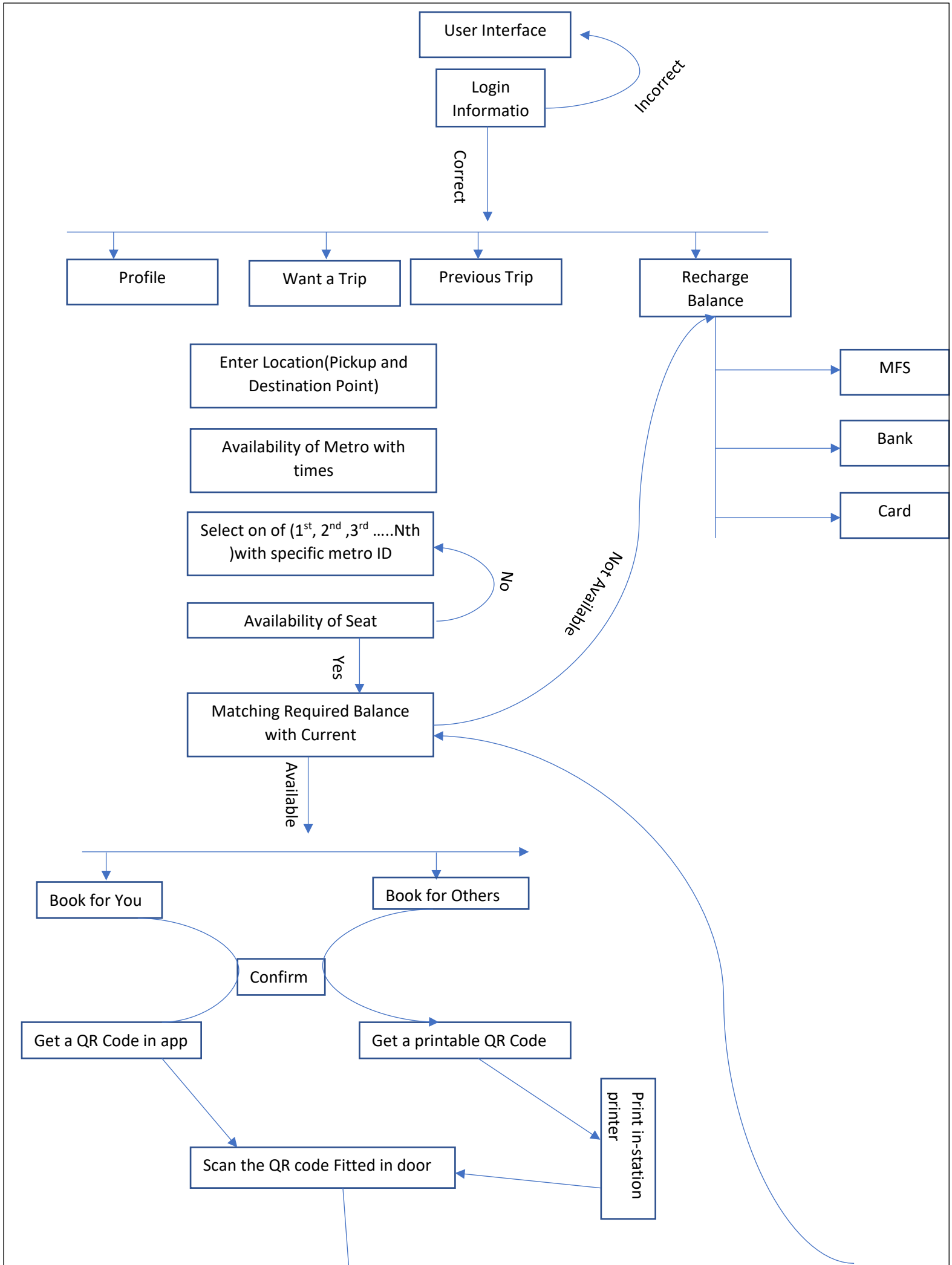

User Interface
Login Informatio
Incorrect
Correct
Profile
Want a Trip
Previous Trip
Recharge Balance
MFS
Bank
Card
Enter Location(Pickup and Destination Point)
Availability of Metro with times
Select on of (1st, 2nd ,3rd .....Nth)with specific metro ID
No
Availability of Seat
Yes
Not Available
Matching Required Balance with Current
Available
Book for You
Book for Others
Confirm
Get a QR Code in app
Get a printable QR Code
Print in-station printer
Scan the QR code Fitted in door

