
System Requirements Specification

Index

For

My Travel Buddy

Version 1.0

TABLE OF CONTENTS

| | |
|--|----|
| BACKEND-EXPRESS NODE APPLICATION | 3 |
| 1 Project Abstract | 3 |
| 2 Assumptions, Dependencies, Risks / Constraints | 4 |
| 2.1 Booking Constraints: | |
| 2.2 Destination Constraints | |
| 2.3 Review Constraints | |
| 2.4 Trip Plan Constraints | |
| 2.5 User Constraints | 4 |
| 3 Rest Endpoints | 5 |
| 3.1 BookingRoutes | |
| 3.2 DestinationRoutes | |
| 3.3 ReviewRoutes | |
| 3.4 TripPlansRoutes | |
| 3.5 UserRoutes | 5 |
| 4 Template Code Structure (modules) | 6 |
| 4.1 controller | 6 |
| 4.2 dao | |
| 4.3 routes | |
| 4.4 service | |
| 4.5 serviceImpl | 9 |
| 5 Considerations | 9 |
| FRONTEND-REACTJ SPA | 10 |
| 1 Problem Statement | 10 |
| 2 Proposed My Time Away Wireframe | 10 |
| 2.1 Home Page | 10 |
| 2.2 Booking Page | 11 |
| 2.3 Destination Page | 11 |
| 2.4 Login Page | |
| 2.5 Review Page | |
| 2.6 Trip Plan Page | |
| 2.7 User Page | |
| 3 Business-Requirement: | 12 |
| 7 Execution Steps to Follow for Backend | 14 |
| 8 Execution Steps to Follow for Frontend | 15 |

MY TRAVEL BUDDY

System Requirements Specification

You need to consume APIs exposed by Backend application and implement authentication (if required) in ReactJs to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

"My Travel Buddy" is a full-stack e-commerce application designed to provide a seamless online travel booking experience. It leverages the MERN stack, with MongoDB as the database, Express.js for the backend, ReactJs for the frontend, and Node.js as the runtime environment. This platform aims to connect travelers with the tour guides to provide unimaginable journeys.

Following is the requirement specifications:

| | |
|--------------------------------|-----------------------|
| | My Travel Buddy |
| | |
| Modules | |
| 1 | Auth |
| 2 | Booking |
| 3 | Destination |
| 4 | Review |
| 5 | Trip plan |
| 6 | User |
| | |
| Auth Module Functionalities | |
| | |
| 1 | Login user |
| 2 | Change the password |
| | |
| Booking Module Functionalities | |
| | |
| 1 | Get all bookings |
| 2 | Create booking |
| 3 | Search booking |
| 4 | Get upcoming bookings |

| | |
|---|----------------------|
| 5 | Get booking by id |
| 6 | Update booking by id |
| 7 | Delete booking by id |
| | |

| | |
|--|-------------------------------|
| Destination Module Functionalities | |
| | |
| 1 | Create destination |
| 2 | Get all destinations |
| 3 | Get top rated destinations |
| 4 | Search destination |
| 5 | Search destination by budget |
| 6 | Get destination details by id |
| 7 | Update destination by id |
| 8 | Delete destination by id |
| | |

| | |
|----------------------------------|------------------------------|
| Review Module Functionalities | |
| | |
| 1 | Get all reviews |
| 2 | Create a review |
| 3 | Search review by destination |
| 4 | Search review by rating |
| 5 | Get review by id |
| 6 | Update review by id |
| 7 | Delete review by id |
| | |

| | |
|-------------------------------------|------------------------|
| Trip Plan Module Functionalities | |
| | |
| 1 | Create a trip plan |
| 2 | Get popular trip plans |
| 3 | Search trip plan |
| 4 | Get my trip plans |
| 5 | Get all trip plans |
| 6 | Get trip plan by id |
| 7 | Update trip plan by id |
| 8 | Delete trip plan by id |
| | |

| User Module Functionalities | |
|-----------------------------|-----------------------------------|
| | |
| 1 | Create user |
| 2 | Get all upcoming trips for a user |
| 3 | Get all past trips for a user |
| 4 | Get all trips for a user |
| 5 | Get all booking for a user |
| 6 | Get all reviews by a user |
| 7 | Get user details |
| 8 | Updated user details |
| 9 | Delete user details |
| | |

2 ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 AUTH CONSTRAINTS

- 2.2 When the login api fails, it must return a proper message like "Authentication failed. User not found" or "Authentication failed. Incorrect password".

