Air India

Practice Problems

Relevel
by Unacademy

Assignment Questions

Problem Statement

Relevel is expanding globally. So, they have to travel worldwide a lot. Hence they have brought an airline service. Now, we want to create an application similar to Air India using which users can book the flight tickets, cancel them and filter them by price, duration. Also, you can contact the airlines for any queries.

Your task is to go through the Problem Statement, Requirements and create an Air India clone for Relevel.

Follow the standard naming conventions and comment on the code well, so that it is easily understandable. We are looking for a basic working prototype so make your assumptions accordingly and pace yourself.

Pre-requisites

- The candidate should make sure GIT, NodeJS and MongoDB are already installed on their system before starting the development.

User Stories

Problem 1 (Estimated Time to complete ~2 Hours)

Story-1 (Define data model)
Create a clear data model for User, Booking Details, Contact.Create mongoDB schema, for example, should be included in your code base.

- Add Appropriate validations and reference keys.
- Spend some time creating some "seed data" that covers all of the use cases you want to show.

Story-2 (Register the user)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):

- Register new Users.
  - Create an API endpoint “/register”
  - Add appropriate request body to register new users.
  - Validate the request. For example if the request has a valid email ID, the password is strong.
  - Store information at the backend in proper tables.
  - For success, return proper response codes.
  - For failure, return proper response codes.
  - Use an appropriate HTTP method out of GET, POST, PUT,PATCH,DELETE, etc.

Story-3 (Login/Logout Functionality)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):

The task is to create an API endpoint “/auth/login” for the authentication of the user created in the above register API:

- The API request should take the input of userId & password.
- Authenticate the values.
- Display status & response code in the response.
- In case of incorrect credentials API response should throw 401 Unauthorized Exception.
- The successful request should return 200 response codes.

Then create an API endpoint “/logout” to safely logout the user.

Problem 1 (Estimated Time to complete ~1.5 Hours)

Story-4 (Flight Ticket Functionality)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):

- Flight ticket.
  - Create an API endpoint “/search”, "/review", "/traveler", "/bookings", "/boarding-pass/:id", "/cancel/:id", "/contact"
  - search: will search for all the flights based on price, duration.
  - review: we can write a review about the tour.
  - traveler: It will give all the details about the traveler.
  - bookings: It will show all the booking details related to the user.
  - boarding-pass: Will fetch the boarding pass for the flight for a particular user.
  - cancel: Will cancel the booking for the respective id.
  - contact: We can send mail to the airline by using contact API.
  - Validate the request. For example if the request has a valid email ID, the password is strong.
  - Store information at the backend in proper tables.
  - For success, return proper response codes.
  - For failure, return proper response codes.
  - Use an appropriate HTTP method out of GET, POST, PUT,PATCH,DELETE, etc.

Air India

Practice Problems

Relevel
by Unacademy

Assignment Solutions

Solution

https://github.com/Gtgstg/Air-India-Clone

User Stories

Problem 1 (Estimated Time to complete ~2 Hours)

Story-1 (Define data model)
Create a clear data model for User, Booking Details, Contact.Create mongoDB schema, for example, should be included in your code base.

- Add Appropriate validations and reference keys.
- Spend some time creating some "seed data" that covers all of the use cases you want to show.

Story-2 (Register the user)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):

- Register new Users.
  - Create an API endpoint “/register”
  - Add appropriate request body to register new users.
  - Validate the request. For example if the request has a valid email ID, the password is strong.
  - Store information at the backend in proper tables.
  - For success, return proper response codes.
  - For failure, return proper response codes.
  - Use an appropriate HTTP method out of GET, POST, PUT,PATCH,DELETE, etc.

Story-3 (Login/Logout Functionality)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):
The task is to create an API endpoint “/auth/login” for the authentication of the user created in the above register API:

- The API request should take the input of userId & password.
- Authenticate the values.
- Display status & response code in the response.
- In case of incorrect credentials API response should throw 401 Unauthorized Exception.
- The successful request should return 200 response codes.

Then create an API endpoint “/logout” to safely logout the user.

Problem 1 (Estimated Time to complete ~1.5 Hours)

Story-4 (Flight Ticket Functionality)
Your APIs should provide the following functionality to downstream consumers of the API (in a RESTful way):

Flight ticket.

- Create an API endpoint “/search”, "/review", "/traveler", "/bookings", "/boarding-pass/:id", "/cancel/:id", "/contact"
- search: will search for all the flights based on price, duration.
- review: we can write a review about the tour.
- traveler: It will give all the details about the traveler.
- bookings: It will show all the booking details related to the user.
- boarding-pass: Will fetch the boarding pass for the flight for a particular user.
- cancel: Will cancel the booking for the respective id.
- contact: We can send mail to the airline by using contact API.
- Validate the request. For example if the request has a valid email ID, the password is strong.
- Store information at the backend in proper tables.
- For success, return proper response codes.
- For failure, return proper response codes.
- Use an appropriate HTTP method out of GET, POST, PUT,PATCH,DELETE, etc.