Website Design Document

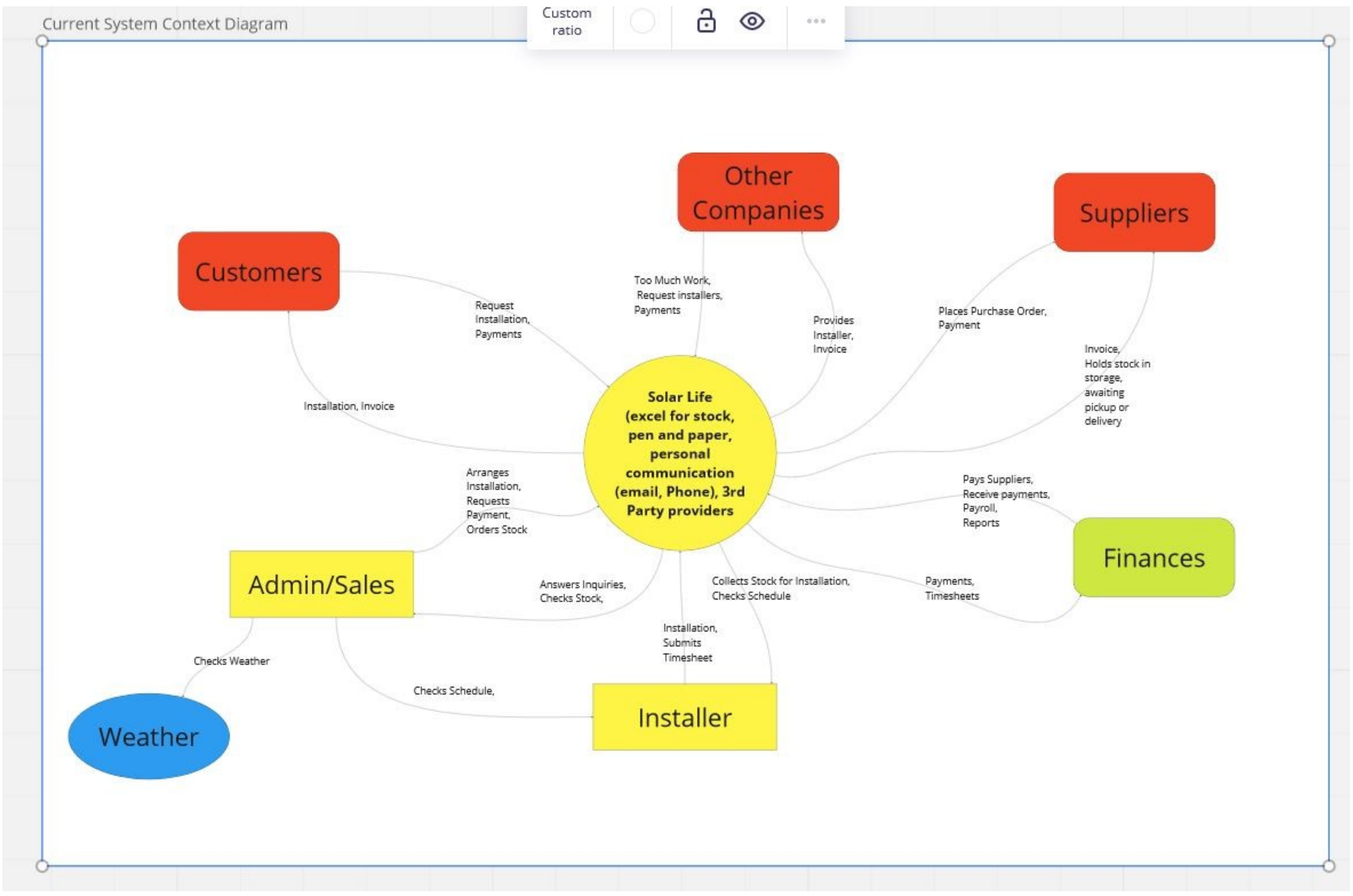
Problem Overview

Solar Life is a solar panel installation company, relatively new with a small structure and limited use of technology in their business practices and procedures. All employees at the moment are part-owners. They would like to implement new systems in place to make the business more efficient in their growth.

Their first aim is to enhance its stock inventory management system to address the following challenges with their current process:

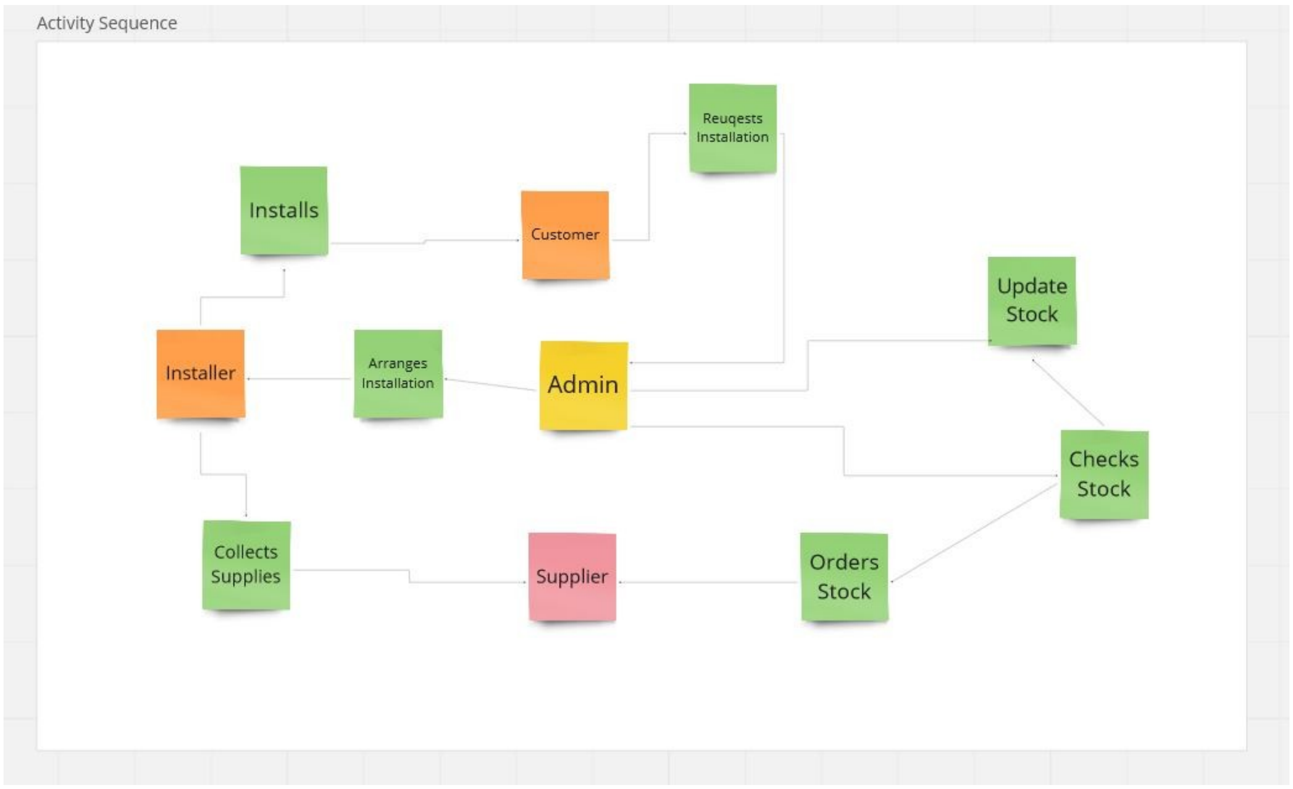
- Inefficiency and errors in using an Excel spreadsheet.
- Limited visibility and lack of real-time updates.
- Inadequate product tracking and information.
- Manual reporting and analysis.
- Limitations on scalability and growth.

Current System Context Diagram



Current System Context Diagram
Custom ratio
Other Companies
Suppliers
Customers
Too Much Work, Request installers, Payments
Request Installation, Payments
Provides Installer, Invoice
Places Purchase Order, Payment
Invoice, Holds stock in storage, awaiting pickup or delivery
Installation, Invoice
Solar Life (excel for stock, pen and paper, personal communication (email, Phone), 3rd Party providers
Arranges Installation, Requests Payment, Orders Stock
Pays Suppliers, Receive payments, Payroll, Reports
Finances
Admin/Sales
Answers Inquiries, Checks Stock,
Collects Stock for Installation, Checks Schedule
Payments, Timesheets
Installation, Submits Timesheet
Checks Weather
Checks Schedule,
Installer
Weather

Business Process for Stock Management



Stakeholders

- **Admin –** currently manages the stock inventory and processes
- **Customer-** want to purchase an installation
- **Supplier-** supplies the goods, manages storage currently until needed
- **Installer-** requires the goods for installation.

Business Requirements:

• **User Authentication**	Only authorized personnel access inventory
• **Inventory Tracking**	Track and record stock levels in real-time
• **Low Stock Alerts**	When stock level falls below threshold, email alert or notification (future potential automatically place order with supplier)
• **Mobile Access**	Mobile view first for access anytime, anywhere
• **Room for Growth**	User Profiles for different access levels for future employee structure.

Project Scope:

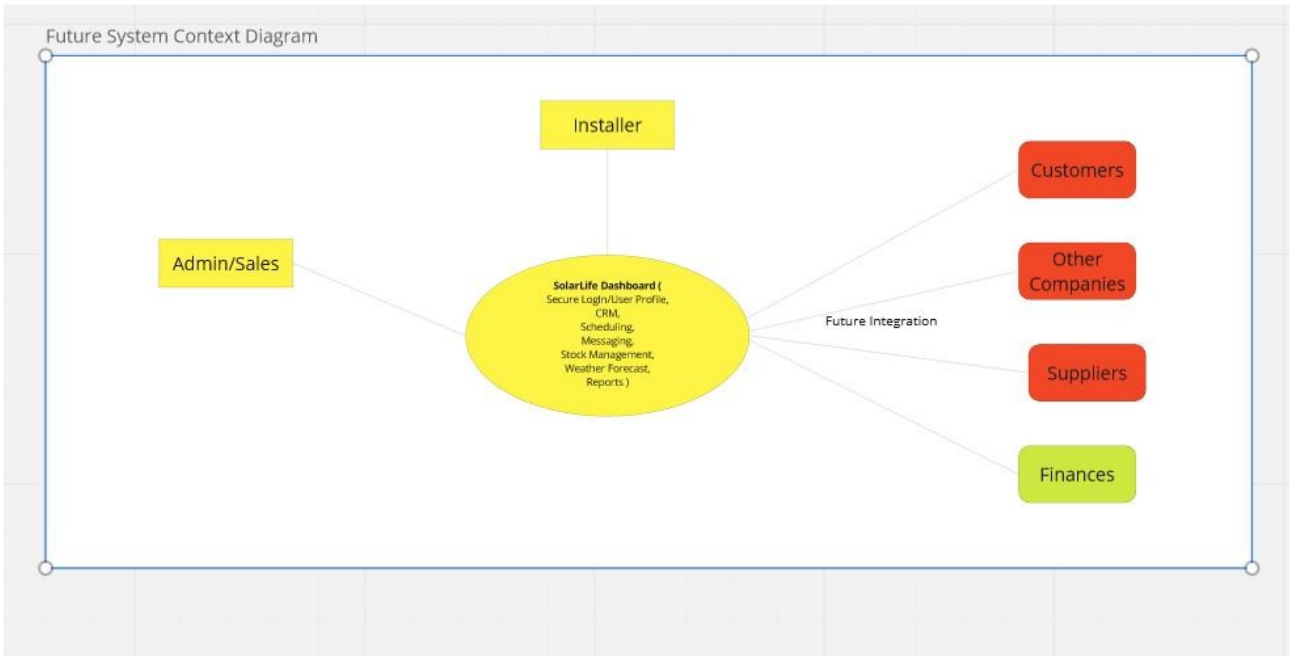
• **Dashboard**	To manage current and future company procedures
• **Authentication**	Create User Roles (admin, staff) permissions for relevant features. (Future Proofing)
• **Inventory Tracking**	Database to store product information and stock levels. Add, Update and view product details.
• **Notifications**	Alert admin when stock level below threshold.
• **Mobile Access**	Responsive design for screen sizes and devices. Optimized for mobile, ensuring usability and functionality.

