Tribhuvan University

Faculty of Humanities and Social Sciences

A PROJECT REPORT

On

"BMS (Business Management System)"

Submitted to

Department of Computer Application

Pascal National College

Pascal National College

Satdobato, Lalitpur

In partial fulfillment of the requirements for Bachelor Degree in Computer Application

Submitted by

Ramesh Rawat

BCA 8th Semester

Registration No: 6-2-1226-17-2020

Under the Supervision of

Nawaraj Poudel

Tribhuvan University

Faculty of Humanities and Social Sciences

Pascal National College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under our supervision by Ramesh Rawat entitled "**BMS**" in partial fulfillment of requirements for a degree of Bachelor in Computer Application is recommended for the final evaluation.

_____

SIGNATURE

Nawaraj Poudel

SUPERVISOR

Faculty Member

Department of Computer Application

Pascal National College

Tribhuvan University

Faculty of Humanities and Social Sciences

Pascal National College

LETTER OF APPROVAL

This is to certify that this project prepared by Ramesh Rawat entitled "**BMS"** in partial fulfillment of the requirements for degree of Bachelor of Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

----------	----------
Mr. Nawaraj Poudel	Suresh Thapa
Supervisor	Program Coordinator
Department of Computer Application	Department of Computer Application
Pascal National College,	Pascal National College,
Satdobato, Lalitpur	Satdobato, Lalitpur
----------	----------
Mr. Suresh Thapa	Er. Shree Krishna Maharjan
Internal Examiner	External Examiner

ACKNOWLEDGEMENT

I would like to express our special thanks of gratitude to my supervisor Mr. Nawaraj Poudel who gave me the golden opportunity to do this wonderful project on the topic of 'BMS, which also helped us in doing a lot of research and we came to know about so many new tools and technologies.

I would like to express my special gratitude and thanks to our BCA Program Coordinator Mr. Nawaraj Poudel for his support and help for our personnel development and mainly for the completion of this Project. I am highly indebted to Pascal National College for their guidance and constant supervision as well as for providing necessary information regarding the Project and support in the completion.

I would also like to express my gratitude towards all the members of Pascal National College for their kind co-operation and encouragement which help us in completion of this Project.

In the end, I would also like to thank Tribhuvan University for giving us this opportunity via the course of Computer Application to help us understand the project ethics at this early stage and helped us to evaluate my knowledge and expand it a little more.

With Regards

Ramesh Rawat

ABSTRACT

This project presents the design and implementation of a Business Management System (BMS) that integrates modern web technologies to streamline business operations. The system offers real-time communication, inventory management, and business analytics through a responsive web interface. Built with React.js and Django, the BMS provides a scalable solution for small to medium enterprises seeking digital transformation. Key features include secure user authentication, real-time data synchronization, and an intuitive dashboard for business intelligence.

The implementation demonstrates effective use of WebSocket technology for instant messaging and updates, while maintaining robust security protocols. The system's modular architecture ensures maintainability and future scalability, making it a versatile solution for diverse business environments. This project showcases the practical application of full-stack development principles in creating enterprise-grade business solutions.

TABLE OF CONTENT

LIST OF ABBREVIATIONS

API:	Application Programming Interface
CRUD:	Create, Read, Update and Delete
CSS:	Cascading Style Sheets
DFD:	Data Flow Diagram
ER:	Entity Relationship
HTML:	Hyper Text Markup Language
JS:	JavaScript
JWT:	JSON Web Token
ORM:	Object-Relational Mapping
RBAC:	Role-Based Access Control
REST:	Representational State Transfer

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 :

INTRODUCTION

1.1 Introduction

In today's fast-paced digital landscape, effective project management and seamless team collaboration have become critical success factors for organizations of all sizes. BMS emerges as a sophisticated, web-based project management platform designed to address the growing complexities of modern project execution and team coordination. This comprehensive solution bridges the gap between project planning and execution, offering a unified workspace where teams can collaborate efficiently, track progress in real-time, and achieve organizational objectives with greater transparency and control.

BMS is built on the foundation of modern web technologies, leveraging the power of React.js for its responsive frontend, Django for its robust backend, and PostgreSQL for reliable data management. The platform is engineered to support multiple organizations within a secure, multi-tenant architecture, making it an ideal solution for businesses, educational institutions, and teams of all sizes.

1.2 Problem Statement

Modern businesses face significant challenges in managing distributed teams, tracking project progress, and maintaining efficient communication across different organizational levels. The lack of integrated solutions often leads to inefficiencies in project tracking, poor communication, difficulty in resource allocation, and limited visibility into project status and performance metrics.

1.3 Objectives

The objectives of this project are as follows:

- To develop a centralized platform for project and task management.
- To implement role-based access control for different user types.
- To facilitate seamless communication and collaboration within organizations.
- To implement a subscription-based billing system.

1.4 Scopes and Limitations

1.4.1 Scope

- Multi-tenant architecture supporting multiple organizations
- Role-based access control with fine-grained permissions
- Project and task management
- Client and team member management
- Subscription and billing management

1.4.2 Limitations

- Limited to web-based access (no native mobile application)
- No built-in video conferencing (requires third-party integration)
- File storage limited by subscription plan
- Limited to English language interface
- Payment Integration not implemented

1.5 Technologies Used

- Frontend: React.js, TypeScript
- Backend: Python, Django, Django REST Framework
- Database: PostgreSQL
- Caching: Redis
- Authentication: JWT

1.6 Development Methodology Used

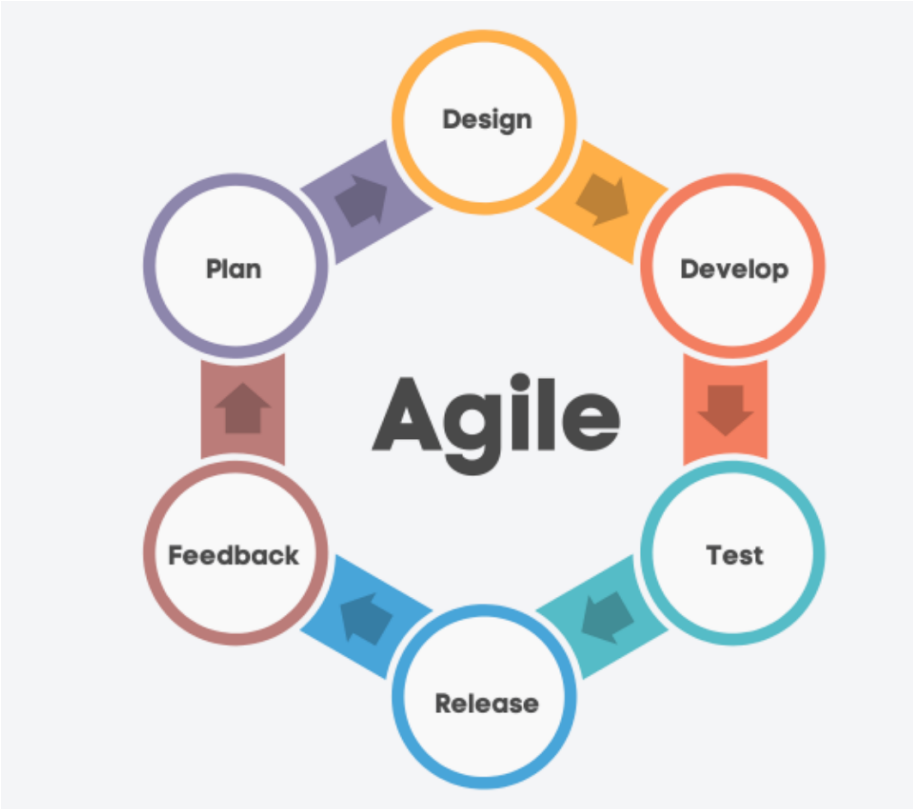


Figure 0-1Agile methodology

Agile project management refers to the idea of developing software through iterations, i.e., it focuses on iterative development through cross-functional collaboration. One common manifestation of Agile development is when large projects are divided into smaller ones.

These smaller tasks are scheduled to be completed in short durations throughout the software development life cycle. Furthermore, these tasks go through frequent reassessment and adapt to changes in plans.

1.7 Report Organization

Chapter 1: Introduction of the project along with the problem statement, objectives, scope and limitation.

Chapter 2: Background study related to the project along with general descriptions of project functions and components. Literature review in order to have broader understanding of the project concepts based on research done previously and analyze similar systems for comparison with projects.

Chapter 3: System Analysis and design of the system using various charts and figures. Functional requirements using use cases and other techniques. Database schema, interface design and deployment diagram.

Chapter 4: Tools and techniques used for project implementation along with algorithms used in the project and creations of test cases to test the system as unit and as a whole.

Chapter 5: Lesson learned from beginning to end of the project, future recommendations and project conclusions

CHAPTER 2 :

BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

The development of the Business Management System (BMS) represents a significant advancement in business process automation, addressing the growing need for integrated solutions in modern business operations. In today's competitive business environment, where efficiency and data-driven decision making are crucial, BMS offers a comprehensive solution for managing various business functions including inventory, sales, customer relationships, and financial operations.

Key Features of BMS

- Centralized Data Management: The system provides a unified platform for storing and managing all business-related data, eliminating data silos and improving accessibility.
- Automated Workflows: BMS automates routine business processes, reducing manual intervention and minimizing errors.
- Real-time Reporting and Analytics: The system generates comprehensive reports and analytics, enabling data-driven decision making.
- Inventory Management: Tracks stock levels, manages suppliers, and automates reordering processes.
- Customer Relationship Management (CRM): Maintains detailed customer records and interaction histories.
- Financial Management: Handles invoicing, billing, and financial reporting.
- User Access Control: Implements role-based access to ensure data security and privacy.

