Project Title: Door & Frame Manufacturing ERP Web Application

Frontend & Backend: React.js + Vite (Full-Stack JavaScript)
Database: Firebase (Cloud Firestore)
Prepared by: Project Team, Computer Engineering Department, AVCOE
Date: November 2025

1. Introduction

1.1 Project Introduction

The *Door & Frame Manufacturing ERP Web Application* digitizes the planning, installation, and monitoring process for a door and frame manufacturing company.
Currently, site supervisors record measurements on paper and manually compile Excel sheets, resulting in errors and delays.
The new ERP system — built with **React + Vite** on the frontend and **Firebase** for backend data management — provides a seamless, real-time, cloud-based solution accessible from any device.

1.2 Purpose

The application's goals are to:

- Enable **supervisors** to create digital building plans (floors, flats, rooms).
- **Capture** and **validate door & frame measurements** without duplication.
- Assign carpenters, **track their progress**, and log issues.
- Allow **managers to review** and **approve projects**, **release payments**, and rate carpenter performance.
- Provide offline capability with **Firebase's** built-in local persistence.
- Deliver **real-time synchronization** when connectivity returns.

1.3 Overview

The system includes:

- **Supervisor Dashboard** – project creation, carpenter assignment, progress updates.
- **Manager Dashboard** – review, approval, payment, and rating.
- **Firebase Backend** – authentication, database, storage, and real-time sync.
- **Offline support** via Service Workers and Firestore persistence.

This architecture reduces manual data handling, ensures data accuracy, and improves communication between site and factory.

2. Functional Requirements

Feature 1: User Authentication & Role Management

- Firebase Authentication with email/password or Google login.
- Role-based access: Supervisor, Manager, Client.
- JWT-like session tokens handled by Firebase SDK.
- Auto logout after 15 minutes of inactivity.

Feature 2: Project & Building Plan Management

- Supervisor creates a new project with project name, location, floors, and flats.
- Room templates can be copied for similar floor plans.
- Each project assigned a unique projectID by Firebase.
- All data stored in Firestore collections for real-time access.

Feature 3: Door & Frame Specification Entry

- Supervisor enters door/frame details (material, size, thickness, label ID).
- Validation ensures uniqueness per room.
- Data automatically synced to Firebase and visible to manager instantly.

Feature 4: Carpenter Assignment

- Add carpenters and team details.
- Assign them to rooms/floors within a project.
- Progress tracked stage-wise (*Frame Installed* → *Door Installed* → *Finishing Done*).

Feature 5: Progress Tracking & Issue Logging

- Supervisors update progress using dropdown statuses.
- Any problem (e.g., wrong frame angle) can be logged as an "Issue Report."
- Managers can comment and mark issues as *Resolved/Rework Needed*.

Feature 6: Payment & Rating Module

- Payment milestones tied to task completion.
- Manager approves payments; transactions recorded in Firestore.
- Managers assign star ratings (1–5) for carpenter quality, timeliness, and behavior.
- Client can view aggregated ratings for future project reference.

Feature 7: Reporting & Exports

- Generate reports for building plan, door/frame specs, and carpenter ratings.
- Export in PDF/CSV using browser print APIs and cloud functions.

3. External Interface Requirements

3.1 User Interface

Aspect	Details
Framework	React.js + Vite
Styling	Tailwind CSS / Material UI
Responsiveness	Mobile-first (PWA)
Offline Support	Firebase persistence + Service Worker
Browser Support	Chrome, Edge, Firefox, Safari

3.2 Hardware Interface

Device	Minimum Specification
Developer PC	Intel i5, 8 GB RAM
Client Device	Android/iOS or PC with browser
Server	Firebase Cloud (serverless hosting)
Connectivity	≥ 5 Mbps recommended

3.3 Software Interface

Component	Description
Frontend	React.js + Vite for UI + Routing
Backend	Firebase Cloud Functions (Node.js)
Database	Cloud Firestore (NoSQL)
Auth	Firebase Authentication
Storage	Firebase Storage for files (labels/reports)
Hosting	Firebase Hosting / Vercel

3.4 Communication Interface

- HTTPS via Firebase Hosting (automatic SSL).
- Real-time sync using Firestore listeners.
- JSON payloads for cloud function triggers.
- Push notifications via Firebase Cloud Messaging (future enhancement).

4. Non-Functional Requirements

4.1 Performance

- Real-time updates < 1 s propagation via Firestore.
- Sync delay ≤ 3 s after reconnecting offline users.
- App load time < 2 s on 4 G network.

4.2 Safety

- Local persistence guarantees no data loss offline.
- Automatic multi-region Firestore replication.
- Regular backups using Firebase Backup Scheduler.

4.3 Security

- Role-based access rules defined in Firestore Security Rules.
- All reads/writes authenticated through Firebase Auth.
- Data encrypted in transit (HTTPS) and at rest (AES-256).
- Admin rights restricted to verified manager accounts.

4.4 Software Quality

Attribute	Requirement
Reliability	Managed backend with 99.99 % uptime
Usability	Intuitive UI and auto-sync behavior
Efficiency	Lightweight React + Vite bundle (< 200 KB)
Maintainability	Modular Firebase collections & React components
Portability	Works on all OS and browsers

4.5 Database Requirements

- **Database:** Firebase Cloud Firestore
- **Structure:**
  - Projects → Floors → Flats → Rooms → DoorFrameDetails
  - Carpenters → Tasks → Payments → Ratings
- **Indexes:** On projectID, floorID, and carpenterID.
- **Data Sync:** Real-time listener updates for Supervisor and Manager dashboards.

4.6 Software Requirements

Tool / Framework	Purpose
React.js + Vite	Full-stack development
Firebase Auth / Firestore / Storage	Backend services
Node.js v20 LTS	Cloud Functions environment
Git & GitHub	Version control
Figma	UI prototyping
VS Code	Development IDE

4.7 Hardware Requirements

Component	Minimum Specification
Developer Laptop	8 GB RAM, 256 GB SSD
Client Device	Smartphone or laptop
Internet	Stable 4 G connection
Storage	Cloud hosted; local caching 50 MB per device

5. Analysis Model

5.1 Process Flow

Supervisor → [Building Plan + Door Data]

↓

Firebase Firestore (Real-time Cloud DB)

↓

Manager → [Review + Approval + Payment]

↓

Client → [Reports & Ratings]

5.2 Offline Flow

- Data temporarily saved in local IndexedDB by Firebase SDK.
- When connection restores → auto-sync to Firestore.

6. System Implementation Plan

Phase	Task	Deliverable
Phase 1	Gather requirements & finalize SRS	Approved SRS
Phase 2	Firebase project setup & configuration	Working backend
Phase 3	Develop UI using React + Vite	Frontend prototype
Phase 4	Integrate Firebase SDKs (Auth, Firestore)	Connected web app
Phase 5	Testing with real site data	QA report
Phase 6	Deploy to Firebase Hosting	Live system
Phase 7	Feedback & Maintenance	Continuous improvement

7. Conclusion

The *Door & Frame Manufacturing ERP Web Application* built using **React + Vite + Firebase** offers a powerful, secure, and flexible platform for managing end-to-end site operations digitally.
By leveraging Firebase's **real-time updates, offline support, and cloud scalability**, the company can ensure accurate data capture, faster project execution, and transparent collaboration between site and management teams.