Future State Out of Scope:

- **Weather API integration -** 7 day/14 day forecasts
- **CRM** - (Manage Customer information, Client Companies for contracting)
- **Mobile App-** (Job and Staff Scheduling, time-tracking)
- **Integrate or rebuild website** to share product information.
- **Messaging –** Users will be able to communicate via the app, send notifications

Solution Statement

Our proposal is a Web App that can be a one stop shop for future company requirements, utilising secure login and user profiles to manage relevant accessibility requirements, scalable for growth and changing needs, potential integration with 3rd party services/providers.



Future System Context Diagram
Installer
Admin/Sales
SolarLife Dashboard (Secure LogIn/User Profile, CRM, Scheduling, Messaging, Stock Management, Weather Forecast, Reports)
Future Integration
Customers
Other Companies
Suppliers
Finances

The first release or MVP (minimum viable product) will focus on an inventory stock management system.

User Stories

Admin- Will have the ability to Create New Products, View Inventory, Update Stock, and get an alert when stocks are low to place an order and view reports such as total stock, total sold.

Installers/Owners - Will be able to view the inventory and reports such as total stock, total sold.

Architecture

MERN Stack

MERN stands for MongoDB, Express, React, and Node, after the four key technologies that make up the stack.

- MongoDB – Document Database
- Express.JS – Node.JS Web Framework
- React.JS – Client-side JavaScript Framework
- Node.JS – Premier JavaScript Web Server

What is MongoDB?

This is the component that is in charge of database management. It's a schema-less, non-relational document-oriented database. The JSON format is used to alter data in this scenario. However, if the developer is familiar with the necessary libraries, converting it to JavaScript data (or vice versa) is rather simple. Because this approach permits database structures to alter over time, data management is made even easier and more versatile. It becomes simple to scale them.

What is Express.JS?

Express is a Node.JS web application framework. Developers may use the Express framework to create sophisticated web apps and APIs. It adds another degree of efficiency to the coding process by allowing developers to create server-side code using Express rather than Node.JS in its entirety. This eliminates the need for the same code to be repeated as it is in the Node.JS HTTP module.

Express allows you to create a REST API that allows you to retrieve data using HTTP queries, which are then executed by React.JS.

What is React.JS?

React.JS is the component of the MERN stack. It is responsible for a website's front-end. It's a popular library for web developers that want to achieve quick page loading times and smooth animations and transitions.

Because views produced in HTML using React are declarative, developers aren't required to handle any changes in the view's state or data. Additionally, React enables the execution of the same code on both the server and the browser.

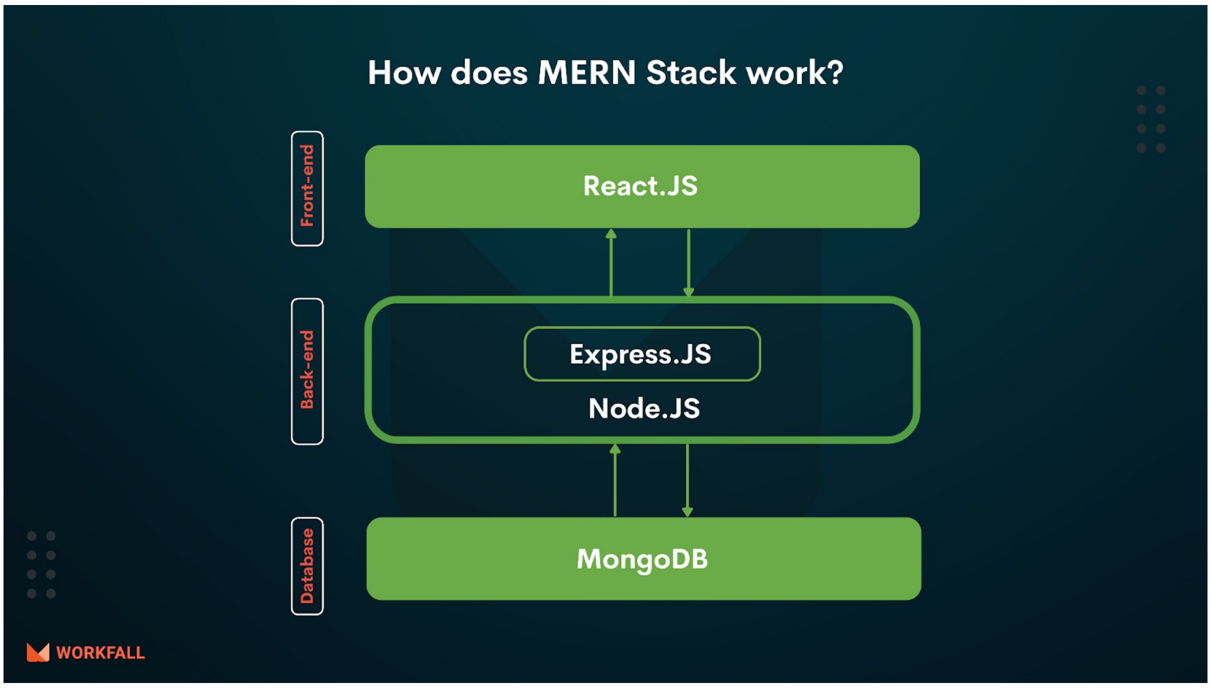
What is Node.JS?