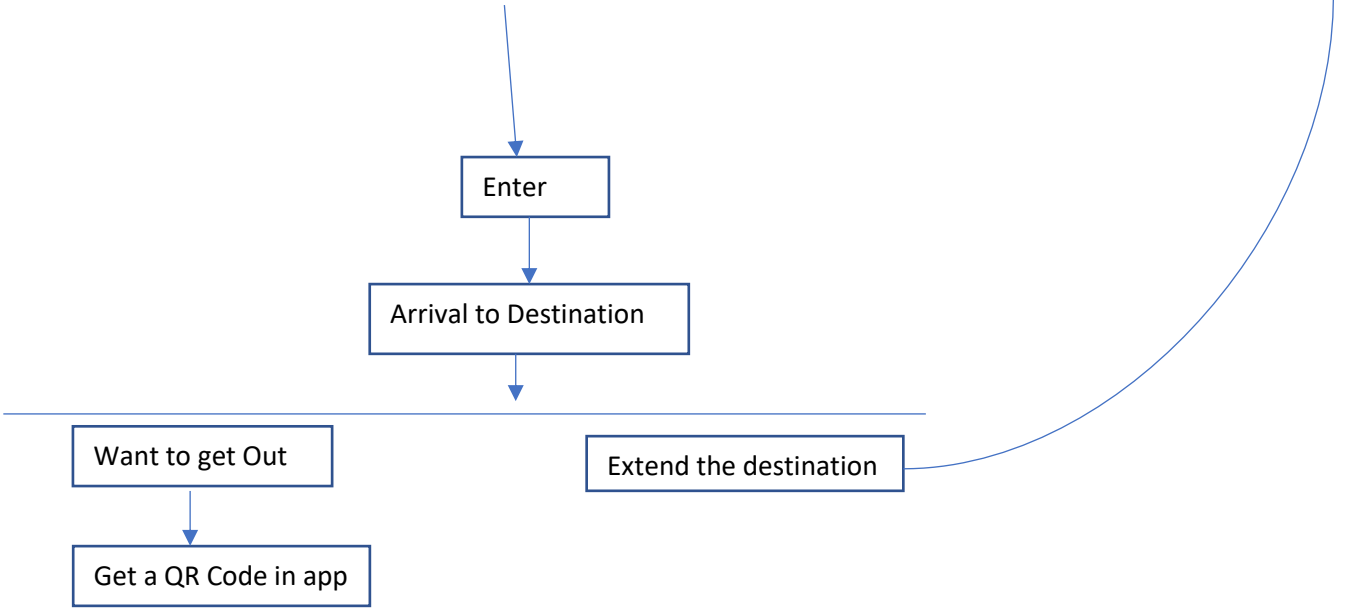


Figure: A high-level design of the service flow

High level architecture of this system

Functional Requirements

We should have database where all the information of the passengers can be stored and it will be secured. And all transactional information will also be stored here.

- Framework should have the option to check data.
- Framework should have the option to store the data in information base.
- Framework should have the option to recover data when needed by administrator.

Metro App

This feature allows the user to connect the server to do all the things. From booking to exit, it will help the user throughout the process and help the administrators to get the customer data to the server automatically online, there-by saving their valuable time. Users need to login with their id and password.

- Password can be changed online.
- It also allows them to view their balance and journey history.
- User id is provided when they register.
- The system must be able to show the users balance and journey history.
- The user must be able to logout after they had finished recharging or after viewing the balance or journey history.

Metro time table

This feature allows the users to view the metro time table. Users are required to enter the source station and destination station, when they enter the data then the system will show the metro time table.

- System must allow the users to enter the source station and destination stations.
- System must be able to process information from database.

Fair and Route map

This feature allows the users to view the fair and route map. Users are required to enter the Pickup and destination station, when they enter the data then the system will display fair details and the route map.

- System must allow the users to enter the source and destination stations.
- System must be able to retrieve information from the database

Administrator

This element permits the administrator to view and answer to grumblings. Administrator can add stations, courses ,train , trip . Administrator can likewise add and update reasonable subtleties, and even add another administrator. In reality, the administrator is a board comprising of a gathering of approved people.

- The framework should permit administrator to add train, stations ,courses, reasonable ,metro schedule and even add another administrator.
- The framework should likewise permit administrator to answer to the grumblings send by the client.
- The framework should be planned so that solitary approved individuals should be permitted to get to some specific modules.
- The records should be adjusted by just directors and nobody else.

