Project 5 : Insurance Service

Project objective:

To build an application to manage the purchase of insurance

Background of the problem statement:

Raindrops Insurances Pvt. Ltd. has hired you as a full stack developer to create a web application for managing health insurance purchases. You're asked to develop a prototype of this application.

As part of this project, you will:

- Build functionality for taking relevant user data
- Build functionality for viewing and purchasing insurance plans
- Create appropriate navigation features within the application

The flow and features of the application:

The following are the primary features of the application:

- This includes an About Us module, where the following are displayed:
    - Application name and developer details
    - User interface and a guide to use the application
    - Key features of the application
    - Contact details and address of the company associated with the application
- This includes a series of questions displayed using pop-ups where the user would be prompted to enter key details required to calculate the premium amount. The details might include the age, earnings, and questions related to the current health conditions of the user.
- Once the user provides these key details, the premium amount should be calculated accordingly and displayed to the user along with other relevant plan details.
- Once the user selects a particular plan, the user should get the option to enter payment details and display a success message upon a successful subscription.
- Once the JSON server is setup (please refer to the Setting Up JSON Server Section), the user should be able to perform transactions with the JSON server by sending a request to the below-mentioned API Endpoints.

API Endpoints:

S NO.	API Description	Endpoint	HTTP Request, Query Parameters, Request/Response Format
1	View all Plans	http://localhost:3333/plans	GET: The response should contain the list of all available insurance plans. Please find the sample response, "**Response 1**."
2	View By Plan Id	http://localhost:3333/plans?planId=1	GET: Insurance plan ID will be passed as a request parameter and the plan details should be fetched as a part of the response. Please find the sample response, "**Response 2.**"
3	Book Now	http://localhost:3333/bookings	POST: Pass the user details and confirm booking details in the request body. Please find the sample request, "**Request Body 3**."

Response 1:

[

{

“planId”: 1,

"planName": “Care Life Plan”,

“description”: “The Care Life Plan safeguards you from unforeseen circumstances by providing instant coverage with no waiting period. Emergency services such as immediate ambulance, specialist consultation and complete assistance are its major features.”,

“baseAmt”: 2500,

},

{

“planId”: 2,

"planName": ”Good Health Plan”,

“description”: “The Good Health Plan is our best-selling plan. Facilities like Air Ambulance and Buy Now and Pay Later options to make it a stand out in the industry.”,

“baseAmt”: 2500

}

]

Response 2:

{

“planId”: 1,

"planName": ”Care Life Plan”,

“description”: “The Care Life Plan safeguards you from unforeseen circumstances by providing instant coverage with no waiting period. Emergency services such as immediate ambulance, specialist consultation, and complete assistance are its major features.”,

“baseAmt”: 2500,

}

Request Body 3:


```
{
   "name":"Justin",
   “city”:”Chennai”,
   “phone”:”2780808989”,
   "email":"hello@gmail.com",
   "age":"35",
   “planId”:1,
   "planName": “Care Life Plan”,
   “validity”: 30,
   “paymentMode”: “credit card”,
   “cardNumber”: “7890-3223-7778”,
   “premiumAmt”: 4500,
   “paymentFreq”: “yearly”
}
```


Technology Specifications:

The following technologies are used in the development of this application:

- HTML
- Bootstrap
- TypeScript
- Angular
- JSON Server

Setting up the JSON server:

- Run the **npm install json-server** command in your project directory
- Create the **db.json** file in the same directory to read and write the data
- Run the **npx json-server —watch -p 3333 db.json** command to start the JSON server
- Provide **db.json** with dummy data for insurance plans to set up the JSON server

The sample dummy data is shown below:


```
{
  "plans": [
    {
      "planId": 1,
      "planName": "Care Life Plan",
    "description": "The Care Life Plan safeguards you from unforeseen circumstances by providing instant coverage with no waiting period. Emergency services such as immediate ambulance, specialist consultation and complete assistance are its major features.",
      "baseAmt": 2500,
      "validity": 15
    },
    {
      "planId": 2,
      "planName": "Good Health Plan",
      "description": "The Good Health Plan is our best selling plan. Facilities like Air Ambulance and Buy Now and Pay Later options make it a stand out in the industry.",
      "baseAmt": 2500,
      "validity": 30
    }
  ],
```

