

**Discovering Knowledge**

**FINAL YEAR PROJECT REPORT**

**2025**

**Builder Management System**

**Group Members**

|  |  |
| --- | --- |
| **Student Name** | **Enrolment#** |
| Muhammad Shoaib Akhter Qadri | 02-131212-009 |

**Supervised by**

Engr Adnan-ur-Rahman

**Department of Software Engineering**

**BAHRIA UNIVERSITY KARACHI CAMPUS**

**Intellectual Property & Submission Policy**

This is to declare that,

1. The project under the supervision of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ having title \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ carried out in partial fulfillment of the requirements of Bachelor of Engineering in Software Engineering degree program and is the sole property of the Bahria University.
2. This report is submitted as the requirement for the project in accordance with the rules laid down by the Bahria University as part of the requirements for the award of the degree of Bachelor of Engineering in Software Engineering.
3. The work presented in this report is our own except where due reference or acknowledgement is given to the work of others.
4. We are aware of Bahira University asserts legal and beneficial ownership rights over all Intellectual Property developed as a result of support either directly from or channeled through Bahria University.
5. We are agreed to assign to Bahira University all of their rights, title and interest in and to Intellectual Property developed as a result of utilization of Bahira University Resources including copyright in any material that is teaching material, computer programs, or created at the request or direction of Bahira University.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Student Name** | **Enrolment #** | **Signatures** |
| **1** | Muhammad Shoaib Akhter Qadri | 02-131212-009 | \_\_\_\_\_\_\_\_\_\_\_ |

**Reference:**

**[1]** R&D policy handbook (BUORIC-P15)

**Submission Proforma**

It is acknowledged that I as a supervisor gone through this report. The report contains all the essential sections as required by the department in accordance with the rules laid down by the Bahria University.

|  |  |
| --- | --- |
| **Supervisor Name:** |  |
|  |  |
| **Supervisor Signatures:** |  |
| **Dated:** |  |

**Acknowledgments**

We would like to express our sincere gratitude to all the individuals and institutions who have contributed to the successful completion of our Final Year Project, "**Builder Management System**".

First and foremost, we would like to thank our supervisor, Engr Adnan-ur-Rahman, for his invaluable guidance, expertise, and continuous support throughout the development process. His insightful feedback, constructive criticism, and prompt assistance have been instrumental in shaping the project and ensuring its successful execution. We are truly grateful for his mentorship and encouragement.

We would also like to extend our heartfelt appreciation to the faculty members of Bahria University for their valuable insights and assistance whenever needed. Their expertise and knowledge in their respective fields have greatly enriched our understanding and helped us overcome various challenges during the project.

Furthermore, we would like to acknowledge the Head of Institute for providing us with the necessary facilities and resources that facilitated the smooth progress of our work. Their support and encouragement have been pivotal in our journey.

We are indebted to the technical support staff who have aided us in setting up the required infrastructure and resolving technical issues promptly. Their dedication and expertise have been invaluable in ensuring the smooth functioning of the project.

Lastly, we would like to express our gratitude to our friends and family for their unwavering support, understanding, and motivation throughout the project. Their encouragement and belief in our abilities have been a constant source of inspiration.

To all those mentioned above and anyone else who has contributed to our project in any way, we extend our deepest appreciation. Your assistance and support have played a significant role in the successful completion of "**Builder Management System”**

**Abstract**

The proposed project aims to develop a **Builder Management Software** tailored for the residential construction industry to address key challenges such as poor communication, lack of transparency, and ineffective project management. The platform will serve as a centralized hub for customers, builders, and tradies to collaborate and track construction projects in real-time. It will feature tools for seamless communication, project tracking, document management, and financial oversight, leveraging advanced technologies such as **React.js** with **Firebase** task automation.

Existing solutions like BuildTools and Procore offer functionalities in project management but fail to provide comprehensive real-time updates and personalized communication. The proposed system distinguishes itself through features such as AI-generated work breakdown structures (WBS), customizable project tracking, and direct communication channels, enhancing customer experience and ensuring accountability.

Adopting an **Agile Development Process**, the project will deliver iterative improvements based on feedback from stakeholders to ensure user satisfaction. By aligning with **Sustainable Development Goals** (SDGs) 9 and 11, the project promotes innovation and infrastructure development while fostering well-managed urban communities.

This software aspires to improve transparency, minimize disputes, and streamline project delivery, setting a new standard in residential construction management.

TABLE OF Contents

[1. INTRODUCTION 11](#_Toc170226041)

[1.1 Introduction 12](#_Toc170226042)

[1.2 Problem Statement 12](#_Toc170226043)

[1.3 Background / Literature Review 12](#_Toc170226044)

[1.4 Problem in existing system 13](#_Toc170226045)

[1.5 Project Solution 13](#_Toc170226046)

[1.5.1 Features of the project 13](#_Toc170226046)

[1.5.2 Methodology/Algorithm 13](#_Toc170226046)

[1.5.3 Technologies to be used 13](#_Toc170226046)

[1.5.4 Project Scope / Deliverables 13](#_Toc170226046)

[1.6 Project Scope / Deliverables 13](#_Toc170226046)

[2. Proposed model 14](#_Toc170226047)

[2.1 Features of the project 15](#_Toc170226048)

[2.2 Methodology/Algorithm 15](#_Toc170226053)

[2.3 Technologies to be used 15](#_Toc170226054)

[3. Software Project Management Plan 16](#_Toc170226055)

[3.1 Software Process Model 17](#_Toc170226056)

[3.2 Roles and Responsibilities 18](#_Toc170226057)

[3.3 Tools and Techniques 18](#_Toc170226058)

[3.3.1 Programming Languages: 18](#_Toc170226059)

[3.3.2 Techniques: 18](#_Toc170226060)

[3.3.3 Tools: 18](#_Toc170226061)

[3.3.4 Coding Standards: 19](#_Toc170226062)

[3.4 Project Management Plan 19](#_Toc170226063)

[3.4.1 Tasks 19](#_Toc170226064)

[3.4.2 Task: SPMP-T001 20](#_Toc170226065)

[3.4.3 Task: SPMP-T002 21](#_Toc170226066)

[3.4.4 Task: SPMP-T003 22](#_Toc170226067)

[3.5 Assignments 24](#_Toc170226068)

[3.6 Timetable 25](#_Toc170226069)

[4. Software Requirements Specifications 26](#_Toc170226070)

[4.1 Introduction 27](#_Toc170226071)

[4.2 Product Overview 27](#_Toc170226072)

[4.3 Specific Requirements 27](#_Toc170226073)

[4.3.1 Image Recognition System Based on Research Findings 27](#_Toc170226074)

[4.3.2 Mobile Application Development 27](#_Toc170226075)

[4.3.3 Recommendations Module 28](#_Toc170226076)

[4.3.4 Testing 28](#_Toc170226077)

[4.4 Functional and Data Requirements 28](#_Toc170226078)

[4.4.1 Image Recognition System 28](#_Toc170226079)

[4.4.2 Mobile Application Development 29](#_Toc170226080)

[4.4.3 Multiclass Classification 29](#_Toc170226081)

[4.4.4 Nutritional Information 30](#_Toc170226082)

[4.4.5 Recommendations Module 30](#_Toc170226083)

[4.5 Non-Functional Requirements 30](#_Toc170226084)

[4.5.1 Reliability 30](#_Toc170226085)

[4.5.2 Availability 30](#_Toc170226086)

[4.5.3 Maintainability 31](#_Toc170226087)

[4.5.4 Portability 31](#_Toc170226088)

[4.5.5 Performance 31](#_Toc170226089)

[4.5.6 Usability Requirements 32](#_Toc170226090)

[4.6 Proposed Solution 32](#_Toc170226091)

[4.6.1 Technology Stack: 33](#_Toc170226092)

[4.7 External Interface Requirements 34](#_Toc170226093)

[4.7.1 User Interfaces 34](#_Toc170226094)

[4.7.2 Hardware Interfaces 36](#_Toc170226095)

[4.7.3 Software Interfaces 37](#_Toc170226096)

[4.7.4 Communications Protocols 38](#_Toc170226097)

[4.8 Database Requirements 38](#_Toc170226098)

[5. Software Design Description 39](#_Toc170226099)

[5.1 Introduction 40](#_Toc170226100)

[5.2 Design Overview 40](#_Toc170226101)

[5.3 Work Flow Diagram 40](#_Toc170226102)

[5.4 Work Breakdown Structure 41](#_Toc170226103)

[5.5 Use case Diagram 42](#_Toc170226104)

[5.6 Sequence Diagram 43](#_Toc170226105)

[5.7 Requirements Traceability Matrix 44](#_Toc170226106)

[5.8 System Architectural Design 45](#_Toc170226107)

[5.9 Chosen System Architecture 45](#_Toc170226108)

[The architecture is outlined below: 46](#_Toc170226109)

[5.10 System Interface Description 47](#_Toc170226110)

[5.10.1 Operating System Interface: 47](#_Toc170226111)

[5.10.2 Files Interface: 48](#_Toc170226112)

[5.10.3 Networking Interface: 48](#_Toc170226113)

[5.10.4 Libraries: 48](#_Toc170226114)

[5.11 Detailed Description of Components 49](#_Toc170226115)

[5.11.1 Component-1: Image Recognition System Based on Research Findings 49](#_Toc170226116)

[5.11.2 Component-2: Mobile Application 49](#_Toc170226117)

[5.11.3 Component 3: Real-time Classification Component 50](#_Toc170226118)

[5.11.4 Component 4: Multiclass Classification Component 50](#_Toc170226119)

[5.11.5 Component 5: User-Friendly Interface Component 50](#_Toc170226120)

[5.11.6 Component 6: Nutritional Information Component 51](#_Toc170226121)

[5.11.7 Component 7: Recommendations Module Component 51](#_Toc170226122)

[5.12 User Interface Design 52](#_Toc170226123)

[6. Implementation 55](#_Toc170226124)

[6.1 Algorithms: 56](#_Toc170226125)

[6.1.1 Resnet-50: 56](#_Toc170226126)

[6.1.2 VGG-16 64](#_Toc170226127)

[6.2 Frontend: 74](#_Toc170226128)

[6.2.1 Classification Component: 74](#_Toc170226129)

[6.2.2 Recommendation Component 79](#_Toc170226130)

[6.2.3 Nutrition Card Component 79](#_Toc170226131)

[6.2.4 Home Screen: 80](#_Toc170226132)

[6.2.5 Calories Box: 81](#_Toc170226133)

[6.3 Backend Code: 82](#_Toc170226134)

[6.3.1 Date Image Consumer Code: 82](#_Toc170226135)

[6.3.2 DockerFile: 83](#_Toc170226136)

[6.3.3 Asgi.py: 84](#_Toc170226137)

[7. Software Test Document 85](#_Toc170226138)

[7.1 System Overview: 86](#_Toc170226139)

[7.1.1 First Model (Date Identification): 86](#_Toc170226140)

[7.1.2 Second Model (Date Classification): 86](#_Toc170226141)

[7.2 System Testing: 86](#_Toc170226142)

[7.2.1 Algorithm Testing: 86](#_Toc170226143)

[7.2.2 WebSocket Testing: 87](#_Toc170226144)

[7.2.3 Software Version: 87](#_Toc170226145)

[7.3 Test Approach 87](#_Toc170226146)

[7.3.1 Major Testing Activities and Techniques 87](#_Toc170226147)

[7.3.2 Major Feature Groups and Combinations 89](#_Toc170226148)

[7.4 Test Plan: 89](#_Toc170226149)

[7.4.1 Scope 89](#_Toc170226150)

[7.4.2 Algorithm Performance Testing: 90](#_Toc170226151)

[7.4.3 Date Identification Testing: 90](#_Toc170226152)

[7.4.4 Date Classification Testing: 90](#_Toc170226153)

[7.4.5 WebSocket Testing: 90](#_Toc170226154)

[7.4.6 Frontend Data Retrieval Testing: 90](#_Toc170226155)

[7.4.7 Resources: 90](#_Toc170226156)

[7.4.8 Items Being Tested: 91](#_Toc170226157)

[7.5 Features to be Tested 91](#_Toc170226158)

[7.5.1 Features to be Tested 91](#_Toc170226159)

[7.5.2 Combination of Features to be Tested 93](#_Toc170226160)

[7.6 Test Cases 94](#_Toc170226161)

[7.6.1 Test Case Specifications 94](#_Toc170226162)

[7.6.2 Test Case: TC-007 100](#_Toc170226163)

[8. Appendix A: Test Results 102](#_Toc170226164)

[8.1.1 Test Results 102](#_Toc170226165)

[8.1.2 VGG-16 117](#_Toc170226166)

[9. Conclusions and Further Work 121](#_Toc170226167)

[9.1 Conclusions: 121](#_Toc170226168)

[9.2 Further Work: 121](#_Toc170226169)

[10. References 122](#_Toc170226170)

# INTRODUCTION

## Introduction

In the residential construction industry, **seamless communication, transparency, and efficient project management** between customers, builders, and tradies are crucial for the success of building projects. This proposal outlines the development of a Builder Management Software aimed at solving key pain points faced by both customers and builders. The proposed solution will offer a **centralized platform** where all parties can track and manage residential construction projects in real-time, ensuring transparency, communication, and accountability at all stages.

## Problem Statement

In the residential building industry, customers face significant challenges in managing communication and expectations with builders during the construction process. After selecting a builder and signing a contract, many customers experience frustration due to a **lack of transparency, poor communication, and missing documented evidence** of important discussions and agreements made during the project. Key project details, such as selected materials, appliances, and finishes, are often discussed verbally but not properly tracked, leading to misunderstandings and disputes over what was promised versus what is delivered.

This lack of transparency and accountability leads to customer dissatisfaction, project delays, and potential cost overruns due to unforeseen variations or changes that weren’t clearly documented or agreed upon.

The proposed web application aims to solve these problems by providing a centralized platform where customers and builders can seamlessly manage their residential construction projects. This platform will allow customers to create and track project details, agree on itemized lists with builders, and ensure that every decision is properly documented and agreed upon before the contract is signed. Additionally, it will provide continuous communication and progress updates throughout the building process, helping both parties stay aligned and informed, thus reducing misunderstandings and disputes.

## Background / Literature Review

Currently, the market offers several builder management software solutions such as BuildTools, Houzz Pro, Procore, CoConstruct, and Buildertrend. While these platforms address some construction management issues, there remain gaps in transparency, real-time communication, and customer-builder collaboration that we aim to fill with our software.

The existing solutions provide functionalities like project management, budget tracking, and progress updates but lack critical features that enhance the customer experience, particularly around personalized communication and real-time documentation of decisions.

Notable Builder Software Platforms:

* BuildTools: <https://www.buildtools.com/>
* Procore: <https://www.procore.com/>
* CoConstruct: <https://www.coconstruct.com/>
* Buildertrend: <https://www.buildertrend.com/>

## Problem in existing system

The existing builder management systems lack transparency, real-time communication, and comprehensive customer-builder collaboration. Customers face challenges in tracking project progress, managing agreements on materials and finishes, and ensuring documentation of important decisions. This often leads to misunderstandings, disputes, delays, and cost overruns. Current solutions also fail to provide personalized communication and real-time updates, leaving customers disconnected and dissatisfied.

## Proposed Solution

Our Builder Management Software will provide an **all-in-one platform** that integrates project management, customer communication, document tracking, and financial oversight. Key features are designed to address the core issues outlined in the problem statement, while offering enhanced functionality not found in existing builder software solutions.

### Features of the Project:

1. **Chatting & Communication:** A real-time messaging system allowing customers, builders, and tradies to communicate seamlessly.
2. **Products Selection & Documentation**: Tracking customer selections for materials, appliances, and finishes with documented agreements.
3. **Profile Management**: Comprehensive profile pages for customers, custom builders, and tradies.
4. **Project Management:** Features for customers and builders to add and track projects, including descriptions, images, and building links.
5. **Companies Section:** A dedicated section for listing companies involved in a project, visible in the sidebar for easy navigation.
6. **Project and Task Dashboard:** Centralized project dashboards for tracking progress, deadlines, payments, and key milestones.
7. **Documentation:** Automatic generation of project-related documents, and search keywords from documents.
8. **Customer Reviews:** A system for customers to leave feedback about builders and companies.
9. **Notifications System:** Real-time alerts for project updates, task completions, and document approvals.
10. **Task Management:** A task assignment system where customers can assign tasks to builders and track progress.
11. **Verified Builders Status:** A status system for verifying builders’ credentials and project compliance with Victorian standards.

# 1.5.2 Methodology/Algorithm:

**Agile Development Process:**

* **Sprints and Iterations**: The project will be divided into **sprints**, with each sprint lasting between 2 to 4 weeks. Each sprint will focus on completing specific deliverables (e.g., frontend design, backend integration, AI functionality).
* **Daily Stand-ups**: Short daily meetings with the development team to discuss progress, roadblocks, and tasks for the day.
* **User Stories and Backlog**: User stories (specific requirements) will be created and prioritized in the product backlog to ensure that the most important features (e.g., real-time communication, AI documentation) are developed first.
* **Continuous Integration and Delivery (CI/CD)**: New code will be continuously integrated, and regular testing will ensure that new features don’t break the system. This allows for frequent, reliable releases.
* **Feedback Loops**: The system will undergo **User Acceptance Testing (UAT)** during multiple stages of the project, incorporating feedback from builders, clients, and tradies to improve the product continuously.

### Technologies to be Used:

* **Frontend:** React.js
* **Backend/Database:** Firebase
* **Real-Time Communication:** Real time Firebase Database
* **Task Scheduling & Progress Tracking:** Firebase Firestore for scheduled tasks

## Project Scope / Deliverables

1. **Chatting & Communication:** A real-time messaging system allowing customers, builders, and tradies to communicate seamlessly.
2. **Products Selection & Documentation**: Tracking customer selections for materials, appliances, and finishes with documented agreements.
3. **Profile Management**: Comprehensive profile pages for customers, custom builders, and tradies.
4. **Project Management:** Features for customers and builders to add and track projects, including descriptions, images, and building links.
5. **Companies Section:** A dedicated section for listing companies involved in a project, visible in the sidebar for easy navigation.
6. **Customer Reviews:** A system for customers to leave feedback about builders and companies.
7. **Notifications System:** Real-time alerts for project updates, task completions, and document approvals.
8. **Verified Builders Status:** A status system for verifying builders’ credentials and project compliance with Victorian standards.

# Software Project Management Plan

## Software Process Model

The **Agile Development Process** is selected for this project to ensure flexibility, collaboration, and continuous improvement. Key components include:

* **Sprints and Iterations**: Development is divided into 2-4 week sprints, focusing on specific deliverables.
* **Daily Stand-ups**: Regular meetings to address progress, challenges, and daily tasks.
* **User Stories and Backlog**: Features are prioritized in a backlog to address critical functionalities first.
* **Continuous Integration and Delivery (CI/CD)**: Regular code integration and testing for reliable updates and releases.
* **Feedback Loops**: User Acceptance Testing (UAT) at various stages to refine the product based on stakeholder input.

This model ensures that the software evolves to meet user needs while maintaining high quality and adaptability.

## Roles and Responsibilities

|  |  |
| --- | --- |
| **Team Members** | **Role** |
| **Muhammad Shoaib Akhter Qadri** | Requirement Analysis, System Design, Development, Database Management, Real-Time Communication, Testing and Debugging, Documentation, Deployment and Stakeholder Communication |

## Tools and Techniques

1. **Frontend Development**: React.js for building a dynamic and responsive user interface.
2. **Backend Development**: Firebase for backend services, including authentication, database, and hosting.
3. **Database Management**: Firebase Firestore for real-time data storage and retrieval.
4. **Version Control**: Git and GitHub for code collaboration and version management.
5. **Task Management**: Jira for organizing and tracking project progress.
6. **Communication**: Slack or Microsoft Teams for team collaboration and communication.
7. **Testing**: React Testing Library for unit and integration testing.

**Techniques:**

1. **Agile Methodology**: Iterative development with regular feedback loops for continuous improvement.
2. **Continuous Integration and Delivery (CI/CD)**: Automating code integration, testing, and deployment.
3. **Real-Time Updates**: Leveraging Firebase for instant synchronization of data across all users.
4. **Responsive Design**: Ensuring the application is accessible on various devices using React.js.
5. **Secure Development Practices**: Implementing secure coding techniques to protect user data and prevent vulnerabilities.

**2.4 Project Management Plan**

**2.4.1 Tasks**

**2.4.1.1 Requirements Analysis and Clarification**

**Description**: Analyze and clarify project requirements to ensure understanding and identify core functionalities.  
**Subtasks**:

* Review existing project documentation.
* Discuss ambiguities within the team.
* Identify necessary features and functionalities.  
  **Task Identifier**: SPMP-T001  
  **Estimated Time**: 1 Week

**2.4.1.2 Task: System Design**

**Description**: Create a comprehensive system design, including architecture, module specifications, and interaction diagrams.  
**Subtasks**:

* Develop system architecture diagrams.
* Specify modules and their functions.
* Create interaction diagrams.
* Document the design in the Software Design Description (SDD).  
  **Task Identifier**: SPMP-T002  
  **Estimated Time**: 2 Weeks

**2.4.1.3 Task: Development - Version 1**

**Description**: Implement the system's initial version based on finalized requirements and design.  
**Subtasks**:

* Set up the development environment.
* Write code for required features.
* Conduct unit testing for components.
* Integrate components and perform system testing.  
  **Task Identifier**: SPMP-T003  
  **Estimated Time**: 5 Weeks
* **2.4.2 Task: SPMP-T001**

**Description**: Conduct requirements analysis and clarification, identifying key features of the system.  
**Deliverables and Milestones**:

* Deliverables: Updated requirements documentation and list of core functionalities.
* Milestones: Completion of requirements review and feature identification.  
  **Resources Needed**: Access to project documents and reference materials.  
  **Dependencies and Constraints**: Dependent on availability of required materials and stakeholder engagement.  
  **Risks and Contingencies**:
* Risk: Lack of resources.
  + Contingency: Use online references or consult experts.
* Risk: Limited stakeholder input.
  + Contingency: Implement regular progress updates and alternative communication channels.
* **2.4.3 Task: SPMP-T002**

**Description**: Develop a detailed system design, including architecture and module specifications.  
**Deliverables and Milestones**:

* Deliverables: System architecture diagrams, module specifications, interaction diagrams, and SDD.
* Milestones: Completion of architecture, diagrams, and SDD.  
  **Resources Needed**: Tools for diagrams and reference materials for design principles.  
  **Dependencies and Constraints**: Requires completion of SPMP-T001.  
  **Risks and Contingencies**:
* Risk: Delays in resources.
  + Contingency: Explore alternative tools.
* Risk: Requirement changes.
  + Contingency: Use a change management process.
* **2.4.4 Task: SPMP-T003**

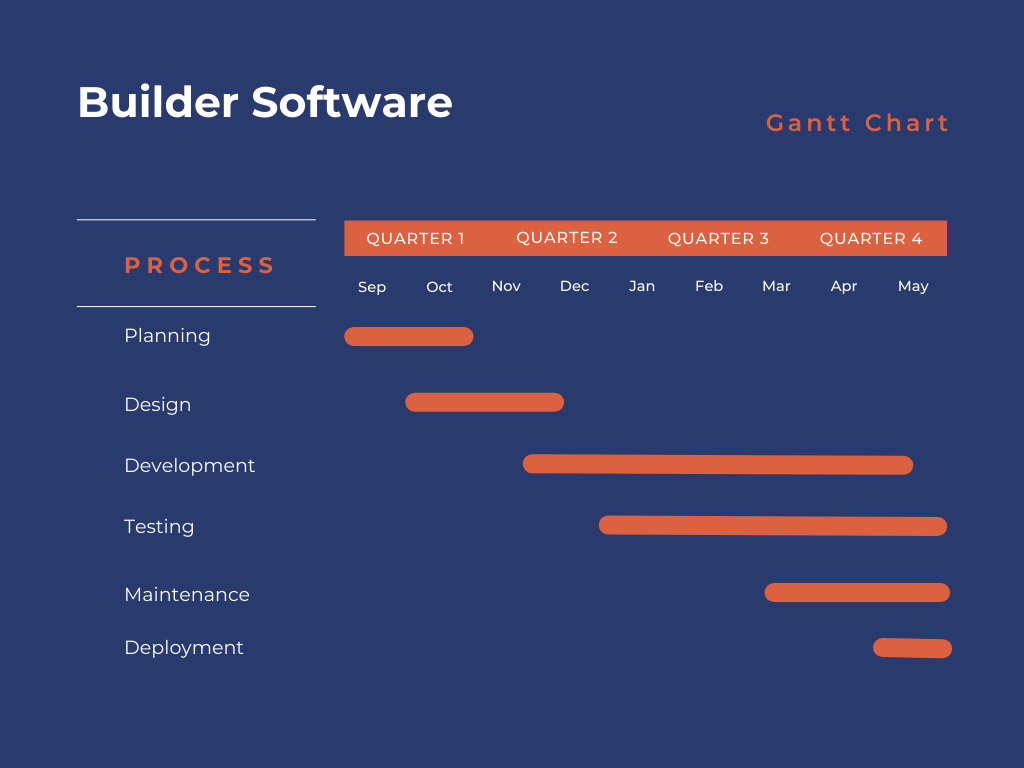
**Description**: Implement the system's functionalities and conduct testing for Version 1.  
**Deliverables and Milestones**:

* Deliverables: Functional Version 1 of the system.
* Milestones: Completion of coding, unit testing, integration, and system testing.  
  **Resources Needed**: Software Design Document (SDD), VS Code.  
  **Dependencies and Constraints**: Requires completion of SPMP-T002.  
  **Risks and Contingencies**:
* Risk: Technical challenges.
  + Contingency: Allocate additional time for debugging.
* Risk: Integration issues.
  + Contingency: Perform thorough integration testing.