2.3 BOOKING CONSTRAINTS

- 2.4 When creating a booking, user and destination fields are must to have.
- 2.5 When searching a booking, we must pass status in the query param.
- 2.6 When getting a booking by id, if id is not found it should return Booking not found.
- 2.7 When updating a booking by id, if id is not found it should return Booking not found.
- 2.8 When deleting a booking by id, if id is not found it should return Booking not found.

2.9 DESTINATION CONSTRAINTS

- 2.10 When creating a destination, the name field is a must to have.
- 2.11 When searching a destination, it is required to pass either name or category.
- 2.12 When searching a destination by budget, min and max values are must to pass.
- 2.13 When getting a destination by id, if id is not found it should return Destination not found.
- 2.14 When updating a destination by id, if id is not found it should return Destination not found.
- 2.15 When deleting a destination by id, if id is not found it should return Destination not found.

2.16REVIEW CONSTRAINTS

- 2.17 When creating a review, user, destination and rating must have fields.
- 2.18 When searching a review by destination, destinationName is a must have field.
- 2.19 When searching review by rating, min and max are must have fields.
- 2.20 When getting a review by id, if id is not found it should return Review not found.
- 2.21 When updating a review by id, if id is not found it should return Review not found.
- 2.22 When deleting a review by id, if id is not found it should return Review not found.

2.23TRIP PLAN CONSTRAINTS

- 2.24 When creating a trip plan, user and destination are must have fields.
- 2.25 When searching a trip plan by destination, destinationName is a must have field.
- 2.26 When getting a trip plan by id, if id is not found it should return Trip plan not found.
- 2.27 When updating a trip plan by id, if id is not found it should return Trip plan not found.
- 2.28 When deleting a trip plan by id, if id is not found it should return Trip plan not found.

2.29USER CONSTRAINTS

- When getting a user by id, if id is not found it should return User not found.
- When updating a user by id, if id is not found it should return User not found.
- When deleting a user by id, if id is not found it should return User not found.

Common Constraints

- All the database operations must be implemented in serviceImpl file only.
- Do not change, add, remove any existing methods in the service file.
- In the service layer, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data in json format.
- Any type of authentication and authorisation must be added in routes file only.

3 REST ENDPOINTS

Rest End-points to be exposed in the routes file and attached with controller method along with method details for the same to be created. Please note, that these all are required to be implemented.

3.1 AUTH CONTROLLER

| URL Exposed | | Purpose |
|--------------------|------|-----------------------------------|
| 1. /api/auth/login | | Logins the user and returns token |
| Http Method | POST | |
| Parameter | - | |

| | | |
|-----------------------------|-------------------------|---------------------------------|
| Return | Token | |
| 2. /api/auth/changePassword | | Changes the password for a user |
| Http Method | POST | |
| Parameter | - | |
| Return | Acknowledgement message | |

3.2 BOOKING CONTROLLER

Note: All routes must be authenticated.

| URL Exposed | | Purpose |
|------------------|------------------|-----------------------|
| 1. /api/bookings | | Fetches all bookings |
| Http Method | GET | |
| Parameter | - | |
| Return | List of bookings | |
| 2. /api/bookings | | Creates a new booking |
| Http Method | POST | |
| Parameter | - | |
| Return | Booking | |

| | | |
|---------------------------|---------------------------|--|
| 3. /api/bookings/search | | Searches a booking by status of destination name |
| Http Method | GET | |
| Parameter | status or destinationName | |
| Return | List of bookings | |
| 4. /api/bookings/upcoming | | Getts list of all upcoming bookings |
| Http Method | GET | |
| Parameter | - | |
| Return | List of bookings | |

| | | |
|-----------------------------|-----------|-----------------------------|
| 5. /api/bookings/:bookingId | | Fetches a booking by its id |
| Http Method | GET | |
| Parameter | bookingId | |
| Return | Booking | |
| 6. /api/bookings/:bookingId | | Updates a booking by id |
| Http Method | PUT | |
| Parameter | bookingId | |
| Return | Booking | |

| | | |
|-----------------------------|-----------|-------------------------|
| 7. /api/bookings/:bookingId | | Deletes a booking by id |
| Http Method | DELETE | |
| Parameter | bookingId | |
| Return | Booking | |