"bookings": [

{

"id": 1,

"name": "Justin",

"city": "Chennai",

"phone": "2780808989",

"email": "hello@gmail.com",

"age": "35",

“planId”: 2,

"planName": "Good Health Plan",

"validity": 30,

"paymentMode": "credit card",

"cardNumber": "7890-3223-7778",

"premiumAmt": 2500,

"paymentFreq": "yearly"

},

{

"id": 2

"name": "Ravi Kumar",

"city": "Faridabad",

"phone": "+919899999990",

"email": "test@gmail.com",

"age": 22,

“planId”: 1,

"planName": "Care Life Plan",

"validity": "2023-12-11",

"paymentMode": "Credit Card",

"cardNumber": "1110-1212-2394",

"premiumAmt": 2500,

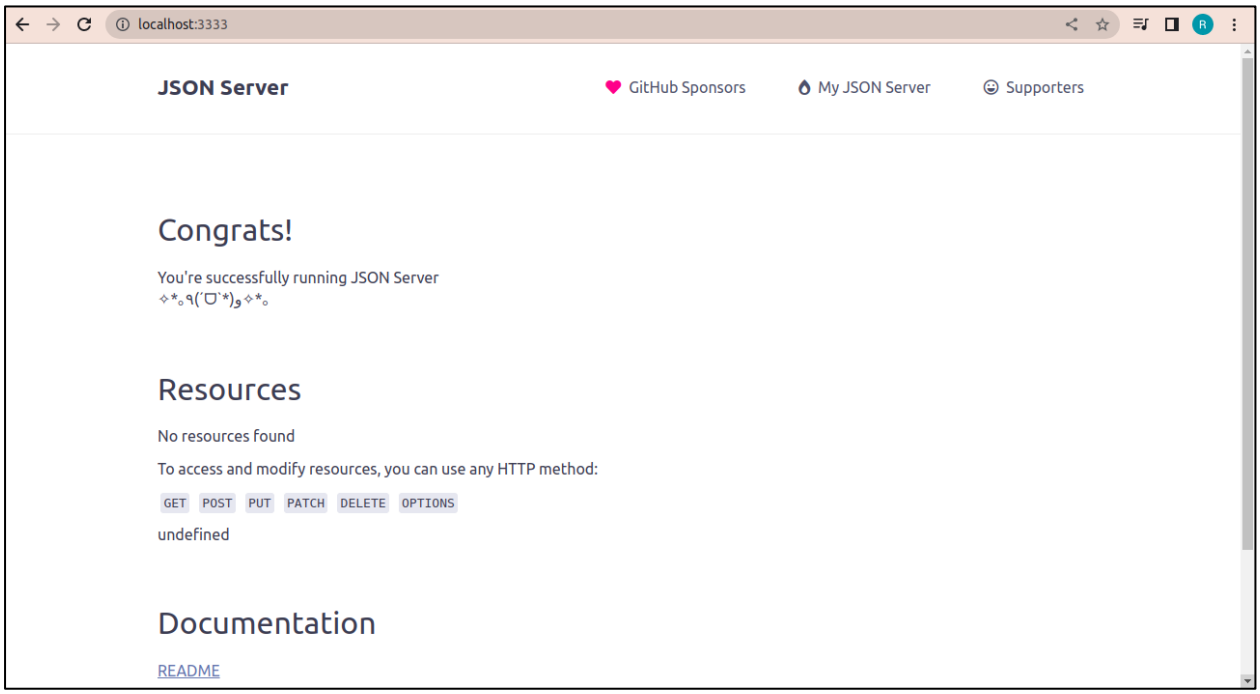
"paymentFreq": "Half Yearly",

}

]

}

- If the server is set up successfully, you should see the following in your browser once you try to access http://localhost:3333.



Validation (Frontend/Backend):

The application must undergo the following validations:

- Implement validation checks at relevant places.
- For example, age should not be negative, contact number should be of 10 digits, email ID should be valid, and so on.
- Additional validation checks may be added as per general understanding.