## Assignments

|  |  |
| --- | --- |
| **Tasks** | **Performed by** |
| Research Work | Muhammad Shoaib Akhter Qadri |
| Data Collection | Muhammad Shoaib Akhter Qadri |
| SRS Document | Muhammad Shoaib Akhter Qadri |
| SRS Document Review | Muhammad Shoaib Akhter Qadri |
| WBS | Muhammad Shoaib Akhter Qadri |
| Use Case Diagram | Muhammad Shoaib Akhter Qadri |
| Architecture Diagram | Muhammad Shoaib Akhter Qadri |
| Sequential Diagram | Muhammad Shoaib Akhter Qadri |
| Workflow Diagram | Muhammad Shoaib Akhter Qadri |
| SDD | Muhammad Shoaib Akhter Qadri |
| Plan Document | Muhammad Shoaib Akhter Qadri |
| Gantt Chart | Muhammad Shoaib Akhter Qadri |
| Research Work | Muhammad Shoaib Akhter Qadri |
| Frontend Development | Muhammad Shoaib Akhter Qadri |
| DB Design and Creation | Muhammad Shoaib Akhter Qadri |
| API Development | Muhammad Shoaib Akhter Qadri |
| Model Training | Muhammad Shoaib Akhter Qadri |
| Model Testing | Muhammad Shoaib Akhter Qadri |
| Model Development | Muhammad Shoaib Akhter Qadri |

## Timetable



|  |  |  |
| --- | --- | --- |
| **Task** | **Start Date** | **Completion Date** |
| Planning | 2024-11-01 | 2024-11-31 |
| Design | 2024-11-01 | 2025-12-07 |
| Frontend Development | 2024-12-08 | 2025-05-01 |
| Backend Development | 2025-02-01 | 2025-05-15 |
| Testing | 2025-03-01 | 2025-04-01 |
| Maintenance | 2025-04-01 | 2025-05-01 |
| Deployment | 2025-05-01 | 2025-05-15 |

# Software Requirements Specifications

## Introduction

## The "Builder Management System" aims to address challenges in the residential construction industry by offering a centralized platform for seamless communication, transparency, and project management among customers, builders, and tradies. It ensures that all parties stay informed and aligned throughout the project lifecycle, mitigating common issues like poor communication, misunderstandings, and delays.

## 3.2 Product Overview:

## The software will integrate project management tools, customer communication features, real-time document tracking, and financial oversight. It will be equipped with key functionalities like task management, chatting systems, progress dashboards and collaboration between stakeholders.

## 3.3 Specific Requirements:

## The system requires functionalities for project tracking, communication, task assignments, document management, and customer reviews. It must support real-time interactions, secure data storage, and compatibility across various devices. Additionally, the software must be intuitive and user-friendly to cater to a wide range of users, from customers to builders and tradies.

## 3.4 Functional and Data Requirements:

## The system must allow for real-time messaging, project updates, document sharing, task assignments, and progress tracking. Data must be securely stored and easily accessible to authorized users. The platform will also manage detailed information on materials, products, and construction phases, ensuring that all decisions are documented and traceable.

## 3.5 Non-Functional Requirements:

## Reliability: The system should be stable, with minimal downtime.

## Availability: It should offer high availability to ensure users can access it anytime.

## Security: Data protection is paramount, with secure login and encryption.

## Maintainability: The system must be easy to maintain and update.

## Portability: It should be accessible on different devices and platforms.

## Performance: It must handle a large number of users and transactions efficiently.

## Usability: The interface must be intuitive and user-friendly.

## 3.6 Proposed Solution:

## The proposed Builder Management System will provide all-in-one functionality to enhance project management, communication, and documentation in the residential construction sector. Key features include real-time chat, project management tools, task tracking, verified builder status, and customer feedback.

## 3.7 Alternative Solution:

## Other existing systems like BuildTools and Procore offer similar features but lack the real-time collaboration that will distinguish your system.

## 3.8 External Interface Requirements:

## User Interfaces: Intuitive and user-friendly design.

## Hardware Interfaces: Compatible with devices such as smartphones, tablets, and desktops.

## Software Interfaces: Integrates with third-party software for financial management and task scheduling.

## Communications Protocols: Real-time communication supported by Firebase Database for instant updates.

## 3.9 Database Requirements:

## The system will rely on Firebase for real-time data syncing, secure storage, and smooth integration across devices. It will store project data, task details, messages, and financial records in a scalable and secure manner.

# Software Design Description

## Work Flow Diagram

The Builder Management System is designed with a modular architecture that separates key functionalities such as project management, communication, task tracking, and documentation. The front-end will be built using React.js for an interactive user experience, while the back-end will use Firebase for real-time data syncing and database management.

## Introduction:

This section introduces the overall design approach of the Builder Management System. It outlines the software architecture, key design principles, and the technologies that will be used to develop the platform. The design aims to provide a user-friendly interface for customers, builders, and tradies, while ensuring seamless communication, real-time updates, and data security.

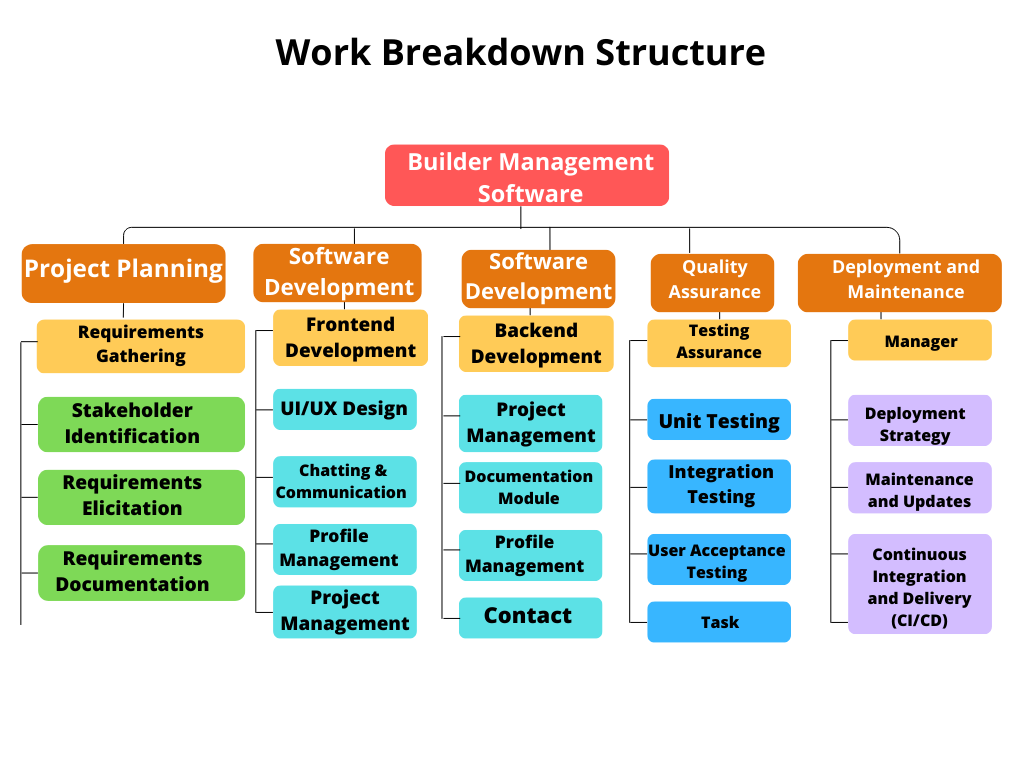
## Work Flow Diagram

A diagram of a software company

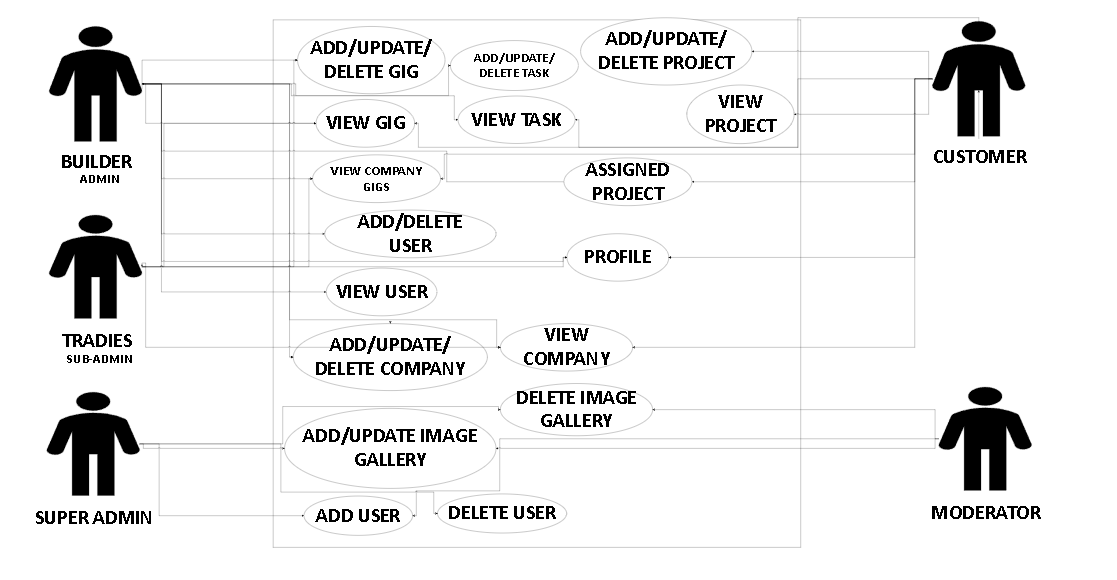
Description automatically generated

Workflow Diagram

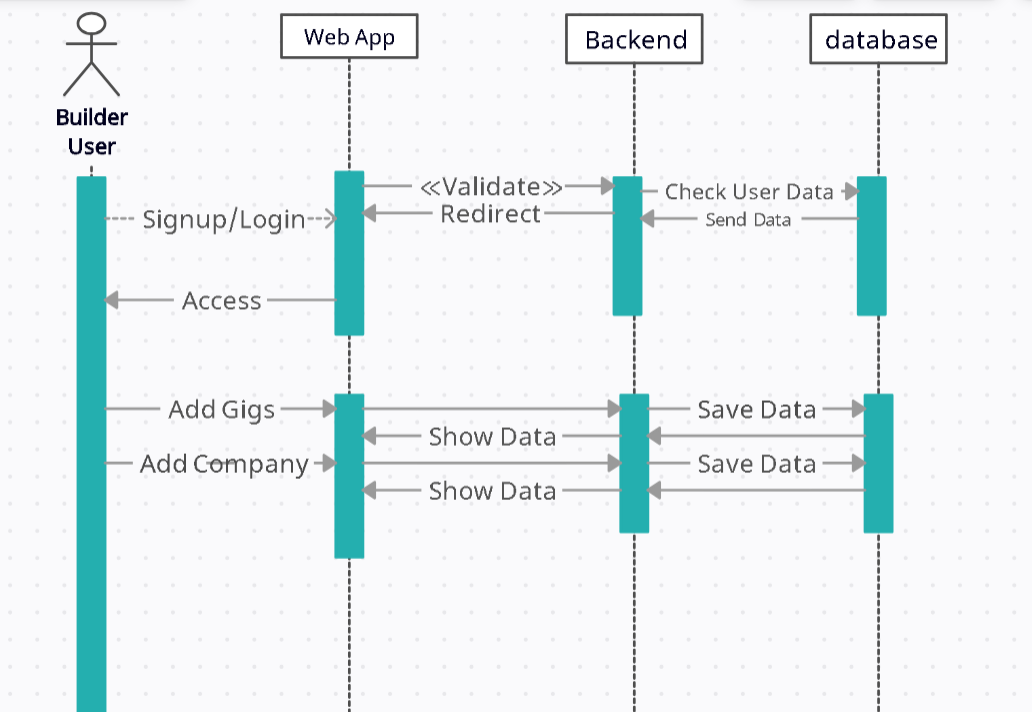
## Work Breakdown Structure



## Use case Diagram

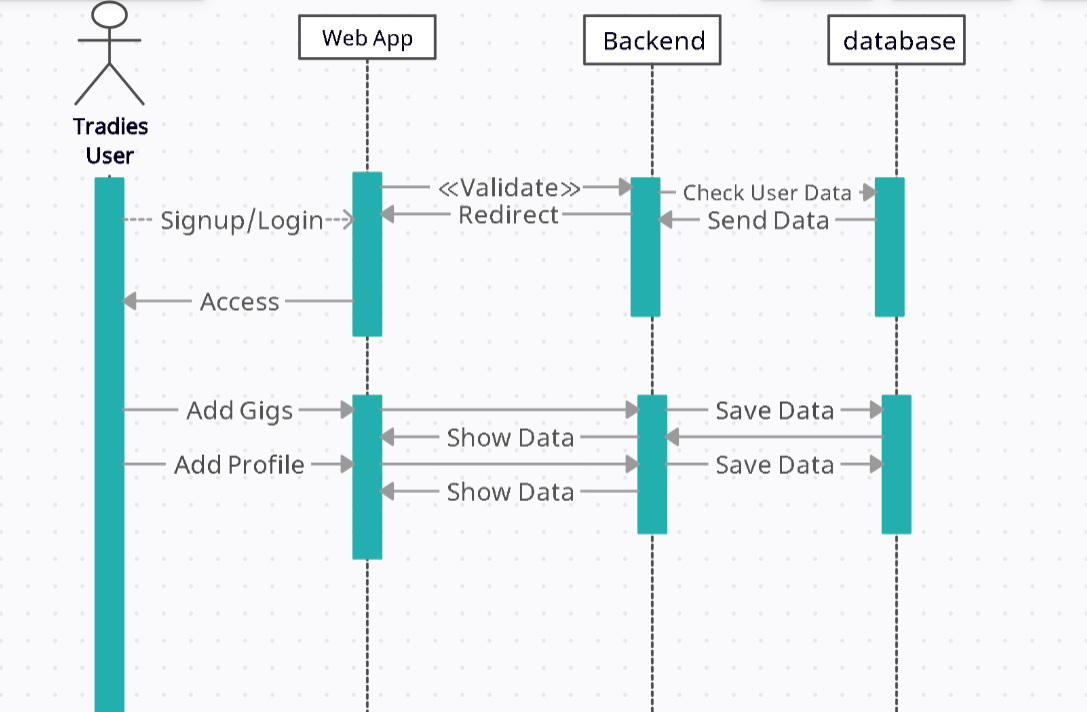


## Sequence Diagram



A diagram of a data flow

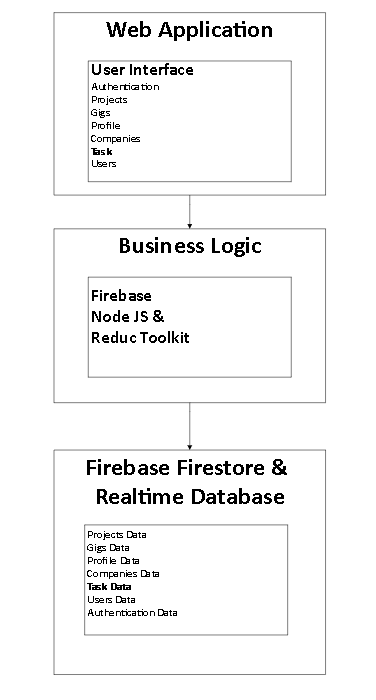
AI-generated content may be incorrect.



## Requirements Traceability Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Components/Requirements | **1**  **(Select Builder)** | **2**  **(Create Project)** | **3**  **(Track Progress)** | **4**  **(Messaging System)** | **5**  **(Task Management)** |
| 1. Real-time Messaging |  |  |  | ✓ |  |
| 2. Project Management |  | ✓ | ✓ |  |  |
| 3. Task Tracking |  |  | ✓ |  | ✓ |
| 4. Document Management |  |  | ✓ |  |  |
| 5. Builder Verification |  |  |  |  |  |
| 6. Customer Feedback |  |  |  |  |  |

## System Architectural Design



**4.9 Chosen System Architecture**

The Builder Management System follows a modular, client-server architecture, designed to separate core functionalities and ensure scalability, security, and real-time interactions.

* **Major Component Groupings:**

1. **Frontend (User Interface):**
   * Technology: React.js
   * Function: Provides an interactive and responsive interface for customers, builders, and tradies.
   * Key Features: Project dashboard, real-time messaging, task management, feedback system.
   * Internal Interface: Connects to the backend via APIs to fetch project data, messages, and tasks.
2. **Backend (Server and Database):**
   * Technology: Firebase (Real-time Database, Firestore for task scheduling, Firebase Authentication)
   * Function: Manages the logic and data flow of the system. Ensures secure data storage, real-time updates, and authentication.
   * Key Features: User authentication, real-time updates, data storage, task management.
   * Internal Interface: Communicates with the frontend to provide updates and process requests.
3. **Notification System:**
   * Technology: Firebase RealTime Database
   * Function: Sends real-time push notifications for project updates, task completions, and system alerts.
   * Internal Interface: Connects to the backend and frontend to trigger notifications.
4. **Interfaces:**

* **Internal Interfaces:** 
  + Frontend ↔ Backend: RESTful APIs using Firebase SDKs for data exchange.
  + Backend ↔ Notification System: Firebase RealTime Database for sending real-time updates to users.
* **External Interfaces:** 
  + Firebase: Handles user authentication, real-time database, and notification management.

1. **Technical Risks & Contingency Plans:**
2. **Real-time Data Syncing:** Delays or issues in real-time updates.
   * Contingency: Implement caching mechanisms and fallback strategies for offline usage.
3. **Scalability:** The system might struggle under heavy traffic or large projects.
   * Contingency: Utilize Firebase's auto-scaling capabilities and ensure optimized database structure.

**4.10 Discussion of Alternative Designs**

Several alternative designs were considered before choosing the current system architecture:

1. **Monolithic Architecture:**
   * **Description:** A single codebase with tightly coupled components for frontend, backend, and database.
   * **Reason for Rejection:** This design can lead to performance bottlenecks, difficult scalability, and harder maintenance. Decoupling components in a modular system enhances maintainability and scalability.
2. **Microservices Architecture:**
   * **Description:** The system is broken down into independent services (e.g., messaging, task management, user authentication).
   * **Reason for Rejection:** Microservices are complex and require significant overhead to manage inter-service communication, especially for a relatively small-scale application. Given the project scope, a modular architecture with integrated services is a more practical solution.

## System Interface Description

**1. Operating System Interface:**

The system will be platform-independent, utilizing modern web technologies such as HTML5, CSS3, and JavaScript (React.js). The backend relies on Firebase, which is cloud-based and does not require specific OS dependencies.

**2. Files Interface:**

The system will handle various file formats (e.g., PDFs, images for project designs, and Word documents for contracts). Files will be stored securely in Firebase's cloud storage and accessed through the frontend for download or viewing.

**3. Networking Interface:**

The system operates over HTTPS to ensure secure communication between the client (frontend) and the server (backend). Firebase real-time databases will be used for data synchronization between the client and server. All API calls between the frontend, backend, will follow RESTful principles.

**4. Libraries:**

* **Frontend Libraries:**
  + React.js for building dynamic user interfaces.
  + Axios for making API requests to the backend.
  + Firebase Realtime Database for real-time messaging features.
* **Backend Libraries:**
  + Firebase SDK for interacting with Firebase's real-time database, Firestore, and Firebase Authentication.
  + Express.js (optional) for managing API routes and backend logic if needed.

**4.12 Detailed Description of Components:**

**4.12.1 Component-1: Real-time Chat & Communication System**

**Responsibilities:**

* Develop real-time messaging functionality to facilitate direct and seamless communication between customers, builders, and tradies.
* Ensure messages are delivered instantly to enhance project coordination.

**Constraints:**

* Must provide immediate response and real-time synchronization.
* Ensure data security and privacy during message exchanges.

**Composition:**

* **Messaging Module:** Implements the real-time messaging capabilities using Firebase Real-time Database.
* **Notification Module:** Sends real-time alerts upon message reception.

**Interactions:**

* Direct interaction between Customers, Builders, and Tradies for communication.

**Resources:**

* Firebase Real-time Database for communication.
* Computational resources for handling simultaneous real-time interactions.

**4.12.2 Component-2: Products Selection & Documentation Component**

**Responsibilities:**

* Track and document selections of materials, appliances, and finishes chosen by customers.
* Generate automated documentation to maintain transparency.

**Constraints:**

* Must ensure all selections and decisions are accurately recorded and retrievable.

**Composition:**

* **Selection Module:** Tracks itemized customer selections.
* **Documentation Module:** Automatically generates and manages project-related documents.

**Interactions:**

* Interacts with Project Management Component to maintain transparency of documented agreements.

**Resources:**

* Storage for documentation.
* Computational resources for automated documentation generation.

**4.12.3 Component-3: Project & Task Management Dashboards**

**Responsibilities:**

* Provide centralized dashboards for tracking project progress, deadlines, payments, and tasks.

**Constraints:**

* Must deliver real-time updates for project status.

**Composition:**

* **Dashboard Module:** Centralized platform for visualization and monitoring project metrics.
* **Task Management Module:** Allows assignment and tracking of tasks.

**Interactions:**

* Communicates with other components for real-time data updates and task tracking.

**Resources:**

* Firebase Firestore for real-time task tracking and scheduled task management.

**4.12.4 Component-4: Verified Builders Status Component**

**Responsibilities:**

* Implement a verification system to certify builders based on compliance with Victorian standards.

**Constraints:**

* Must consistently update builders' verification statuses.