Node.JS is a lightning-fast and efficient programming language that works well with JavaScript to create scalable network applications. To put JavaScript files together, it uses its own module system (each file is treated as a separate module).

In essence, Node.JS unifies web application development by allowing developers to use a single programming language for both the server and client sides, rather than having to utilize separate languages for each.

How does MERN stack work?

The MERN architecture allows you to easily construct a 3-tier architecture (frontend, back-end, database) entirely using JavaScript and JSON.



React.JS Front End

The top tier of the MERN stack is React.JS, the declarative JavaScript framework for creating dynamic client-side applications in HTML. React lets you build complex interfaces through simple Components, connect them to data on your back-end server, and render them as HTML.

Express.JS and Node.JS Server Tier

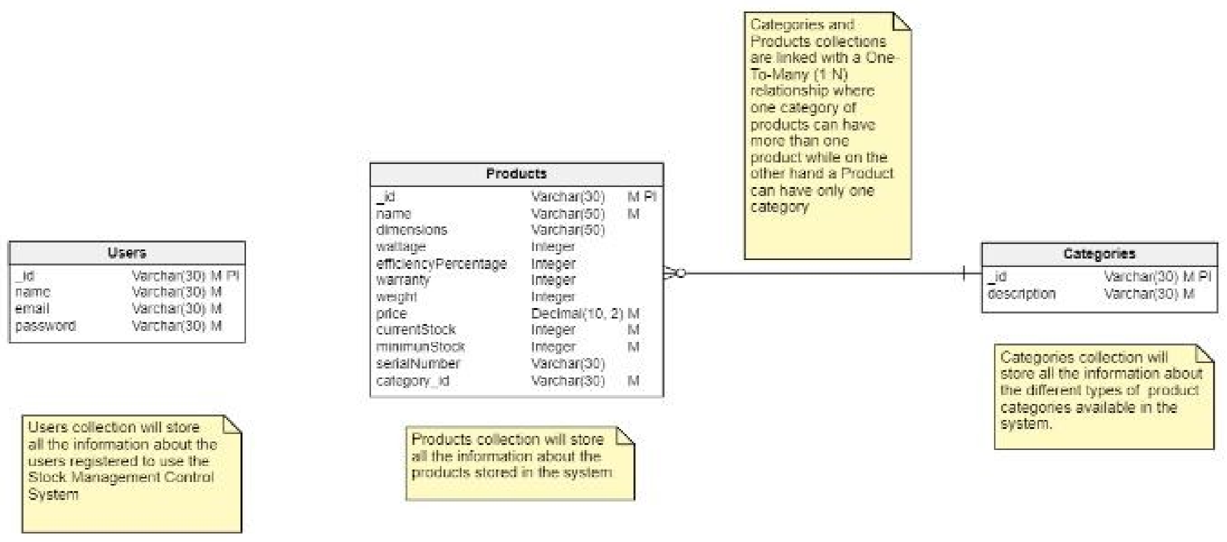
The next level down is the Express.JS server-side framework, running inside a Node.JS server.Express.JS has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

MongoDB Database Tier

If your application stores any data then you're going to use a database that's easy to work with as React, Express, and Node.

That's where MongoDB comes in: JSON documents created in your React.JS front end can be sent to the Express.JS server, where they can be processed and stored directly in MongoDB for later retrieval.

Database Design



Users		
_id	Varchar(30)	M PI
name	Varchar(30)	M
email	Varchar(30)	M
password	Varchar(30)	M

Products		
_id	Varchar(30)	M PI
name	Varchar(50)	M
dimensions	Varchar(50)	
wattage	Integer	
efficiencyPercentage	Integer	
warranty	Integer	
weight	Integer	
price	Decimal(10, 2)	M
currentStock	Integer	M
minimunStock	Integer	M
serialNumber	Varchar(30)	
category_id	Varchar(30)	M

Categories		
_id	Varchar(30)	M PI
description	Varchar(30)	M

Categories and Products collections are linked with a One-To-Many (1 N) relationship where one category of products can have more than one product while on the other hand a Product can have only one category

Users collection will store all the information about the users registered to use the Stock Management Control System

Products collection will store all the information about the products stored in the system

Categories collection will store all the information about the different types of product categories available in the system.

List of Main Features

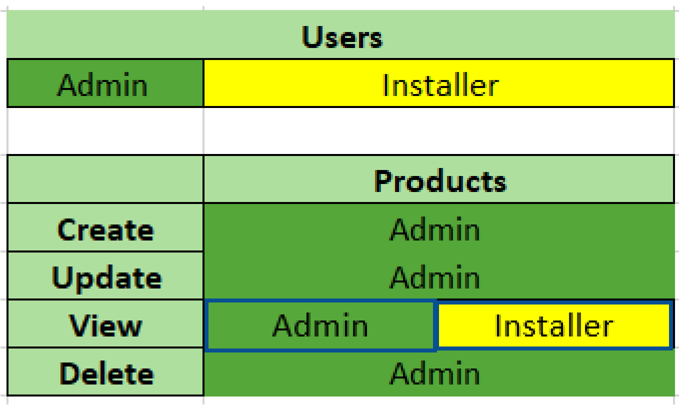
Stock Inventory Management:	Web app platform will provide robust features to effectively manage the company's stock inventory, enabling
efficient	tracking and organization of products.
User Role Management-	The solution will allow the company to define different access levels and permissions based on roles and responsibilities
Dashboard-	Designed with scalability in mind allowing the company to accommodate future
growth	and functionality
User Experience-	Fast, intuitive and user-friendly interface
Cross-Device Accessibility-	Responsive and accessible on both desktop, mobile and tablet, providing flexibility and convenience for users.
Real-Time updates-	Real-time updates applied to stock inventory, ensuring accurate and up-to-date information.
Security and Data Protection-	Only authorized personnel access to system + database.

