# Day 5 - Testing and Backend Refinement - Morent

## **Steps Taken to Build and Integrate Components**

Project Initialization:

Set up the project repository using Git for version control.

Initialized a Node.js application with Express.js as the backend framework.

2. Database Integration:

Integrated MongoDB as the primary database for storing user data and application state.

Utilized Mongoose for object data modeling (ODM) to streamline database interactions.

3. User Authentication:

Implemented Clerk for user authentication and management, allowing for secure sign-up and login processes.

Configured Clerk's API to handle user sessions and access control.

Payment Processing:

Integrated Stripe for payment processing, enabling users to make transactions securely.

Set up Stripe's API to handle payment intents and customer management.

5. Webhook Configuration:

Configured Stripe webhooks to listen for events such as successful payments and subscription updates.

Developed a dedicated endpoint to process incoming webhook events and update the application state accordingly.

Frontend Development:

Developed a React.js frontend to provide a responsive user interface.

Utilized Axios for making API calls to the backend services.

Testing and Deployment:

Conducted unit and integration testing using Jest and Supertest to ensure the reliability of the application.

Deployed the application on a cloud platform (e.g., Heroku or AWS) for production use.

# **Challenges Faced and Solutions Implemented**

1. Challenge: Database Connection Issues:

During the initial setup, we faced intermittent connection issues with MongoDB.

Solution: Implemented retry logic in the database connection code and ensured proper error handling to log connection failures for troubleshooting.

2. Challenge: Stripe Payment Failures:

Users reported failed payment attempts due to incorrect API keys and configuration.

Solution: Conducted a thorough review of the Stripe integration, ensuring that all API keys were correctly set in the environment variables and that the webhook endpoint was properly configured to handle events.

3. Challenge: User Authentication Errors:

Encountered issues with user sessions expiring unexpectedly, leading to a poor user experience.

Solution: Adjusted the session timeout settings in Clerk and implemented a refresh token mechanism to maintain user sessions seamlessly.

Challenge: Webhook Event Handling:

Faced difficulties in processing webhook events due to the asynchronous nature of the requests.

Solution: Utilized a message queue (e.g., RabbitMQ) to handle webhook events asynchronously, ensuring that the application could process events without blocking the main thread.

### **Best Practices Followed During Development**

#### 1. Version Control:

Maintained a clear branching strategy in Git, using feature branches for new developments and merging into the main branch only after thorough code reviews.

#### Code Reviews:

Implemented a peer review process for all code changes to ensure code quality and adherence to coding standards.

#### Documentation:

Documented all API endpoints and integration processes in a centralized repository to facilitate onboarding and knowledge sharing among team members.

#### 4. Error Handling:

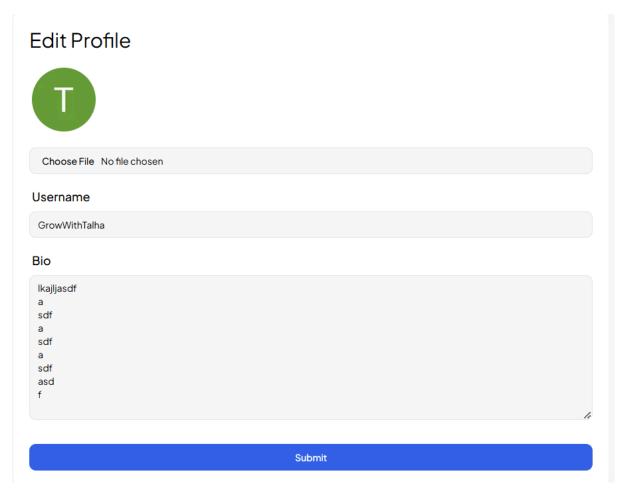
Adopted a consistent error handling strategy across the application to provide meaningful error messages and improve debugging.

Continuous Integration/Continuous Deployment (CI/CD):

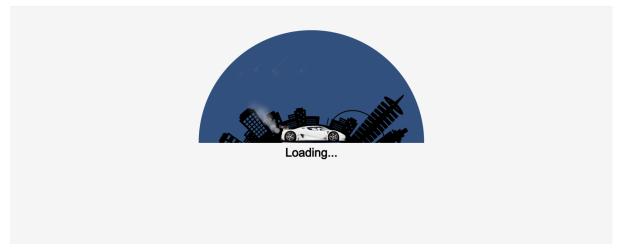
Set up CI/CD pipelines to automate testing and deployment processes, ensuring that code changes are automatically tested and deployed to production.

By following these steps, addressing challenges proactively, and adhering to best practices, we successfully built and integrated the components of our application, resulting in a robust and user-friendly product.

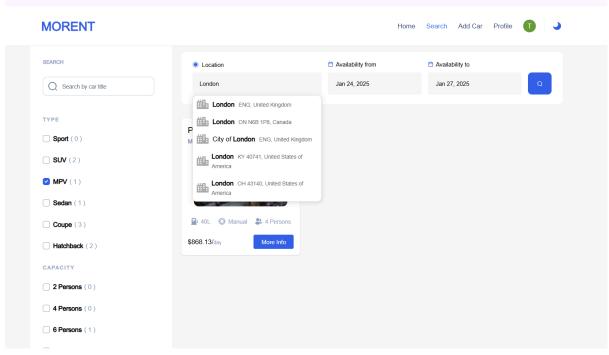
# **Screenshots**



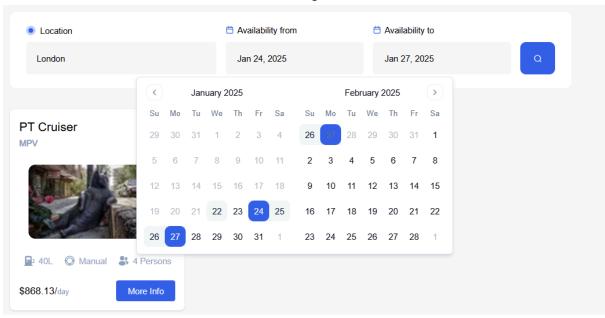
Profile Form

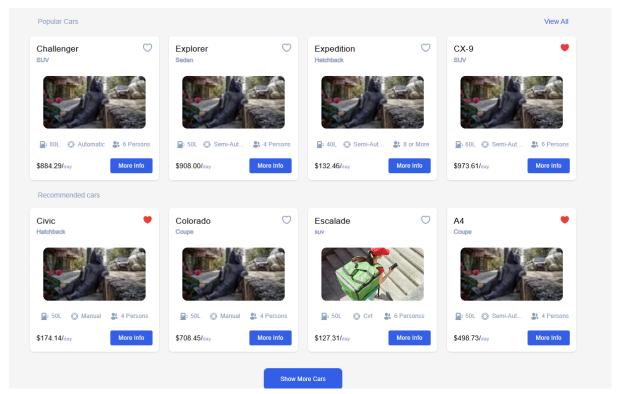


Loading Screen

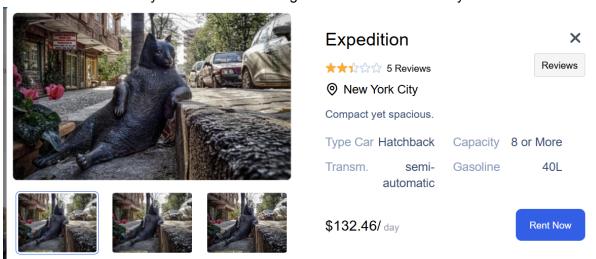


Search Filters with goe location API

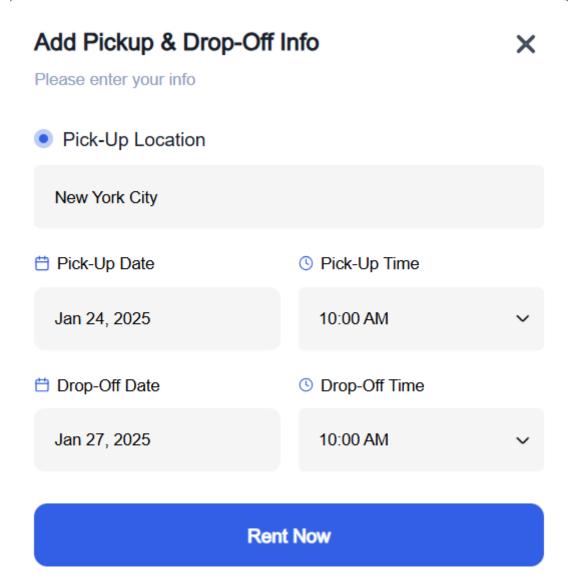




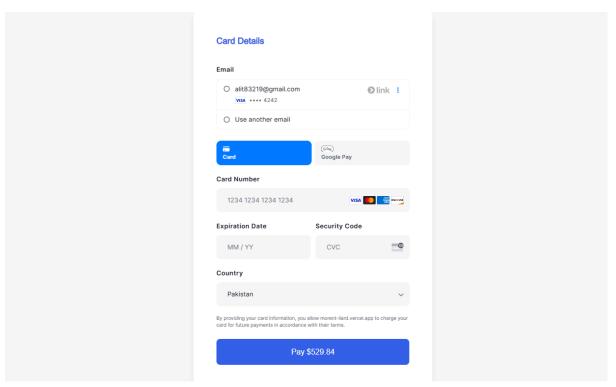
Dynamic Product Listing with Wishlist functionality



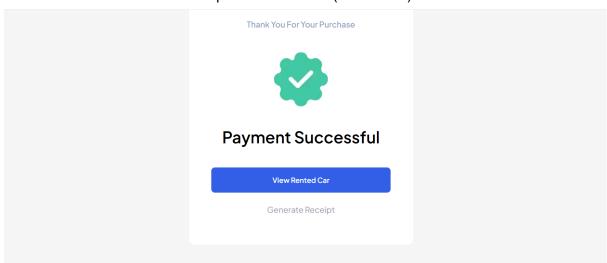
Product description dialog



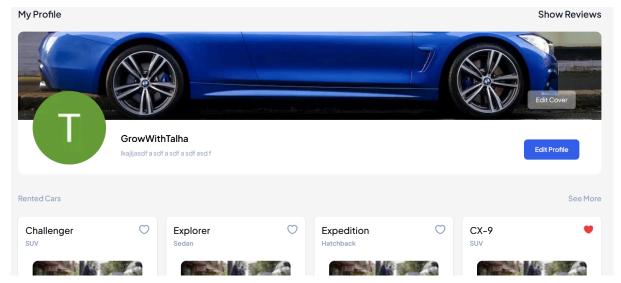
Rent duration selection dialog for checkout



Stripe checkout form (embedded)



Payments success screen



Profile Page

Thats it.