**Composition:**

* **Verification Module:** Handles verification and compliance tracking of builders.

**Interactions:**

* Updates builder profiles and interacts with Profile Management Component.

**Resources:**

* Data sources containing verification standards.
* Compliance checking and verification protocols.

**4.12.5 Component-5: User Interface & Profile Management**

**Responsibilities:**

* Design intuitive, attractive, and user-friendly interfaces.
* Maintain comprehensive profile pages for customers, builders, and tradies.

**Constraints:**

* Interface must adhere to UI/UX best practices for simplicity and accessibility.

**Composition:**

* **UI Design Module:** Creates and manages the visual design and user experience.
* **Profile Module:** Manages detailed user profiles and related information.

**Interactions:**

* Provides interfaces for all user interactions.
* Interacts with Dashboard and Communication modules.

**Resources:**

* Design assets and UI libraries.

**4.12.6 Component-6: Companies Section and Customer Reviews**

**Responsibilities:**

* Manage dedicated listings of companies involved in projects.
* Allow customers to leave feedback and reviews about builders and companies.

**Constraints:**

* Reviews and feedback must be transparently accessible and moderated.

**Composition:**

* **Company Module:** Manages company profiles and listings.
* **Reviews Module:** Handles customer reviews and feedback management.

**Interactions:**

* Interacts with the Profile and Project Management Components.

**Resources:**

* Database for storing reviews and company details.

**4.12.7 Component-7: Notifications and Alert System**

**Responsibilities:**

* Provide real-time alerts for project updates, task completions, and document approvals.

**Constraints:**

* Notifications must be instant and accurate.

**Composition:**

* **Notification Engine:** Generates and dispatches real-time alerts based on system triggers.

**Interactions:**

* Communicates with Project Management, Task Management, and Documentation modules.

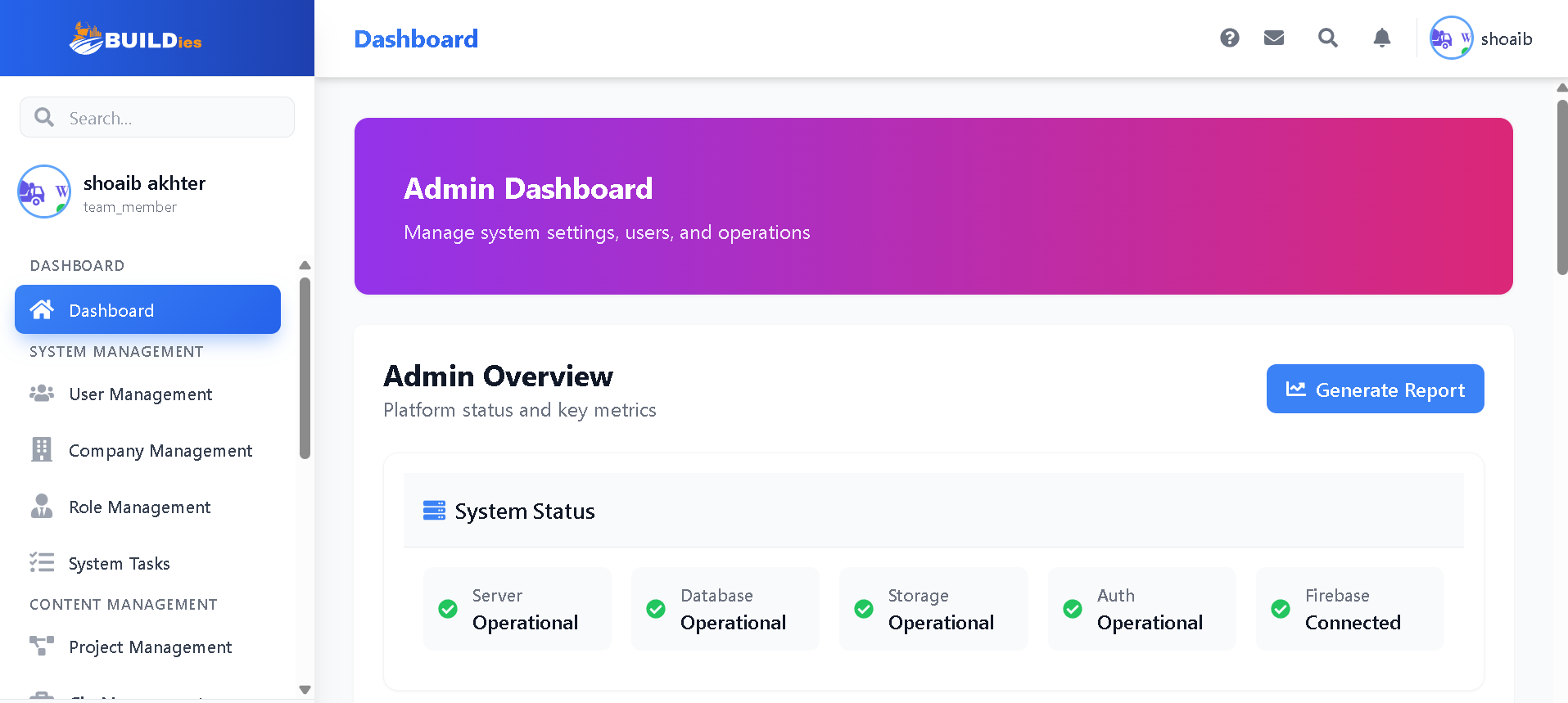
**Resources:**

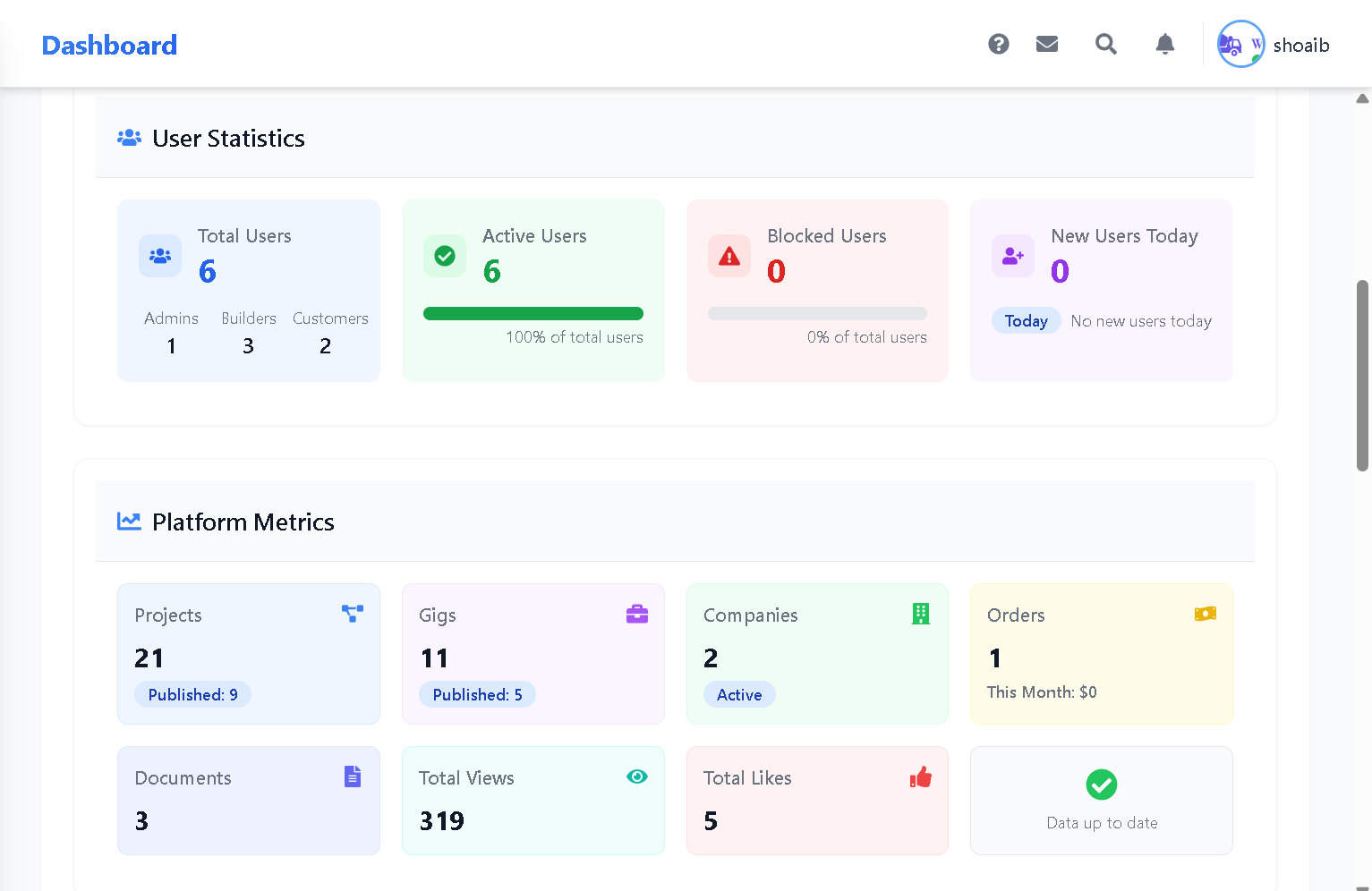
* Firebase real-time infrastructure for immediate notifications.
* **Methodology and Technologies:**
* **Frontend:** React.js
* **Backend:** Firebase
* **Database:** Firebase Firestore

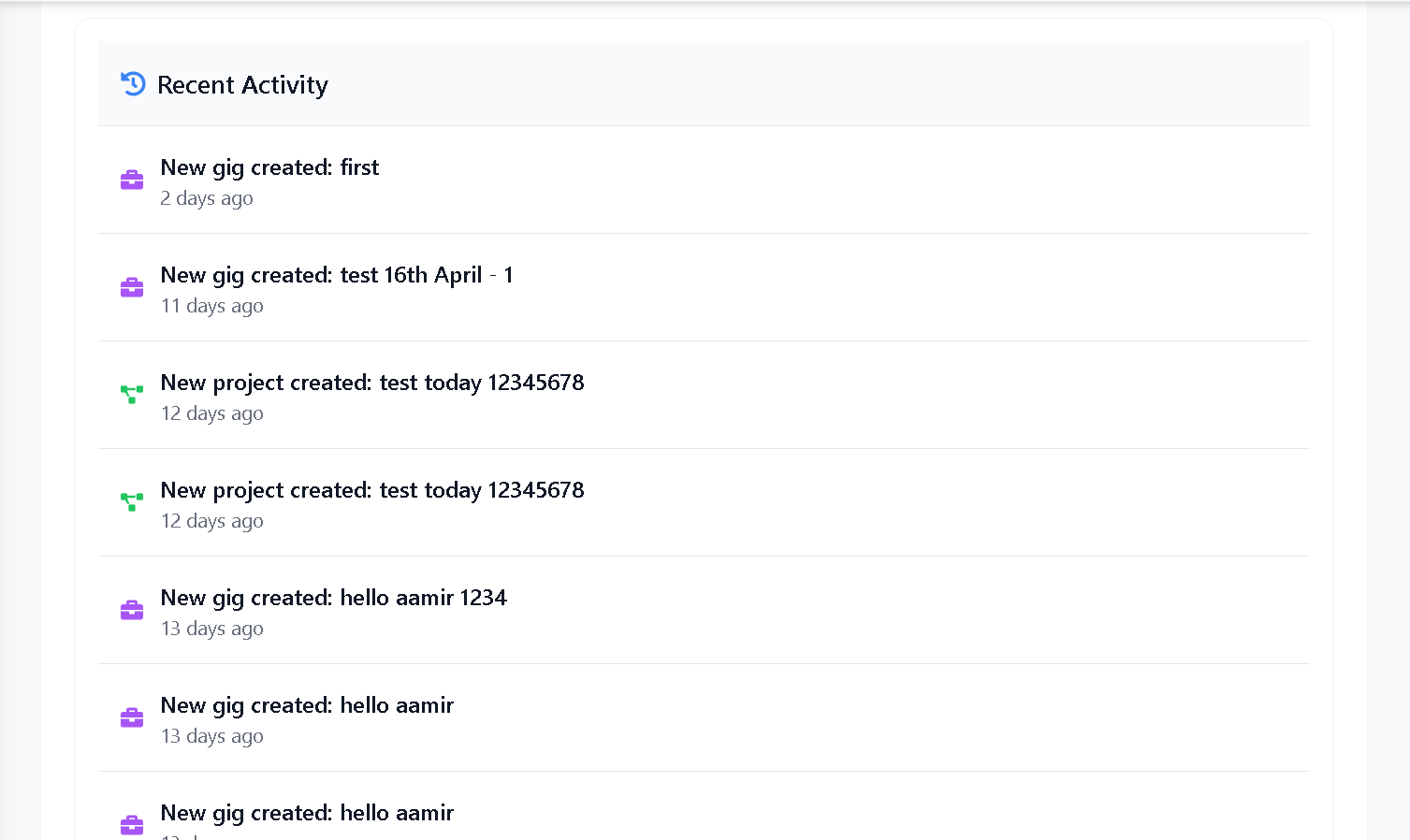
**Alignment with Sustainable Development Goals (SDGs):**

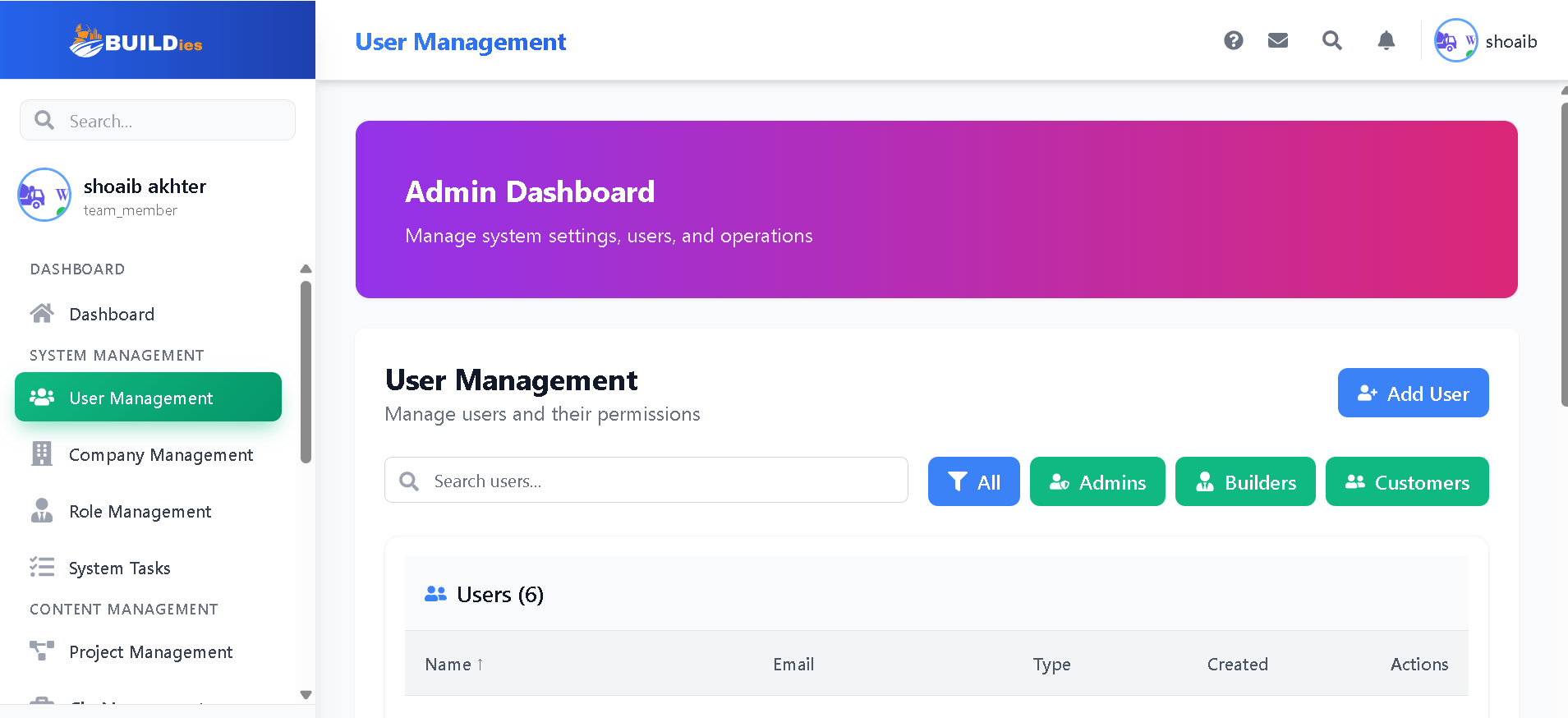
* **Goal 9:** Industry, Innovation, and Infrastructure - Promoting innovative solutions in the construction industry through digital platforms.
* **Goal 11:** Sustainable Cities and Communities - Supporting efficient and transparent management of urban residential construction projects.