Security Features

Secure Authentication-	Using JSON Web Tokens(JWT). Strong password policies and enforce user authentication.
Role-Based Access Control-	RBAC to control access to functionalities and resources. Assign roles and permissions to users based on responsibility.
Secure API Endpoints-	Use middleware and authentication mechanisms to validate requests and verify users identity.

To ensure that this web application is secure and that only the correct user profiles are able to have access to certain aspects, we will create three user profiles and each profile will have the specified access as per the client requirements. All users whatever their profile type will be required to login to gain access, this is to ensure that the data being provided is secure and protected so no unauthorised personal have access to it.

- **Secure Authentication**: JSON Web Tokens(JWT). Strong password policies and enforce user authentication.
- **Role-Based Access Control**: RBAC to control access to functionalities and resources. Assign roles and permissions to users based on responsibility.
- **Secure API Endpoints**: Middleware and authentication mechanisms to validate requests and verify users identity.



Users		
Admin	Installer	

	Products	
Create	Admin	
Update	Admin	
View	Admin	Installer
Delete	Admin	

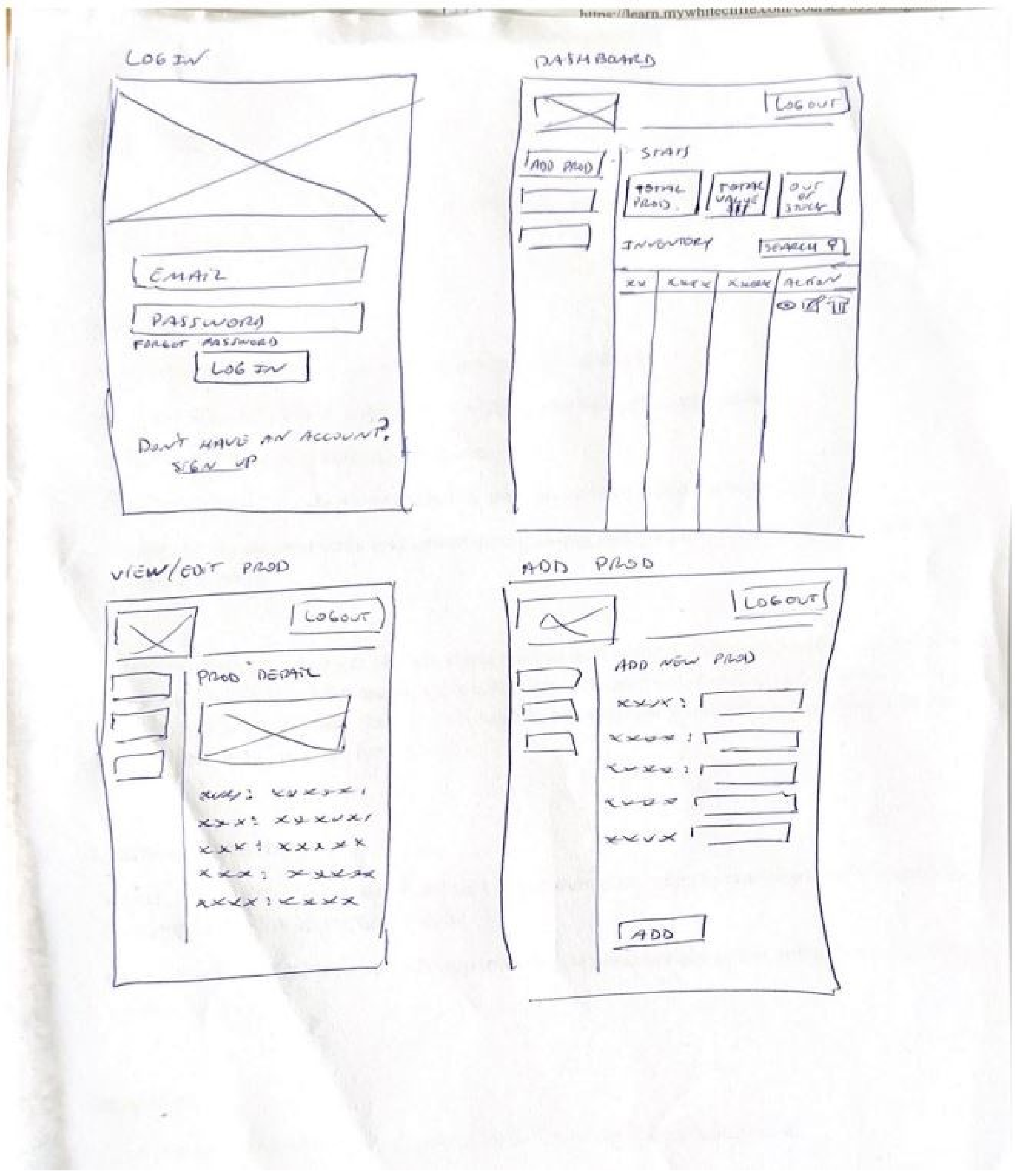
Above is a functional decomposition of the two types of users that will be using this system and also displays the different permissions that they will each have. These permissions were taken by the supplied client requirements:

- **Admin**: Will have the ability to Create New Products, View Inventory, Update Stock, and get an alert when stocks are low to place an order and view reports such as total stock, total sold.