Destination CONTROLLER

Note: Routes which must be authenticated are mentioned with “isSecured” below

| URL Exposed | | Purpose |
|----------------------|----------------------|---------------------------|
| 1. /api/destinations | | Creates a new destination |
| Http Method | POST | |
| Parameter | - | |
| Return | Destination | |
| 2. /api/destinations | | Fetches all destinations |
| Http Method | GET | |
| Parameter | - | |
| Return | List of destinations | |

| | | |
|--------------------------------|----------------------|---|
| 3. /api/destinations/top-rated | | Fetches all top rated destinations |
| Http Method | GET | |
| Parameter | - | |
| Return | List of destinations | |
| 4. /api/destinations/search | | Searches destinations by name or category |
| Http Method | GET | |
| Query param | name category | |
| Return | List of destinations | |

| | | |
|-------------------------------------|----------------------|---|
| 5. /api/destinations/search/budget | | Searches destinations by min and max budget |
| Http Method | GET | |
| Query param | min max | |
| Return | List of destinations | |
| 6. /api/destinations/:destinationId | | Fetches a destination by id |
| Http Method | GET | |
| Parameter | destinationId | |
| Return | Destination | |

| | | |
|-------------------------------------|-----|-----------------------------|
| 7. /api/destinations/:destinationId | | Updates a destination by id |
| Http Method | PUT | |

| | | |
|-------------------------------------|---------------|---|
| Parameter | destinationId | [isSecured] |
| Return | Destination | |
| 8. /api/destinations/:destinationId | | Deletes a destination by id [isSecured] |
| Http Method | DELETE | |
| Parameter | destinationId | |
| Return | Destination | |

REVIEW CONTROLLER

Note: Routes which must be authenticated are mentioned with “isSecured” below.

| URL Exposed | | Purpose |
|-------------------------------|------------------|--|
| 1. /api/reviews | | Fetches all reviews |
| Http Method | GET | |
| Parameter | - | |
| Return | List of reviews | |
| 2. /api/reviews | | Creates a new review [isSecured] |
| Http Method | POST | |
| Parameter | - | |
| Return | Review | |
| 3. /api/reviews/search | | Searches reviews by destination name |
| Http Method | GET | |
| Query param | destination name | |
| Return | List of reviews | |
| 4. /api/reviews/search/rating | | Searches reviews by min and max rating |
| Http Method | GET | |
| Query param | min max | |
| Return | List of reviews | |
| 5. /api/reviews/:reviewId | | Deletes a review by id [isSecured] |
| Http Method | DELETE | |
| Parameter | reviewId | |
| Return | Review | |
| 6. /api/reviews/:reviewId | | Fetches a review by id |
| Http Method | GET | |
| Parameter | reviewId | |
| Return | Review | |

| | | |
|--------------------------|----------|------------------|
| 7. /api/reviews:reviewId | | Updates a review |
| Http Method | PUT | |
| Parameter | reviewId | |
| Return | Review | |

TRIP PLAN CONTROLLER

Note: Routes which must be authenticated are mentioned with “isSecured”.