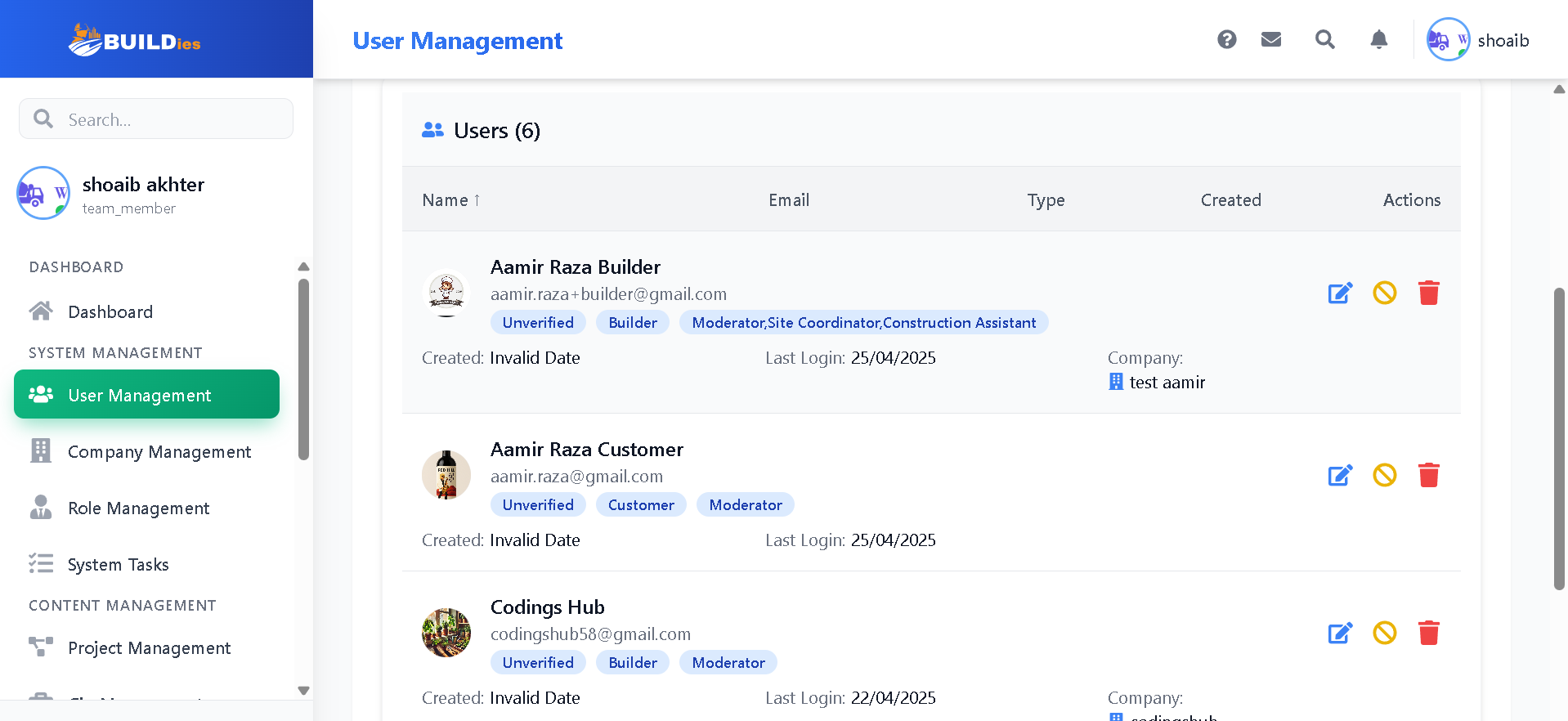
## User Interface Design

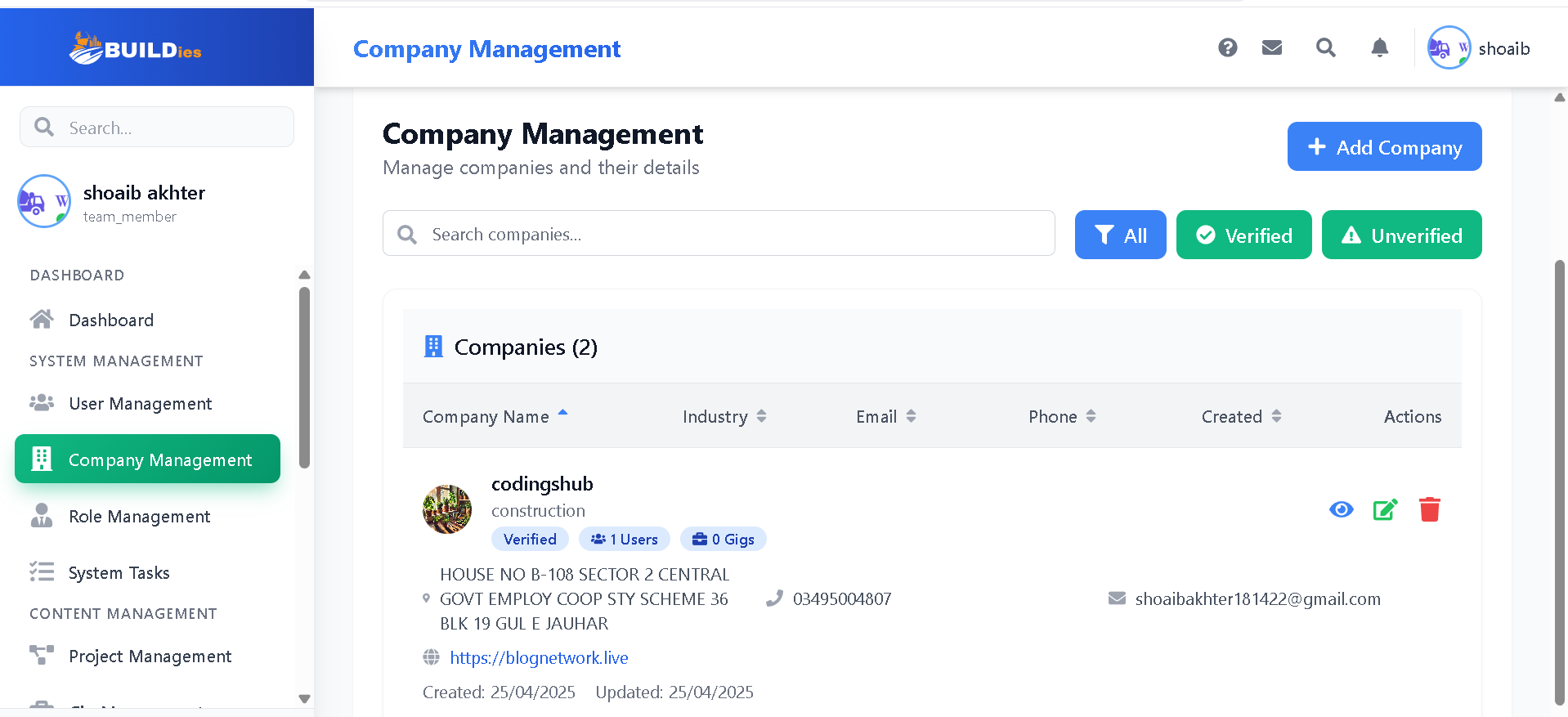


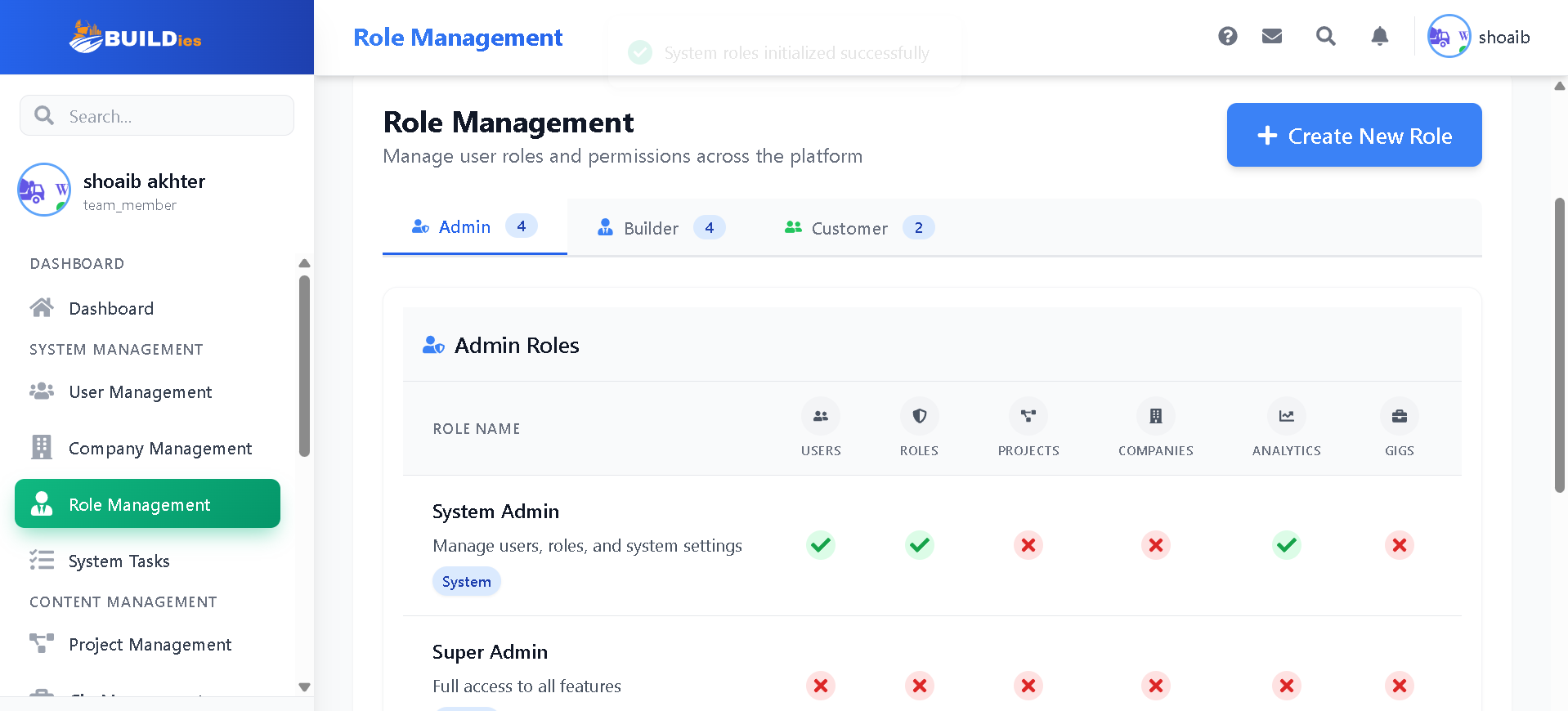


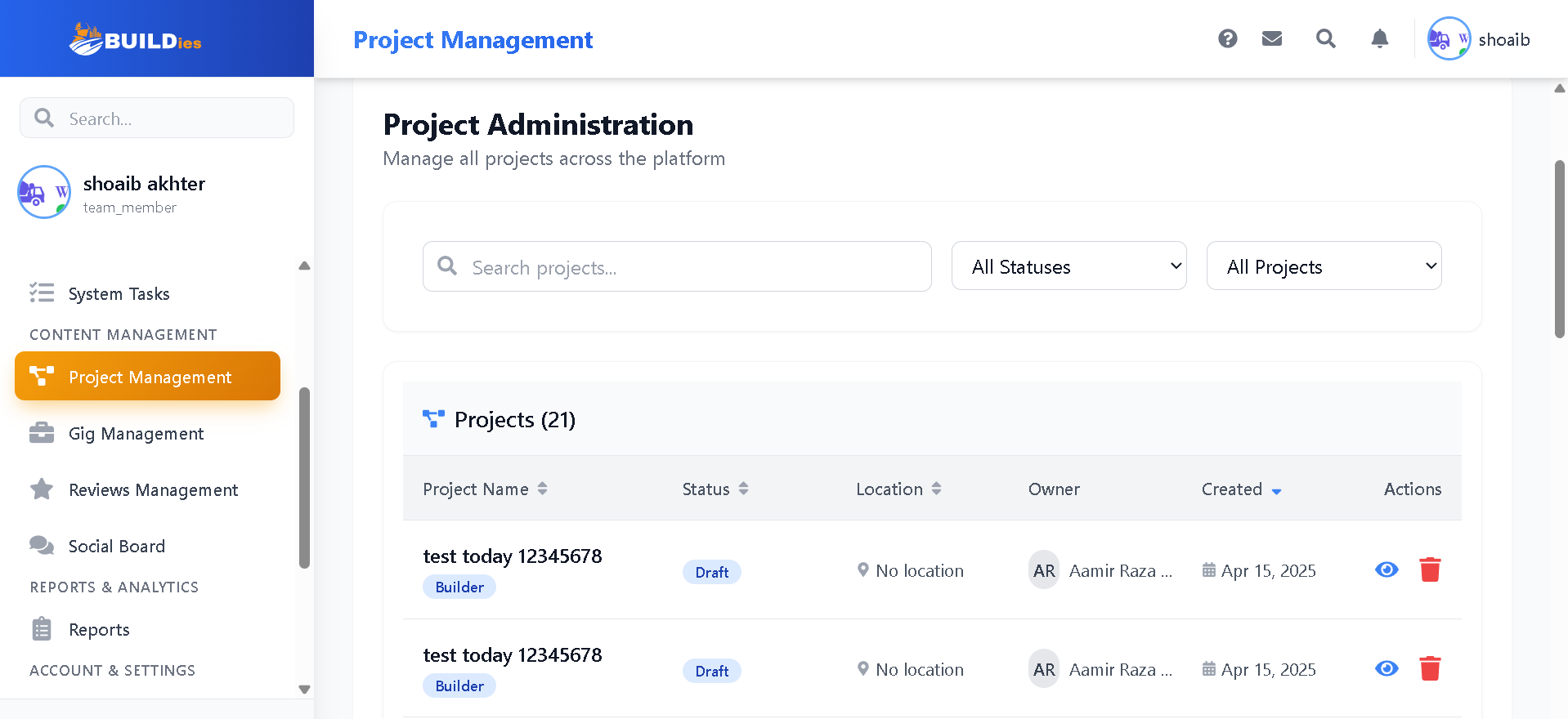


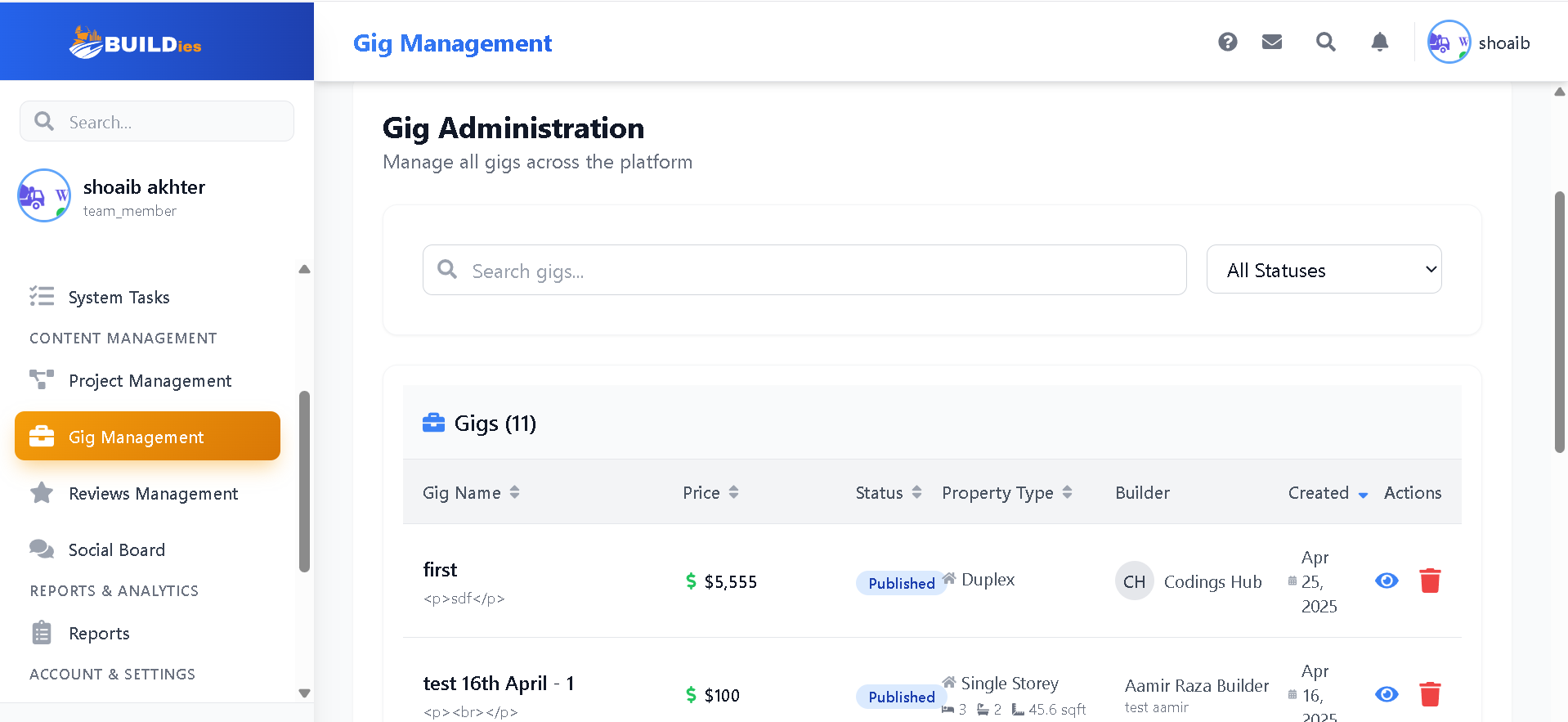


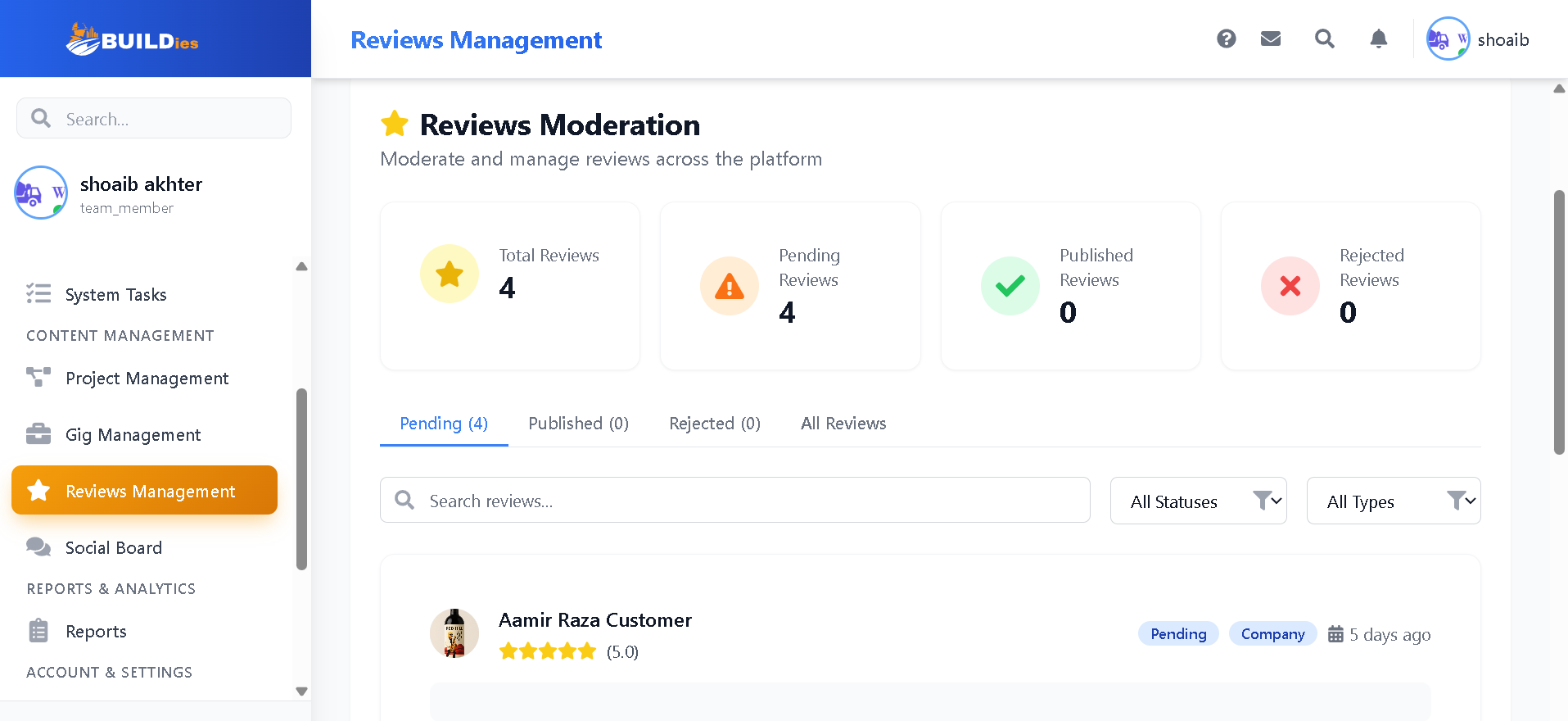


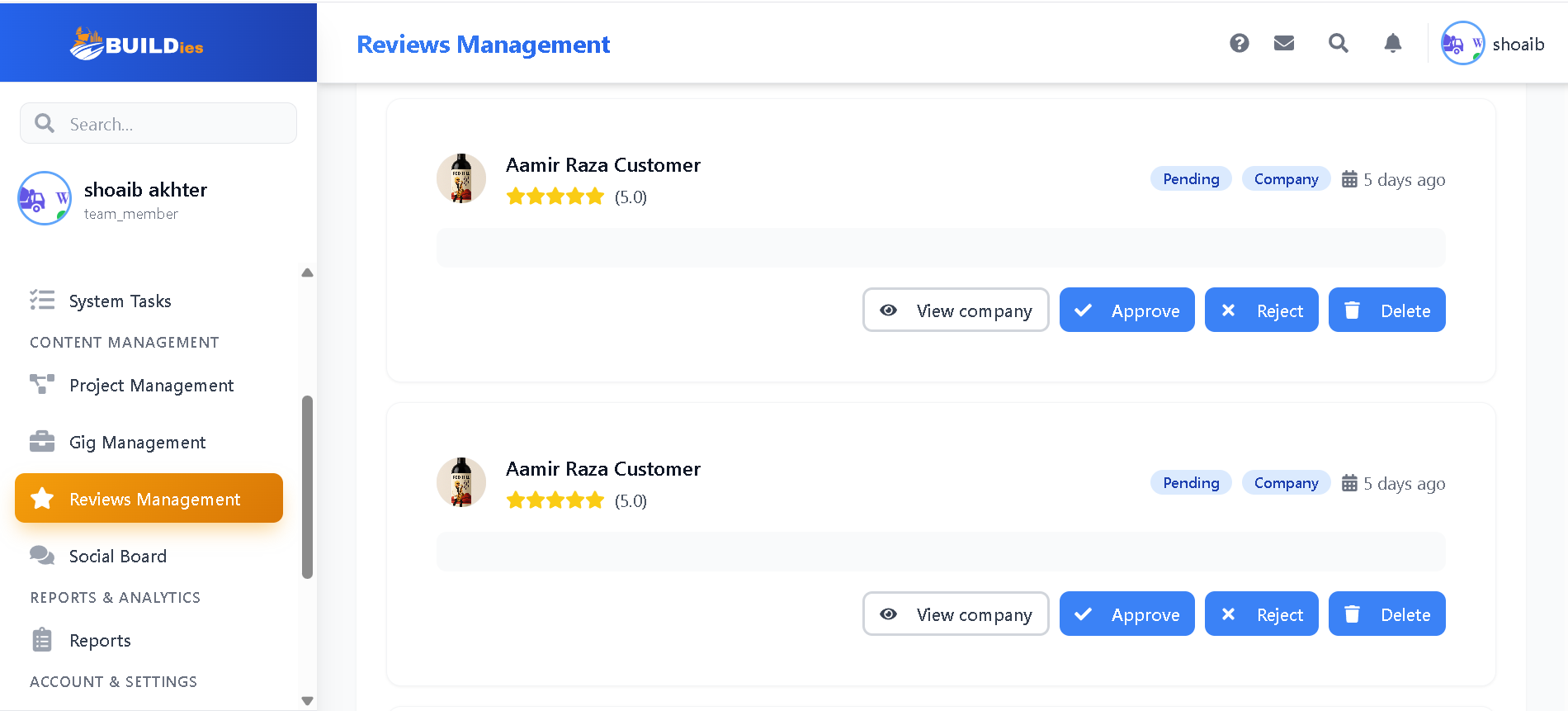


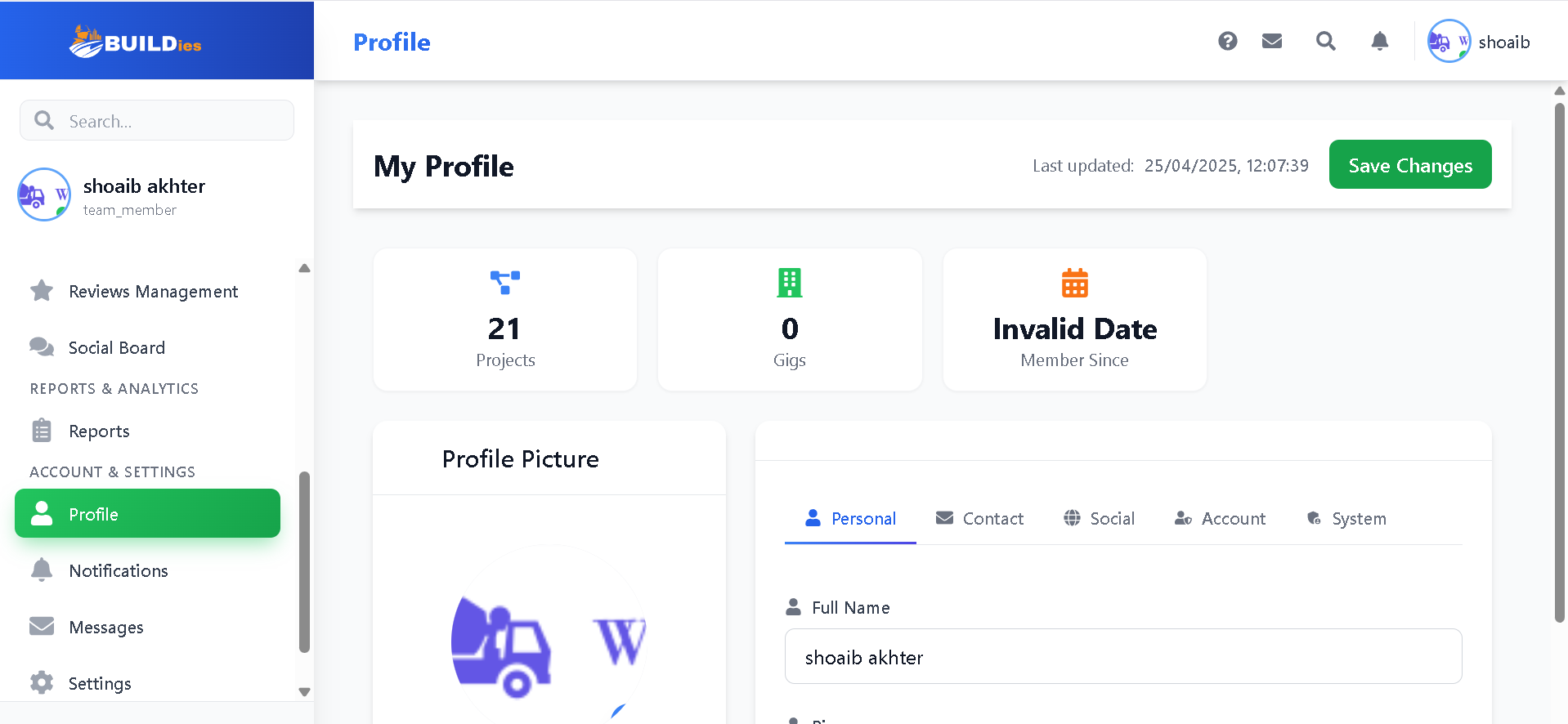


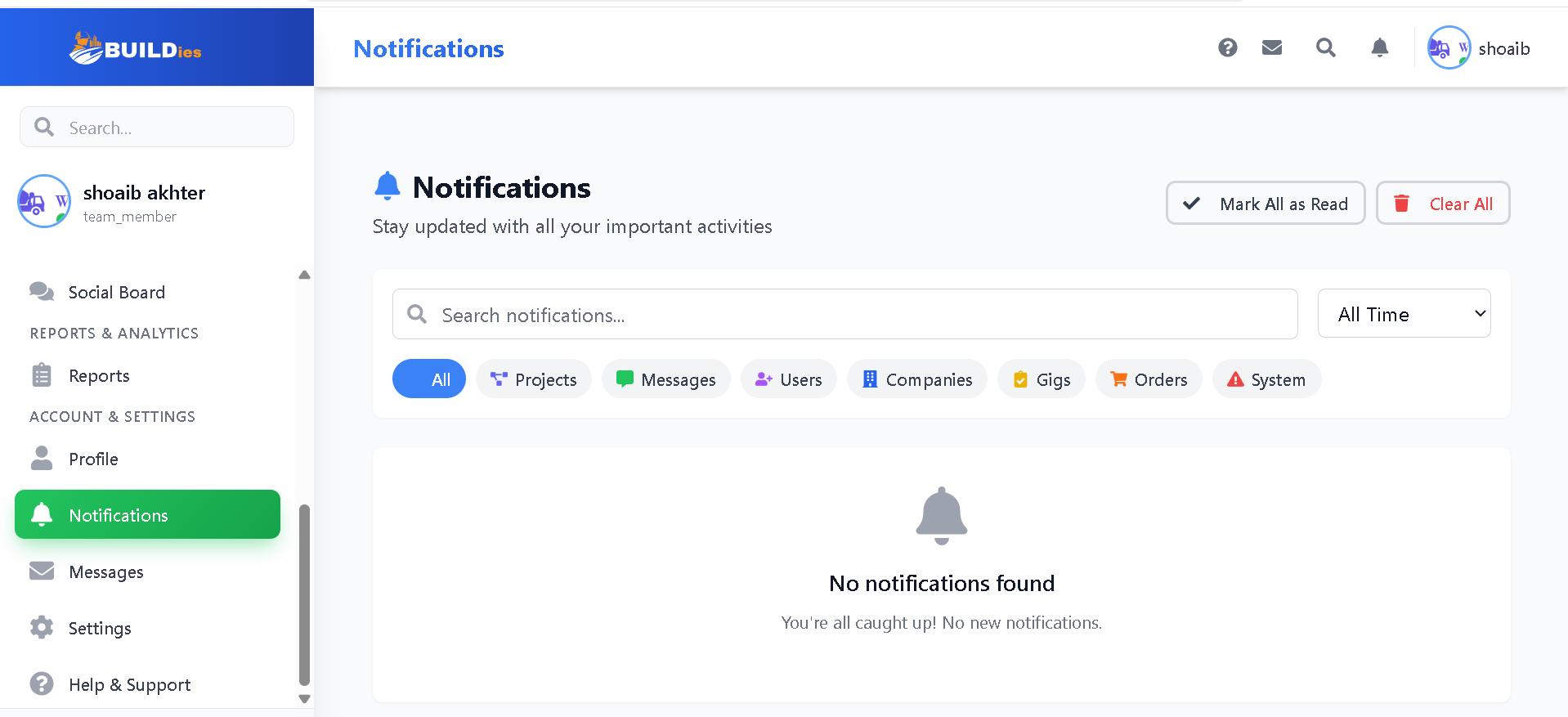


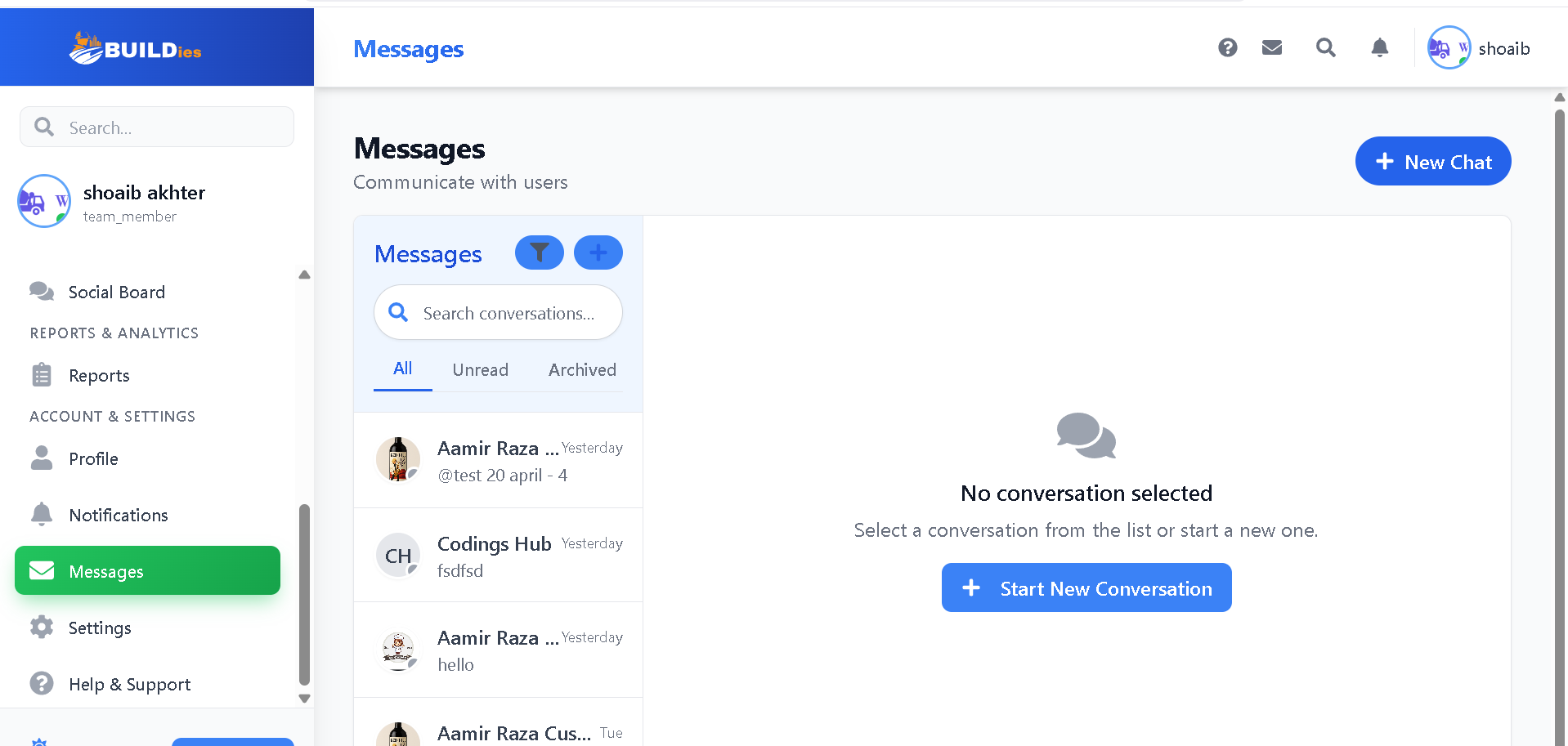












# Implementation

## Frontend:

### Dashboard.jsx:

import React, { useEffect, useMemo, useRef, useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import { openProjectFormAsync } from "../../lib/Project/projectFormSlice";

import { Carousel } from "react-responsive-carousel";

import {

  fetchUsers,

  selectUserLoading,

  selectUserError,

  selectAllUsersData,

} from "../../lib/User/userSlice";

import {

  fetchAllUserProjects,

  selectProjectData,

  selectProjectLoading,

  selectProjectError,

} from "../../lib/Project/projectSlice";

import { Link } from "react-router-dom";

import BuilderGigTabs from "../../Components/Dashboard/Gigs/BuilderGigTabs";

import {

  fetchAllGigs,

  selectGigs,

  selectGigsLoading,

  selectGigsError,

  fetchGigAnalytics,

  selectGigAnalytics,

} from "../../lib/Gigs/gigsSlice";

import BuilderOverview from "../../Components/Dashboard/Overview/BuilderOverview";

import CustomerOverview from "../../Components/Dashboard/Overview/CustomerOverview";

import StatisticsCards from "../../Components/Dashboard/StatisticsCards";

import RecentProjects from "../../Components/Dashboard/RecentProjects";

import ProjectCarousel from "../../Components/Dashboard/Projects/ProjectCarousel";

import FeaturedGigs from "../../Components/Dashboard/Gigs/FeaturedGigs";

import MobileSlider from "./MobileSlider";

import DashboardHeader from "./DashboardHeader"; // Separate header component

import LoadingOverlay from "../../Components/Loader/LoadingOverlay";

import {

  FaQuestionCircle,

  FaChartLine,

  FaCalendarAlt,

  FaUsers,

  FaClock,

  FaEye,

  FaHeart,

  FaUserCheck,

  FaMapMarkerAlt,

  FaChartBar,

  FaChartPie,

  FaHome,

  FaListAlt,

  FaInfoCircle,

  FaProjectDiagram,

  FaStar,

  FaDollarSign,

  FaCheckCircle,

} from "react-icons/fa";

import { Tooltip } from "react-tooltip";

import "react-tooltip/dist/react-tooltip.css";

import TaskSummary from "../../Components/Dashboard/Tasks/TaskSummary";

import ProjectProgress from "../../Components/Dashboard/Projects/ProjectProgress";

import NotificationCenter from "../../Components/Dashboard/Notifications/NotificationCenter";

import { calculateGigPrice } from "../../Utils/utils";

import { db } from "../../Firebase/FirebaseConfig";

import { collection, getDocs, query, where } from "firebase/firestore";

const PROPERTY\_TYPE\_COLORS = {

  "Single Storey": "text-blue-600",

  "Double Storey": "text-green-600",

  Other: "text-gray-600",

};

const DashboardIndex = () => {

  const dispatch = useDispatch();

  const [activeTab, setActiveTab] = useState("overview");

  const [currentGigTab, setCurrentGigTab] = useState("MY\_GIGS");

  const currentUser = useSelector((state) => state.auth.currentUser);

  // Shoaib must remove null below in projectData

  const projectData = useSelector(selectProjectData) || null;

  // const userLoading = useSelector(selectUserLoading);

  const projectLoading = useSelector(selectProjectLoading);

  const userError = useSelector(selectUserError);

  const projectError = useSelector(selectProjectError);

  const users = useSelector(state=>state.user.users);

  const gigs = useSelector(selectGigs);

  const gigsLoading = useSelector(selectGigsLoading);

  const gigsError = useSelector(selectGigsError);

  const cardWidthRef = useRef(null);

  const [cardSlideWidth, setCardSlideWidth] = useState(0);

  const analytics = useSelector((state) => state.gigs.analytics);

  const [timeRange, setTimeRange] = useState("week");

  const userProjects = useMemo(() => {

    if (!currentUser) return [];

    if (currentUser.userType.toLowerCase() === "builder") {

      if (!currentUser.companyId) return [];

      const companyUserIds = users

        ?.filter((u) => u.companyId === currentUser.companyId)

        .map((u) => u.uid);

      return projectData?.filter(

        (p) =>

          companyUserIds.includes(p.userId) ||

          companyUserIds.includes(p.assignedId)

      );

    } else if (currentUser.userType.toLowerCase() === "customer") {

      return projectData?.filter(

        (p) => p.userId === currentUser.uid || p.assignedId === currentUser.uid

      );

    }

    return [];

  }, [currentUser, users, projectData]);

  useEffect(() => {

    if (currentUser) {

      dispatch(fetchAllUserProjects());

    }

  }, [currentUser, dispatch]);

  const activeCount = userProjects.filter(

    (project) =>

      project.status.toLowerCase() !== "draft" &&

      project.status.toLowerCase() !== "completed"

  ).length;

  const completedCount = userProjects.filter(

    (project) => project.status.toLowerCase() === "completed"

  ).length;

  const assignedCount = userProjects.filter(

    (project) => project.assignedId

  ).length;

  const statusCards = [

    {

      title: "Active",

      color: "indigo",

      icon: "M9 17v-6a2 2 0 012-2h2a2 2 0 012 2v6m-6 0h6",

      count: activeCount,

    },

    {

      title: "Completed",

      color: "green",

      icon: "M5 13l4 4L19 7",

      count: completedCount,

    },

    {

      title: "Assigned",

      color: "yellow",

      icon: "M8 12h.01M12 12h.01M16 12h.01M21 12c0 4.418-3.582 8-8 8a8.001 8.001 0 01-8-8c0-4.418 3.582-8 8-8a8.001 8.001 0 018 8z",

      count: assignedCount,

    },

  ];

  const statusCardsData = statusCards;

  const trendingProjects = [...userProjects]

    .sort((a, b) => new Date(b.updatedAt) - new Date(a.updatedAt))

    .slice(0, 5);

  // Mock analytics data - replace with real data from your backend

  const analyticsData = {

    activeProjects:

      userProjects?.filter((p) => p.status === "active").length || 0,

    teamMembers: users?.length || 0,

    completedTasks: userProjects?.reduce(

      (acc, project) =>

        acc +

        (project.tasks?.filter((task) => task.status === "completed").length ||

          0),

      0

    ),

    avgResponseTime: "2.5h",

  };

  // Mock notifications - replace with real notifications from your backend

  const notifications = userProjects?.slice(0, 5)?.map((project) => ({

    type: "info",

    title: `Project Update: ${project.title}`,

    message: `Status changed to ${project.status}`,

    timestamp: project.updatedAt,

  }));

  useEffect(() => {

    if (cardWidthRef.current) {

      setCardSlideWidth(cardWidthRef.current.offsetWidth);

    }

  }, [statusCardsData]);

  // Get available tabs based on user type

  const getAvailableTabs = () => {

    const commonTabs = ["overview", "projects", "tasks"];

    switch (currentUser?.userType) {

      case "Admin":

        return [...commonTabs, "analytics", "users", "platform"];

      case "Builder":

        return [...commonTabs, "analytics", "team", "gigs"];

      case "Customer":

        return [...commonTabs, "payments", "builders"];

      default:

        return commonTabs;

    }

  };

  // Add this near your other state declarations

  const [scrollPosition, setScrollPosition] = useState(0);

  const tabsContainerRef = useRef(null);

  // Add this function to handle scroll

  const handleTabScroll = (direction) => {

    if (tabsContainerRef.current) {

      const scrollAmount = 120; // Adjust this value based on your tab width

      const newPosition =

        direction === "right"

          ? scrollPosition + scrollAmount

          : scrollPosition - scrollAmount;

      tabsContainerRef.current.scrollTo({

        left: newPosition,

        behavior: "smooth",

      });

      setScrollPosition(newPosition);

    }

  };

  // Update the tabs navigation JSX

  const renderTabsNavigation = () => (

    <div className="border-b border-gray-200 dark:border-gray-700">

      {/\* Desktop view \*/}

      <nav className="hidden md:flex mb-6 flex-wrap justify-start gap-2 overflow-x-auto pb-2 px-4">

        {getAvailableTabs().map((tab) => (

          <button

            key={tab}

            onClick={() => setActiveTab(tab)}

            className={`

                                whitespace-nowrap

                                px-4 py-2

                                rounded-lg

                                text-sm

                                transition-colors

                                duration-200

                                min-w-[120px]

                                ${

                                  activeTab === tab

                                    ? "bg-blue-600 dark:bg-blue-500 text-white"

                                    : "bg-gray-200 dark:bg-gray-700 text-gray-700 dark:text-gray-200 hover:bg-gray-300 dark:hover:bg-gray-600"

                                }

                            `}

          >

            {tab.charAt(0).toUpperCase() + tab.slice(1)}

          </button>

        ))}

      </nav>

      {/\* Mobile view with slider \*/}

      <div className="md:hidden relative px-2">

        <div className="flex items-center gap-2">

          <button

            onClick={() => handleTabScroll("left")}

            className="p-2 bg-gray-100 dark:bg-gray-700 text-gray-700 dark:text-gray-200 rounded-full shadow-md z-10 hover:bg-gray-200 dark:hover:bg-gray-600 transition-colors duration-200"

            aria-label="Scroll left"

          >

            <svg

              className="w-4 h-4"

              fill="none"

              stroke="currentColor"

              viewBox="0 0 24 24"

            >

              <path

                strokeLinecap="round"

                strokeLinejoin="round"

                strokeWidth={2}

                d="M15 19l-7-7 7-7"

              />

            </svg>

          </button>

          <div

            ref={tabsContainerRef}

            className="flex overflow-x-auto scrollbar-hide scroll-smooth py-4 gap-2 flex-1"

            style={{ scrollBehavior: "smooth" }}

          >

            {getAvailableTabs().map((tab) => (

              <button

                key={tab}

                onClick={() => setActiveTab(tab)}

                className={`

                                        flex-shrink-0