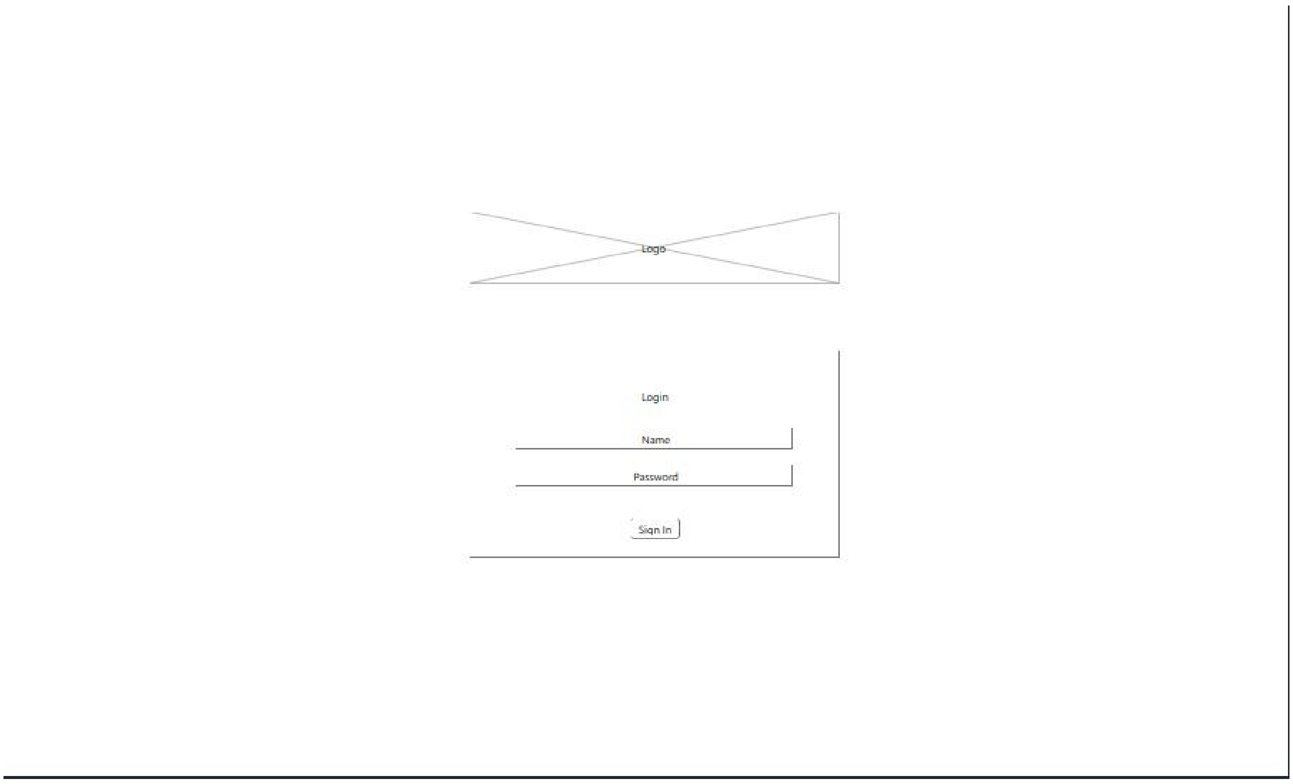
- **Installers**: Will be able to view the inventory and reports such as total stock, total sold.