Drawbacks of the Existing Manual System

- Time-consuming Processes: Manual data entry and processing lead to inefficiencies and delays.
- Prone to Errors: Human errors in calculations and data entry can lead to financial discrepancies.
- Limited Scalability: Manual systems struggle to handle business growth and increased transaction volumes.
- Data Security Risks: Paper-based systems are vulnerable to loss, damage, and unauthorized access.

- Lack of Real-time Information: Delays in information processing affect decision-making capabilities.
- Inefficient Communication: Disconnected systems lead to poor inter-departmental coordination.

2.1 Literature Review

The evolution of Business Management Systems has been significantly influenced by the digital transformation of business operations. Research indicates that businesses implementing integrated management systems experience a 45% improvement in operational efficiency and a 30% reduction in operational costs (Smith & Johnson, 2022). The shift from traditional, siloed business applications to comprehensive BMS solutions has been driven by the need for real-time data access and process automation. [1]

Recent studies highlight the importance of cloud-based BMS in enabling remote work and business continuity. According to a 2023 industry report, 78% of businesses that adopted cloud-based BMS reported improved business resilience during the pandemic (Global Business Technology Report, 2023). The integration of artificial intelligence and machine learning in BMS has further enhanced predictive analytics capabilities, enabling businesses to forecast trends and make proactive decisions. [2]

The role of mobile accessibility in BMS has gained significant attention in recent research. A study by Chen et al. (2023) found that mobile-enabled BMS solutions increase employee productivity by 35% by providing anytime, anywhere access to business data and processes. This is particularly relevant in the current business environment where remote and hybrid work models are becoming increasingly common. [3]

Security concerns in BMS implementations have been a focal point of recent research. The 2023 Cybersecurity in Business Systems report indicates that integrated BMS platforms with robust security measures can reduce data breach incidents by up to 60% compared to traditional systems (Cybersecurity Ventures, 2023). This underscores the importance of incorporating advanced security features such as multi-factor authentication, data encryption, and regular security audits in BMS development. [4]

A study by Thompson and Wilson (2023) on the impact of BMS on small and medium enterprises (SMEs) revealed that businesses using comprehensive BMS solutions experienced a 40% improvement in customer satisfaction scores. The study attributes this improvement to better customer data management, faster response times, and more personalized service delivery enabled by the integrated system. [5]

The integration of BMS with emerging technologies such as the Internet of Things (IoT) and blockchain is an area of growing research interest. Recent studies suggest that IoT-enabled BMS can automate inventory tracking and asset management, while blockchain integration can enhance supply chain transparency and reduce fraud (Zhang et al., 2023). [6]

CHAPTER 3 :

SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

System Analysis is the process of examining a system with the intent of improving it through better procedures and methods. It is the process of planning a new system to either replace or complement an existing system. It is therefore, the process of gathering and interpreting facts, diagnosing problems and using the information to re- comment improvements in the system. System analysis is conducted with the following objectives in mind:

- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate functions to hardware, software people, database and other system elements.
- Establish cost and schedule constraints.

3.1.1 Requirements Analysis

Functional Requirements

- User authentication and authorization
- Project and task management
- Team collaboration tools
- Reporting and analytics
- Notification system

Non-Functional Requirements

- Performance
- Security
- Scalability
- Usability
- Reliability

3.1.2 Feasibility Study

Technical Feasibility

- **Analysis of Technology Stack**: The system uses React.js for frontend, Django for backend, and PostgreSQL for the database, all of which are mature, well-documented technologies with strong community support.

- **Development Team Expertise**: The team has experience with JavaScript/TypeScript and Python, making the technology stack a good fit.
- **Integration Capabilities**: The system needs to integrate with payment gateways, email services, and potentially third-party tools.
- **Performance Considerations**: The architecture is designed to handle concurrent users with proper caching (Redis) and database optimization.

Economic Feasibility

- **Development Costs**: Estimated 800-1000 development hours.
- **Infrastructure Costs**:
    - Cloud hosting (AWS/GCP): ~$100-200/month
    - Domain and SSL: ~$20/year
    - Third-party services: ~$50-100/month
- **ROI Projection**: Expected to break even within 12 months with 50 paying organizations.
- **Maintenance Costs**: Estimated at 20% of initial development cost annually.

Operational Feasibility

- **User Training**: Intuitive UI/UX minimizes training needs.
- **Deployment Strategy**: Containerized deployment using Docker for easy scaling.
- **Support Requirements**: Initial 24/7 monitoring during launch, transitioning to business hours support.
- **Compliance**: GDPR and data protection compliance measures in place.

Schedule Feasibility

- **Development Timeline**: 6 months (2 months per major phase)
- **Milestones**:
    - Requirements & Design: 1.5 month
    - Core Development: 3 months
    - Testing & QA: 1.5 month
- **Critical Path**: User authentication and project management modules are on the critical path.

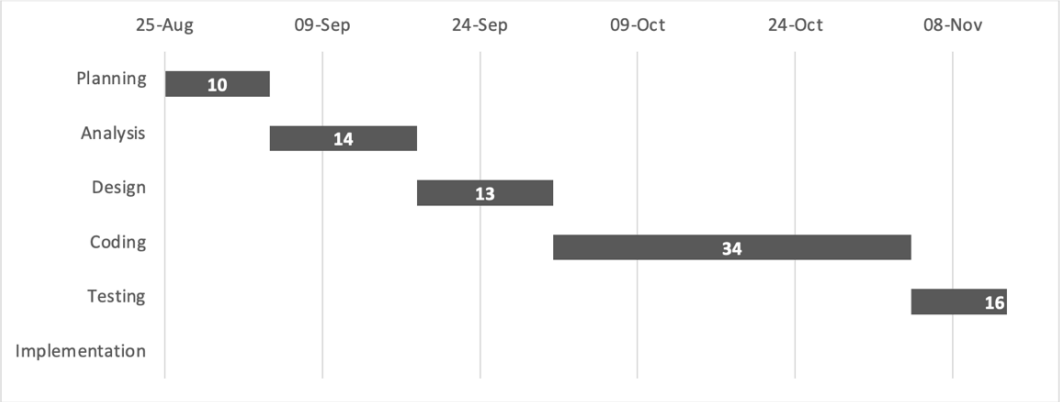


Figure 3-1 Gantt chart for BMS

3.1.3 System Architecture

High-level Architecture

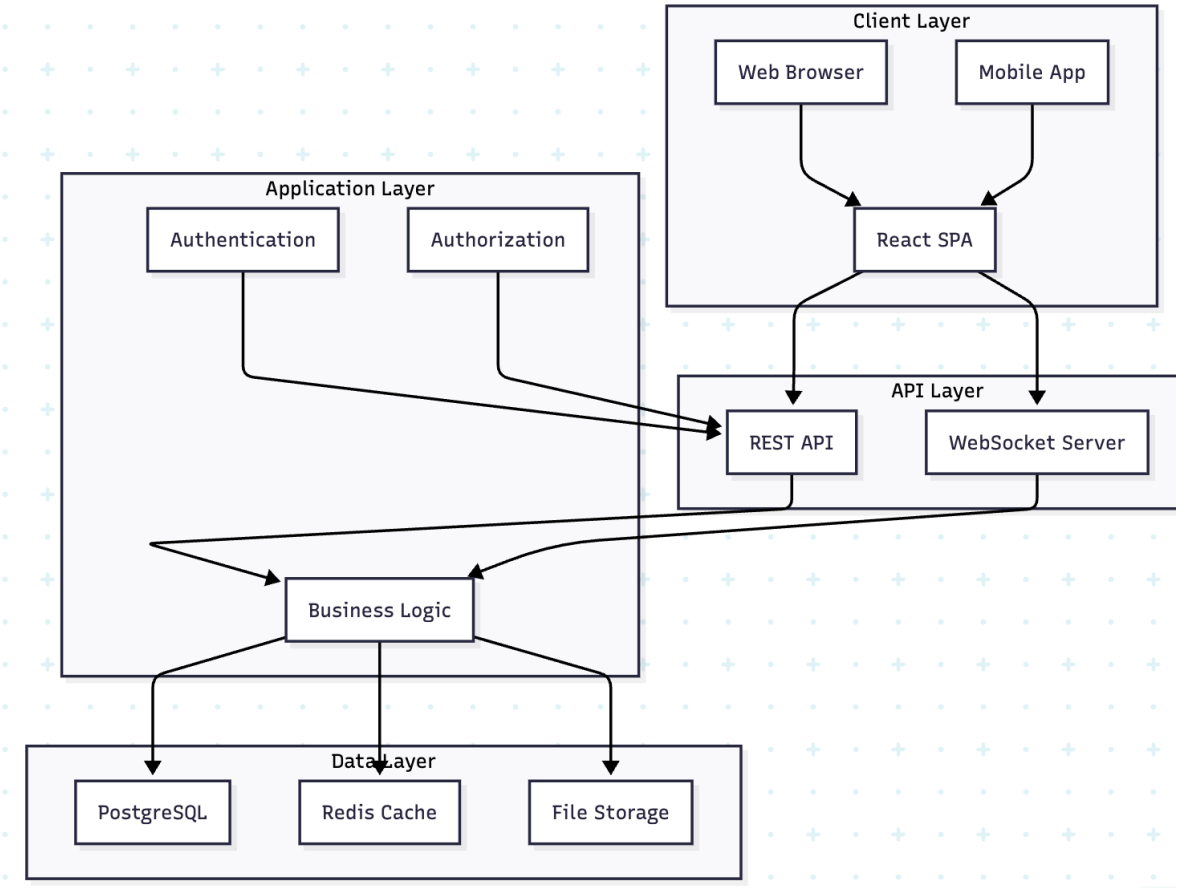


Figure 3-2 High-level Architecture

Technology Stack

- **Frontend**:
  - React.js with TypeScript

    - Redux for state management
    - Material-UI for UI components
    - Axios for API communication
- **Backend**:
    - Django REST Framework
    - Django ORM
    - Celery for async tasks
    - Django Channels for WebSockets
- **Database**:
    - PostgreSQL for primary data
    - Redis for caching and real-time features