                                        whitespace-nowrap

                                        px-4 py-2

                                        rounded-lg

                                        text-sm

                                        transition-colors

                                        duration-200

                                        min-w-[100px]

                                        ${

                                          activeTab === tab

                                            ? "bg-blue-600 dark:bg-blue-500 text-white shadow-md"

                                            : "bg-gray-200 dark:bg-gray-700 text-gray-700 dark:text-gray-200 hover:bg-gray-300 dark:hover:bg-gray-600"

                                        }

                                    `}

              >

                {tab.charAt(0).toUpperCase() + tab.slice(1)}

              </button>

            ))}

          </div>

          <button

            onClick={() => handleTabScroll("right")}

            className="p-2 bg-gray-100 dark:bg-gray-700 text-gray-700 dark:text-gray-200 rounded-full shadow-md z-10 hover:bg-gray-200 dark:hover:bg-gray-600 transition-colors duration-200"

            aria-label="Scroll right"

          >

            <svg

              className="w-4 h-4"

              fill="none"

              stroke="currentColor"

              viewBox="0 0 24 24"

            >

              <path

                strokeLinecap="round"

                strokeLinejoin="round"

                strokeWidth={2}

                d="M9 5l7 7-7 7"

              />

            </svg>

          </button>

        </div>

      </div>

    </div>

  );

  // Add these new state and effect hooks at the top of your component

  const [teamMetrics, setTeamMetrics] = useState({

    totalRevenue: 0,

    onTime: 0,

    clientSatisfaction: 0,

    teamUtilization: 0,

  });

  const refreshMetrics = async () => {

    if (!currentUser?.companyId) return;

    try {

      // Fetch fresh data from Firestore

      const projectsRef = collection(db, "projects");

      const projectQuery = query(

        projectsRef,

        where("companyId", "==", currentUser.companyId)

      );

      const projectSnapshot = await getDocs(projectQuery);

      const projects = projectSnapshot.docs.map((doc) => ({

        id: doc.id,

        ...doc.data(),

      }));

      // Recalculate metrics

      const revenue = projects.reduce((total, project) => {

        return (

          total + (project.status === "completed" ? project.budget || 0 : 0)

        );

      }, 0);

      const completedProjects = projects.filter(

        (p) => p.status === "completed"

      );

      const onTimeProjects = completedProjects.filter((p) => {

        const completionDate = new Date(p.completedAt);

        const deadlineDate = new Date(p.deadline);

        return completionDate <= deadlineDate;

      });

      const onTimeRate =

        completedProjects.length > 0

          ? ((onTimeProjects.length / completedProjects.length) \* 100).toFixed(

              1

            )

          : 0;

      const reviews = projects

        .flatMap((p) => p.reviews || [])

        .map((r) => r.rating || 0);

      const avgSatisfaction =

        reviews.length > 0

          ? (reviews.reduce((a, b) => a + b, 0) / reviews.length).toFixed(1)

          : 0;

      // Calculate team utilization

      const tasksRef = collection(db, "tasks");

      const taskQuery = query(

        tasksRef,

        where("companyId", "==", currentUser.companyId)

      );

      const taskSnapshot = await getDocs(taskQuery);

      const tasks = taskSnapshot.docs.map((doc) => doc.data());

      const totalTeamMembers = users?.filter(

        (u) => u.companyId === currentUser.companyId

      ).length;

      const assignedTasks = tasks.filter(

        (t) => t.assignedTo && t.status !== "completed"

      );

      const utilization =

        totalTeamMembers > 0

          ? ((assignedTasks.length / totalTeamMembers) \* 100).toFixed(1)

          : 0;

      // Update metrics state

      setTeamMetrics({

        totalRevenue: revenue.toLocaleString("en-US", {

          style: "currency",

          currency: "USD",

        }),

        onTime: `${onTimeRate}%`,

        clientSatisfaction: `${avgSatisfaction}/5`,

        teamUtilization: `${utilization}%`,

      });

    } catch (error) {

      console.error("Error refreshing metrics:", error);

      // Optionally show an error toast/notification here

    }

  };

  // Add an effect to refresh metrics when timeRange changes

  useEffect(() => {

    refreshMetrics();

  }, [timeRange, currentUser?.companyId]);

  // Get analytics data based on user type

  const getAnalyticsData = () => {

    switch (currentUser?.userType) {

      case "Admin":

        return {

          totalUsers: users?.length,

          totalBuilders: users?.filter((u) => u.userType === "Builder")

            .length,

          totalCustomers: users?.filter((u) => u.userType === "Customer")

            .length,

          totalProjects: projectData.length,

          platformMetrics: {

            activeUsers: users?.filter((u) => u.status === "Active")

              .length,

            totalRevenue: "$50,000", // Replace with actual data

            growthRate: "15%",

            userRetention: "85%",

          },

        };

      case "Builder":

        const companyUsers = users?.filter(

          (u) => u.companyId === currentUser.companyId

        );

        const companyProjects = userProjects;

        return {

          teamSize: companyUsers?.length,

          activeProjects: companyProjects.filter((p) => p.status !== "draft")

            .length,

          completedProjects: companyProjects.filter(

            (p) => p.status === "completed"

          ).length,

          totalRevenue: teamMetrics.totalRevenue,

          projectMetrics: {

            onTime: teamMetrics.onTime,

            clientSatisfaction: teamMetrics.clientSatisfaction,

            teamUtilization: teamMetrics.teamUtilization,

          },

        };

      case "Customer":

        const customerProjects = userProjects;

        const completedProjects = customerProjects.filter(

          (p) => p.status === "completed"

        ).length;

        const activeProjects = customerProjects.filter(

          (p) => p.status !== "completed" && p.status !== "draft"

        ).length;

        const totalSpent = customerProjects.reduce(

          (acc, project) => acc + (project.totalCost || 0),

          0

        );

        return {

          totalProjects: customerProjects.length,

          completedProjects,

          activeProjects,

          totalSpent: totalSpent.toLocaleString("en-AU", {

            style: "currency",

            currency: "AUD",

          }),

          projectMetrics: {

            onTrack: customerProjects.filter((p) => !isProjectDelayed(p))

              .length,

            delayed: customerProjects.filter((p) => isProjectDelayed(p)).length,

            satisfaction: calculateAverageSatisfaction(customerProjects),

          },

        };

      default:

        return {};

    }

  };

  const renderUserTypeSpecificContent = () => {

    switch (currentUser?.userType) {

      case "Admin":

        return (

          <div className="grid grid-cols-1 lg:grid-cols-2 gap-4">

            <div className="bg-white p-4 rounded-lg shadow">

              <h3 className="text-lg font-semibold mb-3">Platform Overview</h3>

              <div className="space-y-2">

                <p>Total Users: {getAnalyticsData().totalUsers}</p>

                <p>Total Builders: {getAnalyticsData().totalBuilders}</p>

                <p>Total Customers: {getAnalyticsData().totalCustomers}</p>

                <p>Total Projects: {getAnalyticsData().totalProjects}</p>

              </div>

            </div>

            <div className="bg-white p-4 rounded-lg shadow">

              <h3 className="text-lg font-semibold mb-3">Platform Metrics</h3>

              <div className="space-y-2">

                <p>

                  Active Users: {getAnalyticsData().platformMetrics.activeUsers}

                </p>

                <p>

                  Total Revenue:{" "}

                  {getAnalyticsData().platformMetrics.totalRevenue}

                </p>

                <p>

                  Growth Rate: {getAnalyticsData().platformMetrics.growthRate}

                </p>

                <p>

                  User Retention:{" "}

                  {getAnalyticsData().platformMetrics.userRetention}

                </p>

              </div>

            </div>

          </div>

        );

      case "Builder":

        const companyUsers = users?.filter(

          (u) => u.companyId === currentUser.companyId

        );

        const companyProjects = userProjects;

        const analytics = getAnalyticsData();

        return (

          <div className="space-y-6">

            {/\* Company Overview Cards \*/}

            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">

              <div className="bg-gradient-to-br from-blue-50 to-blue-100 dark:from-blue-900/50 dark:to-blue-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-blue-600 dark:text-blue-400">

                      Team Size

                    </p>

                    <h3 className="text-2xl font-bold text-blue-700 dark:text-blue-300 mt-1">

                      {analytics.teamSize}

                    </h3>

                  </div>

                  <div className="p-3 bg-blue-100 dark:bg-blue-800 rounded-lg">

                    <FaUsers className="w-6 h-6 text-blue-600 dark:text-blue-400" />

                  </div>

                </div>

                <div className="mt-4 flex items-center text-sm">

                  <FaChartLine className="w-4 h-4 text-green-500 mr-1" />

                  <span className="text-green-500 font-medium">+12%</span>

                  <span className="text-gray-600 dark:text-gray-400 ml-2">

                    from last month

                  </span>

                </div>

              </div>

              <div className="bg-gradient-to-br from-green-50 to-green-100 dark:from-green-900/50 dark:to-green-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-green-600 dark:text-green-400">

