Sprint Backlog

Goals:

- [x] Enhance the PostgreSQL database to support new features: Settings, Inventory, Jobs, Purchases, and Dashboard, including setting up database tables and filling them with data.
- [x] Expand the React front-end application to include new pages (Settings, Inventory, Jobs, Purchases, and Dashboard), ensuring they communicate effectively with the backend through API calls.
- [x] Extend the REST API using Node.js and Express.js to handle requests from the new front-end components, addressing security concerns and getting API endpoints up.
- [x] Presentation: Create a presentation for the Sprint Deliverable that encapsulates all the work done, including database enhancements, front-end expansions, and API extensions.

Frontend Development:

- [x] Create baseline page: Ensure the baseline page aligns with the expanded goals, possibly revising it to incorporate new elements or functionalities.
- [x] Create a page for each existing tab: Develop pages for Settings, Inventory, Jobs, Purchases, and Dashboard.
- [x] Add tabs (if needed): Assess if additional tabs are required for comprehensive navigation or functionality, especially considering the introduction of new features.
- [x] Populate created tabs: Ensure all newly created tabs are populated with dynamic data from the backend, reflecting real-time changes and interactions.
- [ ] Implement Admin Role: Expand the admin role capabilities to include user registration, inventory management, job listings management, purchase order approvals, and dashboard analytics.

Backend Development:

- [x] API for inventory tab: Enhance or create new endpoints to manage inventory items, including CRUD operations.
- [x] API for orders tab: Develop endpoints to handle order processing, tracking, and history.
- [x] API for customers tab: Create APIs to manage customer data, interactions, and history.
- [x] API for dashboard: Ensure the dashboard has the necessary endpoints to aggregate and display data from various features effectively.
- [x] Sanitization: Implement data validation and sanitization across all endpoints to address security concerns.
- [ ] Docker and AWS Integration:
- [ ] Deploy on AWS: Consider using Amazon S3 and CloudFront for static site hosting. This involves deploying the updated React application and ensuring it's accessible.
- [ ] Use AWS Postgres database in Docker?: Evaluate the feasibility of integrating the AWS PostgreSQL database with Docker for development and testing environments.