User Interface Design

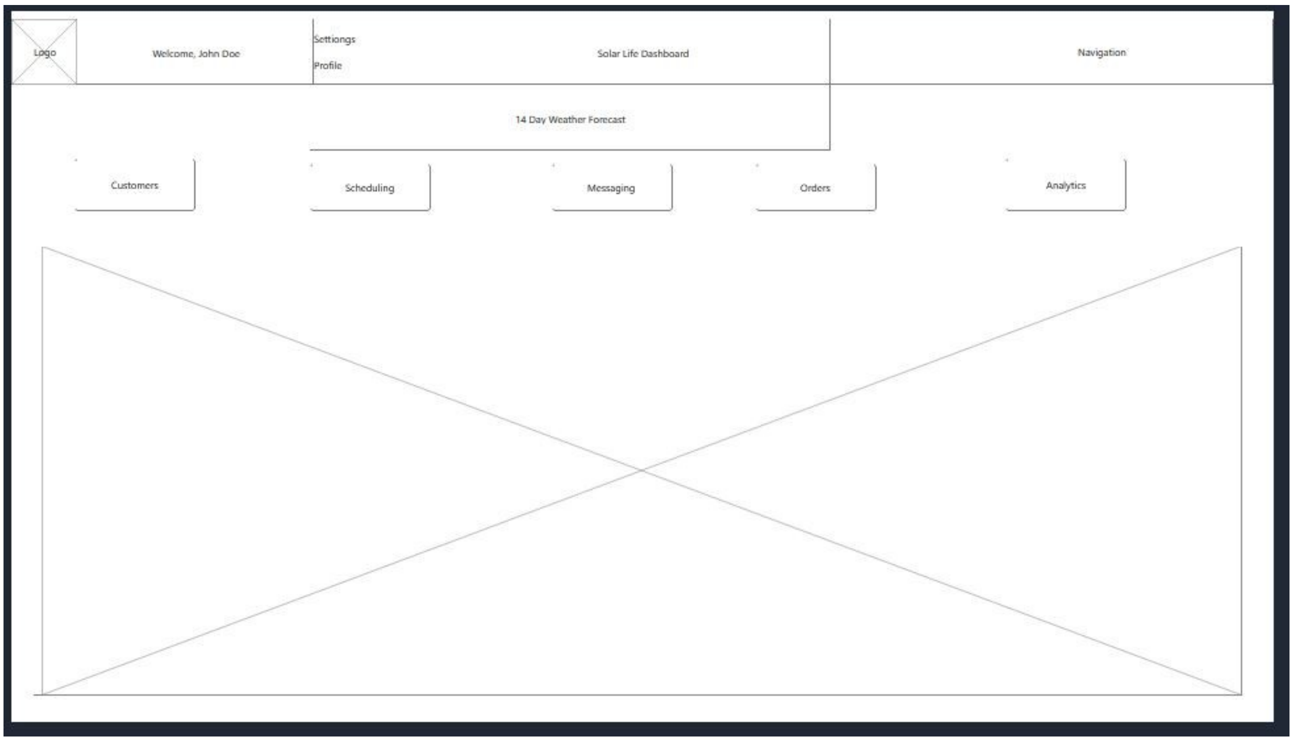
Sketches



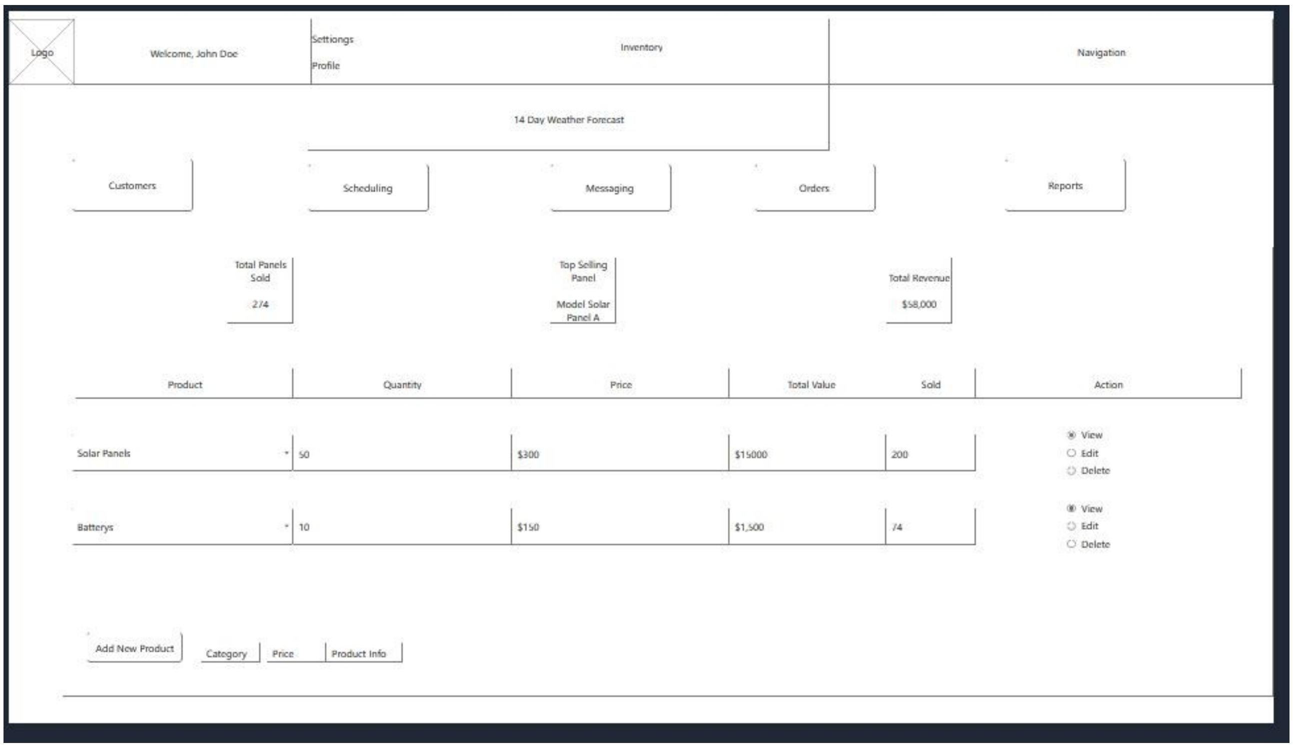
Login



Dashboard



Inventory



Logo

Welcome, John Doe

Settiongs

Profile

Inventory

Navigation

14 Day Weather Forecast

Customers

Scheduling

Messaging

Orders

Reports

Total Panels Sold

274

Top Selling Panel

Model Solar Panel A

Total Revenue

$58,000

Product	Quantity	Price	Total Value	Sold	Action
Solar Panels	50	$300	$15000	200	[✓ View] Edit Delete
Batterys	10	$150	$1,500	74	[✓ View] Edit Delete