| URL Exposed | Purpose | | | | | | |
|---|--------------------|------|-----------|---|--------|--------------------|----------------------------|
| 1. /api/trips <table> <tr><td>Http Method</td><td>POST</td></tr> <tr><td>Parameter</td><td>-</td></tr> <tr><td>Return</td><td>TripPlan</td></tr> </table> | Http Method | POST | Parameter | - | Return | TripPlan | Creates a new trip plan |
| Http Method | POST | | | | | | |
| Parameter | - | | | | | | |
| Return | TripPlan | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 2. /api/trips/popular <table> <tr><td>Http Method</td><td>GET</td></tr> <tr><td>Parameter</td><td>-</td></tr> <tr><td>Return</td><td>List of trip plans</td></tr> </table> | Http Method | GET | Parameter | - | Return | List of trip plans | Fetches popular trip plans |
| Http Method | GET | | | | | | |
| Parameter | - | | | | | | |
| Return | List of trip plans | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|--|--|-----|-------------|--|--------|--------------------|---|
| 3. /api/trips/search <table> <tr><td>Http Method</td><td>GET</td></tr> <tr><td>Query param</td><td>destination name min and max budget</td></tr> <tr><td>Return</td><td>List of trip plans</td></tr> </table> | Http Method | GET | Query param | destination name min and max budget | Return | List of trip plans | Searches list of trip plans by destination name of min and max budget |
| Http Method | GET | | | | | | |
| Query param | destination name min and max budget | | | | | | |
| Return | List of trip plans | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 4. /api/trips/me <table> <tr><td>Http Method</td><td>GET</td></tr> <tr><td>Parameter</td><td>-</td></tr> <tr><td>Return</td><td>List of trip plans</td></tr> </table> | Http Method | GET | Parameter | - | Return | List of trip plans | Gets a trip plan by user [isSecured] |
| Http Method | GET | | | | | | |
| Parameter | - | | | | | | |
| Return | List of trip plans | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|--|-------------|--------|-----------|------------|--------|----------|---------------------------|
| 5. /api/trips <table> <tr><td>Http Method</td><td>DELETE</td></tr> <tr><td>Parameter</td><td>id</td></tr> <tr><td>Return</td><td>TripPlan</td></tr> </table> | Http Method | DELETE | Parameter | id | Return | TripPlan | Gets a trip plans |
| Http Method | DELETE | | | | | | |
| Parameter | id | | | | | | |
| Return | TripPlan | | | | | | |
| | | | | | | | |
| | | | | | | | |
| 6. /api/trips/:tripPlanId <table> <tr><td>Http Method</td><td>GET</td></tr> <tr><td>Parameter</td><td>tripPlanId</td></tr> <tr><td>Return</td><td>TripPlan</td></tr> </table> | Http Method | GET | Parameter | tripPlanId | Return | TripPlan | Fetches a trip plan by id |
| Http Method | GET | | | | | | |
| Parameter | tripPlanId | | | | | | |
| Return | TripPlan | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|---|-------------|-----|-----------|------------|--------|----------|---------------------------|
| 7. /api/trips/tripPlanId <table> <tr><td>Http Method</td><td>PUT</td></tr> <tr><td>Parameter</td><td>tripPlanId</td></tr> <tr><td>Return</td><td>TripPlan</td></tr> </table> | Http Method | PUT | Parameter | tripPlanId | Return | TripPlan | Updates a trip plan by id |
| Http Method | PUT | | | | | | |
| Parameter | tripPlanId | | | | | | |
| Return | TripPlan | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | |
|---------------------------|------------|---------------------------|
| 8. /api/trips/:tripPlanId | | Deletes a trip plan by id |
| Http Method | DELETE | |
| Parameter | tripPlanId | |
| Return | TripPlan | |

USER CONTROLLER

Note: Routes which must be authenticated are mentioned with “isSecured”.

| URL Exposed | | Purpose |
|------------------------------|--------------------|---|
| 1. /api/users | | Creates a new user |
| Http Method | POST | |
| Parameter | - | |
| Return | User | |
| 2. /api/users/upcoming-trips | | Fetches upcoming trips booked by passed token user [isSecured] |
| Http Method | GET | |
| Parameter | - | |
| Return | List of trip plans | |

| | | | |
|--------------------------|--------------------|--|--|
| 3. /api/users/past-trips | | | Fetches past trips done by passed token user [isSecured] |
| Http Method | GET | | |
| Parameter | - | | |
| Return | List of trip plans | | |
| 4. /api/users/all-trips | | | Fetches all booking done by passed token user [isSecured] |
| Http Method | GET | | |
| Parameter | - | | |
| Return | List of trip plans | | |

| | | |
|------------------------|------------------|---|
| 5. /api/users/bookings | | Fetches bookings done by passed token user [isSecured] |
| Http Method | GET | |
| Parameter | - | |
| Return | List of bookings | |
| 6. /api/users/reviews | | Fetches reviews given by passed token user [isSecured] |
| Http Method | GET | |
| Parameter | - | |
| Return | List of reviews | |

| | | |
|---------------|-----|--------------------------------|
| 7. /api/users | | Fetches a user by passed token |
| Http Method | GET | |

| | | |
|---------------|------|--|
| Parameter | - | [isSecured] |
| Return | User | |
| 8. /api/users | | Updates a user by passed token [isSecured] |
| Http Method | PUT | |
| Parameter | - | |
| Return | User | |

| | | |
|---------------|--------|--|
| 9. /api/users | | Deletes a user by passed token [isSecured] |
| Http Method | DELETE | |
| Parameter | - | |
| Return | User | |