Programming prerequisites

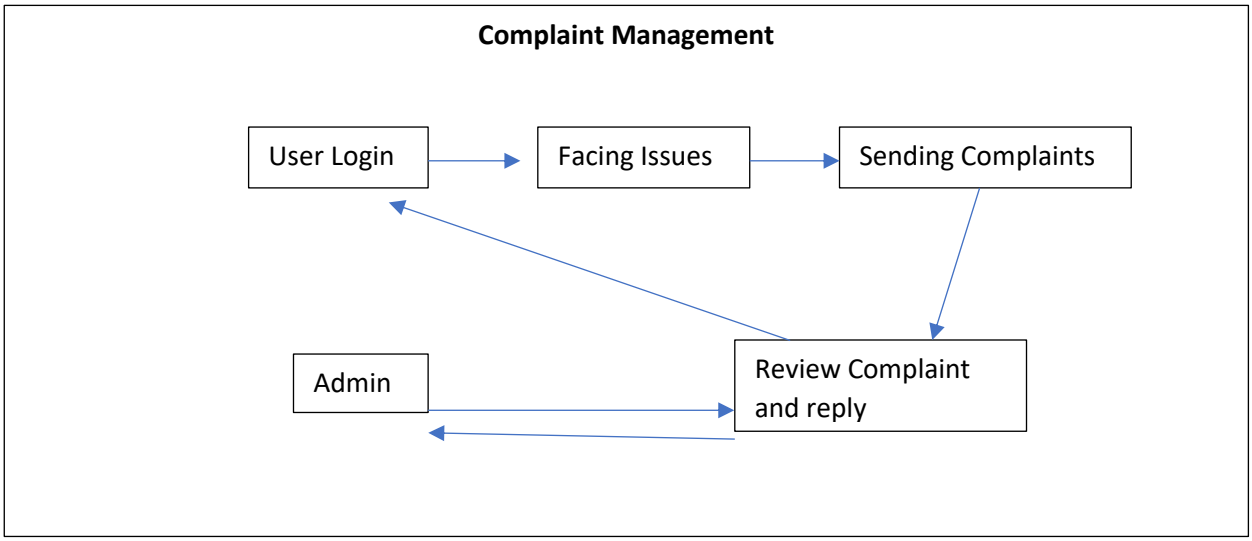
Working framework Windows 10 is utilized as the working framework as it is steady and supports more highlights and is easier to understand

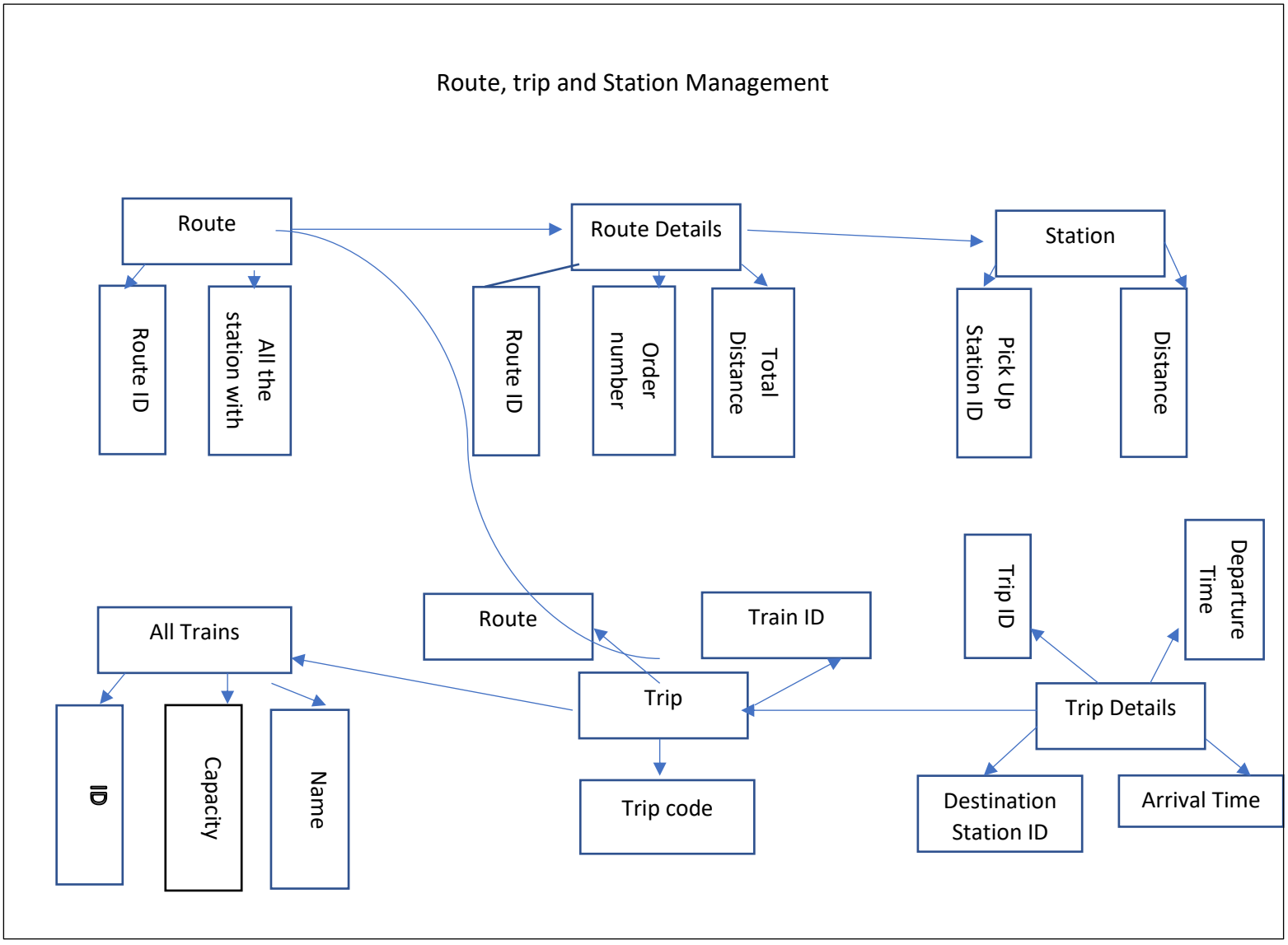
Information base MYSQL-MYSQL is utilized as data set as it simple to keep up and recover records by straightforward inquiries which are in English language which are straightforward and simple to compose. Improvement apparatuses and Programming language-HTML is utilized to compose the entire code and create pages with CSS, java content for styling work and php for cut off side scripting.