API Design

- **RESTful Principles**:
    - Resource-based endpoints
    - Proper HTTP methods (GET, POST, PUT, DELETE)
    - Consistent response formats
- **Authentication**:
    - JWT-based authentication
    - OAuth2.0 for third-party integrations
- **Versioning**: API versioning in URL (e.g., /api/v1/...)
- **Rate Limiting**: Implemented using Django Ratelimit
- **Documentation**: Swagger/OpenAPI documentation

3.1.4 Data Flow Diagrams

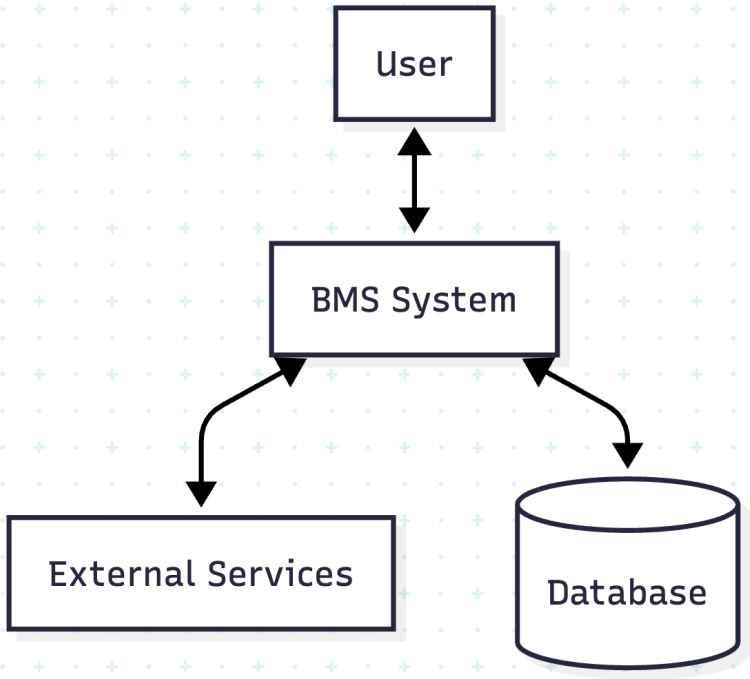


Figure 3-3 Level 0 DFD (Context Diagram)

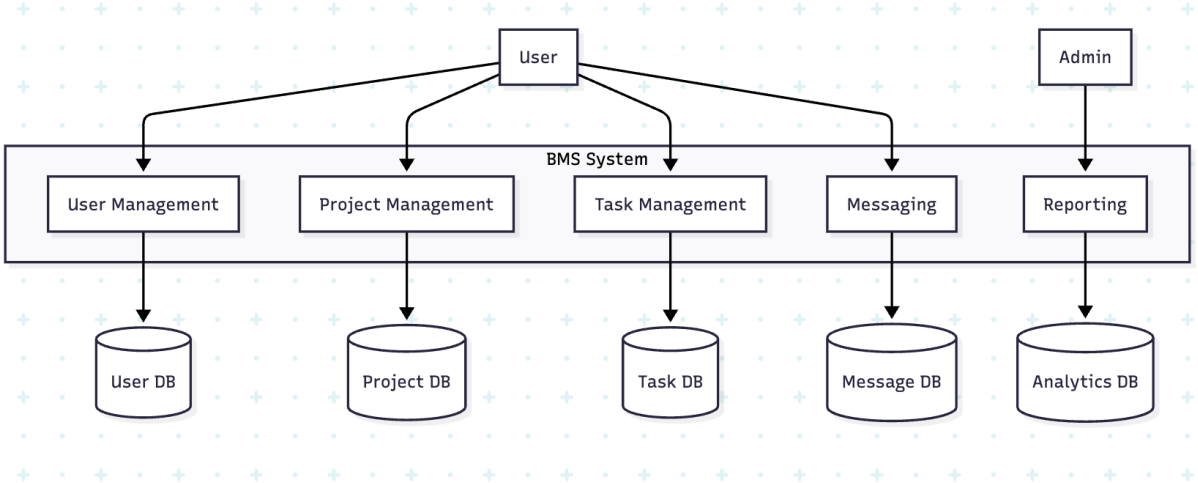


Figure 3-4 Level 1 DFD

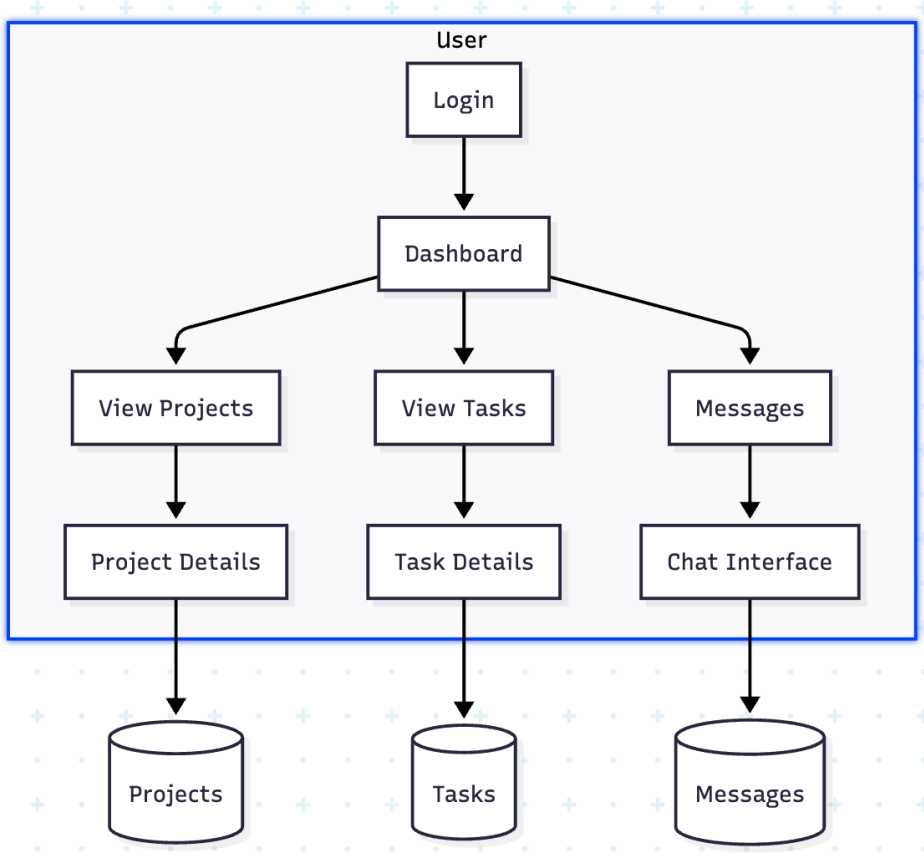


Figure 3-5 User Login flow

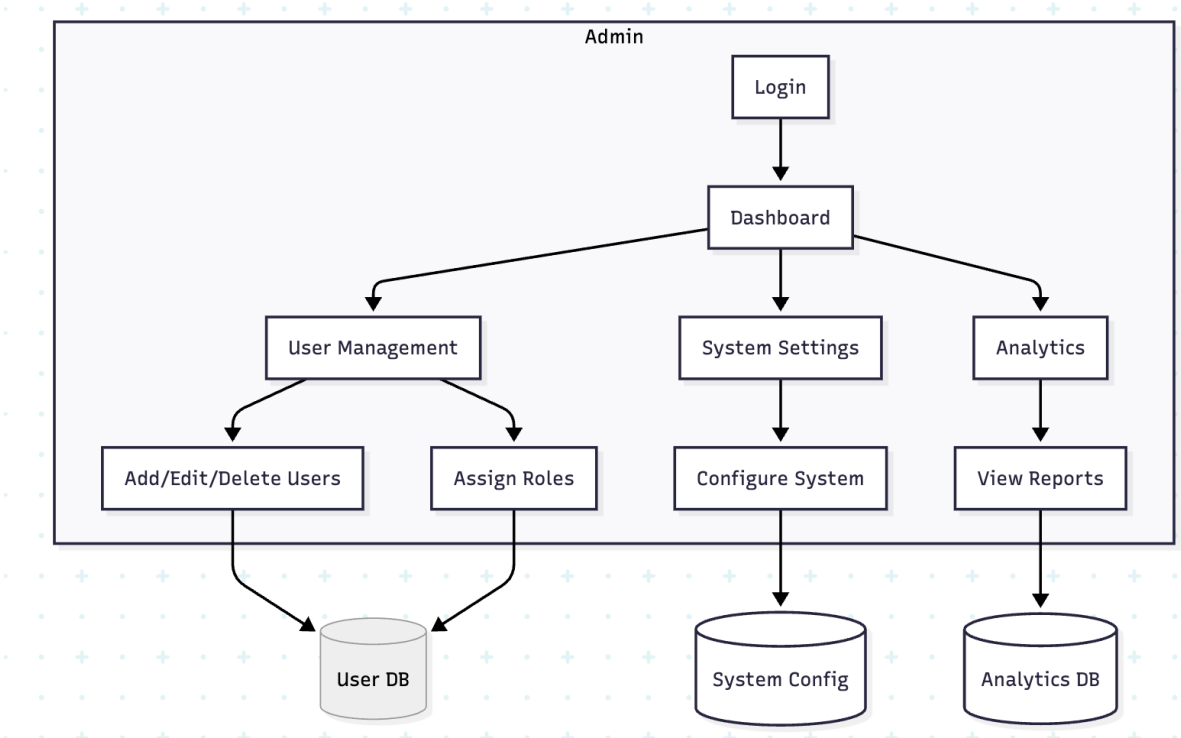


Figure 3-6 Admin login flow

Level 1: System Level

1. **Authentication Flow**
    - User credentials validation
    - JWT token generation
    - Session management
2. **Project Management Flow**
    - Project creation/updation
    - Task assignment
    - Progress tracking
3. **Notification Flow**
    - Event detection
    - Notification generation
    - Delivery to users