4 TEMPLATE CODE STRUCTURE

4.1 Auth code structure

1) MODULES/AUTH: controller

Resources

| | | |
|----------------------------------|---|-------------------|
| AuthController (Class) | This is the controller class for the auth module. | To be implemented |
|----------------------------------|---|-------------------|

2) MODULES/AUTH: middleware

Resources

| File | Description | Status |
|--------------------------------|----------------------|-------------------|
| authGuard (function) | Authentication guard | To be implemented |

3) MODULES/AUTH: routes

Resources

| File | Description | Status |
|--------------------|-----------------|------------------------|
| Auth routes | Routes for auth | Partially implemented. |

4.1 Booking code structure

1) MODULES/BOOKING: controller

Resources

| | | |
|-------------------------------------|--|-------------------|
| BookingController (Class) | This is the controller class for the booking module. | To be implemented |
|-------------------------------------|--|-------------------|

2) MODULES/BOOKING: dao

Resources

| File | Description | Status |
|------------------------|---------------------|---------------------|
| models/booking model | Models for booking | Already implemented |
| schemas/booking schema | Schemas for booking | Already implemented |

3) MODULES/BOOKING: routes

Resources

| File | Description | Status |
|----------------|--------------------|------------------------|
| Booking routes | Routes for booking | Partially implemented. |

4) MODULES/BOOKING: service

Resources

| Class | Description | Status |
|----------------|--|----------------------|
| BookingService | <ul style="list-style-type: none">Defines BookingService | Already implemented. |

5) MODULES/BOOKING: service/impl

Resources

| Class | Description | Status |
|--------------------|--|--------------------|
| BookingServiceImpl | <ul style="list-style-type: none">Implements BookingService. | To be implemented. |

4.1 Destination code structure

1) MODULES/DESTINATION: controller

Resources

| | | |
|----------------------------------|--|-------------------|
| DestinationController (Class) | This is the controller class for the destination module. | To be implemented |
|----------------------------------|--|-------------------|

2) MODULES/DESTINATION: dao

Resources

| File | Description | Status |
|----------------------------|-------------------------|---------------------|
| models/destination model | Models for destination | Already implemented |
| schemas/destination schema | Schemas for destination | Already implemented |

3) MODULES/DESTINATION: routes

Resources

| File | Description | Status |
|--------------------|------------------------|------------------------|
| Destination routes | Routes for destination | Partially implemented. |

4) MODULES/DESTINATION: service

Resources

| Class | Description | Status |
|--------------------|--|----------------------|
| DestinationService | <ul style="list-style-type: none">Defines DestinationService | Already implemented. |

5) MODULES/DESTINATION: service/impl

Resources

| Class | Description | Status |
|------------------------|--|--------------------|
| DestinationServiceImpl | <ul style="list-style-type: none">Implements DestinationService. | To be implemented. |

4.1 Review code structure

1) MODULES/REVIEW: controller

Resources

| | | |
|-----------------------------|---|-------------------|
| ReviewController (Class) | This is the controller class for the review module. | To be implemented |
|-----------------------------|---|-------------------|

2) MODULES/REVIEW: dao

Resources

| File | Description | Status |
|-----------------------|--------------------|---------------------|
| models/review model | Models for review | Already implemented |
| schemas/review schema | Schemas for review | Already implemented |

3) MODULES/REVIEW: routes

Resources

| File | Description | Status |
|---------------|-------------------|------------------------|
| Review routes | Routes for review | Partially implemented. |

4) MODULES/REVIEW: service

Resources

| Class | Description | Status |
|---------------|---|----------------------|
| ReviewService | <ul style="list-style-type: none">Defines ReviewService | Already implemented. |

5) MODULES/REVIEW: service/impl

Resources

| Class | Description | Status |
|-------------------|---|--------------------|
| ReviewServiceImpl | <ul style="list-style-type: none">Implements ReviewService. | To be implemented. |

TripPlan code structure

1) MODULES/TRIPPLAN: controller

Resources

| | | |
|-------------------------------|--|-------------------|
| TripPlanController (Class) | This is the controller class for the trip plan module. | To be implemented |
|-------------------------------|--|-------------------|

2) MODULES/TRIPPLAN: dao

Resources

| File | Description | Status |
|------------------------|----------------------|---------------------|
| models/trip plan model | Models for trip plan | Already implemented |

| | | |
|---------------------------------|-----------------------|---------------------|
| schemas/trip plan schema | Schemas for trip plan | Already implemented |
|---------------------------------|-----------------------|---------------------|