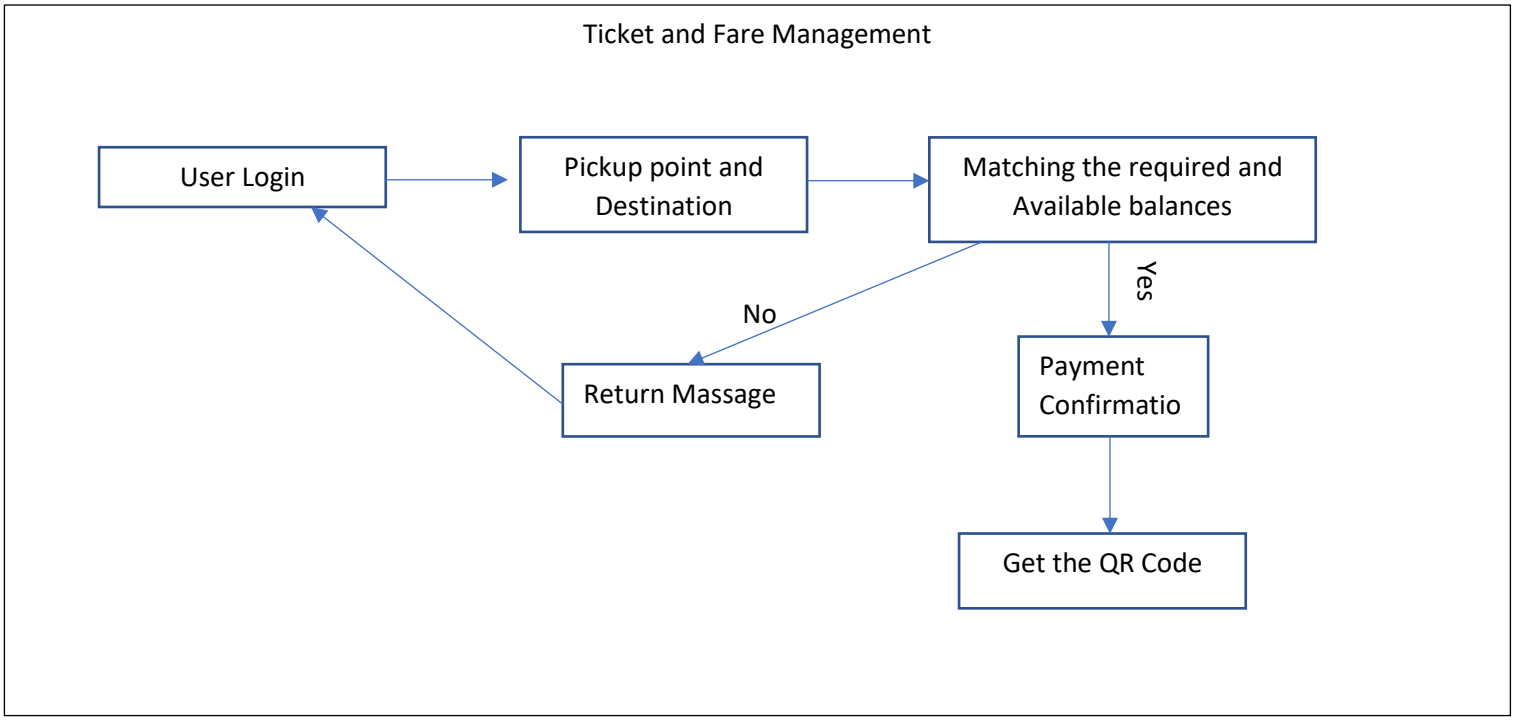
Equipment prerequisites

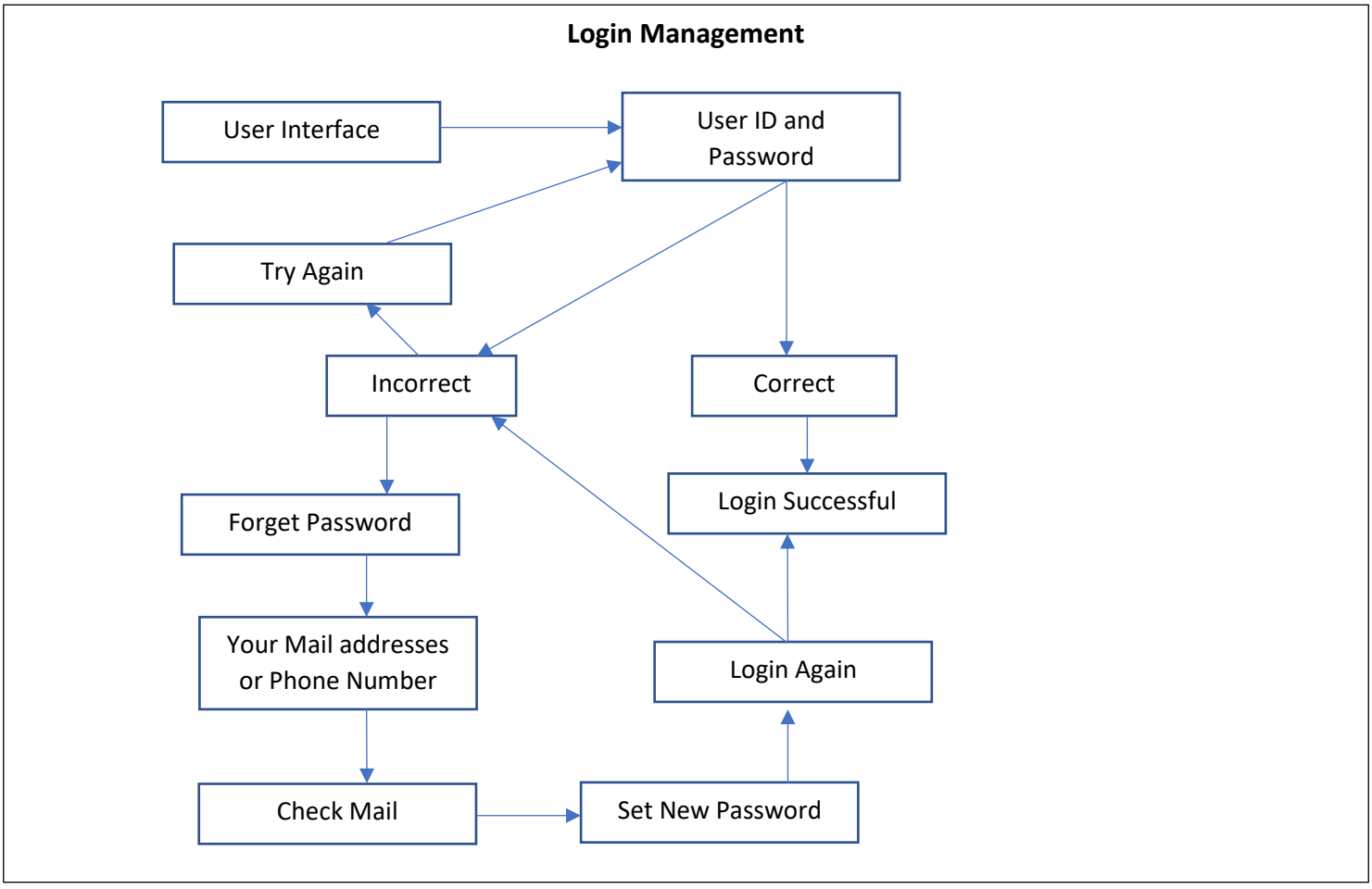
Update and Most Powerful Processor is utilized as a processor to offer solid and stable Assistance by utilizing this processor we can continue building up our undertaking with no concerns.

UML diagram


Complaint Management
User Login
Facing Issues
Sending Complaints
Admin
Review Complaint and reply


Route, trip and Station Management
Route
Route ID
All the station with
Route Details
Route ID
Order number
Total Distance
Station
Pick Up Station ID
Distance
All Trains
ID
Capacity
Name
Route
Train ID
Trip
Trip code
Trip ID
Departure Time
Trip Details
Destination Station ID
Arrival Time


Ticket and Fare Management
User Login
Pickup point and Destination
Matching the required and Available balances
Yes
No
Payment Confirmatio
Return Massage
Get the QR Code


Login Management
User Interface
User ID and Password
Try Again
Incorrect
Correct
Forget Password
Login Successful
Your Mail addresses or Phone Number
Login Again
Check Mail
Set New Password

High Level DB Architecture (ERD)

This ER (Entity Relationship) Diagram represents the model of Metro System Entity. The entity-relationship diagram of Metro Rail Ticket Booking System shows all the visual instrument of database tables and the relations between Routes, Fare, Metro, Stations etc. It used structure data and to define the relationships between structured data groups of Metro System functionalities. The main entities of System are Metro, Routes, Tickets, Fare, Booking Counter and Stations.