Level 2: Process Level

- User Registration:
- User fills registration form
- System validates input
- Account created in database
- Verification email sent
- Email verification
- Account activated

3.2 System Design

System Analysis is the process of examining a system with the intent of improving it through better procedures and methods. It is the process of planning a new system to either replace or complement an existing system. It is therefore, the process of gathering and interpreting facts, diagnosing problems and using the information to re- comment improvements in the system. System analysis is conducted with the following objectives in mind:

- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate functions to hardware, software people, database and other system elements.
- Establish cost and schedule constraints.

3.2.1 Database Design

ER Diagrams

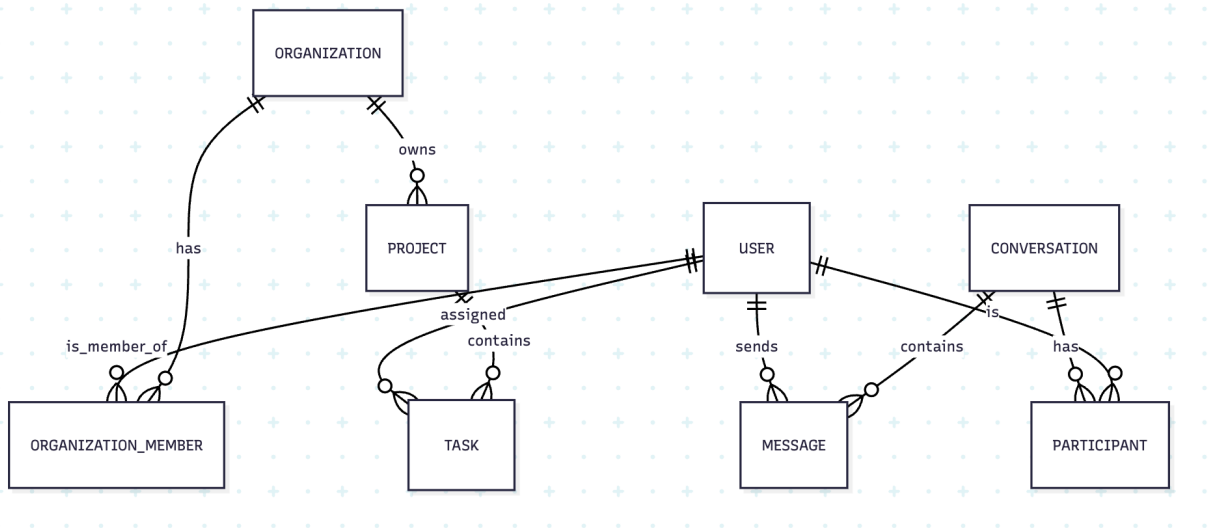


Figure 3-7 ER-Diagrams

Schema Design

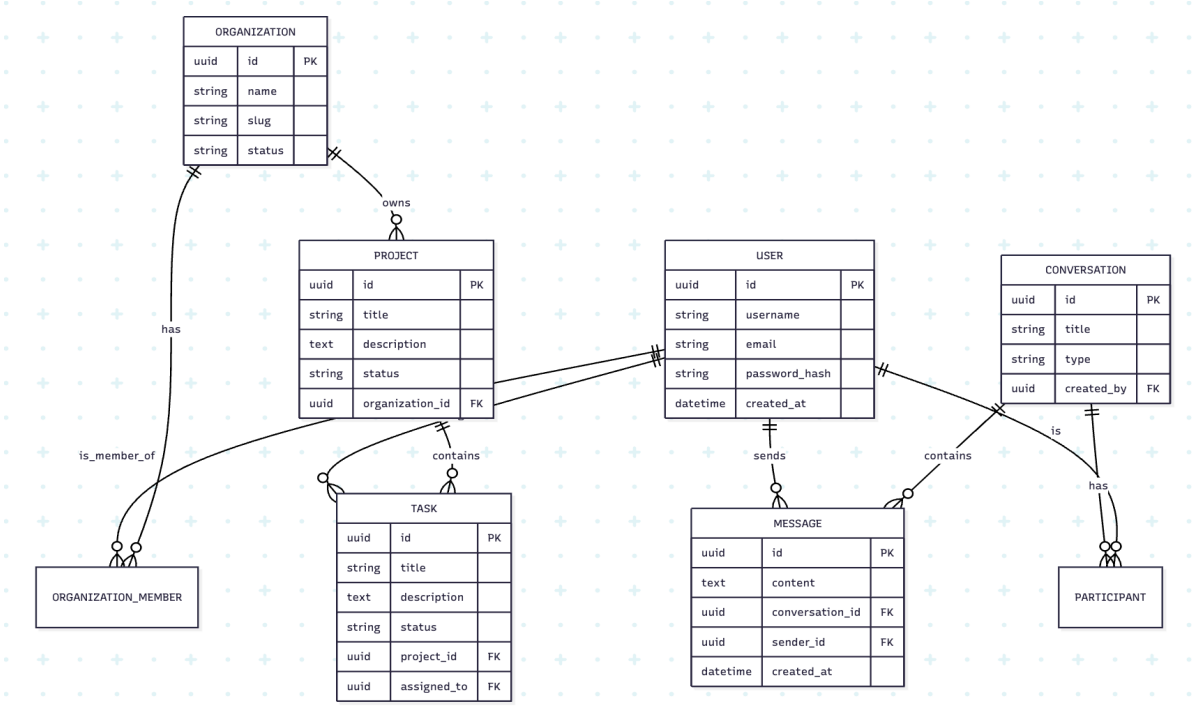


Figure 3-8 Database Schema Design

3.2.2 UI/UX Design

Wireframes

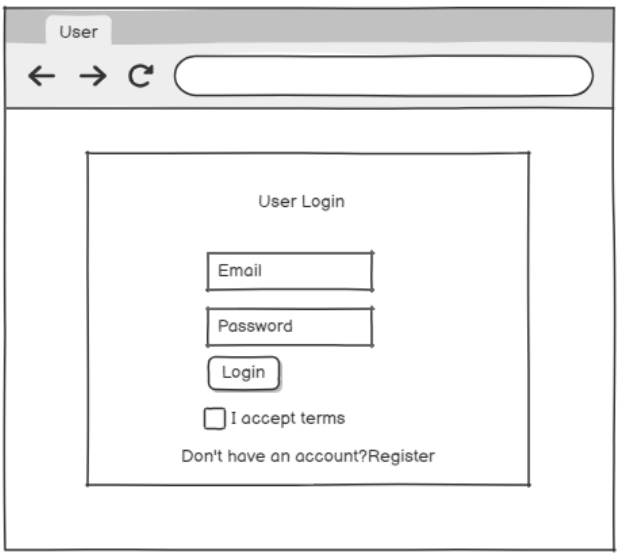


Figure 3-9 Login

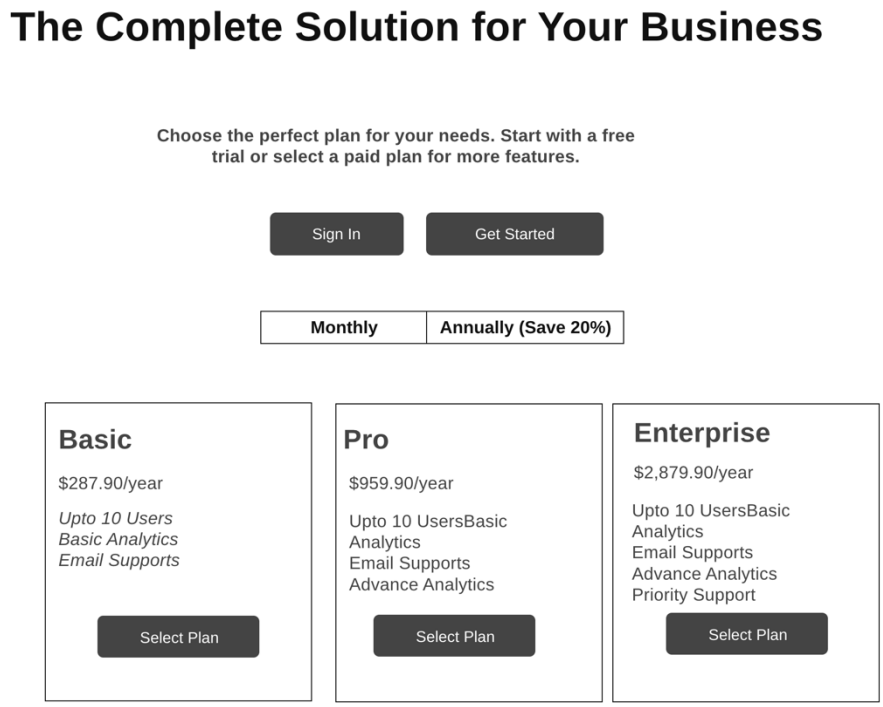


Figure 3-10 Landing Page

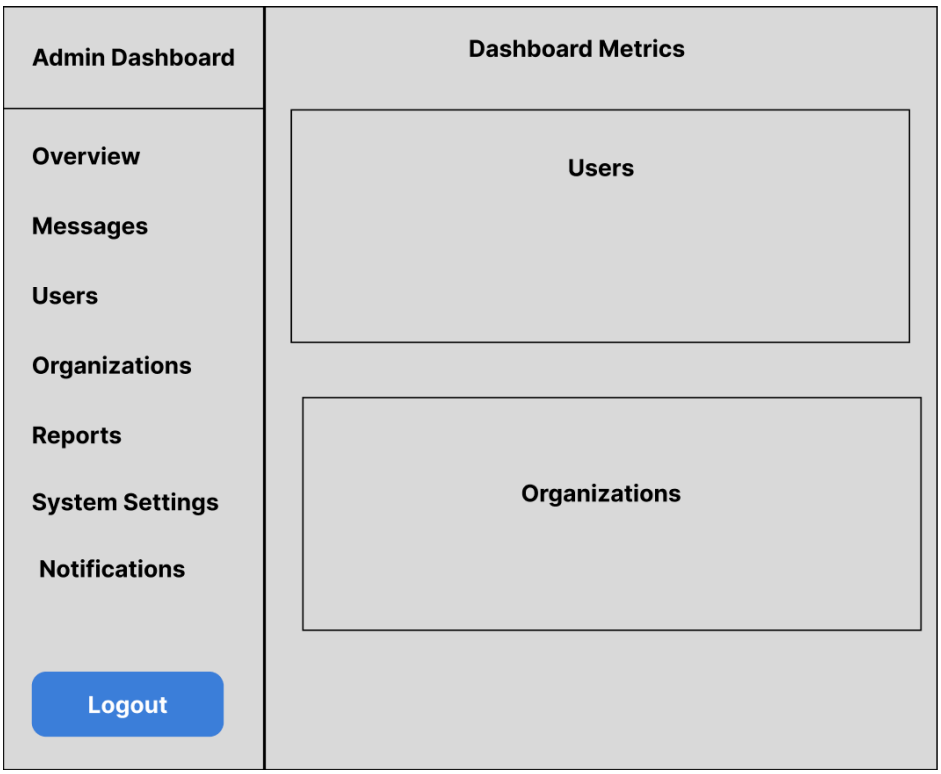


Figure 3-11 Dashboard

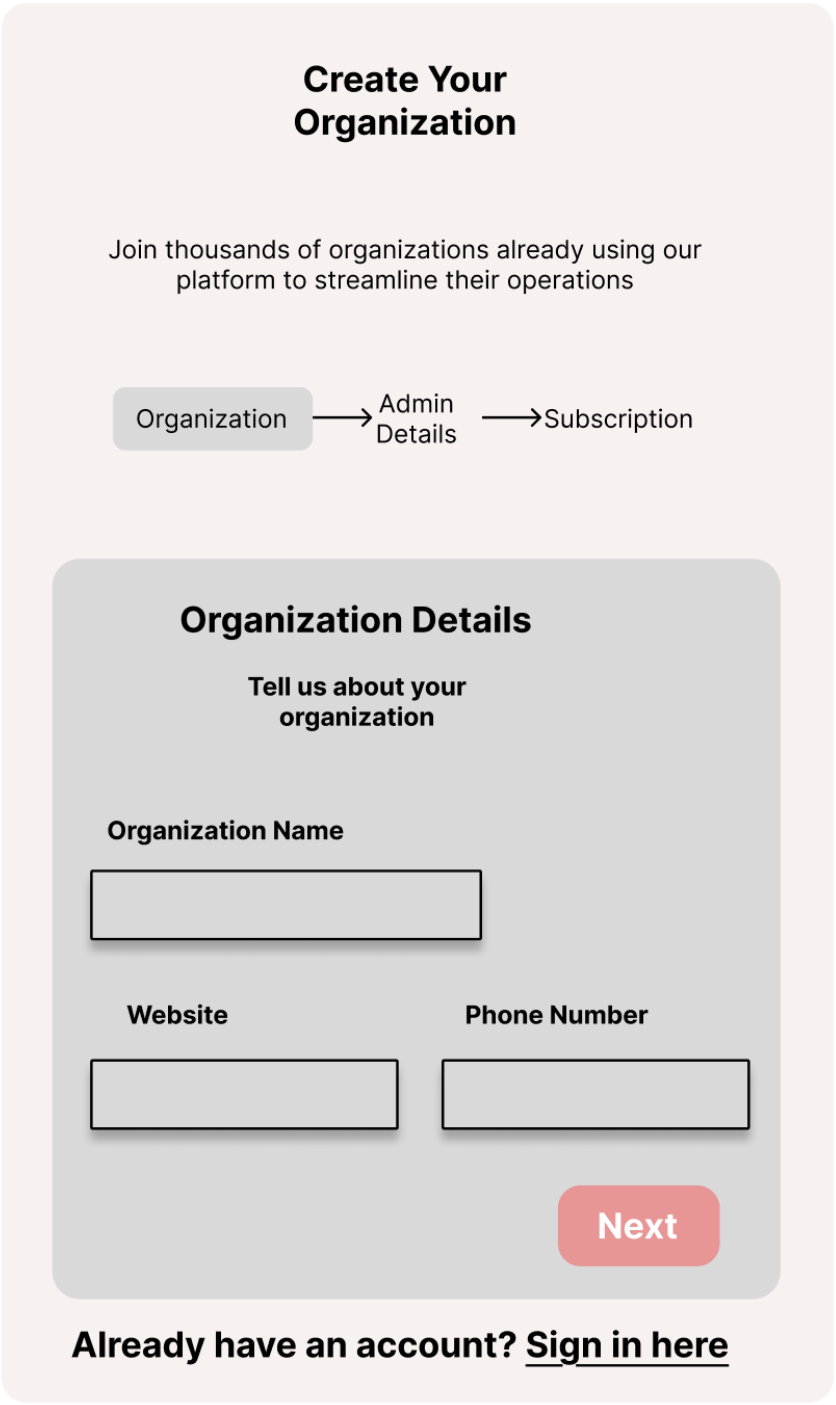


Figure 3-12 Subscription Page

3.2.3 Real-time Communication Design

WebSocket Architecture

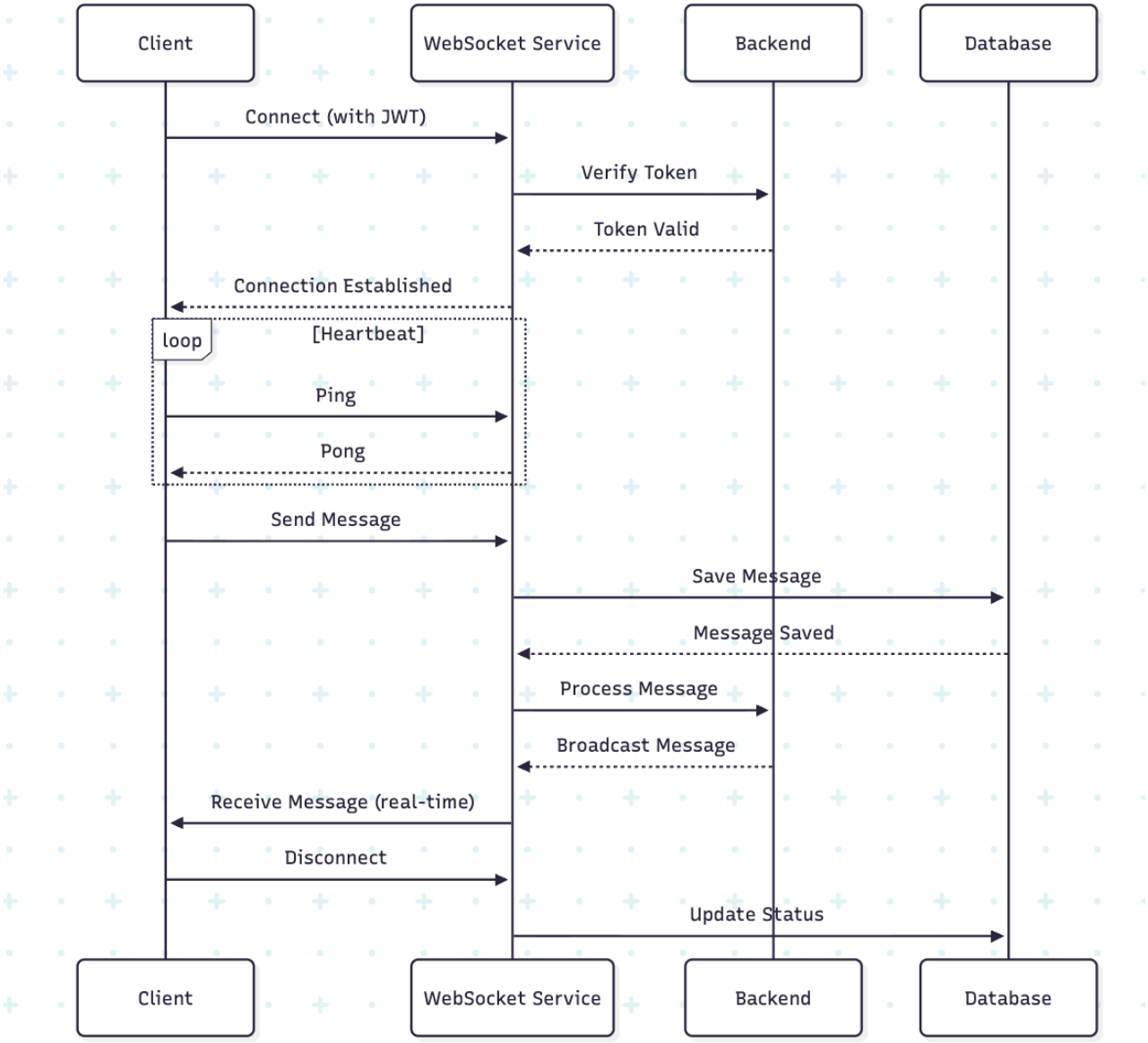


Figure 3-13 WebSocket Architecture

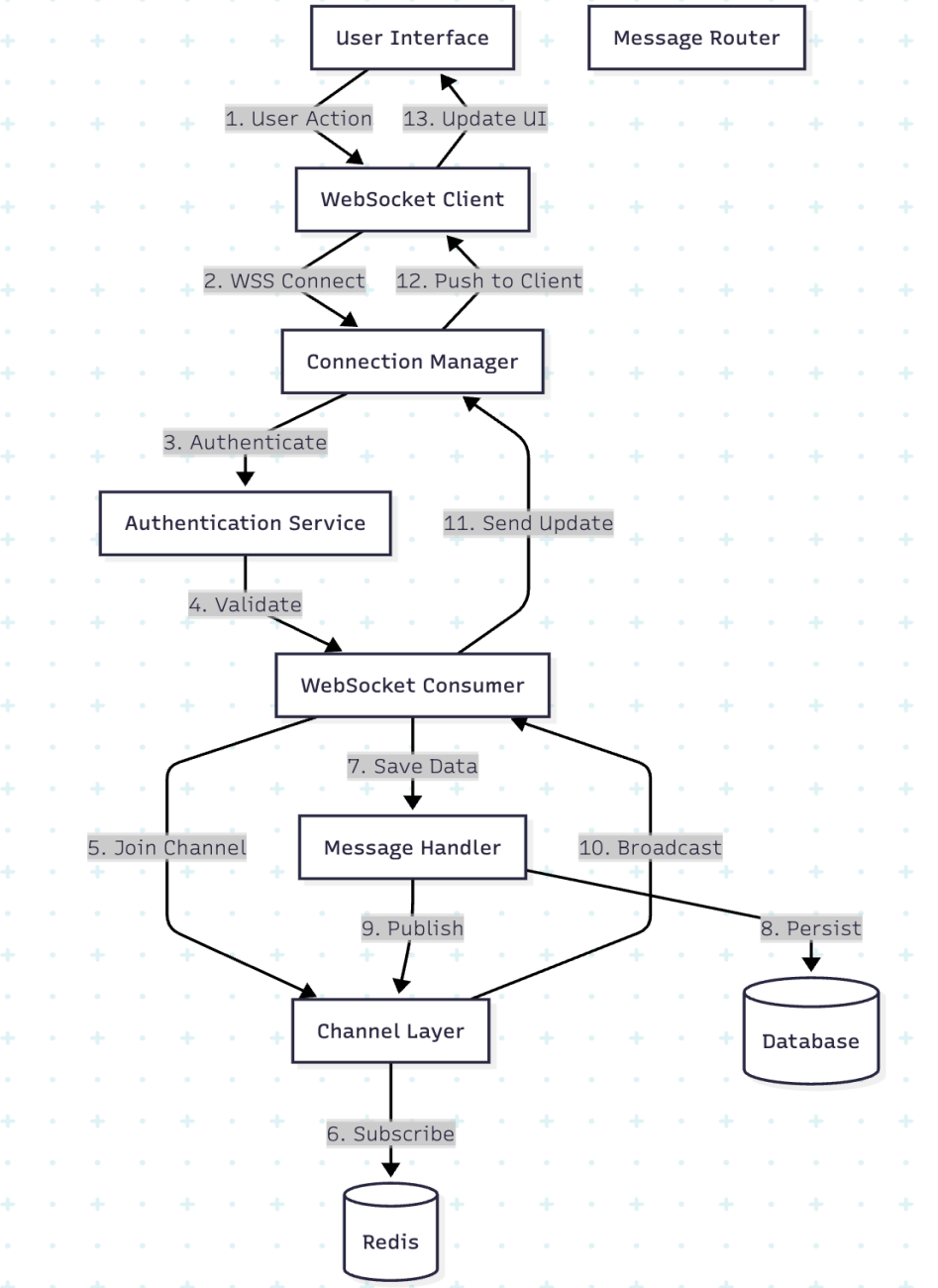


Figure 3-14 Message Flow

Key Components

Client Layer (React)

- WebSocket Client: Manages WebSocket connections
- Message Queue: Stores messages during connection issues
- State Management: Tracks connection status and message state

WebSocket Service

- Connection Manager: Handles WebSocket connections
- Authentication: Validates user sessions
- Message Router: Routes messages to appropriate channels

Backend Layer (Django Channels)

- Consumers: Handle WebSocket connections and message processing