3) MODULES/TRIPPLAN: routes

Resources

| File | Description | Status |
|------------------------|----------------------|------------------------|
| TripPlan routes | Routes for trip plan | Partially implemented. |

4) MODULES/TRIPPLAN: service

Resources

| Class | Description | Status |
|------------------------|---|----------------------|
| TripPlanService | <ul style="list-style-type: none"> Defines TripPlanService | Already implemented. |

5) MODULES/TRIPPLAN: service/impl

Resources

| Class | Description | Status |
|----------------------------|---|--------------------|
| TripPlanServiceImpl | <ul style="list-style-type: none"> Implements TripPlanService. | To be implemented. |

User code structure

1) MODULES/USER: controller

Resources

| | | |
|-----------------------------------|---|-------------------|
| UserController (Class) | This is the controller class for the user module. | To be implemented |
|-----------------------------------|---|-------------------|

2) MODULES/USER: dao

Resources

| File | Description | Status |
|---------------------|------------------|---------------------|
| models/user model | Models for user | Already implemented |
| schemas/user schema | Schemas for user | Already implemented |

3) MODULES/USER: routes

Resources

| File | Description | Status |
|-------------|-----------------|------------------------|
| User routes | Routes for user | Partially implemented. |

4) MODULES/USER: service

Resources

| Class | Description | Status |
|-------------|---|----------------------|
| UserService | <ul style="list-style-type: none">Defines UserService | Already implemented. |

5) MODULES/USER: service/impl

Resources

| Class | Description | Status |
|-----------------|---|--------------------|
| UserServiceImpl | <ul style="list-style-type: none">Implements UserService. | To be implemented. |

FRONTEND-REACTJS SPA

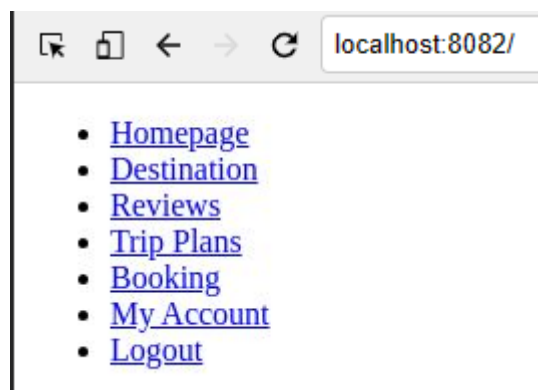
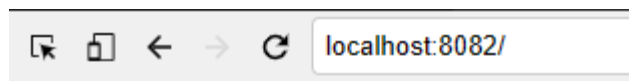
1 PROBLEM STATEMENT

"My Travel Buddy" is a SPA in ReactJs with full-stack e-commerce application designed to provide a seamless online travel booking experience

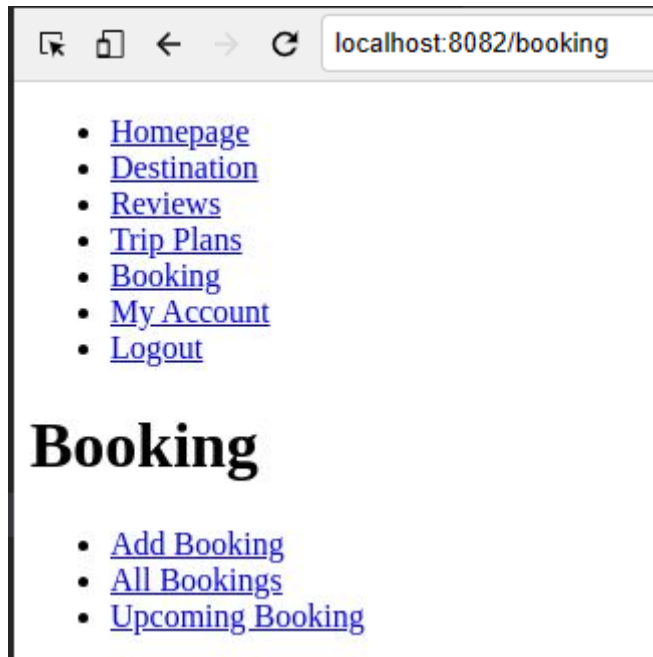
2 PROPOSED MY TRAVEL BUDDY

UI needs improvisation and modification as per given use case and to make test cases passed.