Metro Rail Ticket Booking System entities and them

attributes :

Metro Entity : Attributes of Metro are Metro-id, Metro name, Metro number, Metro-seat-number, Metro-ticket, Metro-type, Metro-description

Routes Entity : Attributes of Routes are Metro-route-id, Metro-route-name, Metro-route-type,

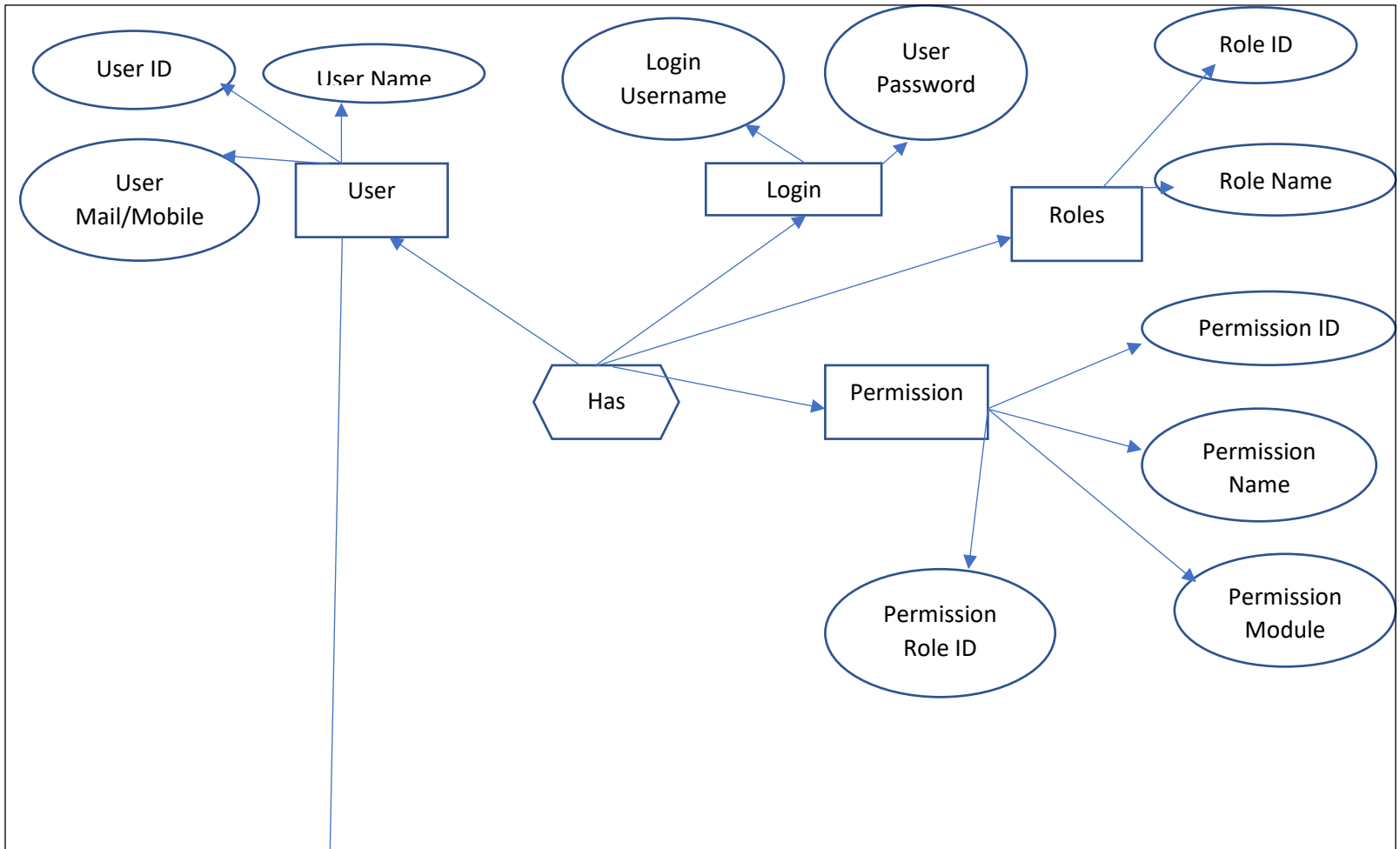
Metro-route-description

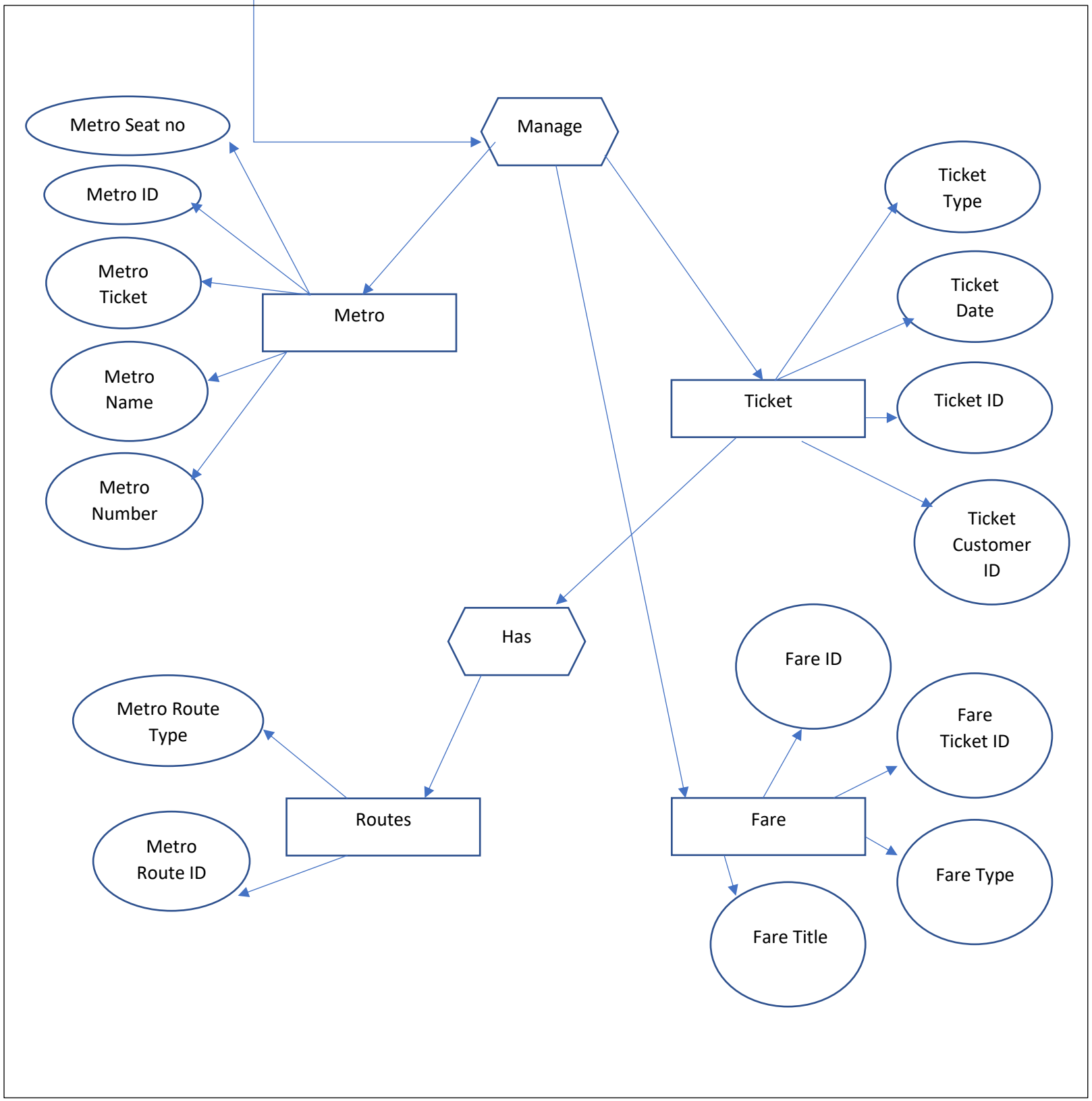
Tickets Entity : Attributes of Tickets are ticket-id, ticket-type, ticket-date, ticket-description

Fare Entity : Attributes of Fare are fare-id, fare-ticket-id, fare-title, fare-type, fare-description

Booking Counter Entity : Attributes of Booking Counter are booking-id, booking-title, booking-type, booking-ticket, booking-date, booking-description

Stations Entity : Attributes of Stations are station, station name, station type, station description

