Ticket Booking Application

Gamage Danushka - AE9196

YTSP0300-3004- Data Modelling and Back-end Development

26th April 2024

Master in Full Stack Software Development

Instructor(s): Pasi Manninen

A black background with purple text

Description automatically generated

Contents

[1 Introduction 3](#_Toc164886051)

[2 Planning Documeation 3](#_Toc164886052)

[2.1 Purpose 3](#_Toc164886053)

[2.2 Scope 3](#_Toc164886054)

[2.3 Objectives 3](#_Toc164886055)

[2.4 Requirements 4](#_Toc164886056)

[2.4.1 Functional Requirements 4](#_Toc164886057)

[2.4.2 Non-functional Requirements 4](#_Toc164886058)

[2.5 Flow Chart 4](#_Toc164886059)

[2.6 Functionality 5](#_Toc164886060)

[2.6.1 Admin User Functionality 5](#_Toc164886061)

[2.6.2 Normal User functionality 5](#_Toc164886062)

[2.7 Architecture And Dependencies 5](#_Toc164886063)

[2.7.1 Design Architecture 5](#_Toc164886064)

[2.7.2 Dependencies 6](#_Toc164886065)

[2.8 Development plan 6](#_Toc164886066)

[2.8.1 Key milestones 6](#_Toc164886067)

[2.8.2 Timeline 6](#_Toc164886068)

[2.9 Test Strategy 7](#_Toc164886069)

[2.10 Conclusion 7](#_Toc164886070)

[3 Design Solution 8](#_Toc164886071)

[3.1 Solution Architecture 8](#_Toc164886072)

[3.2 Database Design Architecture 8](#_Toc164886073)

[3.2.1 MongoDB Collection Design 8](#_Toc164886074)

[3.2.2 ER diagram 9](#_Toc164886075)

[4 REST API Endpoints 10](#_Toc164886076)

[5 Structure of the Application 11](#_Toc164886077)

[6 Time Management 11](#_Toc164886078)

[7 Developed Solution structure 12](#_Toc164886079)

[7.1 App.js 12](#_Toc164886080)

[7.2 Login.js 13](#_Toc164886081)

[7.3 Signup\_route.js 13](#_Toc164886082)

[7.4 WishlistAdust.js 14](#_Toc164886083)

[7.5 MovieFilter.js 14](#_Toc164886084)

[8 Additional Information 15](#_Toc164886085)

[9 Conclusion 16](#_Toc164886086)

# Introduction

In this documentation, Basically covered the Backend architecture of the ticket booking application and in this part, I will focus on the back end level. In the backend level architecture design on the Node.js and the MondoDB. In the Design application will focus on the Ticket Booking application and it will ensure the basic functionality of the booking application has. In the application basically covered the RestAPI and JWT token optimization.

# Planning Documeation

## Purpose

The purpose of this document is to provide a comprehensive plan for developing a ticket booking application backend that allows users to register in the system, browse tickets, view additional information, and book tickets via the web platform.

## Scope

The scope of this application includes:

* User registration and authentication
* Ticket browsing and searching
* Showtime and additional information search
* Ticket booking and edit history

## Objectives

* To provide a user-friendly interface for booking tickets.
* To integrate with user registration systems and show available ticket count.

## Requirements

### Functional Requirements

* User registration and login
* Ticket listing with details
* Ticket scheduling
* Payment processing
* Booking confirmation and ticket generation

### Non-functional Requirements

* Security: Secure user data and payment information.
* Performance: Efficient loading and response times.
* Scalability: Able to handle many concurrent users.
* Usability: Intuitive and easy-to-use interface.

## A diagram of a company Description automatically generatedFlow Chart

## Functionality

In the application has two specific user types;

1. Admin User
2. Normal User

### Admin User Functionality

In the application level, user must always register with as a normal user. But, after that existing can change the user as an administrator. In this case, admin user can do few specific things when comparing other users.

* Add/Delete the ticket information
* Change the booking information
* Delete and update the user information
* Also possible to work as a normal user

### Normal User functionality

Normal users mean the users who visiting the application to check the ticket and booking the slots. In that case, they can simply work with the following functionality.

* Login/ sign up to the system
* Find out the relevant ticket information according to the define settings
* Check the ticket information according to selected ticket
* Book the ticket slot
* Update/ delete the already booked slot

## Architecture And Dependencies

### Design Architecture

* Language : Nodejs
* Database : MongoDB
* Version control - Git

### Dependencies

* Body-parser
* Cookie parser
* Cors
* Express
* Express-jwt
* MongoDB
* Passport

These are the required npm module dependencies that we can use for the design solution.