Business logic/exceptional scenarios:

The application shall have the following exception scenarios:

- The user must input the relevant details before submitting a request to buy an insurance plan. The **submit** button should be disabled until all the mandatory fields have been provided.
- Other exception scenarios may be added as per general understanding.

The following requirements should be met:

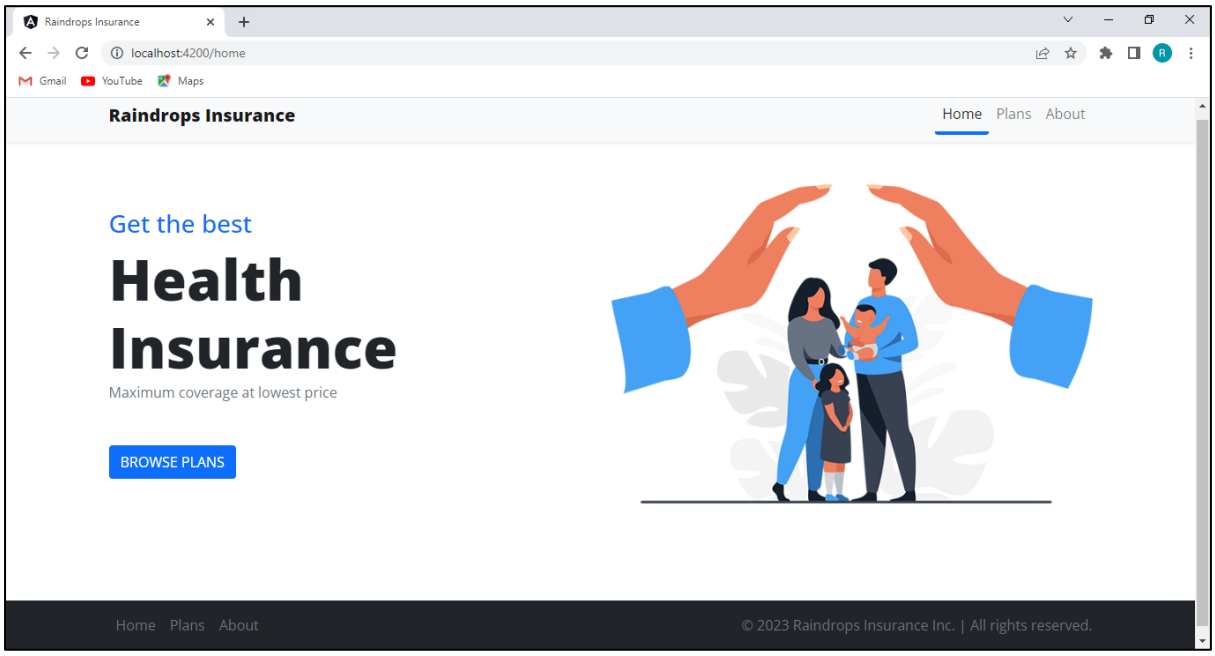
Ensure you have accomplished the following by the end of your application completion:

- Create an attractive and appealing user interface for buying insurance plans
- Check if all the planned business functionalities work as expected
- Ensure that the JSON server is set up properly to facilitate access and storage of data