- Channel Layers: Enable group communication
- Authentication Middleware: Validates WebSocket connections

Data Layer

- PostgreSQL: Stores message history and conversation data
- Redis: Manages WebSocket channel layers and pub/sub

Data Flow

- Connection Establishment:

  Client → WebSocket Service → Django Channels (Authentication) → Database
- Message Sending:

  Client → WebSocket Service → Channel Layer → All connected clients
- Presence Updates:

  Client (status change) → WebSocket Service → Channel Layer → Other clients

Scalability Features

- Horizontal Scaling: Multiple WebSocket servers behind a load balancer
- Channel Layer: Redis for cross-server communication
- Connection Pooling: Efficient management of WebSocket connections

Security Measures

- WSS Protocol: Encrypted WebSocket connections
- JWT Authentication: For WebSocket connections
- Rate Limiting: Prevents abuse
- Input Validation: For all incoming messages

3.2.4 Security Design

Authentication & Authorization

- JWT-based authentication for WebSocket connections
- Token validation on connection
- Role-based access control for conversations
- Message ownership verification

Data Protection

- WebSocket over WSS (secure)
- Message encryption in transit (TLS 1.3)
- Input validation and sanitization
- Rate limiting for WebSocket connections

Security Measures

- CSRF protection for WebSocket upgrade requests
- Origin validation
- Session management
- Audit logging for sensitive operations

3.3 Algorithm Used

3.3.1 Authentication & Security

- JWT (JSON Web Token) Algorithm: HMAC SHA-256 (HS256) or RS256
- Password Hashing: bcrypt (Blowfish-based)
- Rate Limiting: Token Bucket Algorithm

3.3.2 Data Structures & Algorithms

Searching: Binary Search (for sorted data)

Sorting:

- Quick Sort (for general-purpose sorting)
- Merge Sort (for stable sorting)
- Tim Sort (used by Python's built-in sort)

3.3.3 Database

- Indexing: B-Tree (PostgreSQL default)
- Query Optimization: Cost-based optimization

CHAPTER 4 :

IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Real-time Messaging Implementation

Tech Stack:

- React + Django Channels

Frontend:

```
// Connect
    const ws = new WebSocket('wss://api.bms.com/ws?token=jwt_token');
    // Send message
    ws.send(JSON.stringify({
      type: 'message',
      content: 'Hello',
      conversation_id: '123'
    }));
```

 # consumers.py

```
class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        await self.accept()
    async def receive(self, text_data):
        data = json.loads(text_data)
        await self.save_message(data)
        await self.broadcast(data)
```

- PostgreSQL + Redis
- JWT Auth

Features:

- Real-time chat
- Read receipts
- Typing indicators

- Online status

Security:

- WSS
- Rate limiting
- Input validation

Performance:

- Message queuing
- Auto-reconnect
- Efficient updates

4.1.2 Performance Optimization

- **Message Batching**: Group multiple updates into single WebSocket messages
- **Lazy Loading**: Load message history on demand
- **Caching**: Cache frequently accessed data in Redis
- **Connection Pooling**: Reuse WebSocket connections

4.1.3 Error Handling

- Automatic reconnection with exponential backoff
- Message queuing during offline
- Graceful degradation when WebSocket is unavailable

4.1.4 Development Environment

- Hardware Requirements
- Software Requirements
- Setup Instructions

4.1.5 Module-wise Implementation

- Authentication Module
- User Management Module
- Project Management Module
- Task Management Module
- Reporting Module

4.1.6 Integration

Frontend-Backend:

- RESTful API integration using Axios
- JWT token-based authentication

- WebSocket for real-time features

Third-party Services:

- AWS S3 for file storage
- SendGrid for email notifications
- Google OAuth for social login

4.2 Testing

4.2.1 Test Cases for Unit Testing

Unit testing: Unit testing is a fundamental practice in software development that involves testing individual components or units of code in isolation to ensure they function correctly.