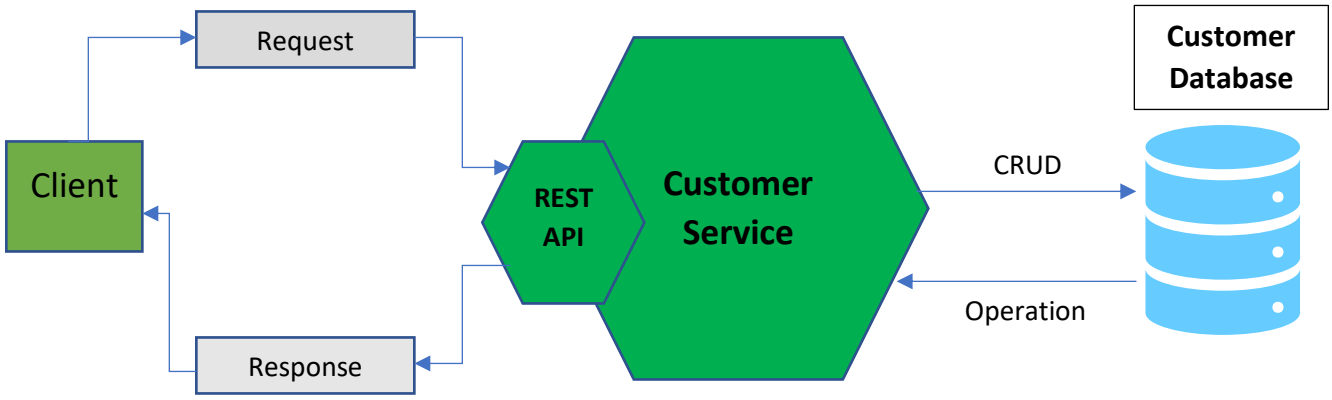



Metro Seat no
Manage
Metro ID
Ticket Type
Metro Ticket
Ticket Date
Metro
Metro Name
Ticket
Ticket ID
Metro Number
Ticket Customer ID
Has
Fare ID
Metro Route Type
Fare Ticket ID
Routes
Fare
Metro Route ID
Fare Type
Fare Title

High Level API Designs (RESTful)

This step is one of the essential steps before we begin the REST API development. Without obviously characterizing the design of the framework we can't assemble it with no issue.

The beneath Diagram speaks to the framework that we are building. Our framework comprises of a RESTful help and an information base. The RESTful assistance, Customer Service, will be created in Node.js with Express.js. MongoDB will be utilized as an information base for the administration to connect.



Engineering – Customer Information System

Low-Level Design

So far, we have characterized the engineering for the client data framework. Presently it's an ideal opportunity to do a profound jump into the low-level plan of the API. This progression is the following one after the engineering.

In the low-level plan, we will go to the API endpoint level plan. After the solicitation has been gotten by the RESTful assistance what will end up adjusting that solicitation is the thing that we will examine in the low-level plan of every one of the API endpoints.

Normally, we should make the succession graph for every endpoint at this stage.

Before we hop onto the grouping graphs it's smarter to see what are on the whole the various parts will be included right from the solicitation got stage until the reaction is sent back to the requester.

Layers Or Components Of Node.js REST API

Normally, RESTful help will achieve the undertaking of serving the solicitation in the layered methodology. In the Node.js world, there are various parts accessible for us to execute the API usefulness. They are course, middleware, administration, and model.

Some of them may not be Node.js system related layers, however they not new to the soothing help world. In item arranged dialects, the administrations are most presumably actualized in quite a layered methodology. We could get that approach here also.

The following are the various segments that we will use to execute the Customer Service. We should have a short gander at these segments and see what each will do.

1. Regulator

Normally, a regulator will deal with the solicitation, summon administrations to play out that activity, and cycle reaction to sending back to the requester. Frequently regulator will make a grouping of administration brings in coordination to achieve the solicitation as planned. In fact, it handles the progression of the middleware calls before it sends the reaction.

2. Middleware

Middleware in a NodeJS world, is a capacity that approaches the solicitation object, reaction object, and next capacity. Here the following() work is utilized to conjure the following middleware in the stack.

Middleware capacities can play out the accompanying undertakings:

- Execute any code.
- Make changes to the solicitation and the reaction objects.
- End the solicitation reaction cycle.
- Call the following middleware in the stack.

On the off chance that the current middleware work doesn't end the solicitation reaction cycle, it should call straightaway() to pass control to the following middleware work. Something else, the solicitation will be left hanging.

3. Administration

A help is any capacity that can play out any undertaking, such as figuring some recipe, getting to the information base to peruse or compose. Here we will utilize an assistance capacity to get to the information base for recovering and putting away the client data.

Additionally, the administration won't approach the solicitation and reaction object. So anything should be done on the solicitation and reaction article should be done in the regulator as it were.

At that point, the regulator should pass that data as boundaries to the administration capacities to play out the undertaking. This path there is an away from of obligations among all the segments in the RESTful assistance.

4. Model

This part is the information access layer to bring and save the archives. The administration layer will summon the models to play out any activities on the record in the information base by means of the model.

A model speaks to the archive that can be made, refreshed, eliminated and brought from the information base.

TPS (transaction per second) Calculation and required infrastructure

Metro Rail will be busy mainly in the morning Time(7 am to 11am) and evening time(5Pm to 8pm) because in Dhaka, schools, colleges, university, office, shopping mall mainly start from 7 am to 11am and closes from 5 pm to 8pm and during these two-slot people mainly travel in huge margin but it is not like that metro rail will not have any passenger during rest of the time rather metro will be less-busier rest of the time.

According to the assumption from the case given;

- Daily 100k app login
- Daily 80k booking
- Daily 50k purchase
- Daily 25k journey complete

If 50000 Purchase the Service Daily then TPS(Transaction Per Second) will be:

50000/86400= 0.58 Transaction Per Second.

If we exclude the 7 hours, mainly 11pm to 6 am, the TPS will be 0.82 Per second. If we only consider Up time then TPS will be 1.98.

It will be calculated by our sever where oracle will be work as base.