Home page



Booking page

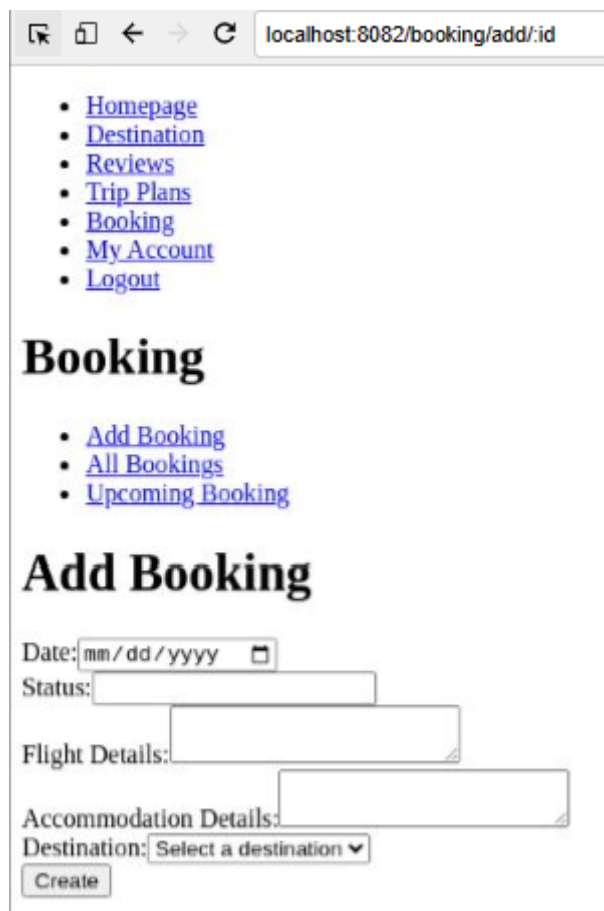


A screenshot of a web browser showing the 'Booking' page. The address bar displays 'localhost:8082/booking'. The page features a sidebar with a list of links: 'Homepage', 'Destination', 'Reviews', 'Trip Plans', 'Booking', 'My Account', and 'Logout'. The main content area has a large heading 'Booking' followed by a list of links: 'Add Booking', 'All Bookings', and 'Upcoming Booking'.

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Booking

- [Add Booking](#)
- [All Bookings](#)
- [Upcoming Booking](#)




A screenshot of a web browser showing the 'Add Booking' page. The address bar displays 'localhost:8082/booking/add/.id'. The page features a sidebar with a list of links: 'Homepage', 'Destination', 'Reviews', 'Trip Plans', 'Booking', 'My Account', and 'Logout'. The main content area has a large heading 'Add Booking' followed by a form with the following fields: 'Date' (with a date picker icon), 'Status', 'Flight Details', 'Accommodation Details', and 'Destination' (a dropdown menu). A 'Create' button is located at the bottom left of the form.

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Booking

- [Add Booking](#)
- [All Bookings](#)
- [Upcoming Booking](#)

Add Booking

Date: 

Status:

Flight Details:

Accommodation Details:

Destination:

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Booking

- [Add Booking](#)
- [All Bookings](#)
- [Upcoming Booking](#)

Bookings

Search Bookings

Destination Name:

Status:

No bookings available.

Destination Page



- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Destination

- [Add Destination](#)
- [All Destinations](#)
- [TopRated Destinations](#)

Edit Destination

Name:

Description:

Category:

Budget:

Image URL:

localhost:8082/destination/all

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Destination

- [Add Destination](#)
- [All Destinations](#)
- [TopRated Destinations](#)

All Destinations

Search Destination

Name:

Category:

Min Budget:

Max Budget:

No Destination

Login Page

localhost:8082/login

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Login](#)

Login Account

Email:

Password:

localhost:8082/login

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Login](#)

Create New Account

Username:

Email:

Password:

Create Account

Back to Login

localhost:8082/trips/add/:id

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Trips

- [Add Trip](#)
- [All Trips](#)
- [Popular Trips](#)
- [My Trips](#)

Add Trip

Destination:

Select a destination ▾

Start Date:

mm / dd / yyyy

End Date:

mm / dd / yyyy

Activities:

+ Add Activity

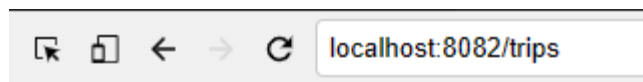
Accommodations:

Hotel Name

mm / dd / yyyy

mm / dd / yyyy

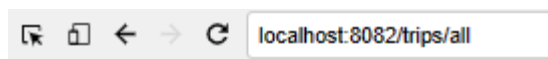
Create Trip



- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Trips

- [Add Trip](#)
- [All Trips](#)
- [Popular Trips](#)
- [My Trips](#)



- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

Trips

- [Add Trip](#)
- [All Trips](#)
- [Popular Trips](#)
- [My Trips](#)

All Trips

No trips available.

localhost:8082/user

- [Homepage](#)
- [Destination](#)
- [Reviews](#)
- [Trip Plans](#)
- [Booking](#)
- [My Account](#)
- [Logout](#)

User Details

Username:

Email:

Password:

Other pages i.e Review, Trip Plan and User are similar like above one.

3 BUSINESS-REQUIREMENT:

As an application developer, develop the E-Mall ProConnect (Single Page App) with below guidelines:

| User Story # | User Story Name | User Story |
|--------------|-----------------|---|
| US_01 | Home Page | <p>As a user I should be able to visit the Home page as the default page and below links are accessible to me:</p> <ol style="list-style-type: none"> 1. Homepage 2. Destination 3. Reviews 4. Trip Plans 5. Login 6. Booking (only when logged in) |

| | | |
|-------|------------------|--|
| | | <p>7. My Account (only when logged in)</p> <p>8. Logout (only when logged in)</p> |
| US_02 | Login Page | <p>As a user I should be able to visit the Login page once I click on the Login button. And there should be a create account button, on clicking it, I should be able to see the form to create a new account.</p> |
| US_03 | Destination Page | <p>As a user I should be able to see the destination page (when clicked on homepage anchor tag) and perform all operations:</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> 1. Be able to see the Add Destination link and on clicking, it must open the appropriate page also. 2. Be able to see the All Destinations link and on clicking, it must open the appropriate page also. 3. Be able see TopRated Destinations link and on clicking, it must open the appropriate page also |
| US_04 | Review Page | <p>As a user I should be able to see the review page as:</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> 1. Have All Reviews link when and on clicking, it must open the appropriate page also |

| | | |
|-------|------------|--|
| US_05 | Trips Page | <p>As a user I should be able to see the trips page as:</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> 2. Be able see Add Trip link and on clicking, it must open the appropriate page also 3. Be able see All Trips link and on clicking, it must open the appropriate page also 4. Be able see Popular Trips link and on clicking, it must open the appropriate page also 5. Be able see My Trips link and on clicking, it must open the appropriate page also |
|-------|------------|--|

| | | |
|-------|--------------|--|
| US_06 | Booking Page | <p>As a user I should be able to see the booking page as:</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> 6. Be able see Add Booking link and on clicking, it must open the appropriate page also 7. Be able see All Booking link and on clicking, it must open the appropriate page also 8. Be able see Upcoming Booking link and on clicking, it must open the appropriate page also |
|-------|--------------|--|

| | | |
|-------|-----------------|--|
| US_07 | My Account Page | <p>As a user I should be able to see the booking page as:</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none"> 9. Be able see Add Booking link and on clicking, it must open the appropriate page also 10. Be able see All Booking link and on clicking, it must open the appropriate page also 11. Be able see Upcoming Booking link and on clicking, it must open the appropriate page also |
|-------|-----------------|--|

EXECUTION STEPS TO FOLLOW FOR BACKEND

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. This editor Auto Saves the code.
4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B**-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.
7. You can follow series of command to setup express environment once you are in your project-name folder:
 - a. npm install -> Will install all dependencies -> takes 10 to 15 min
 - b. npm run start -> To compile and run the project.
 - c. npm run jest -> to run all test cases and see the summary of all passed and failed test cases.
 - d. npm run test -> to run all test cases and register the result of all test cases. **It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
8. You need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

EXECUTION STEPS TO FOLLOW FOR FRONTEND

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. This is a web-based application, to run the application on a browser, use the internal browser in the environment.
4. You can follow series of command to setup ReactJs environment once you are in your project-name folder:
 - a. npm install -> Will install all dependencies -> takes 10 to 15 min
 - b. npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:8082 to open project in browser -> takes 2 to 3 min
 - c. npm run jest -> To run all test cases and check the summary.
 - d. npm run test -> to run all test cases. **It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. You need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.