                      Active Projects

                    </p>

                    <h3 className="text-2xl font-bold text-green-700 dark:text-green-300 mt-1">

                      {analytics.activeProjects}

                    </h3>

                  </div>

                  <div className="p-3 bg-green-100 dark:bg-green-800 rounded-lg">

                    <FaProjectDiagram className="w-6 h-6 text-green-600 dark:text-green-400" />

                  </div>

                </div>

                <div className="mt-4">

                  <div className="flex items-center justify-between text-sm">

                    <span className="text-gray-600 dark:text-gray-400">

                      Completion Rate

                    </span>

                    <span className="text-green-600 dark:text-green-400 font-medium">

                      {analytics.projectMetrics.onTime}

                    </span>

                  </div>

                  <div className="mt-2 h-2 bg-green-200 dark:bg-green-700 rounded-full">

                    <div

                      className="h-full bg-green-500 rounded-full"

                      style={{ width: analytics.projectMetrics.onTime }}

                    />

                  </div>

                </div>

              </div>

              <div className="bg-gradient-to-br from-purple-50 to-purple-100 dark:from-purple-900/50 dark:to-purple-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-purple-600 dark:text-purple-400">

                      Client Satisfaction

                    </p>

                    <h3 className="text-2xl font-bold text-purple-700 dark:text-purple-300 mt-1">

                      {analytics.projectMetrics.clientSatisfaction}

                    </h3>

                  </div>

                  <div className="p-3 bg-purple-100 dark:bg-purple-800 rounded-lg">

                    <FaHeart className="w-6 h-6 text-purple-600 dark:text-purple-400" />

                  </div>

                </div>

                <div className="mt-4 flex items-center space-x-1 text-sm">

                  {[1, 2, 3, 4, 5].map((star) => (

                    <FaStar

                      key={star}

                      className={`w-4 h-4 ${

                        star <=

                        parseFloat(analytics.projectMetrics.clientSatisfaction)

                          ? "text-yellow-400"

                          : "text-gray-300 dark:text-gray-600"

                      }`}

                    />

                  ))}

                </div>

              </div>

              <div className="bg-gradient-to-br from-orange-50 to-orange-100 dark:from-orange-900/50 dark:to-orange-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-orange-600 dark:text-orange-400">

                      Revenue

                    </p>

                    <h3 className="text-2xl font-bold text-orange-700 dark:text-orange-300 mt-1">

                      {analytics.totalRevenue}

                    </h3>

                  </div>

                  <div className="p-3 bg-orange-100 dark:bg-orange-800 rounded-lg">

                    <FaChartBar className="w-6 h-6 text-orange-600 dark:text-orange-400" />

                  </div>

                </div>

                <div className="mt-4">

                  <div className="flex items-center text-sm">

                    <span className="text-orange-600 dark:text-orange-400 font-medium">

                      {analytics.projectMetrics.teamUtilization} Utilization

                    </span>

                    <FaInfoCircle

                      className="ml-2 w-4 h-4 text-gray-400 cursor-help"

                      data-tooltip-id="revenue-tooltip"

                      data-tooltip-content="Team resource utilization rate"

                    />

                  </div>

                </div>

              </div>

            </div>

            {/\* Project Timeline and Team Performance \*/}

            <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

              <div className="bg-white dark:bg-gray-800 rounded-xl shadow-sm p-6">

                <div className="flex justify-between items-center mb-6">

                  <h3 className="text-lg font-semibold text-gray-800 dark:text-gray-200">

                    Team Performance

                  </h3>

                  <div className="flex items-center space-x-2">

                    <select

                      className="text-sm border rounded-md p-1 dark:bg-gray-700 dark:border-gray-600"

                      onChange={(e) => setTimeRange(e.target.value)}

                    >

                      <option value="week">This Week</option>

                      <option value="month">This Month</option>

                      <option value="quarter">This Quarter</option>

                    </select>

                  </div>

                </div>

                <TeamPerformanceChart metrics={teamMetrics} />

                <div className="mt-4 pt-4 border-t dark:border-gray-700">

                  <div className="flex items-center justify-between text-sm text-gray-600 dark:text-gray-400">

                    <span>Last updated: {new Date().toLocaleTimeString()}</span>

                    <button

                      className="text-blue-600 dark:text-blue-400 hover:underline"

                      onClick={() => refreshMetrics()}

                    >

                      Refresh

                    </button>

                  </div>

                </div>

              </div>

            </div>

            <Tooltip id="revenue-tooltip" />

          </div>

        );

      case "Customer":

        const customerAnalytics = getAnalyticsData();

        return (

          <div className="space-y-6">

            {/\* Customer Overview Cards \*/}

            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">

              {/\* Total Projects Card \*/}

              <div className="bg-gradient-to-br from-blue-50 to-blue-100 dark:from-blue-900/50 dark:to-blue-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-blue-600 dark:text-blue-400">

                      Total Projects

                    </p>

                    <h3 className="text-2xl font-bold text-blue-700 dark:text-blue-300 mt-1">

                      {customerAnalytics.totalProjects}

                    </h3>

                  </div>

                  <div className="p-3 bg-blue-100 dark:bg-blue-800 rounded-lg">

                    <FaProjectDiagram className="w-6 h-6 text-blue-600 dark:text-blue-400" />

                  </div>

                </div>

              </div>

              {/\* Active Projects Card \*/}

              <div className="bg-gradient-to-br from-green-50 to-green-100 dark:from-green-900/50 dark:to-green-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-green-600 dark:text-green-400">

                      Active Projects

                    </p>

                    <h3 className="text-2xl font-bold text-green-700 dark:text-green-300 mt-1">

                      {customerAnalytics.activeProjects}

                    </h3>

                  </div>

                  <div className="p-3 bg-green-100 dark:bg-green-800 rounded-lg">

                    <FaClock className="w-6 h-6 text-green-600 dark:text-green-400" />

                  </div>

                </div>

              </div>

              {/\* Total Spent Card \*/}

              <div className="bg-gradient-to-br from-purple-50 to-purple-100 dark:from-purple-900/50 dark:to-purple-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-purple-600 dark:text-purple-400">

                      Total Spent

                    </p>

                    <h3 className="text-2xl font-bold text-purple-700 dark:text-purple-300 mt-1">

                      {customerAnalytics.totalSpent}

                    </h3>

                  </div>

                  <div className="p-3 bg-purple-100 dark:bg-purple-800 rounded-lg">

                    <FaDollarSign className="w-6 h-6 text-purple-600 dark:text-purple-400" />

                  </div>

                </div>

              </div>

              {/\* Project Status Card \*/}

              <div className="bg-gradient-to-br from-amber-50 to-amber-100 dark:from-amber-900/50 dark:to-amber-800/50 p-6 rounded-xl shadow-sm">

                <div className="flex items-center justify-between">

                  <div>

                    <p className="text-sm font-medium text-amber-600 dark:text-amber-400">

                      Completed Projects

                    </p>

                    <h3 className="text-2xl font-bold text-amber-700 dark:text-amber-300 mt-1">

                      {customerAnalytics.completedProjects}

                    </h3>

                  </div>

                  <div className="p-3 bg-amber-100 dark:bg-amber-800 rounded-lg">

                    <FaCheckCircle className="w-6 h-6 text-amber-600 dark:text-amber-400" />

                  </div>

                </div>

              </div>

            </div>

          </div>

        );

      default:

        return null;

    }

  };

### MarketPlace (Gigs):

import React, { useState, useEffect, useMemo } from "react";

import { useSelector, useDispatch } from "react-redux";

import { fetchAllGigs, selectGigs, deleteGig } from "../../lib/Gigs/gigsSlice";

import { fetchUsers, selectAllUsersData } from "../../lib/User/userSlice";

import {

  fetchAllCompanies,

  selectCompanies,

} from "../../lib/Company/companySlice";

import {

  MdDelete,

  MdSwapVert,

  MdArrowUpward,

  MdArrowDownward,

} from "react-icons/md";

import Pagination from "../Dashboard/Layout/Pagination";

import toast from "react-hot-toast";

import dayjs from "dayjs";

import DeletePopup from "../../Components/Dashboard/Layout/DeletePopup";

const ITEMS\_PER\_PAGE = 10;

const ManageGigs = () => {

  const dispatch = useDispatch();

  const currentUser = useSelector((state) => state.auth.currentUser);

  const gigs = useSelector(selectGigs) || [];

  const allUsersData = useSelector(selectAllUsersData) || [];

  const companies = useSelector(selectCompanies) || [];

  // State hooks.

  const [loading, setLoading] = useState(true);

  const [filterCompany, setFilterCompany] = useState("");

  const [currentPage, setCurrentPage] = useState(1);

  const [showDeletePopup, setShowDeletePopup] = useState(false);

  const [gigToDelete, setGigToDelete] = useState(null);

  const [sortConfig, setSortConfig] = useState({

    key: null,

    direction: "asc",

  });

  const [showLoader, setShowLoader] = useState(false);

  const handleSort = (key) => {

    setSortConfig((prev) => ({

      key,

      direction: prev.key === key && prev.direction === "asc" ? "desc" : "asc",

    }));

  };

  const filteredGigs = useMemo(() => {

    return gigs.filter((gig) => {

      if (filterCompany.trim() === "") return true;

      if (gig.companyId) {

        const company = companies.find((c) => c.id === gig.companyId);

        if (company && company.companyName) {

          return company.companyName

            .toLowerCase()

            .includes(filterCompany.toLowerCase());

        }

      }

      return false;

    });

  }, [gigs, companies, filterCompany]);

  const sortedAndFilteredGigs = useMemo(() => {

    let sortedGigs = [...filteredGigs];

    if (sortConfig.key) {

      sortedGigs.sort((a, b) => {

        let aValue, bValue;

        switch (sortConfig.key) {

          case "name":

            aValue = a.Introduction?.gigName || "";

            bValue = b.Introduction?.gigName || "";

            break;

          case "creator":

            const creatorA = allUsersData.find((u) => u.uid === a.builderId);

            const creatorB = allUsersData.find((u) => u.uid === b.builderId);

            aValue = creatorA ? creatorA.fullName || creatorA.email : "";

            bValue = creatorB ? creatorB.fullName || creatorB.email : "";

            break;

          case "company":

            const companyA = companies.find((c) => c.id === a.companyId);

            const companyB = companies.find((c) => c.id === b.companyId);

            aValue = companyA?.companyName || "";

            bValue = companyB?.companyName || "";

            break;

          case "createdAt":

            aValue = a.createdAt

              ? new Date(a.createdAt.toDate?.() || a.createdAt).getTime()

              : 0;

            bValue = b.createdAt

              ? new Date(b.createdAt.toDate?.() || b.createdAt).getTime()

              : 0;

            break;

          case "updatedAt":

            aValue = a.updatedAt

              ? new Date(a.updatedAt.toDate?.() || a.updatedAt).getTime()

              : 0;

            bValue = b.updatedAt

              ? new Date(b.updatedAt.toDate?.() || b.updatedAt).getTime()

              : 0;

            break;

          case "status":

            aValue = a.status || "";

            bValue = b.status || "";

            break;

          default:

            return 0;

        }

        const comparison = aValue > bValue ? 1 : aValue < bValue ? -1 : 0;

        return sortConfig.direction === "asc" ? comparison : -comparison;

      });

    }

    return sortedGigs;

  }, [filteredGigs, sortConfig, allUsersData, companies]);

  useEffect(() => {

    const fetchData = async () => {

      try {

        await dispatch(fetchAllGigs());

        await dispatch(fetchUsers());

        await dispatch(fetchAllCompanies());

      } catch (error) {

        toast.error("Error fetching data");

      } finally {

        setLoading(false);

      }

    };

    fetchData();

  }, [dispatch]);

  const totalItems = sortedAndFilteredGigs.length;

  const totalPages = Math.ceil(totalItems / ITEMS\_PER\_PAGE);

  const currentItems = sortedAndFilteredGigs.slice(

    (currentPage - 1) \* ITEMS\_PER\_PAGE,

    currentPage \* ITEMS\_PER\_PAGE

  );

  // Delete confirmation logic.

  const confirmDeleteGig = async () => {

    if (!gigToDelete) return;

    setShowLoader(true);

    try {

      await dispatch(deleteGig({ gigId: gigToDelete })).unwrap(); // Fix: Pass object with gigId property

      toast.success("Gig deleted successfully");

      setShowDeletePopup(false);

      setGigToDelete(null);

      await dispatch(fetchAllGigs()); // Refresh the gigs list after deletion

    } catch (error) {

      toast.error(`Failed to delete gig: ${error.message}`);

    } finally {

      setShowLoader(false);

    }

  };

  // Ensure only Admin users access this page.

  if (currentUser?.userType !== "Admin") {

    return (

      <div className="p-4">

        <h2 className="text-xl font-bold">Access Denied</h2>

        <p>You do not have permission to view this page.</p>

      </div>

    );

  }

  const handleDeleteClick = (gigId) => {

    if (!gigId) return; // Add validation

    setGigToDelete(gigId);

    setShowDeletePopup(true);

  };

  // Modify the renderSortIcon function

  const renderSortIcon = (key) => {

    if (sortConfig.key === key) {

      return sortConfig.direction === "asc" ? (

        <MdArrowUpward className="inline ml-1 text-purple-600" />

      ) : (

        <MdArrowDownward className="inline ml-1 text-purple-600" />

      );

    }

    return <MdSwapVert className="inline ml-1 text-gray-400" />;

  };

  // Update the table headers section

  const tableHeaders = [

    { key: "name", label: "Name", sortable: true },

    { key: "creator", label: "Created By", sortable: true },

    { key: "company", label: "Company Name", sortable: true },

    { key: "createdAt", label: "Created At", sortable: true },

    { key: "updatedAt", label: "Updated At", sortable: true },

    { key: "status", label: "Status", sortable: true },

    { key: "action", label: "Action", sortable: false },

  ];

  return (

    <div className="w-full h-full">

      <div className="max-w-7xl mx-auto px-2 sm:px-4 py-2 sm:py-4">

        {/\* Header Section \*/}

        <div className="bg-white dark:bg-darkBackground rounded-lg shadow p-4 mb-4">

          <h1 className="text-2xl font-bold mb-6 text-gray-900 dark:text-white">

            Manage Gigs

          </h1>

          {/\* Filter by Company Name \*/}

          <div className="mb-6">

            <label className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-2">

              Filter by Company Name

            </label>

            <input

              type="text"

              value={filterCompany}

              onChange={(e) => {

                setFilterCompany(e.target.value);

                setCurrentPage(1);

              }}

              placeholder="Enter company name"

              className="w-full sm:w-1/3 p-2 border border-gray-300 dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-purple-500 dark:bg-gray-800 dark:text-white"

            />

          </div>

          {loading ? (

            <div className="flex justify-center items-center h-64">

              <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-purple-500"></div>

            </div>

          ) : (

            <>

              {sortedAndFilteredGigs.length === 0 ? (

                <p className="text-gray-700 dark:text-gray-300">

                  No gigs found.

                </p>

              ) : (

                <div className="overflow-x-auto bg-white dark:bg-gray-800 shadow-sm rounded-lg">

                  <table className="min-w-full divide-y divide-gray-200 dark:divide-gray-700">

                    <thead className="bg-gray-50 dark:bg-gray-700">

                      <tr>

                        {tableHeaders.map((header) => (

                          <th

                            key={header.key}

                            onClick={() =>

                              header.sortable && handleSort(header.key)

                            }

                            className={`px-4 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider ${

                              header.sortable

                                ? "cursor-pointer hover:bg-gray-100 dark:hover:bg-gray-600"

                                : ""

                            } group`}

                          >

                            <div className="flex items-center space-x-1">

                              <span>{header.label}</span>

                              {header.sortable && renderSortIcon(header.key)}

                            </div>

                          </th>

                        ))}

                      </tr>

                    </thead>

                    <tbody className="divide-y divide-gray-200 dark:divide-gray-700">

                      {currentItems.map((gig) => {

                        // Find creator using gig.builderId.

                        const creator = allUsersData.find(

                          (u) => u.uid === gig.builderId

                        );

                        // Get company name from gig data using gig.companyId.

                        const company = gig.companyId

                          ? companies.find((c) => c.id === gig.companyId)

                          : null;

                        const latestUpdate =

                          gig.lastUpdated &&

                          Array.isArray(gig.lastUpdated) &&

                          gig.lastUpdated.length > 0

                            ? gig.lastUpdated.reduce((latest, current) => {

                                const currentTime = current.updatedAt

                                  ? new Date(

                                      current.updatedAt.toDate?.() ||

                                        current.updatedAt

                                    ).getTime()

                                  : 0;

                                const latestTime = latest.updatedAt

                                  ? new Date(

                                      latest.updatedAt.toDate?.() ||

                                        latest.updatedAt

                                    ).getTime()

                                  : 0;

                                return currentTime > latestTime

                                  ? current

                                  : latest;

                              }, gig.lastUpdated[0])

                            : null;

                        return (

                          <tr

                            key={gig.id}

                            className="hover:bg-gray-50 dark:hover:bg-gray-700"

                          >

                            <td className="px-4 py-3 text-sm font-medium text-gray-900 dark:text-gray-100">

                              {gig.Introduction?.gigName || "N/A"}

                            </td>

                            <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                              {creator

                                ? creator.fullName || creator.email

                                : "Unknown"}

                            </td>

                            <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                              {company ? company.companyName : "N/A"}

                            </td>

                            <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                              {gig.createdAt

                                ? dayjs(

                                    gig.createdAt.toDate

                                      ? gig.createdAt.toDate()

                                      : gig.createdAt

                                  ).format("YYYY-MM-DD HH:mm")

                                : "N/A"}

                            </td>

                            <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                              {latestUpdate?.updatedAt

                                ? dayjs(

                                    latestUpdate.updatedAt.toDate

                                      ? latestUpdate.updatedAt.toDate()

                                      : latestUpdate.updatedAt

                                  ).format("YYYY-MM-DD HH:mm")

                                : "N/A"}

                            </td>

                            <td className="px-4 py-3 text-sm">

                              <span

                                className={`px-2 py-1 rounded-full text-xs font-bold ${

                                  gig.status?.toLowerCase() === "published"

                                    ? "bg-green-100 text-green-800"

                                    : gig.status?.toLowerCase() === "draft"

                                    ? "bg-yellow-100 text-yellow-800"

                                    : "bg-gray-100 text-gray-800"

                                }`}

                              >

                                {gig.status || "N/A"}

                              </span>

                            </td>

                            <td className="px-4 py-3 text-sm">

                              <button

                                onClick={() => handleDeleteClick(gig.id)}

                                className="text-red-600 hover:text-red-800 transition-colors duration-200"

                                aria-label="Delete Gig"

                              >

                                <MdDelete size={20} />

                              </button>

                            </td>

                          </tr>

                        );

                      })}

                    </tbody>

                  </table>

                </div>

              )}

              {/\* Pagination \*/}

              {totalPages > 1 && (

                <div className="mt-6">

                  <Pagination

                    currentPage={currentPage}

                    totalPages={totalPages}

                    onPageChange={setCurrentPage}

                  />

                </div>

              )}

            </>

          )}

          {/\* Delete Confirmation Popup \*/}

          {showDeletePopup && (

            <DeletePopup

              title="Delete Gig"

              message="Are you sure you want to delete this gig? This action cannot be undone."

              onCancel={() => {

                setShowDeletePopup(false);

                setGigToDelete(null);

              }}

              onConfirm={confirmDeleteGig}

              showLoader={showLoader}

            />

          )}

        </div>

      </div>

    </div>

  );

};

export default ManageGigs;

export default Classification;

### Projects Component

import React, { useState, useEffect, useMemo } from "react";

import { useSelector, useDispatch } from "react-redux";

import {

  fetchAllProjects,

  deleteProject,

  selectProjectData,

} from "../../lib/Project/projectSlice";

import { fetchUsers, selectAllUsersData } from "../../lib/User/userSlice";

import {

  MdDelete,

  MdSwapVert,

  MdArrowUpward,

  MdArrowDownward,

} from "react-icons/md";

import Pagination from "../Dashboard/Layout/Pagination";

import toast from "react-hot-toast";

import dayjs from "dayjs";

import {

  fetchAllCompanies,

  selectCompanies,

} from "../../lib/Company/companySlice";

import InPageLoader from "../../Components/Loader/InPageLoader";

const ITEMS\_PER\_PAGE = 10;

const ManageProjects = () => {

  const dispatch = useDispatch();

  const currentUser = useSelector((state) => state.auth.currentUser);

  const projects = useSelector(selectProjectData) || [];

  const allUsersData = useSelector(selectAllUsersData) || [];

  const companies = useSelector(selectCompanies) || [];

  const [loading, setLoading] = useState(true);

  const [filterCompany, setFilterCompany] = useState("");

  const [currentPage, setCurrentPage] = useState(1);

  const [showDeletePopup, setShowDeletePopup] = useState(false);

  const [projectToDelete, setProjectToDelete] = useState(null);

  const [showLoader, setShowLoader] = useState(false);

  const [sortConfig, setSortConfig] = useState({

    key: null,

    direction: "asc",

  });

  const handleSort = (key) => {

    setSortConfig((prev) => ({

      key,

      direction: prev.key === key && prev.direction === "asc" ? "desc" : "asc",

    }));

  };

  const getBuilderCompanyName = (project, creator) => {

    if (project.assignedId) {

      const assignedBuilder = allUsersData.find(

        (u) => u.uid === project.assignedId

      );

      if (assignedBuilder && assignedBuilder.companyId) {

        const company = companies.find(

          (c) => c.id === assignedBuilder.companyId

        );

        return company ? company.companyName : "N/A";

      }

    }

    if (creator && creator.userType === "Builder" && creator.companyId) {

      const company = companies.find((c) => c.id === creator.companyId);

      return company ? company.companyName : "N/A";

    }

    return "N/A";

  };

  const filteredProjects = useMemo(() => {

    return projects.filter((project) => {

      if (filterCompany.trim() === "") return true;

      const creator = allUsersData.find((user) => user.uid === project.userId);

      if (creator && creator.userType === "Builder" && creator.companyId) {

        const company = companies.find((c) => c.id === creator.companyId);

        if (company && company.companyName) {

          return company.companyName

            .toLowerCase()

            .includes(filterCompany.toLowerCase());

        }

      }

      return false;

    });

  }, [projects, allUsersData, companies, filterCompany]);

  const getLatestUpdateTime = (lastUpdated) => {

    if (

      !lastUpdated ||

      !Array.isArray(lastUpdated) ||

      lastUpdated.length === 0

    ) {

      return 0;

    }

    return Math.max(

      ...lastUpdated.map((update) =>

        update.updatedAt

          ? new Date(update.updatedAt.toDate?.() || update.updatedAt).getTime()

          : 0

      )

    );

  };

  const sortedAndFilteredProjects = useMemo(() => {

    let sortedProjects = [...filteredProjects];

    if (sortConfig.key) {

      sortedProjects.sort((a, b) => {

        let aValue, bValue;

        switch (sortConfig.key) {

          case "name":

            aValue = a.projectOverview?.projectName || "";

            bValue = b.projectOverview?.projectName || "";

            break;

          case "creator":

            const creatorA = allUsersData.find((u) => u.uid === a.userId);

            const creatorB = allUsersData.find((u) => u.uid === b.userId);

            aValue = creatorA ? creatorA.fullName || creatorA.email : "";

            bValue = creatorB ? creatorB.fullName || creatorB.email : "";

            break;

          case "userType":

            aValue = a.userType || "";

            bValue = b.userType || "";

            break;

          case "company":

            const creatorCompanyA = allUsersData.find(

              (u) => u.uid === a.userId

            );

            const creatorCompanyB = allUsersData.find(

              (u) => u.uid === b.userId

            );

            aValue = getBuilderCompanyName(a, creatorCompanyA);

            bValue = getBuilderCompanyName(b, creatorCompanyB);

            break;

          case "createdAt":

            aValue = a.createdAt

              ? new Date(a.createdAt.toDate?.() || a.createdAt).getTime()

