**Projects**

**E-commerce Website**

**User Microservice:**

**Working Video:** <https://github.com/AdhithyaRS/trackerWebJavaAPI/tree/main/Working%20Video>

**Backend:**

Old Code link: <https://github.com/AdhithyaRS/trackerWebJavaAPI>

I recognized some structural issues in the initial project setup and have since reconstructed it from scratch to ensure proper usage and alignment with best practices.

New Code link: <https://github.com/adhithya-rs/UserMicroService/>

**Frontend:**

Code link: <https://github.com/AdhithyaRS/trackerWebReact>

**Product Microservice(Backend Only):**

Code link: <https://github.com/adhithya-rs/ProductMicroService>

**Summary:**

Currently in development, this e-commerce platform is designed to provide a seamless online shopping experience. I have successfully completed the backend and frontend for the user service, as well as the backend for the product service.

**Key Contributions and Technologies**:

1. **User Microservice**:
   * **Functionality**: Enables functionalities such as user registration, sign-in, and password reset.
   * **Security**: Implements HttpOnly JWT tokens for secure session tracking and incorporates OTP verification via mobile number and email during signup and password resets.
   * **Technologies**: Developed using Java and Spring Boot for the backend, with MongoDB for data management, including sharding to enhance scalability. The frontend is built using React, ensuring a responsive and interactive user interface.
2. **Backend Architecture**:
   * **Development**: Crafted with Java and Spring Boot, focusing on an annotation-based configuration to streamline backend operations.
   * **Authentication and Security**: Utilizes JWT for session management and role-based authentication, integrating HttpOnly cookies to bolster security measures.
   * **Verification Processes**: Implemented robust mechanisms for email and phone number verification during user registration and password recovery, enhancing the security and reliability of user data.
3. **Frontend Design**:
   * **User Interface**: Developed using React, providing a dynamic and responsive design that adapts to different devices and screen sizes.
   * **API Integration**: Managed seamless API interactions to ensure efficient data exchange between the frontend and backend, optimizing the user experience.
4. **Error Handling**:
   * **Robustness**: Incorporated comprehensive exception handling strategies using try-catch blocks to maintain system stability and user experience during unexpected scenarios.

**Current Status**: The project is actively progressing, with substantial parts of the user interaction interfaces and backend services fully functional and in testing phases.

**Amazon Website Testing**

Code link: <https://github.com/AdhithyaRS/AmazonTesting>

**Summary:**

This project involved comprehensive testing of the Amazon e-commerce platform, focusing on automating functional and integration tests to validate user interactions and transactional processes.

**Key Contributions and Technologies**:

1. **Testing Scope**:
   * **Functional Testing**: Conducted detailed functional tests covering critical features such as sign-in, sign-up, and product search functionalities.
   * **Integration Testing**: Performed end-to-end integration testing from login through to product purchase, ensuring all components interact seamlessly.
2. **Test Automation**:
   * **Selenium with Java**: Utilized Selenium WebDriver and Java for test automation, adhering to the Page Object Model (POM) design pattern to enhance maintainability and readability of the test code. Each key functionality, like sign-in, sign-up, and product search, was encapsulated within dedicated classes.
   * **TestNG Framework**: Leveraged TestNG to structure tests and manage test suites, facilitating efficient testing cycles and results aggregation.
3. **Test Reporting and Documentation**:
   * **Allure Test Reporting**: Integrated Allure Test Listener to generate comprehensive test reports, capturing detailed execution results and metrics.
   * **Enhanced Test Outputs**: Developed functionality to capture screenshots and detailed test outputs upon both pass and failure events, which are automatically attached to the Allure reports for thorough documentation.
   * **Input Visibility**: Implemented enhancements to display specific input data used during tests within the Allure reports, providing clear insights into test conditions and outcomes.
4. **Data-Driven Testing**:
   * **CSV Integration**: Created and utilized a CSV file for inputting test cases, which streamlines the association of test cases with their respective methods. This setup supports scalability, allowing for easy integration of additional test cases as the application evolves.

**Project Impact**: The automation framework developed during this project significantly reduced manual testing efforts and improved the reliability and speed of the testing process. The detailed reports generated have been instrumental in identifying issues early in the development cycle, ensuring a higher quality product.

**Console Base Movie Booking**

**Working Video:** <https://github.com/AdhithyaRS/MovieBokkingJavaConsole/tree/master/Working%20Video>

Code link: <https://github.com/AdhithyaRS/MovieBokkingJavaConsole>

**Summary:**

The Movie Booking Application is a console-based system designed to streamline the ticket booking process, eliminating the need for manual intervention at ticket counters. It enhances user experience by providing a user-friendly interface with distinct roles for administrators and regular users.

User Authentication:

• The application supports two types of logins: User and Admin.

• User details, including first name, last name, login ID, password, contact number, and user type, are stored in a User class with an associated Enum for user types.

• User objects are created only once at the start and termination of the application, with new objects created only during a switch between user types.

Functionality for Users:

Users have various options to interact with the application, including:

1. View all movies

2. View released movies

3. View upcoming movies

4. Search movies by name

5. Book a movie

6. View past bookings

7. View upcoming bookings

8. Reset password

9. Sign out

Functionality for Admins:

Admins have additional functionalities for managing the system:

1. Add/Update a movie

2. Remove a movie

3. Check ticket status

4. View upcoming tickets

5. View expired tickets

6. Reset application data

7. Backup data

8. View application logs

9. Reset password

10. Sign out

11. Terminate the application (exclusive to admin)

Security Measures:

• Only admins have the authority to terminate the application for security purposes.

• Admins can view logs, allowing them to analyze console records on specific dates, aiding in troubleshooting and fixing issues after an application crash.

Technical Implementation:

• Real-time seat availability and layout are displayed through a 2D array in the console.

• JSON files are used for data storage, managed by a separate data service class.

• Data processing and retrieval are efficiently handled using HashMaps and ArrayLists.

**Auto-Complete Search Box**

Link:

<https://github.com/AdhithyaRS/AutocompleteTrie>

**Console Base Movie Booking**

Link:

<https://github.com/AdhithyaRS/MovieBokkingJavaConsole>

**In-Memory Parking Lot**

Link:

<https://github.com/adhithya-rs/parkingLot>

**TicTacToe**

Link:

<https://github.com/adhithya-rs/TicTacToe>