dynamic pricing included; update/cancellation incurs charges).
- Providers can create new transportation services.
- Profile details update functionality for users and providers.

5.2 Non-Functional Features

- Scalability to support 100 daily active users and 250+ daily bookings.
- Prevention of overbooking and double-bookings.

6 Problem and issues faced

6.1 Key and Certificate Based Authentication Challenges

Initially, we aimed for full key-based authentication where users would receive a private key and certificate upon signup. Users were expected to sign each blockchain transaction individually using their credentials.

However, we faced major challenges:

- Managing keys securely on the frontend was complex.
- Every transaction required re-signing, complicating UX.
- Session management via traditional tokens was not compatible with Fabric's signature verification model.
- Full frontend cryptographic handling exceeded project timelines.

6.1.1 Final Approach: Email and Password Sign-In

Due to these challenges, we shifted to email-password based authentication. The backend now verifies users using JWT tokens and submits blockchain transactions using an application identity.

All critical operations (booking, payment, registration) are still securely recorded and verifiable on the blockchain, while keeping the system simple and user-friendly.

6.2 Structured Output Parsing from Terminal Commands

When executing chaincode via terminal, the raw output contained unnecessary logs and multiple lines, not just the JSON payload. Example:

```
2025-04-27 19:03:13.464 IST 0001 INFO [chaincodeCmd]
    chaincodeInvokeOrQuery ->
Chaincode invoke successful. result: status:200 payload:"{\"Name
    \":\"deepanshu\",
\"Email\":\"deenfdmmddddfu@gmail.com\",\"Phone
    \":\"7082575299\",\"PastTravels\":null,
\"UpcomingTravels\":null,\"BankBalance\":0,\"IsAnonymous\":false
    ,\"PaymentID\":[]}"
```

This made direct JSON unmarshalling fail with errors like `invalid character ' ' looking for beginning of object key string`. Proper structuring and clean payload extraction were needed before parsing into Go maps or structs.

7 Roles and Responsibilities

- **Narendra Singh (210649)**

- Hyperledger setup and complete logic.
- Dynamic pricing and ticket booking chaincode development.
- Update and delete ticket chaincode (work in progress).
- Frontend designing.
- Testing.

- **Deepanshu (210310)**
  - Frontend development.
  - Backend API integration with the hyperledger.
  - Chaincode development for user and provider registration and profile management.
  - adding the payment and account balance part