              : 0;

            bValue = b.createdAt

              ? new Date(b.createdAt.toDate?.() || b.createdAt).getTime()

              : 0;

            break;

          case "updatedAt":

            aValue = getLatestUpdateTime(a.lastUpdated);

            bValue = getLatestUpdateTime(b.lastUpdated);

            break;

          case "status":

            aValue = a.status || "";

            bValue = b.status || "";

            break;

          default:

            return 0;

        }

        const comparison = aValue > bValue ? 1 : aValue < bValue ? -1 : 0;

        return sortConfig.direction === "asc" ? comparison : -comparison;

      });

    }

    return sortedProjects;

  }, [filteredProjects, sortConfig, allUsersData]);

  useEffect(() => {

    const fetchData = async () => {

      try {

        await dispatch(fetchAllProjects());

        await dispatch(fetchUsers());

        await dispatch(fetchAllCompanies());

      } catch (error) {

        toast.error("Error fetching data");

      } finally {

        setLoading(false);

      }

    };

    fetchData();

  }, [dispatch]);

  // Update pagination to use sortedAndFilteredProjects

  const totalItems = sortedAndFilteredProjects.length;

  const totalPages = Math.ceil(totalItems / ITEMS\_PER\_PAGE);

  const currentItems = sortedAndFilteredProjects.slice(

    (currentPage - 1) \* ITEMS\_PER\_PAGE,

    currentPage \* ITEMS\_PER\_PAGE

  );

  const renderSortIcon = (key) => {

    if (sortConfig.key === key) {

      return sortConfig.direction === "asc" ? (

        <MdArrowUpward className="inline ml-1 text-purple-600" />

      ) : (

        <MdArrowDownward className="inline ml-1 text-purple-600" />

      );

    }

    return <MdSwapVert className="inline ml-1 text-gray-400" />;

  };

  const tableHeaders = [

    { key: "name", label: "Name", sortable: true },

    { key: "creator", label: "Created By", sortable: true },

    { key: "userType", label: "User Type", sortable: true },

    { key: "company", label: "Builder Company", sortable: true },

    { key: "createdAt", label: "Created At", sortable: true },

    { key: "updatedAt", label: "Updated At", sortable: true },

    { key: "status", label: "Status", sortable: true },

    { key: "action", label: "Action", sortable: false },

  ];

  // Ensure only Admin users access this page.

  if (currentUser?.userType !== "Admin") {

    return (

      <div className="p-4">

        <h2 className="text-xl font-bold">Access Denied</h2>

        <p>You do not have permission to view this page.</p>

      </div>

    );

  }

  // Delete confirmation logic.

  const confirmDeleteProject = async () => {

    if (!projectToDelete) return;

    setShowLoader(true);

    try {

      await dispatch(

        deleteProject({ uid: currentUser.uid, projectId: projectToDelete })

      ).unwrap();

      toast.success("Project deleted successfully!");

      setShowLoader(false);

      setShowDeletePopup(false);

      setProjectToDelete(null);

      await dispatch(fetchAllProjects());

    } catch (error) {

      toast.error("Failed to delete project");

    }

  };

  const handleDeleteClick = (projectId) => {

    setProjectToDelete(projectId);

    setShowDeletePopup(true);

  };

  return (

    <div className="p-4 sm:p-6 bg-gray-50 dark:bg-gray-900 min-h-screen">

      <h1 className="text-2xl font-bold mb-6 text-gray-900 dark:text-white">

        Manage Projects

      </h1>

      {/\* Filter by Company Name \*/}

      <div className="mb-6">

        <label className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-2">

          Filter by Company Name

        </label>

        <input

          type="text"

          value={filterCompany}

          onChange={(e) => {

            setFilterCompany(e.target.value);

            setCurrentPage(1);

          }}

          placeholder="Enter company name"

          className="w-full sm:w-1/3 p-2 border border-gray-300 dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-purple-500 dark:bg-gray-800 dark:text-white"

        />

      </div>

      {loading ? (

        <InPageLoader />

      ) : (

        <>

          {filteredProjects.length === 0 ? (

            <p className="text-gray-700 dark:text-gray-300">

              No projects found.

            </p>

          ) : (

            <div className="overflow-x-auto bg-white dark:bg-gray-800 shadow-sm rounded-lg">

              <table className="min-w-full divide-y divide-gray-200 dark:divide-gray-700">

                <thead className="bg-gray-50 dark:bg-gray-700">

                  <tr>

                    {tableHeaders.map((header) => (

                      <th

                        key={header.key}

                        onClick={() =>

                          header.sortable && handleSort(header.key)

                        }

                        className={`px-4 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider ${

                          header.sortable

                            ? "cursor-pointer hover:bg-gray-100 dark:hover:bg-gray-600"

                            : ""

                        } group`}

                      >

                        <div className="flex items-center space-x-1">

                          <span>{header.label}</span>

                          {header.sortable && renderSortIcon(header.key)}

                        </div>

                      </th>

                    ))}

                  </tr>

                </thead>

                <tbody className="divide-y divide-gray-200 dark:divide-gray-700">

                  {currentItems.map((project) => {

                    const creator = allUsersData.find(

                      (u) => u.uid === project.userId

                    );

                    const builderCompany = getBuilderCompanyName(

                      project,

                      creator

                    );

                    // Disable delete if project is assigned.

                    const isDeleteDisabled =

                      project?.status &&

                      (project.status.toLowerCase() === "assigned to builder" ||

                        project.status.toLowerCase() ===

                          "assigned to customer");

                    const latestUpdate =

                      project.lastUpdated &&

                      Array.isArray(project.lastUpdated) &&

                      project.lastUpdated.length > 0

                        ? project.lastUpdated.reduce((latest, current) => {

                            const currentTime = current.updatedAt

                              ? new Date(

                                  current.updatedAt.toDate?.() ||

                                    current.updatedAt

                                ).getTime()

                              : 0;

                            const latestTime = latest.updatedAt

                              ? new Date(

                                  latest.updatedAt.toDate?.() ||

                                    latest.updatedAt

                                ).getTime()

                              : 0;

                            return currentTime > latestTime ? current : latest;

                          }, project.lastUpdated[0])

                        : null;

                    return (

                      <tr

                        key={project.id}

                        className="hover:bg-gray-50 dark:hover:bg-gray-700"

                      >

                        <td className="px-4 py-3 text-sm font-medium text-gray-900 dark:text-gray-100">

                          {project.projectOverview?.projectName || "N/A"}

                        </td>

                        <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                          {creator

                            ? creator.fullName || creator.email

                            : "Unknown"}

                        </td>

                        <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                          {project.userType || "N/A"}

                        </td>

                        <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                          {builderCompany}

                        </td>

                        <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                          {project.createdAt

                            ? dayjs(

                                project.createdAt.toDate

                                  ? project.createdAt.toDate()

                                  : project.createdAt

                              ).format("YYYY-MM-DD HH:mm")

                            : "N/A"}

                        </td>

                        <td className="px-4 py-3 text-sm text-gray-900 dark:text-gray-100">

                          {latestUpdate?.updatedAt

                            ? dayjs(

                                latestUpdate.updatedAt.toDate

                                  ? latestUpdate.updatedAt.toDate()

                                  : latestUpdate.updatedAt

                              ).format("YYYY-MM-DD HH:mm")

                            : "N/A"}

                        </td>

                        <td className="px-4 py-3 text-sm">

                          <span

                            className={`px-2 py-1 rounded-full text-xs font-bold ${

                              project.status?.toLowerCase() === "published"

                                ? "bg-green-100 text-green-800"

                                : project.status?.toLowerCase() === "draft"

                                ? "bg-yellow-100 text-yellow-800"

                                : "bg-gray-100 text-gray-800"

                            }`}

                          >

                            {project.status || "N/A"}

                          </span>

                        </td>

                        <td className="px-4 py-3 text-sm">

                          <button

                            onClick={() => {

                              if (!isDeleteDisabled)

                                handleDeleteClick(project.id);

                            }}

                            disabled={isDeleteDisabled}

                            title={

                              isDeleteDisabled

                                ? "Project is assigned so cannot delete"

                                : ""

                            }

                            className={`transition-colors duration-200 ${

                              isDeleteDisabled

                                ? "text-gray-400 cursor-not-allowed"

                                : "text-red-600 hover:text-red-800"

                            }`}

                            aria-label="Delete Project"

                          >

                            <MdDelete size={20} />

                          </button>

                        </td>

                      </tr>

                    );

                  })}

                </tbody>

              </table>

            </div>

          )}

          {/\* Pagination \*/}

          {totalPages > 1 && (

            <div className="mt-6">

              <Pagination

                currentPage={currentPage}

                totalPages={totalPages}

                onPageChange={setCurrentPage}

              />

            </div>

          )}

        </>

      )}

      {/\* Delete Confirmation Popup \*/}

      {showDeletePopup && (

        <div className="fixed inset-0 flex items-center justify-center z-50 p-4">

          <div className="absolute inset-0 bg-black opacity-50"></div>

          <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6 z-10 max-w-lg w-full">

            <div className="mb-4">

              <p className="text-red-600 font-bold text-sm sm:text-base">

                ALERT: Deleting this project is permanent.

              </p>

            </div>

            <p className="text-gray-800 dark:text-gray-200 mb-4 text-sm">

              Deleting this project will permanently remove it from the system

              and unassign any builder or shared associations. Are you sure you

              want to delete this project?

            </p>

            <div className="flex justify-end space-x-3">

              <button

                onClick={() => {

                  setShowDeletePopup(false);

                  setShowLoader(false);

                }}

                className="px-4 py-2 rounded-md bg-gray-300 text-gray-800 hover:bg-gray-400 transition-colors text-sm"

              >

                Cancel

              </button>

              <button

                onClick={confirmDeleteProject}

                className="px-3 py-1 rounded-md bg-red-600 text-white hover:bg-red-700 transition-colors text-xs sm:text-sm"

              >

                {showLoader ? (

                  <div className="flex items-center justify-center space-x-2">

                    <InPageLoader color="white" width={5} height={5} />

                    <span>Deleting...</span>

                  </div>

                ) : (

                  "Delete"

                )}

              </button>

            </div>

          </div>

        </div>

      )}

    </div>

  );

};

export default ManageProjects;

## Backend Code:

### Auth.Controller.js:

const { admin } = require('../config/db.config');

const jwt = require('jsonwebtoken');

const {

  registerUser,

  loginUser,

  logoutUser,

  sendPasswordResetEmail,

  resetPassword,

  getUserFromSessionCookie,

  googleLoginUser,

  verifyEmailForRegistration,

  sendVerificationEmail,

  completeRegistration,

  googleRegistrationService

} = require('../services/auth.service');

const { generateToken } = require('../utils/jwt');

// POST /register

async function register(req, res, next) {

  try {

    const { email, password, fullName, userType } = req.body;

    const user = await registerUser({ email, password, fullName, userType });

    const token = generateToken(user);

    return res.status(201).json({ success: true, user,token });

  } catch (error) {

    return next(error);

  }

}

// POST /login

async function login(req, res, next) {

  try {

    const { email, password } = req.body;

    const { user, token } = await loginUser({ email, password, res });

    return res.status(200).json({

      success: true,

      user,

      token

    });

  } catch (error) {

    return res.status(401).json({

      success: false,

      message: error.message || 'Invalid credentials'

    });

  }

}

// GET /me

// Uses the session cookie to identify the user

async function getMe(req, res, next) {

  try {

    const sessionCookie = req.cookies.session;

    const userData = await getUserFromSessionCookie(req.headers.authorization);

    return res.json({ success: true, user: userData });

  } catch (error) {

    return res.status(401).json({ error: 'Not authenticated' });

  }

}

// POST /logout

async function logout(req, res) {

  try {

    // Clear the session cookie

    logoutUser(res);

    return res.json({

      success: true,

      message: 'Logged out successfully'

    });

  } catch (error) {

    return res.status(500).json({

      success: false,

      message: error.message || 'Logout failed'

    });

  }

}

// POST /forgot-password

async function forgotPassword(req, res, next) {

  try {

    const { email } = req.body;

    await sendPasswordResetEmail(email);

    return res.json({ success: true, message: 'Password reset email sent.' });

  } catch (error) {

    return next(error);

  }

}

// POST /reset-password

async function doResetPassword(req, res) {

  try {

    const { token, newPassword } = req.body;

    if (!token || !newPassword) {

      return res.status(400).json({

        success: false,

        message: 'Missing required fields'

      });

    }

    await resetPassword(token, newPassword);

    return res.json({

      success: true,

      message: 'Password updated successfully'

    });

  } catch (error) {

    return res.status(400).json({

      success: false,

      message: error.message

    });

  }

}

async function googleLogin(req,res,next) {

  try {

    const {idToken}=req.body;

    const { user, token }=await googleLoginUser({idToken,res});

    return res.status(200).json({

      success: true,

      token,

      user

    });

  } catch (error) {

    return res.status(401).json({

      success:false,

      message:error.message || 'Google Authentication Failed'

    })

  }

}

async function googleRegister(req, res) {

  try {

    const { idToken } = req.body;

    const userData = await googleRegistrationService({ idToken, res });

    const token = generateToken({ uid, ...userData })

    return res.status(200).json({

      success: true,

      user: {

        ...userData,

        isRegistered: Boolean(userData.userType && userData.fullName), // Check if required fields exist

      },

      token

    });

  } catch (error) {

    return res.status(401).json({

      success: false,

      message: error.message || 'Google registration failed'

    });

  }

}

async function checkEmail(req,res) {

  try {

    const {email}=req.body;

    const existingUser=await verifyEmailForRegistration(email);

    if (existingUser) {

      return res.status(400).json({

        success: false,

        message: 'Email already registered'

      });

    }

    const redirectUrl = `${process.env.FRONTEND\_URL}/register-password`;

    const verificationLink = await sendVerificationEmail(email, redirectUrl);

    res.json({

      success: true,

      message: 'Verification email sent'

    });

  } catch (error) {

    res.status(500).json({

      success: false,

      message: error.message

    });

  }

}

async function completeUserRegistration(req,res) {

  try {

    const {email,fullName,userType}=req.body;

    if (!email || !fullName || !userType) {

      return res.status(400).json({

        success: false,

        message: 'Missing required registration fields'

      });

    }

    const userData=await completeRegistration({

      email,

      fullName,

      userType

    });

    const token = generateToken(userData);

    res.status(201).json({

      success: true,

      user: userData,

      token

    });

  } catch (error) {

    res.status(500).json({

      success: false,

      message: error.message

    });

  }

}

module.exports = {

  register,

  login,

  getMe,

  logout,

  forgotPassword,

  doResetPassword,

  googleLogin,

  completeUserRegistration,

  checkEmail,

  googleRegister

};

### User.Controller.js:

const UserService = require('../services/user.service');

const getUsers = async (req, res) => {

  try {

    const users = await UserService.getUsers(req.user);

    return res.status(200).json({

      success: true,

      data: users

    });

  } catch (error) {

    return res.status(500).json({

      success: false,

      message: error.message

    });

  }

};

const deleteUser = async (req, res) => {

  try {

    const result = await UserService.deleteUser(req.params.uid, req.user);

    res.json(result);

  } catch (error) {

    res.status(error.status || 500).json({ error: error.message });

  }

};

const inviteUser = async (req, res) => {

  try {

    const result = await UserService.inviteUser(req.body, req.user);

    res.json(result);

  } catch (error) {

    res.status(error.status || 500).json({ error: error.message });

  }

};

const updateUser = async (req, res) => {

  try {

    const result = await UserService.updateUser(

      req.params.userId,

      req.body,

      req.user

    );

    return res.status(200).json({

      success: true,

      message: result.message,

      user: result.user

    });

  } catch (error) {

    console.error('Update user error:', error);

    return res.status(500).json({

      success: false,

      message: error.message || 'Error updating user'

    });

  }

};

module.exports = {

  getUsers,

  deleteUser,

  inviteUser,

  updateUser

};

### Auth.Middleware.js:

// Auth validation middleware

const { admin, db } = require('../config/db.config');

const { UnauthorizedError, ForbiddenError } = require('../utils/errorHandler');

const { verifyToken } = require('../utils/jwt');

const verifyDeletionPermission = async (req, res, next) => {

  const authHeader = req.headers.authorization;

  // 1. Check for bearer token

  if (!authHeader || !authHeader.startsWith('Bearer ')) {

    return res.status(401).json({ error: 'Unauthorized: No token provided.' });

  }

  const token = authHeader.split(' ')[1];

  try {

    // 2. Verify token and get caller UID

    const decodedToken = await admin.auth().verifyIdToken(token);

    const callerUid = decodedToken.uid;

    if (!callerUid) {

      return res.status(401).json({ error: 'Unauthorized: Invalid token.' });

    }

    // 3. Get the caller's document

    const callerDoc = await db.collection('users').doc(callerUid).get();

    if (!callerDoc.exists) {

      return res.status(403).json({ error: 'Forbidden: Caller user doc not found.' });

    }

    const callerData = callerDoc.data();

    const callerRole = callerData.role;           // e.g. "Moderator", "Admin", etc.

    const callerUserType = callerData.userType;     // e.g. "Admin", "Builder", "Customer"

    const callerCompanyId = callerData.companyId || null;

    // 4. Get the target user's document (the one to be deleted)

    const { uid: targetUid } = req.params;

    const targetDoc = await db.collection('users').doc(targetUid).get();

    if (!targetDoc.exists) {

      return res.status(404).json({ error: 'Target user not found.' });

    }

    const targetData = targetDoc.data();

    const targetRole = targetData.role;

    const targetUserType = targetData.userType;

    const targetCompanyId = targetData.companyId || null;

    // 5. Determine deletion permission

    // Self-deletion is not allowed.

    if (callerUid === targetUid) {

      return res.status(403).json({ error: 'Forbidden: You cannot delete your own account.' });

    }

    // For deletion of other users:

    // a) If caller is a Moderator in Admin:

    //    - They can delete any user except another Moderator (with the same userType).

    if (callerUserType === 'Admin' && callerRole.includes('Moderator')) {

      if (targetRole === 'Moderator' && callerUserType === targetUserType) {

        return res.status(403).json({

          error: 'Forbidden: A Admin Moderator cannot delete another Moderator.'

        });

      }

      req.user = { callerUid, callerRole, callerUserType, callerCompanyId };

      return next();

    }

    // b) If caller is a Moderator in Builder:

    //    - They can delete a user in the same company only if that user is not also a Moderator.

    if (callerUserType === 'Builder' && callerRole.includes('Moderator')) {

      if (callerCompanyId && targetCompanyId && callerCompanyId === targetCompanyId) {

        if (targetRole === 'Moderator') {

          return res.status(403).json({

            error: 'Forbidden: A Builder Moderator cannot delete another Moderator within the same company.'

          });

        }

        req.user = { callerUid, callerRole, callerUserType, callerCompanyId };

        return next();

      } else {

        return res.status(403).json({

          error: 'Forbidden: You can only delete users within your company.'

        });

      }

    }

    // c) All other deletion attempts are forbidden.

    return res.status(403).json({

      error: 'Forbidden: You are not authorized to delete this account.'

    });

  } catch (error) {

    console.error('Error verifying token or permissions:', error);

    return res.status(401).json({ error: 'Unauthorized: Invalid token.' });

  }

};

const authenticateToken = async (req, res, next) => {

  try {

    const authHeader = req.headers.authorization;

    if (!authHeader?.startsWith('Bearer ')) {

      throw UnauthorizedError('No token provided');

    }

    const token = authHeader.split('Bearer ')[1];

    console.log("token from authHeader: ",token);

    if (!token) {

      throw new UnauthorizedError('No token provided');

    }

    const decoded = verifyToken(token);

    // Add user data to request

    const userDoc = await db.collection('users').doc(decoded.uid).get();

    if (!userDoc.exists) {

      throw new Error('User not found');

    }

    req.user = {

      decoded,

      ...userDoc.data()

    };

    next();

  } catch (error) {

    res.status(401).json(UnauthorizedError(error.message));

  }

};

const authorizeRoles = (roles) => {

  return (req, res, next) => {

    if (!roles.includes(req.user.role)) {

      return res.status(403).json(ForbiddenError('Insufficient permissions'));

    }

    next();

  };

};

module.exports = { verifyDeletionPermission,authenticateToken,authorizeRoles };

# Software Test Document

* **SOFTWARE TEST DOCUMENT**
* **6.1 System Overview:**

The software system to be tested is the web-based "Builder Management System." The application is designed to streamline communication, documentation, and project management between customers, builders, and tradies. It features real-time chat, documentation tracking, task management dashboards, and builder verification.