## Development plan

### Key milestones

1. Requirement analysis and documentation
2. API level design and identify the type
3. Design the API inputs and outputs
4. Develop the API
5. Test the API and finalize the API endpoints

### Timeline

1. Week 03 - 04 : Requirement analysis and documentation
2. Week 05 : Discuss with supervisor and get approval for the project
3. Week 05-09: Learn about the NodeJS and MongoDB
4. Week 10-12: Design the API endpoint and finalize API formats
5. Week 12-14 : Develop the API endpoint
6. Week 14 -16 :Integrate with the frontend
7. Week 17 : Test and finalize the documentation

## Test Strategy

The testing of the API’s body and result will ensure with the Postman application. In that case, we can simply ensure the result is working fine.

## Conclusion

In this part, will describe the plan according to the backend level of the application. The overall documentation related to planning it will ensure the features, requirements and architecture according to design rest API endpoints.

# Design Solution

## Solution Architecture

In the design application, backend level is designed by using the NodeJS and MongoDB.

## Database Design Architecture

### MongoDB Collection Design

Basically, in there I used major five (5) collections to store information.

1. LoginInfo – Store the information related to logging information
2. Movies – Store the information related to ticket
3. User\_Booking - Store the information related to user booked ticket information
4. User\_info – Store the information related to user registration
5. WhishList – Store the information related to wish list (favorite)

A screenshot of a computer

Description automatically generated

### ER diagram

A screenshot of a computer

Description automatically generatedHere, I used the MongoDB, and I attached this ER diagram only for the identifying the basic entities that I used in my application.

# REST API Endpoints

These are the basic Rest Endpoints in the design application.

Get Request :

1. Login
2. MovieInfoLoad
3. UserHistoryLoad
4. UserHistoryDelete

Post Request :

1. SignUp
2. MovieAdd
3. MovieLoad
4. MovieFilter
5. UserAdd
6. UserLoad
7. BookTickets

Delete Request:

1. MovieDelete
2. UserDelete

Put Request:

1. MovieUpdate
2. UserUpdate

Patch Request:

1. wishlistAdjust

# Structure of the Application

A diagram of a software system

Description automatically generatedIn the backend structure of the application like this.

In the design application, basically work as a monolithic based design and it will ensure the basic functionality that the ticket booking application has.

# Time Management

In the application level, I used the Git as the version control system, and it will show all the things like this.

Use : <https://github.com/Nandalochana/Application-Framework-CA-MAS-/commits/main/Server>

* Start date : 2024- January 29
* End date : 2024 April 26
* Total time : 88 Days
* Learning duration : 50 + days
* Development time : 30+ days

It takes some time to learn the nodejs and MongoDB. Because this is the first time that I have worked with Node.js and MongoDB. I took more time to learn this. But I think I did my best to learn, and right now I have a good knowledge of Node.js and MongoDB.

# Developed Solution structure

Here, I attached the basic and few routes and logic parts only. If you need more information, please go to the server level in here: https://github.com/Nandalochana/Application-Framework-CA-MAS-/tree/main/Server/routes

## App.js

The basic routs are like this.

A screen shot of a computer program

Description automatically generated

## Login.js

Login end point is like this.

A screenshot of a computer program

Description automatically generated

## A screen shot of a computer code Description automatically generatedSignup\_route.js

## WishlistAdust.js

A screen shot of a computer code

Description automatically generated

## MovieFilter.js

A screen shot of a computer code

Description automatically generated

# Additional Information

A screen shot of a computer

Description automatically generatedIn the backend level, I used JWT as token authentication library and it will ensure the data security of the application. Here are few additional libraries that I used.

# Conclusion

In the design application, the backend level will focus on Node.js and MongoDB. Basically, I use the JWT as the authentication library. In the application development phase, more than 60% of the time was spent learning the framework and the MongoDB concept. This concept is new for me, and I think I achieved my target very well. The authentication part is the hardest part that I must manage; in the present code, there are also some points to handle.

In this application, I learn the basic design principles related to Node.js and MongoDB. I think this will be a great opportunity to work with this framework. My great experience was the first time that I connected my router to MongoDB. I did heavy research for it. Also, I believe if I had a change, I would like to connect with Swagger. Then I can represent my API clearly. That is one drawback we now have. The easiest part is the deciding the router paths and I could manage it very easily with my experience. The hardest part is the connect with MongoDB and identify the query pattern. I can give the final grade as 4 (out of 5). Because of my hard work and research skills. As an application. The implemented functions are 100% fully working and finalized.

I think this assignment will be a turning point in my life.