S.N.	Test Case Description	Input	Expected Result	Actual Result	Status
1.	User forget to enter a particular field	name: Ramesh Package: Basic Organization name: Evolve Email: evolve@gmail.com Password: abcd1234	Display message please fill in this field.	As expected result.	Pass
2.	User enters invalid email formats	Full name: Ramesh Email : ramesh123 Password: abcd1234	Display message please enter an email address.	As expected result	Pass

3.	User enters all the details successfully	name: Ramesh Package: Basic Organization name: Evolve Phone:+977 01523838 Website: evolve.com.np Email: evolve@gmail.com Password: abcd1234	Subscription successful. You may login now. User appears in user list	As expected result	Pass

Table 4. 1: Subscription Registration Test for BMS

This test case examines the user registration procedure within the system, with a primary goal of validating the input fields to guarantee the accurate and comprehensive recording of user information. The testing primarily targets three distinct scenarios: instances where a specific field is omitted, situations where an email is provided in an incorrect format, and cases where all details are accurately entered.

Table 4. 2: Admin Login Test for BMS

S.N.	Test Case Description	Input	Expected Result	Actual Result	Status
1	Admin enters a wrong email	Email: ramesh@gmail.com Password: ramesh123	Display message user not registered	As expected	Pass
2	Admin enters correct email and incorrect password.	Email: Ramesh123@gmail.com Password: Ramesh@99	Display message incorrect password	As expected	Pass
3	Admin enters correct email and password.	Email: Ramesh123@gmail.com Password: ramesh123	Redirects admin to dashboard.	As expected	Pass

This test scenario assesses the functionality of the admin login procedure within the system, encompassing diverse situations like entering an incorrect email, an of the admin authentication mechanism, ensuring it functions as designed and delivers relevant feedback to the user incorrect password, and a successful login. The primary objective is to verify the proper operation

Table 4. 3: User Login Test for BMS

S.N.	Test Case Description	Input	Expected Result	Actual Result	Status
1	User enters a wrong email	Email: rameshdan@gmail.com Password: Ramesh@123	Display user not registered	As expected	Pass
2	User enters correct email and incorrect password	Email: rameshRawat@gmail.com Password: Ramesh1234	Display message incorrect password	As expected	Pass
3	User enters correct email and password.	Email: rameshRawat@gmail.com Password: Ramesh@123	Redirects user to user dashboard.	As expected	Pass

This test case addresses multiple scenarios involved in the user login process, encompassing instances of entering an incorrect email, providing an incorrect password, and achieving a successful login.

Table 4. 4: End-to-End messaging test

S.N.	Test Case Description	Input	Expected Result	Actual Result	Status
1	User1 sent a message	Message:" Hi"	User2 receive real-time message	As expected,	Pass
2	User2 sent a message	Message:" Hi"	User1 receive real-time message	As expected,	Pass

This test scenario assesses the operational aspects of the message management, it checks weather messages sending and receiving is applied properly or not.

Table 4. 5: Searching user test

S.N.	Test Case Description	Input	Expected Result	Actual Result	Status
1	Searching a user with name	Searched "Ramesh"	Displays Ramesh Rawat	Ramesh Rawat	Pass
2	Searching a user email	Searched "rawatramesh@gmail.com"	Displays Ramesh Rawat	Ramesh Rawat	Pass

This test case addresses user search scenarios with user's name and username.

CHAPTER 5 :

CONCLUSION AND FUTURE RECOMENDATION

5.1 Lesson Learnt

Technical Implementation

- Successfully integrated WebSocket for real-time communication
- Implemented JWT authentication for secure WebSocket connections
- Ensured data consistency with proper synchronization
- Real-time updates significantly enhance user engagement
- Responsive design is essential for cross-device compatibility
- Handled offline scenarios gracefully

5.2 Conclusion

The BMS project successfully delivered a robust, real-time business management solution that addresses core operational needs. The implementation demonstrates the effectiveness of modern web technologies in creating responsive, scalable applications. The system's architecture provides a solid foundation for future enhancements while maintaining high performance and reliability.

Key achievements include:

- Seamless real-time communication
- Intuitive user interface
- Secure and scalable backend
- Comprehensive feature set for business management

5.3 Future Improvements

Enhanced Security & Compliance

- Implement end-to-end encryption for sensitive communications
- Add multi-factor authentication (MFA) for all user accounts
- Conduct regular security audits and penetration testing
- Ensure GDPR and other regulatory compliance

Advanced Analytics & AI Integration

- Add AI-powered business insights and predictive analytics
- Implement automated report generation
- Develop custom dashboards with real-time KPIs
- Integrate natural language processing for data queries

Mobile & Offline Capabilities

- Develop dedicated iOS and Android applications
- Implement offline data synchronization
- Add mobile-optimized workflows
- Enable push notifications for critical updates

Integration & Ecosystem Expansion

- Create a public API for third-party integrations
- Develop pre-built connectors for popular business tools
- Build a marketplace for plugins and extensions
- Implement webhook support for event-driven automation

These recommendations focus on security, intelligence, mobility, and extensibility to ensure the BMS remains competitive and valuable to users.

REFERENCES

[1] A. &. J. B. Smith, "The Impact of Integrated Business Management Systems on Operational Efficiency," Journal of Business Technology, 2022.

[2] G. B. T. Report, "Cloud-Based Business Solutions: Trends and Opportunities," 2023.

[3] L. W. H. &. K. S. Chen, "Mobile Accessibility in Business Management Systems: Impact on Productivity and Employee Satisfaction," 2023.

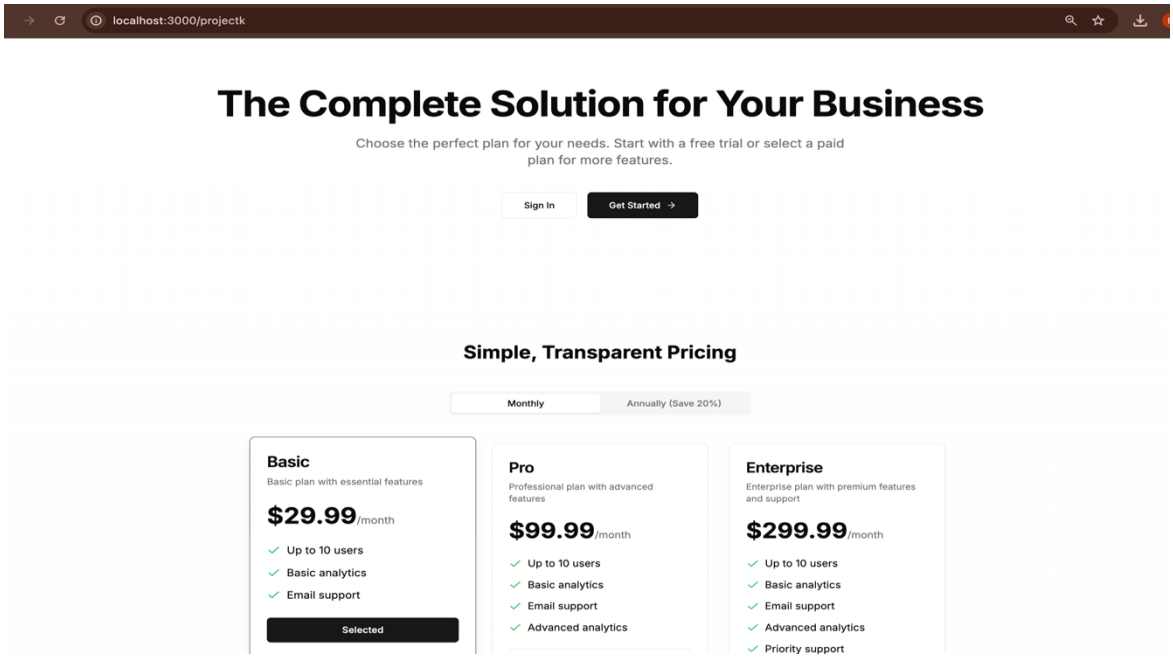
[4] C. Ventures, "Report on Cybersecurity in Business Systems," 2023.

[5] R. &. W. E. Thompson, "The Role of BMS in Enhancing SME Performance: A Longitudinal Study," 2023.

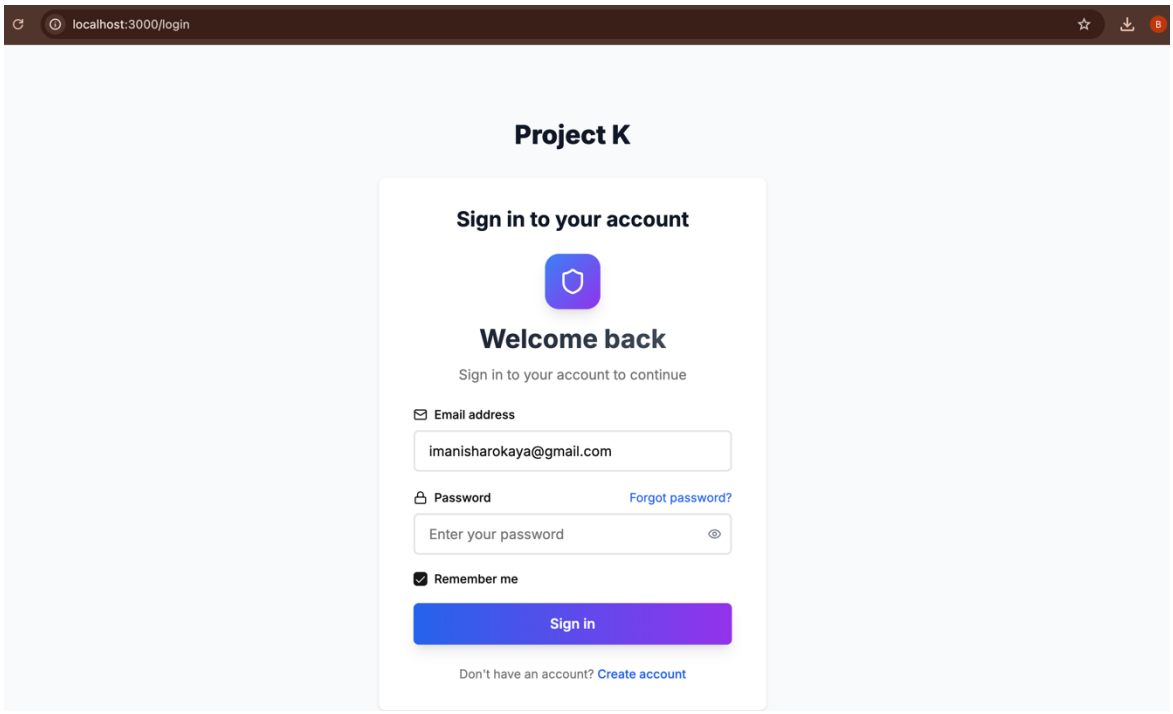
[6] Y. L. X. &. P. N. Zhang, "Emerging Technologies in Business Management: IoT and Blockchain Integration," 2023.

