A sample CAPSTONE case - MERN Stack

Problem Statement

Build a MERN Application to search for books, open the books, add book to users favorite list using microservices.

Requirements

- A frontend where the user can register/login to the application. The login page should have a link for registration using which the user

should be able to register. On Successful registration the user should be taken to the login page.

- Proper navigation links for all the pages should be available within pages

- Error handling should be implemented across pages. Appropriate messages should be displayed for the same.

- Success messages should be displayed for the User Registration.

- Logout feature should be implemented.

Tech Stack

- Node

- MySQL / MongoDB

- React

User can add an book to favourite list and should be able to view the favourite list.
User can recommend a book to recommendation list and should be able to view global recommendations.

Modules

UI (User interface), React - should be able to

1. List books.

2. View or open a book.

3. Search for a book.

4. Add a book into favorite list.

5. View favorite books.

6. View global recommendations from recommendations list

7. View books by categories (Latest, Drama, Fiction, Comedy, Philosophy, Horror, Thriller, Art, Science, Top-Rated) .

8. User gets a welcome mail on registration.

9. User can comment on books after loging in.

10. UI should be appealing and user friendly.

Backend (Node)

1.User Service

2.Favorite Service

3.Recommendation Service

4.Comment Service

Flow of Modules

Building frontend

1. Register/Login.

2. Show list of books based on categories - populating from external API.

3. Show book details - populating from external API.

4. Build a view to show favorite and recommendation list.

5. Add Pagination for favorite/recommendation list.

6. Search for books by any keyword.

7. Display Book details

for example - Author name, Book Image, Book Title, Description, Language etc.

8. Create a view for displaying books on search.

- Using Services to populate these data in views

- Stitching these views using Routes and Guards

- Unit Tests should be created

Building backend

Building the

UserService

- Creating an api in Node to facilitate user registration and login with MySQL as backend. Upon login, JWT token has to be generated.

- Unit Testing of the entire code which involves the positive and negative scenarios.

Building the favourite Service

- Building an api in Node to facilitate CRUD operation over favorite books stored in MongoDB. JWT Filter should be

applied for all the API calls of the googleapiservice. JWT token should be used to authorize the user access to all the resources.

- Write Unit Test Cases and get it executed.

Building the recommendation Service

- Building an api in Node to facilitate CRUD operation over recommended books stored in MongoDB. JWT Filter should be

applied for all the API calls of the googleapiservice. JWT token should be used to authorize the user access to all the resources.

- Write Unit Test Cases and get it executed.

Building the comment Service

- Building an api in Node to facilitate CRUD operation over comments stored in MongoDB. JWT Filter should be

applied for all the API calls of the googleapiservice. JWT token should be used to authorize the user access to all the resources.

- Write Unit Test Cases and get it executed.

Demonstrate the entire application

1. Make sure all the functionalities are implemented.

2. Make sure both the UI (Component and Services) and server side (For all layers) codes are unit tested.