**EcoPanel Marketplace Software Requirements Document**



1. Introduction

This document outlines the software requirements for the EcoPanel Marketplace project, a React-based web application designed to facilitate buying, selling, and management of solar panels and renewable energy solutions.

2. Functional Requirements

2.1 User Authentication and Authorization

- Implement a signup screen with dropdown options for admin, supplier, customer, and internal/external team roles [DONE]

- Store user type ('type') in the database based on signup selection [DONE]

- During login, authenticate user and redirect based on stored type [yet DONE for ‘suppliers’ only. **‘admin’, ‘customer’, & ‘teams’ to be done after separate webpages of them are created i.e, CustomerPage.js(/customerPage), AdminPage.js(/adminPage) & teamPage.js(/teamPage)]**

- Implement role-based access control for different user types, allowing functionalities of admin only for admin, functionalities of suppliers only for suppliers, etc for teams & customers. The functionalities that are common will be visible to both. **[I have created separate webpages as SupplierPage, AdminPage. One option is to create separate of customerPage and teamPage too, while another option is to create a single page with well-defined roll-based functionalities. This is more optimized but also complex to implement because of the security risks as overlooking a single issue and allow customer to use functionality of admin if mechanisms not properly implemented].**

- Payment verification system and authentication

2.2 Database Management

- Design MySQL database schema to store user credentials and details [DONE] {id, name, email, password, phone\_number, type, confirmation\_token, confirmed}

- Create tables for storing products with their specification columns {product\_name, product\_id, price, status(approved/disapproved/pending until admin decides of supplier uploaded), **[other detailed specs of solar panel based upon what the supervisor tells]** }

- Implement tables for order history, payment records, and net-metering details for integration of payment methods

- Deploy the DB on cloud or online service like Hostinger or GitHub Codespace

2.3 Frontend Components

- Develop home screen, contact page, and about page APIs and beautified webpage upon traversal visible for all with/without login

- Display products listed on the website accessible for all with prices. The products when clicked open a dialog with detailed specifications and price with a buy button. Clicking the buy button checks if login with customer account or not, if not, traverses to login screen else proceed to payment page. Customer will have the option to see his order history, available products with images, specs & prices, and promotional authenticity content like product reviews.

- Create supplier dashboard for uploading panel descriptions(specs) and images. The status column in the product table(not created) in DB would set to ‘pending’ when uploaded. The admin(s) would get the ‘product available’ notification in his notifications with option of ‘approve’ or ‘disapprove’. If disapproved, generate notification for the supplier in his products section where all his uploaded products would be shown as disapproved & stay in DB though. If approved, the product would be shown as approved and would be available to be displayed among products on the website.

- Develop customer dashboard for viewing orders, payments, and net-metering details

- Implement internal/external team checklist feature with image upload capability of using Cloudinary. The checklist would be passed by the admin to the internal/external team. When passed, it will be visible to all accounts that are signed-up as internal/external ‘teams’. The checklist when ticked by any member of the internal/external team would open a camera/gallery for him to snap/upload the image before ticking and thus it displayed to the admin with the internal/external team uploaders’ ‘name’ and uploaded pic and ticked task. Admin could add more tasks to the checklist of one project and pass it to the internal/external team. ‘internal\_teams’ could see all the projects passed to them along with the checklist.

- Admin could see payment details of all the customers and total orders. Supplier details and the products they have uploaded supplier-wise are visible.

2.4 Backend Functionality

- Set up Express.js backend with RESTful APIs

- Implement API endpoints for user authentication, product management, and order processing

- Integrate Cloudinary for image uploads and storage as when the supplier uploads the image of product with specs, the product through cloudinary would be stored in DB but not be displayed. When approved by admin, only then displayed.

- Develop email verification functionality using Nodemailer

***{further details explained above in ‘Frontend Components’ section}***

3. Non-Functional Requirements

3.1 Performance

- Optimize database queries for efficient data retrieval

- Implement caching strategies where appropriate

- Design scalable backend architecture to handle increased load

3.2 Security

- Implement bcrypt.js for password hashing [DONE]

- Use HTTPS for secure communication

- Implement CORS middleware for cross-origin resource sharing

- Encrypt sensitive data both at rest and in transit

- Disallow accessing of endpoints dedicated to only a specific ‘type’ of user (/adminPage can’t be accessed by anyone unless not logined as admin, etc for /suppliersPage, /teamPage, /customersPage, & others if made)

- Signup for Admin & Internal/external teams would MUST require approval by the admin through admin email.

3.3 Usability

- Utilize React hooks (useState) for state management

- Implement CSS-in-JS solutions (styled-components or Emotion) for styling

- Optimize image loading using Cloudinary

3.4 Scalability

- Implement Docker containers for containerization or Kubernetes for orchestration in production

- Utilize GitHub Codespaces for collaborative development

4. Technologies and Frameworks

4.1 Frontend

- React.js as the primary frontend framework

- Material-UI for UI components

- React Router v6 for routing

- styled-components or Emotion for CSS-in-JS styling

4.2 Backend

- Node.js with Express.js framework

- MySQL database for storing application data (users information table [DONE], products specifications table, payment data and history table, )

- Cloudinary for image storage and processing

4.3 Development Tools

- Create React App for project setup

- Git for version control

- NPM as package manager

- Webpack as module bundler

4.4 Testing Frameworks

- Jest for JavaScript testing

5. Deployment Strategy

- Local development environment using Create React App

- Containerization using Docker

- Orchestration using Kubernetes (for production)

- Continuous Integration/Continuous Deployment (CI/CD) pipeline

6. Maintenance and Updates

- Implement automated testing suite for continuous integration

- Set up monitoring and alerting systems for critical components

- Plan for regular updates and maintenance of dependencies

7. Assumptions and Dependencies

- Availability of solar panel marketplaces and renewable energy sources

- Compliance with relevant industry regulations

- Integration with payment gateways for secure transactions

8. Constraints

- Budget constraints for development and hosting

- Timeline constraints for initial launch and ongoing development

- Technical limitations of chosen technologies

*9. Glossary*

- **Bcrypt.js**: Password hashing algorithm

- **Cloudinary**: Image storage and processing service

- **CORS**: Cross-Origin Resource Sharing

- **Docker**: Containerization platform

- **Emotion**: CSS-in-JS solution

- **Express.js**: Node.js web application framework

- **GitHub Codespaces**: Cloud-based development environments

- **Jest**: JavaScript testing framework

- **Kubernetes**: Container orchestration platform

- **Material-UI**: React component library

- **MySQL**: Relational database management system

- **Nodemailer**: Email sending library

- **npm**: Package manager for JavaScript

- **React.js**: JavaScript library for building user interfaces

- **React Router v6**: React navigation library

- **styled-components**: CSS-in-JS solution

- **Webpack**: Module bundler for JavaScript applications

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Compiled by Internee* ***Humza Asrar*** *under Supervision of* ***Junaid Nasir*** *at* ***Infinite Algos)***