Add New Product

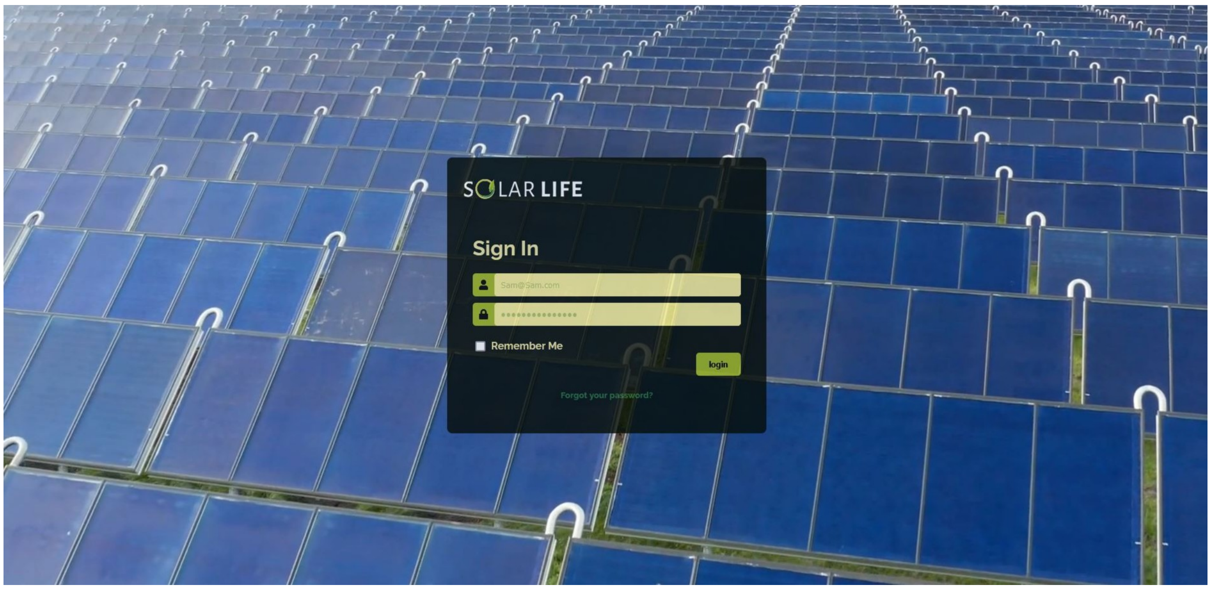
Category

Price

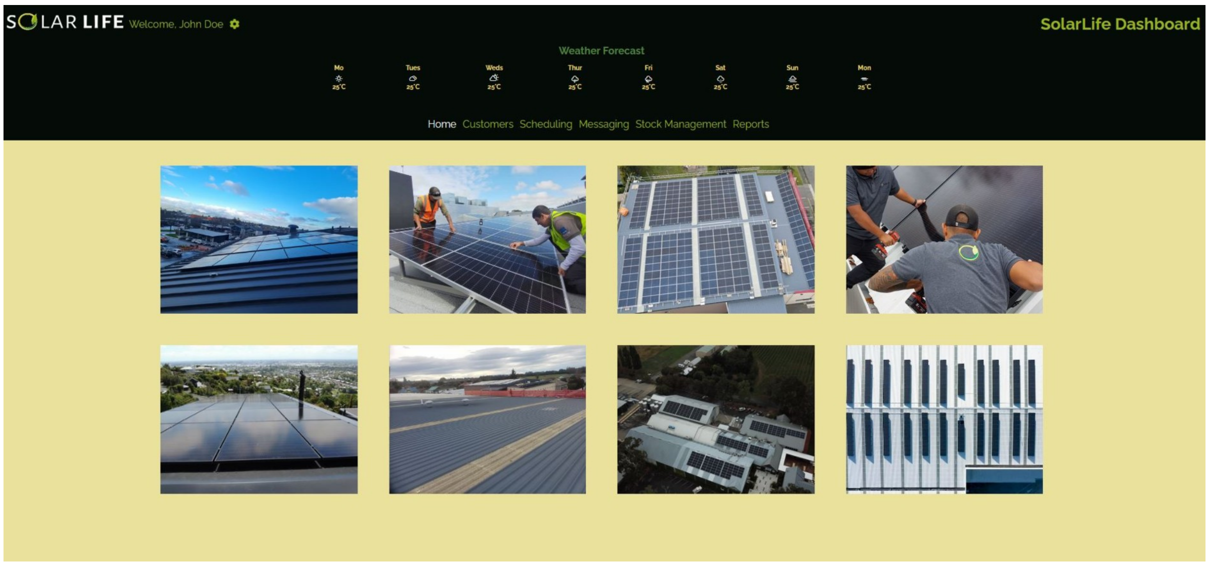
Product Info

Prototype

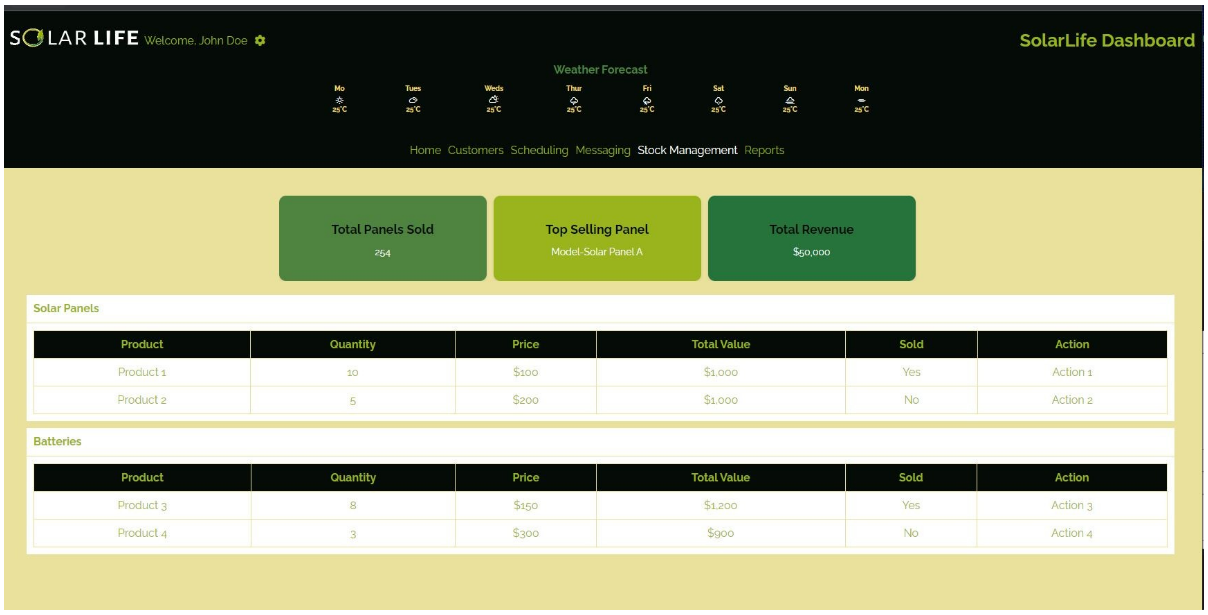
Login Page



Dashboard Page



Inventory Page



Total Panels Sold

254

Top Selling Panel

Model-Solar Panel A

Total Revenue

$50,000

Solar Panels

Product	Quantity	Price	Total Value	Sold	Action
Product 1	10	$100	$1,000	Yes	Action 1
Product 2	5	$200	$1,000	No	Action 2

Batteries

Product	Quantity	Price	Total Value	Sold	Action
Product 3	8	$150	$1,200	Yes	Action 3
Product 4	3	$300	$900	No	Action 4