The system consists of several main components:

* **6.1.1 Communication and Documentation:**
* **Technology**: Firebase Real-time Database
* **Functionality**: Facilitates real-time messaging and automatic generation of documented agreements.
* **6.1.2 Task Management and Project Dashboard:**
* **Technology**: Firebase Firestore
* **Functionality**: Provides centralized task management, progress tracking, and milestone management.
* **6.1.3 Verified Builder Status:**
* **Technology**: Custom Verification Protocol
* **Functionality**: Ensures builders meet Victorian standards through a structured verification process.
* **6.2 System Testing:**

The testing process includes the following phases:

* **6.2.1 Functional Testing:**
* Evaluate all user interface and backend functionalities to ensure correctness.
* **6.2.2 Real-time Communication Testing:**
* Verify real-time messaging and notification delivery integrity using Firebase Real-time Database.
* **6.2.3 Document Generation and Retrieval Testing:**
* Test the automated document creation process and keyword-based retrieval.
* **6.2.4 Software Version:**
* Initial version: v1.0
* Feature-complete version: v1.1
* Deployment-ready version: v2.0
* **6.3 Test Approach:**

Testing includes structured and methodical techniques to comprehensively cover each functionality:

* **6.3.1 Major Testing Activities:**
* **Functional Testing:** Validate all system components.
* **Real-time Communication:** Confirm seamless and instant data transmission.
* **Document Generation:** Ensure accuracy and accessibility of automatically generated documents.
* **6.3.2 Testing Techniques:**
* End-to-end scenario testing
* Stress and load testing for real-time features
* Usability testing for UI/UX
* **6.4 Test Plan:**
* **6.4.1 Scope:**
* Real-time chat and notifications
* Automated documentation
* Project and task dashboards
* Builder verification mechanism
* **6.4.2 Resources:**
* Tools: React Testing Library, Firebase Emulator Suite, Jest

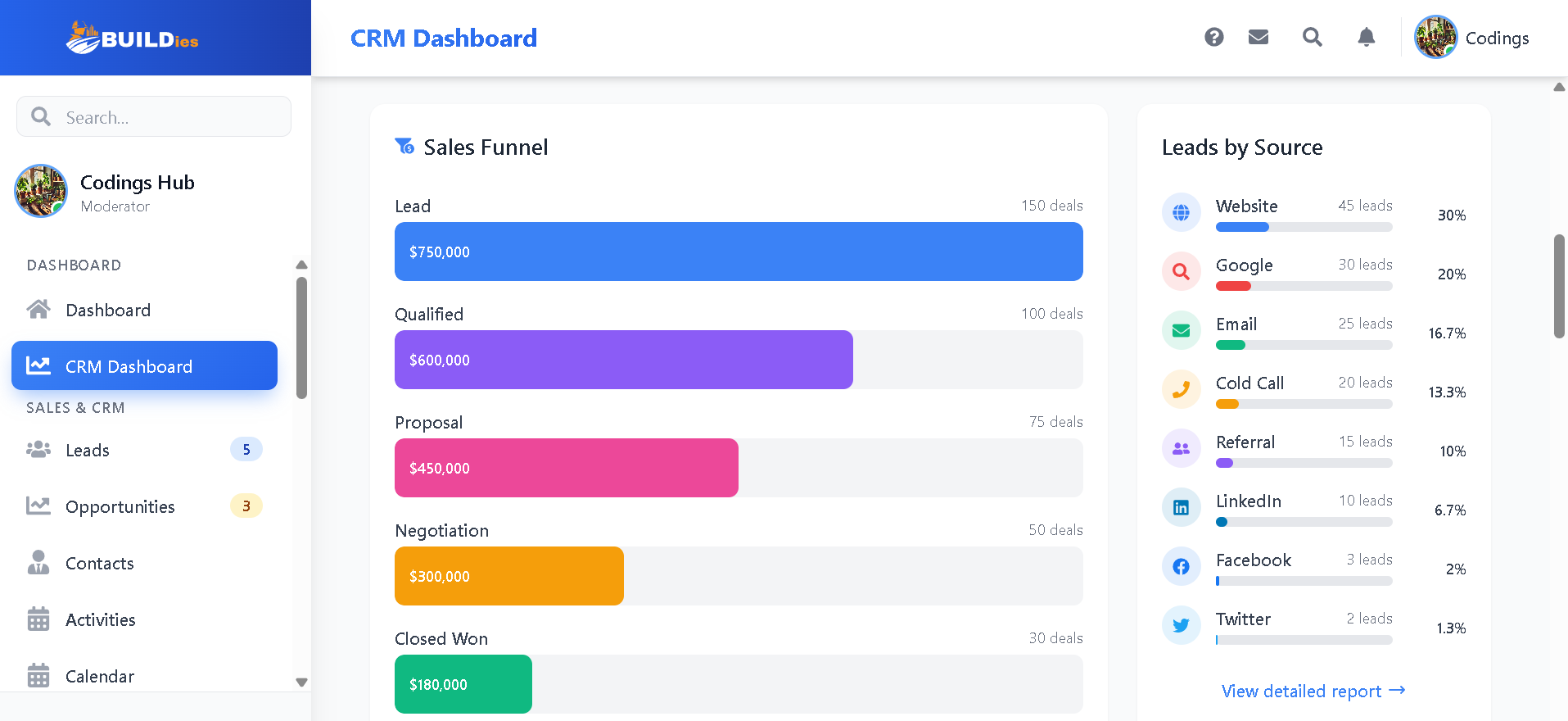
| **Task** | **Duration** | **Responsible Person** |
| --- | --- | --- |
| Functional Testing | 2 Weeks | Muhammad Shoaib Akhter |
| Real-time Communication Testing | 1 Week | Muhammad Shoaib Akhter |
| Document Generation Testing | 1 Week | Muhammad Shoaib Akhter |
| Frontend UI and Dashboard Testing | 1 Week | Muhammad Shoaib Akhter |

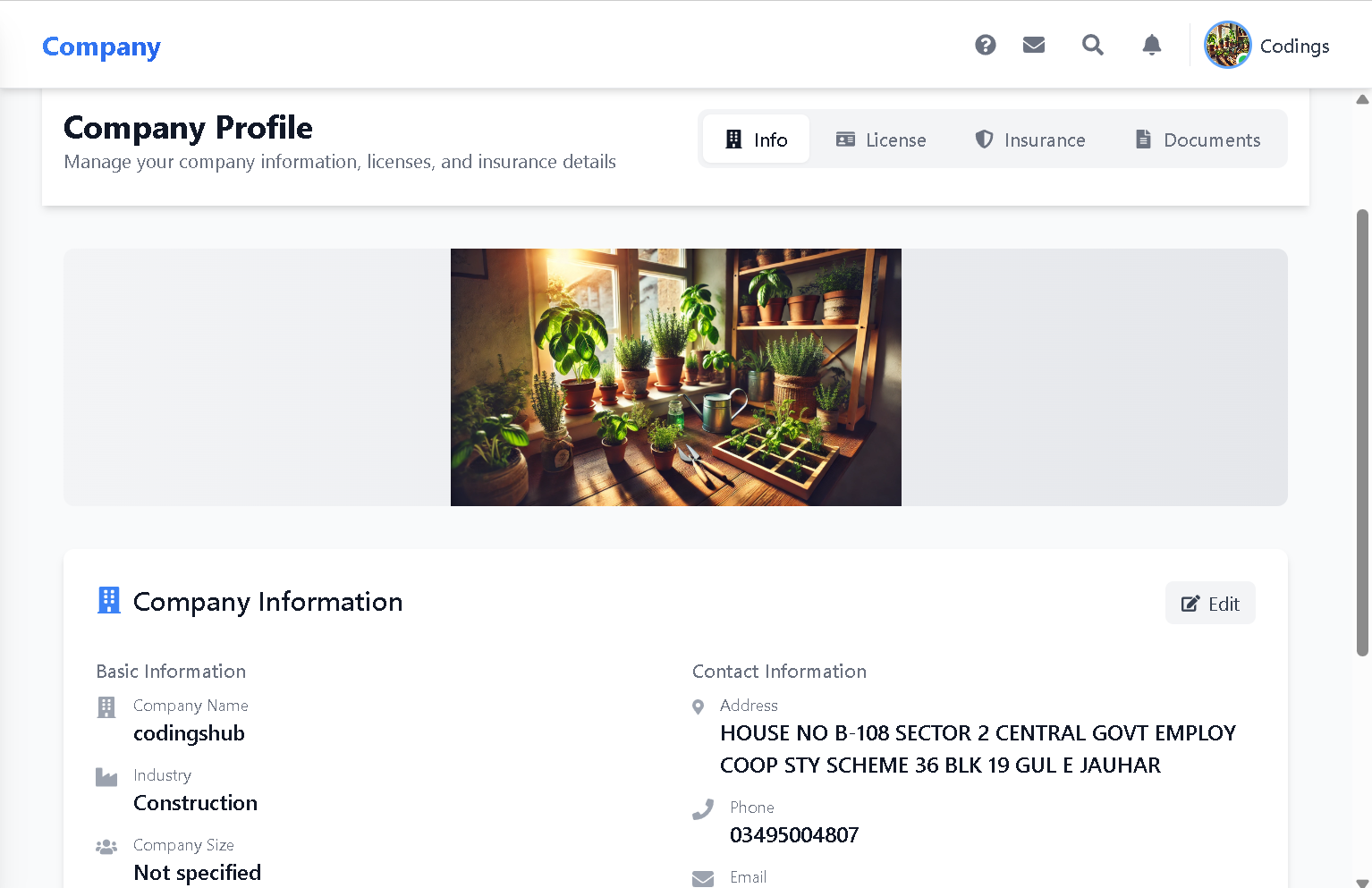
* **6.5 Features to be Tested:**
* **6.5.1 Functional Features:**
* **Real-time Messaging:**
  + TC-001: Verify message delivery accuracy and timeliness.
  + TC-002: Validate notification system functionality.
* **Automated Documentation:**
  + TC-003: Test automated document generation accuracy.
  + TC-004: Verify keyword-based document retrieval functionality.
* **Task Management Dashboard:**
  + TC-005: Validate task creation, assignment, and status updates.
  + TC-006: Ensure project milestones update accurately.
* **Builder Verification:**
  + TC-007: Validate verification process and status updates.
* **6.5.2 Combinations of Features:**
* **Real-time Messaging and Notification Integration:**
  + TC-008: Confirm notifications trigger correctly upon message reception.
* **Document Generation and Task Management Integration:**
  + TC-009: Validate that generated documents accurately reflect task statuses.
* **6.6 Test Cases:**
* **6.6.1 TC-001: Real-time Messaging**
* **Procedure:**
  1. Initiate real-time chat session.
  2. Send and verify messages.
* **Expected Output:** Instant message delivery without delays.
* **Pass/Fail Criteria:** Messages delivered within 1 second.
* **6.6.2 TC-003: Automated Document Generation**
* **Procedure:**
  1. Trigger document generation upon data entry.
  2. Validate generated document against input data.
* **Expected Output:** Documents accurately reflect input data.
* **Pass/Fail Criteria:** 100% accuracy in generated documents.
* **6.6.3 TC-007: Builder Verification**
* **Procedure:**
  1. Perform verification process.
  2. Check builder status update.
* **Expected Output:** Status updates correctly after verification process.
* **Pass/Fail Criteria:** Builder status reflects verification outcome correctly.

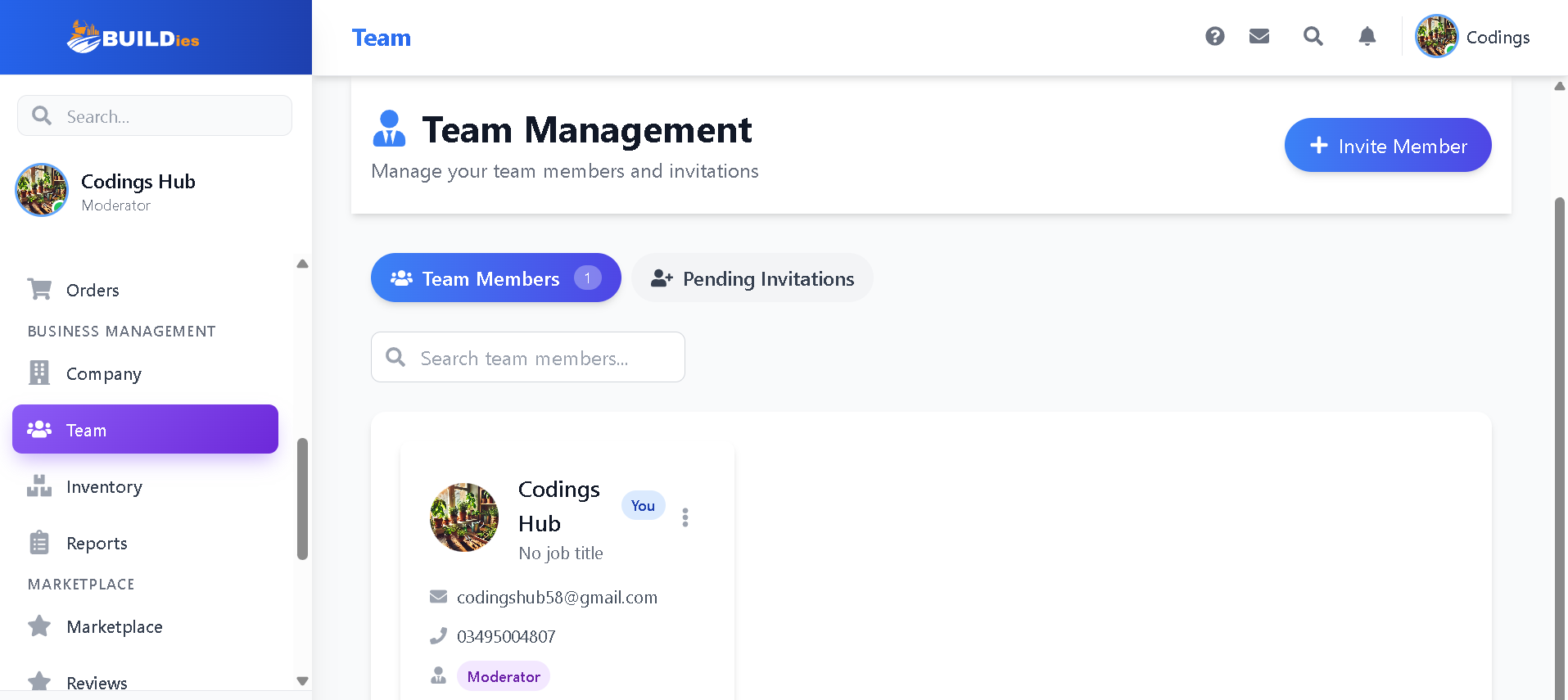
# Appendix A: Test Results

### Test Results

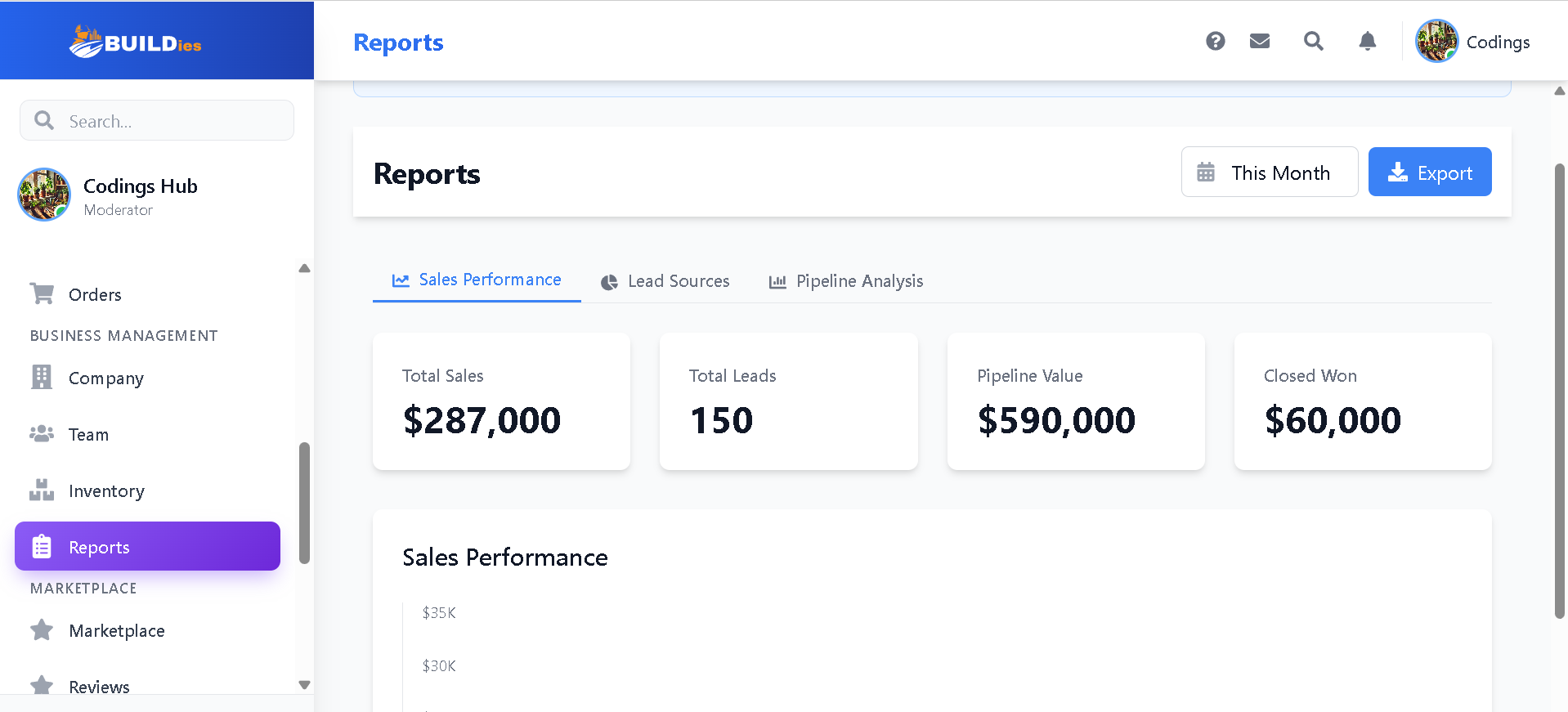
#### 

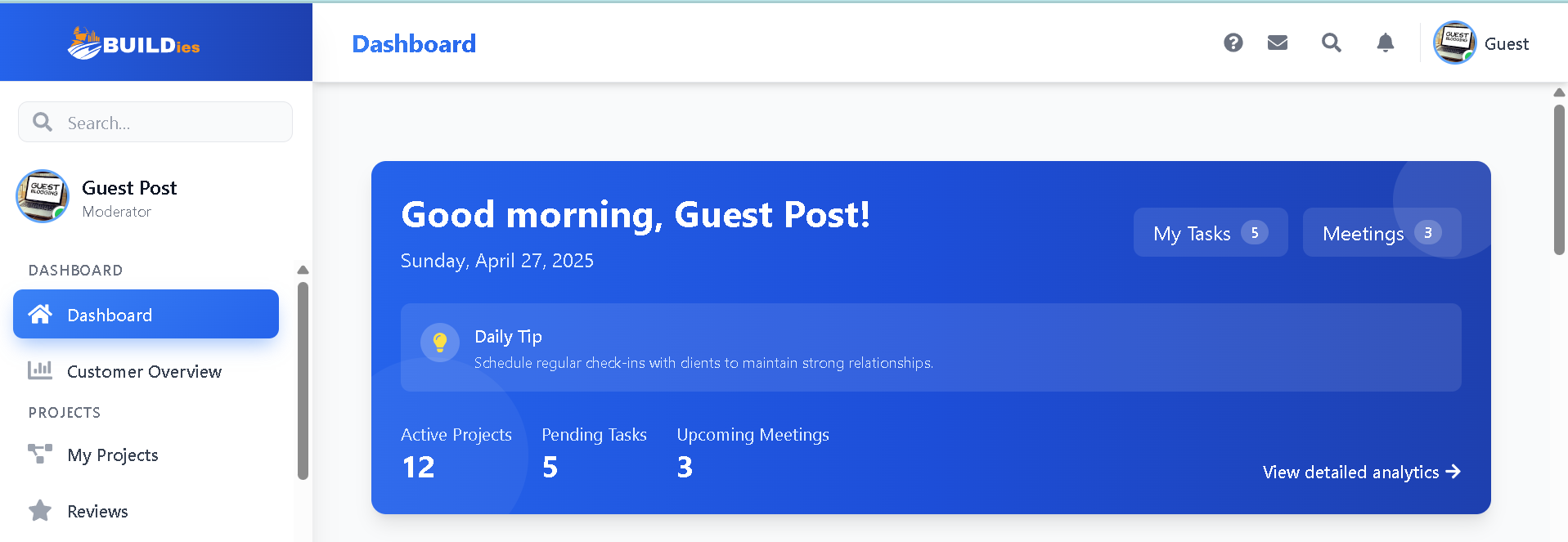


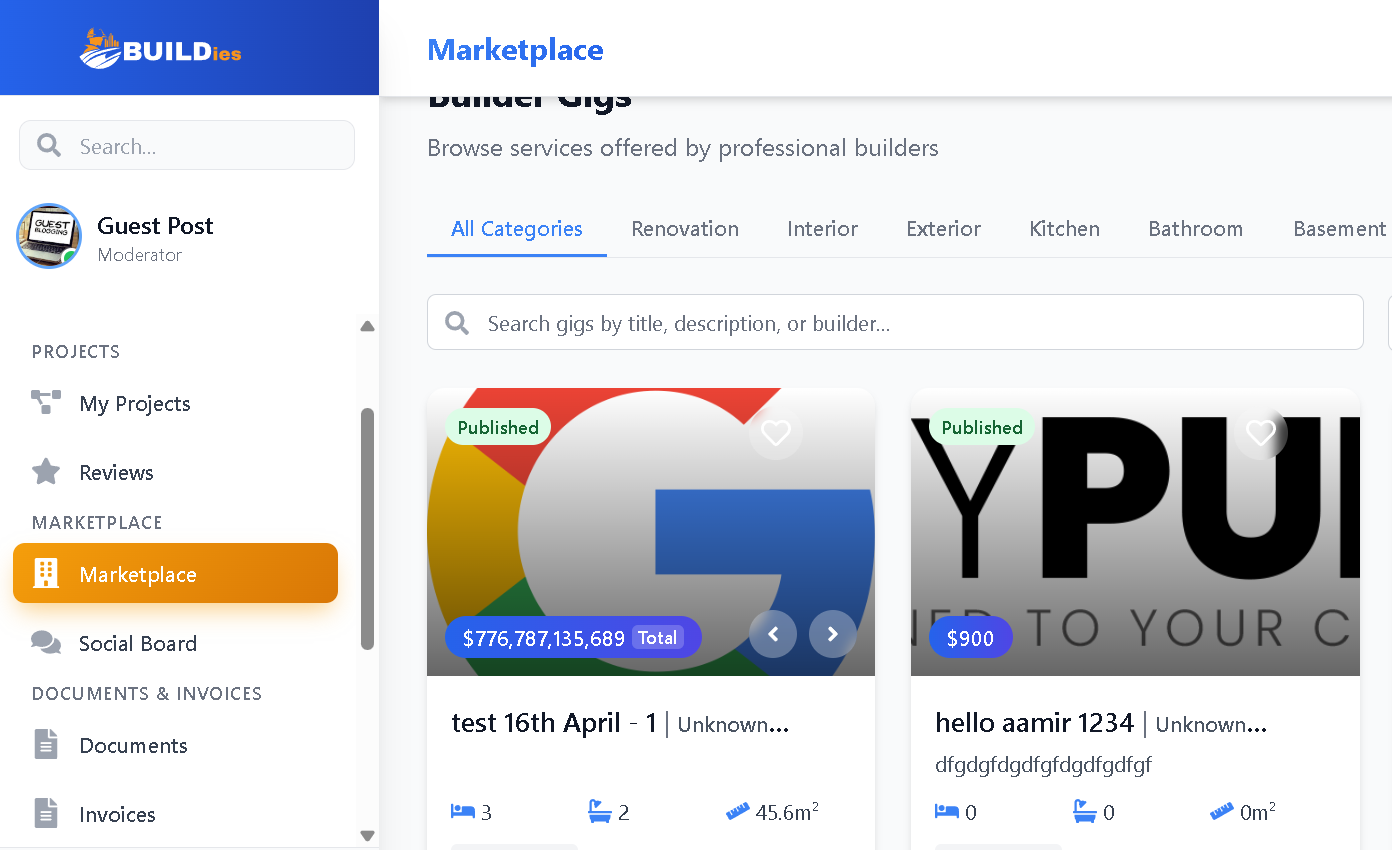




### VGG-16







# Conclusions and Further Work

## Conclusions:

The Builder Management System project has successfully established an efficient and transparent platform for enhancing communication, documentation, and project management among customers, builders, and tradies. Utilizing advanced technologies such as Firebase Real-time Database and Firebase Firestore, the system has significantly streamlined real-time interactions, documentation processes, and task management. Comprehensive testing procedures ensured reliability, responsiveness, and user-friendliness of the platform.

The integration of real-time messaging and automated document generation has addressed critical issues of miscommunication and lack of transparency prevalent in the residential construction industry. The system's robust verification process ensures builders meet specific Victorian standards, thereby enhancing trust and accountability within the construction community.

By implementing this system, we have successfully provided a centralized solution that supports better decision-making, reduces project delays, and ensures high-quality project outcomes through improved accountability and transparency.

## Further Work:

While the Builder Management System has effectively addressed key challenges, there remain several opportunities for enhancement and future development:

* **User Feedback System**: Implement a robust user feedback mechanism to continuously collect data on system performance, usability, and user satisfaction. Utilizing this feedback will enable iterative improvements and enhance overall user experience.
* **Continuous Learning and Updates**: Establish a framework for continuous learning and regular system updates based on emerging technologies, regulatory changes, and user requirements. This adaptability will ensure the system remains current, relevant, and capable of addressing evolving needs in the residential construction industry.

# References

1. BuildTools - <https://www.buildtools.com/>
2. Procore - <https://www.procore.com/>
3. CoConstruct - <https://www.coconstruct.com/>
4. Buildertrend - <https://www.buildertrend.com/>