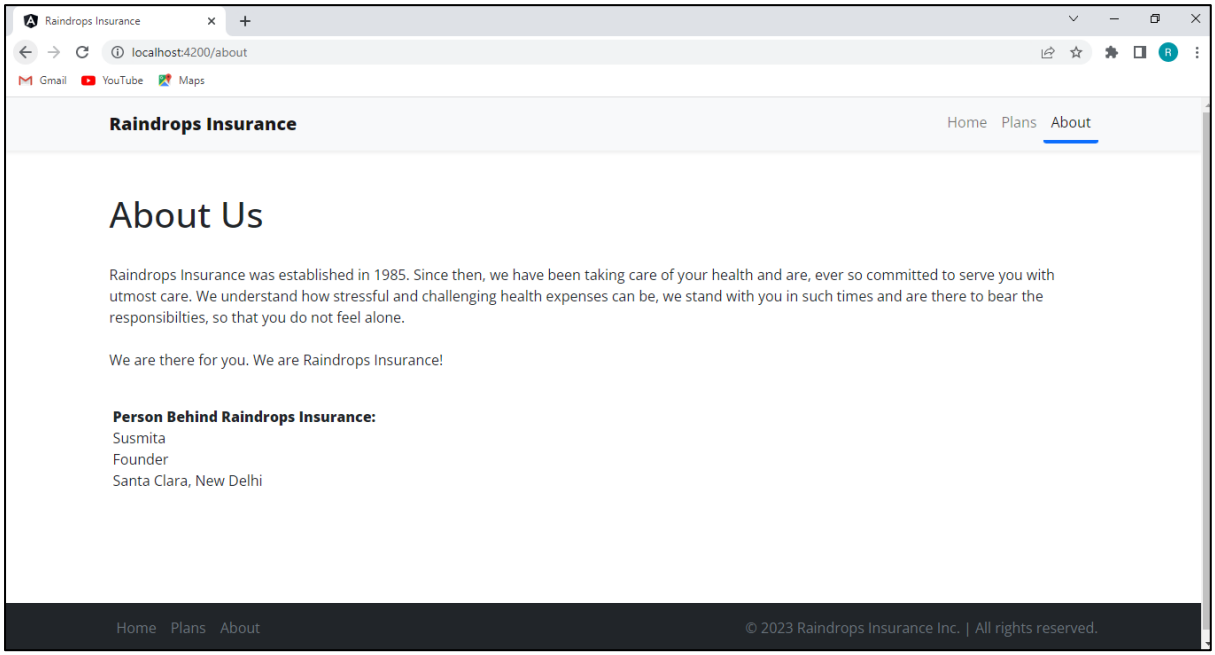
The application should not close, exit, or throw an unhandled exception if the user specifies invalid input.

These are the following screenshots:

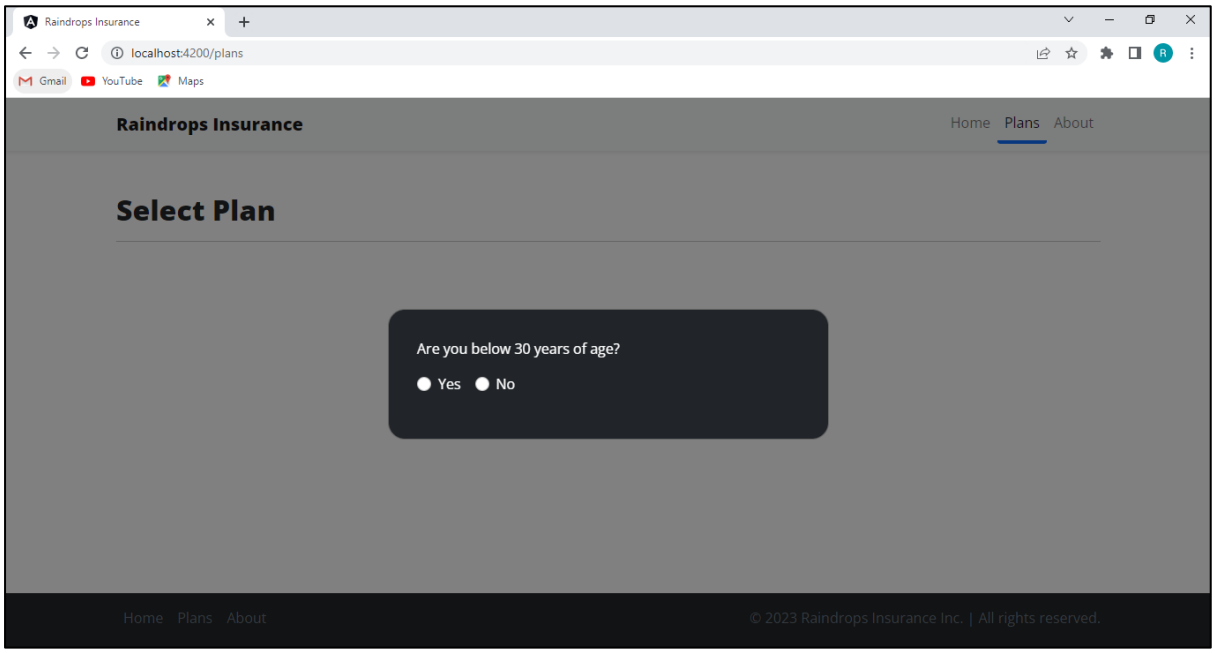
Home Page:



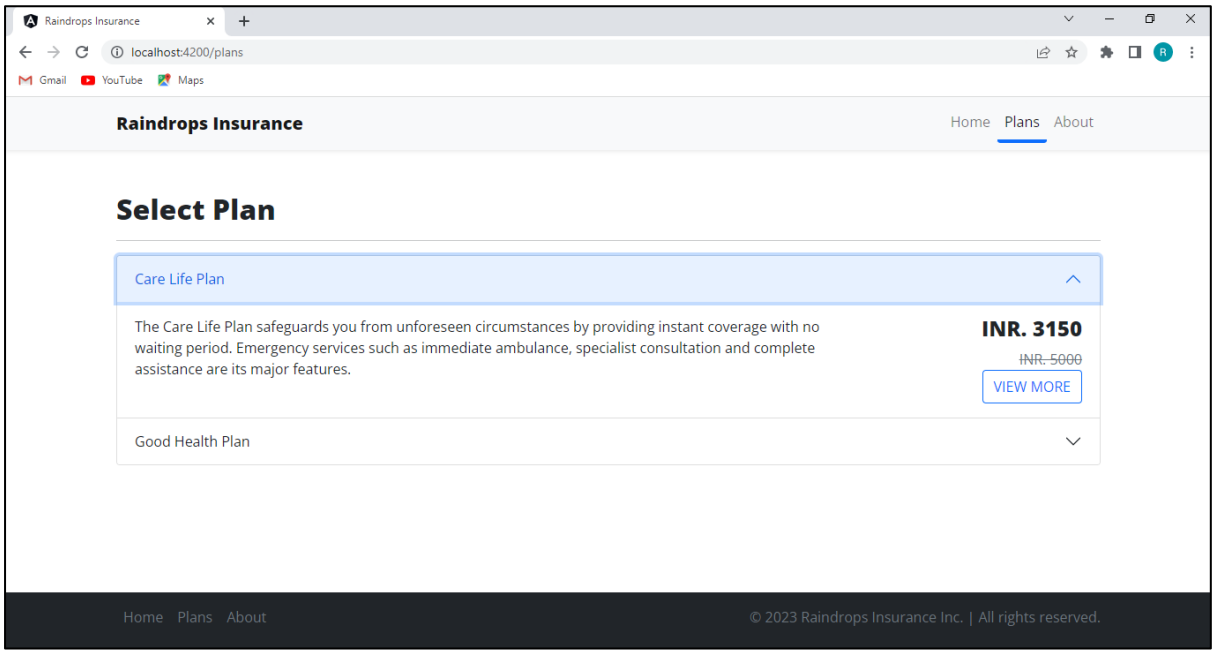
About Us Page:



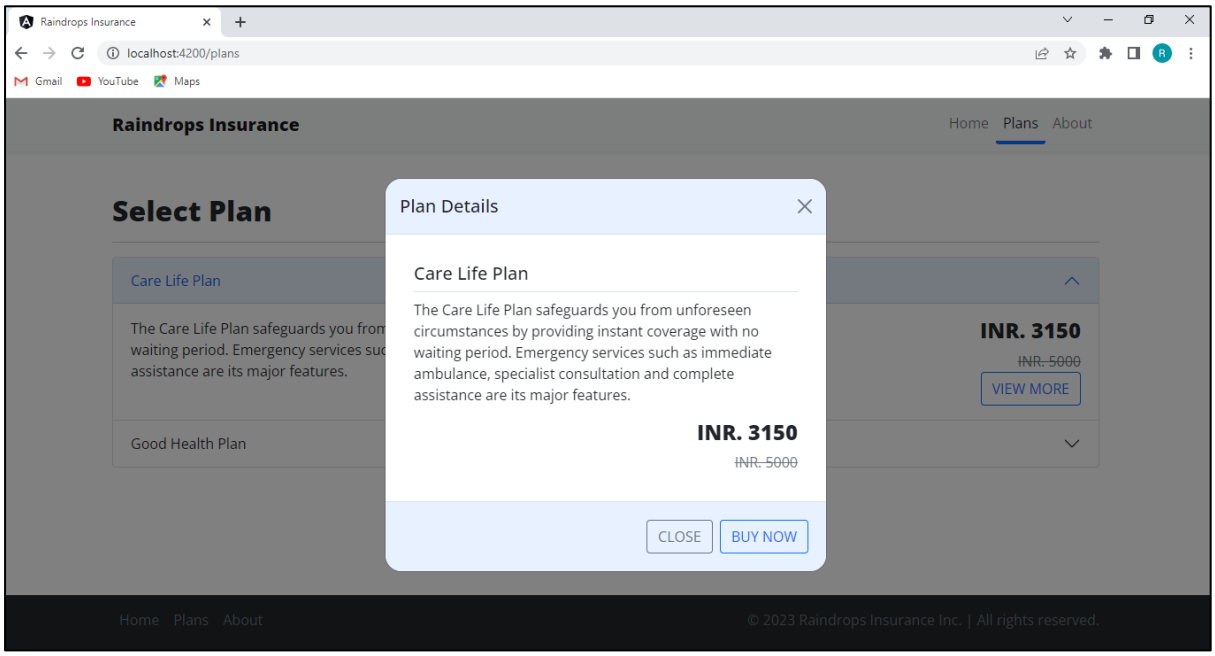
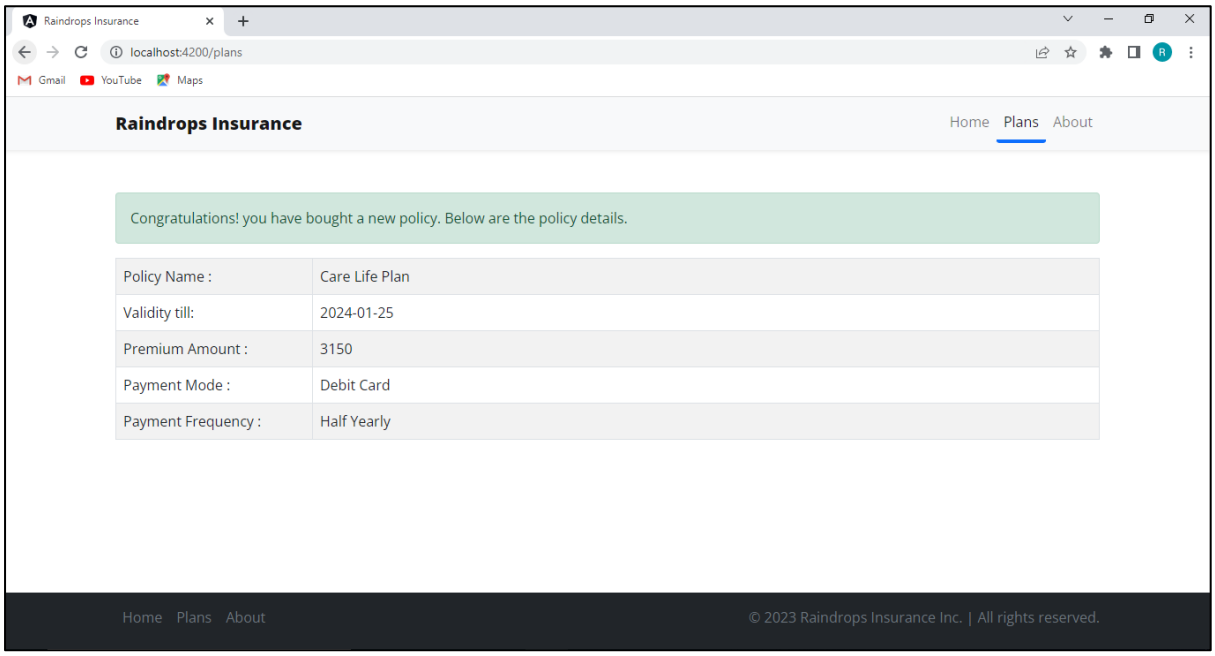
Choices Pop-up:



View Plans:



Select Plan:



Buy Plan: