Weekly Report 3
CIS 453 M001
Group 10 - Hussein, Zichen, Chang, Ryan

Team Name / Members: Rental10 Squad

1. Objectives for this Week:

- **Establish a Robust Development Baseline:** Our primary goal was to move from conceptual diagrams to a functional Full-Stack environment using Node.js for the backend and React for the frontend.
- **Initialize the System Architecture:** We aimed to build the "Client-Server" directory structure to allow the team to work on frontend and backend modules in parallel.
- **Containerize the Full Stack:** We aimed to containerize the API, MySQL database, and client using Docker Compose to ensure repeatable setup across teammates.
- **Develop Core Functional Prototypes:** Specifically, we targeted the initial implementation of User Registration (FR1) and the Vehicle Catalog (FR2) to validate our data flow.
- **Repository and Environment Sync:** We planned to set up the GitHub repository and configure npm manifests to manage our engineering dependencies like Axios and Express.

2. Work Completed:

Repository and Version Control Setup:

- **GitHub Initialization:** We successfully initialized the remote repository on GitHub and established the branch protection strategy for the Rental10 Squad.
- **Environment Standardization:** We configured the .gitignore files for both the client and server directories to ensure that node_modules and sensitive .env files are not tracked, preventing repository bloat and security leaks.
- **Project Decoupling:** We structured the repository into two distinct root folders (/client and /server), allowing the frontend and backend to scale independently and facilitating smoother collaboration.

Backend Infrastructure (Port 5000):

- We initialized an Express.js server and integrated dotenv to manage environment variables, ensuring our database credentials remain secure.

- We configured CORS middleware to allow our React frontend (Port 3000) to communicate with the Node.js backend without security blocks.

Frontend Prototyping (Port 3000):

- **Modular Routing:** Using react-router-dom, we implemented a Single Page Application (SPA) architecture for seamless navigation between the Catalog, Login, and Sign Up pages.
- **Catalog Implementation:** We developed the Catalog.js component using React Hooks (useState, useEffect) to handle asynchronous API calls for the vehicle fleet.
- **Service Layer:** We created a centralized api.js using Axios, providing a reusable interface for all future network requests.

Containerized Full-Stack Deployment (Docker Compose):

- We containerized the backend API and MySQL database using Docker Compose, enabling one-command startup for the entire stack (API + DB + Client).

- We validated end-to-end connectivity by successfully registering a user through the API and retrieving vehicle data via /api/cars.

3. Challenges Encountered:

- **PowerShell Execution Restrictions:** A major roadblock was the PSSecurityException which prevented npm scripts from running on Windows. Resolution: We troubleshot this by migrating the terminal environment to Command Prompt (CMD) and manually adjusting the system's ExecutionPolicy.
- **Directory and Manifest Integrity:** We faced several ENOENT errors because the environment was missing standard React files. Resolution: I manually reconstructed the package.json files and the public folder structure to satisfy the Webpack entry-point requirements.
- **Module Resolution Errors:** During the integration of pages, we hit "Module not found" errors. Resolution: We resolved this by standardizing our directory naming conventions (e.g., lowercase pages folder) and fixing relative import paths in App.js.

4. Next Steps:

- **Security & Authentication Logic:** Our next sprint involves implementing password hashing with bcryptjs and session persistence with JWT for the registration flow.
- **UI/UX Skinning:** We will begin integrating the vehicle image assets and applying CSS layouts to the car cards we built today.
- **Finalize MySQL schema & seed data:** We will complete Vehicles/User/Booking tables and add initial fleet records for consistent demo behavior.

5. Team Contribution:

- **Hussein:** Was responsible for the full environment configuration, solving the npm security blocks, and building the core App.js, Catalog.js, and api.js logic.
- **Ryan:** Scheduled to take the lead on frontend styling and image asset management in the next phase.
- **Chang:** Currently preparing the backend authentication logic and security protocols for our upcoming User Management sprint.
- **Zichen Shen:** Using Docker Compose to containerize the MySQL database and backend API, making it easier for the team to run the same development environment and prepare for future schema and data work.