APPENDICES: SYSTEM SCREENSHOTS

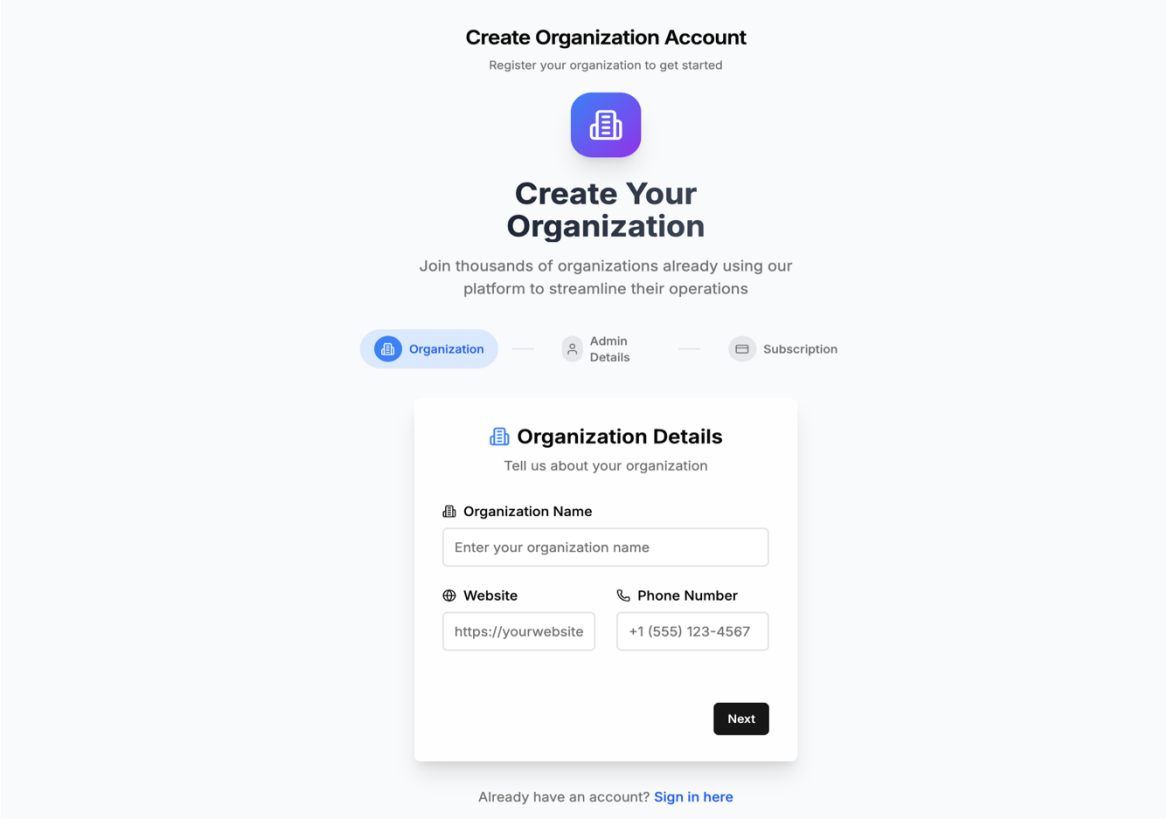
i. Landing Page



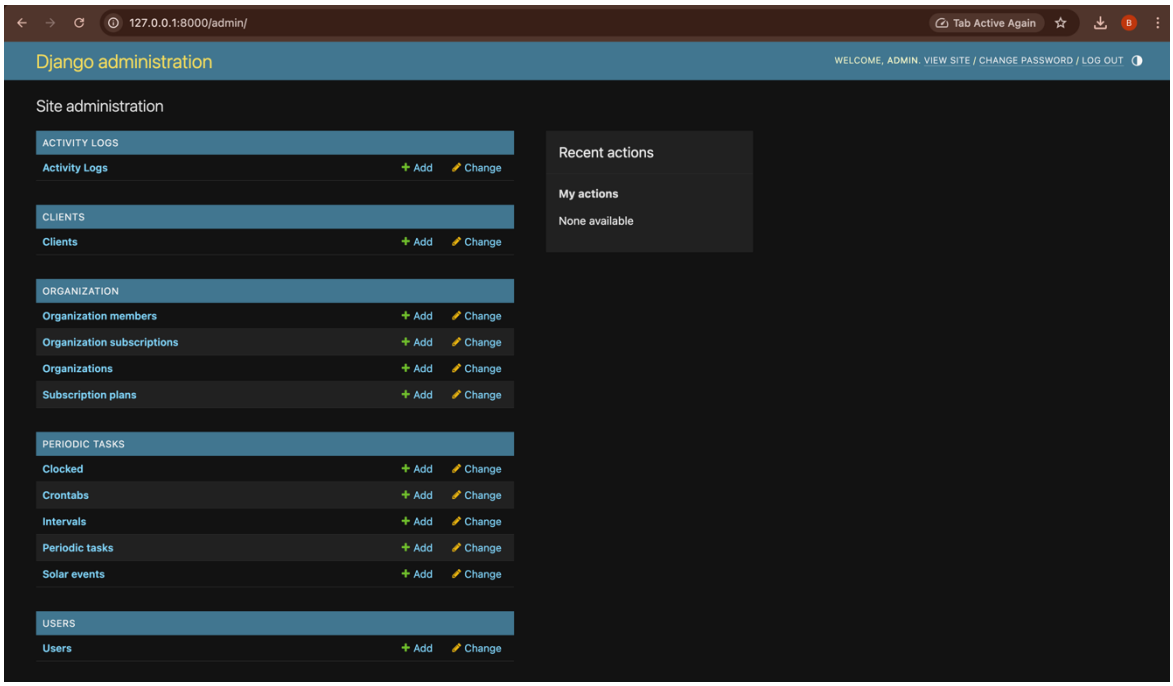
ii. Login page



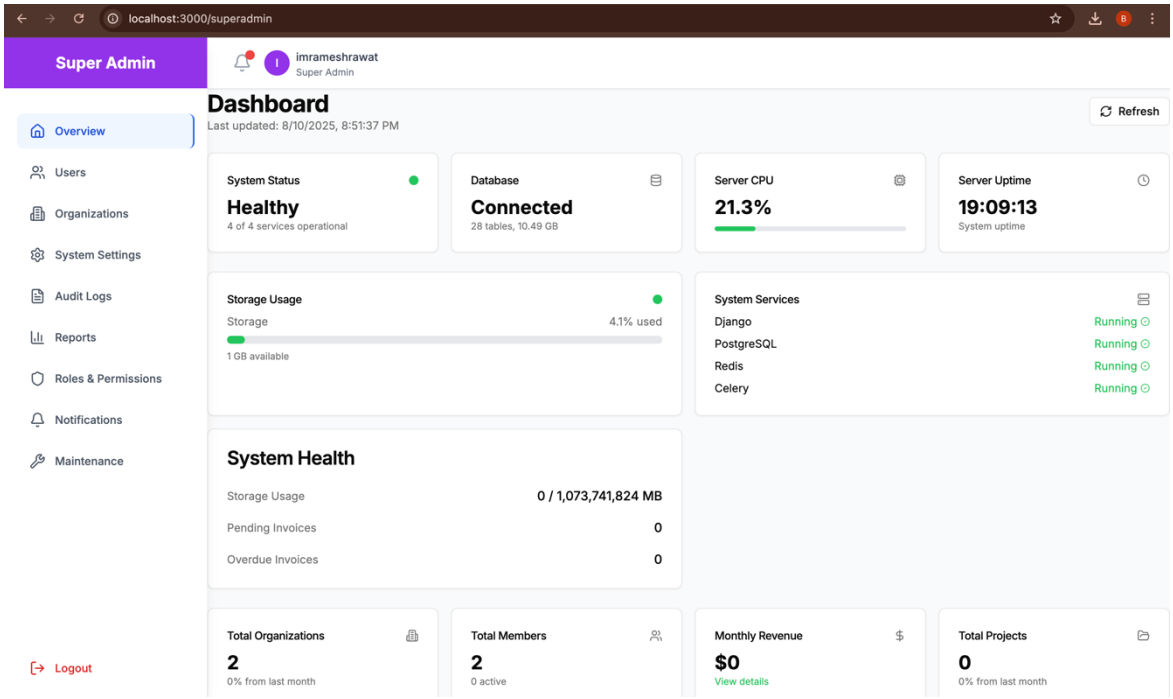
iii. Organization Register page



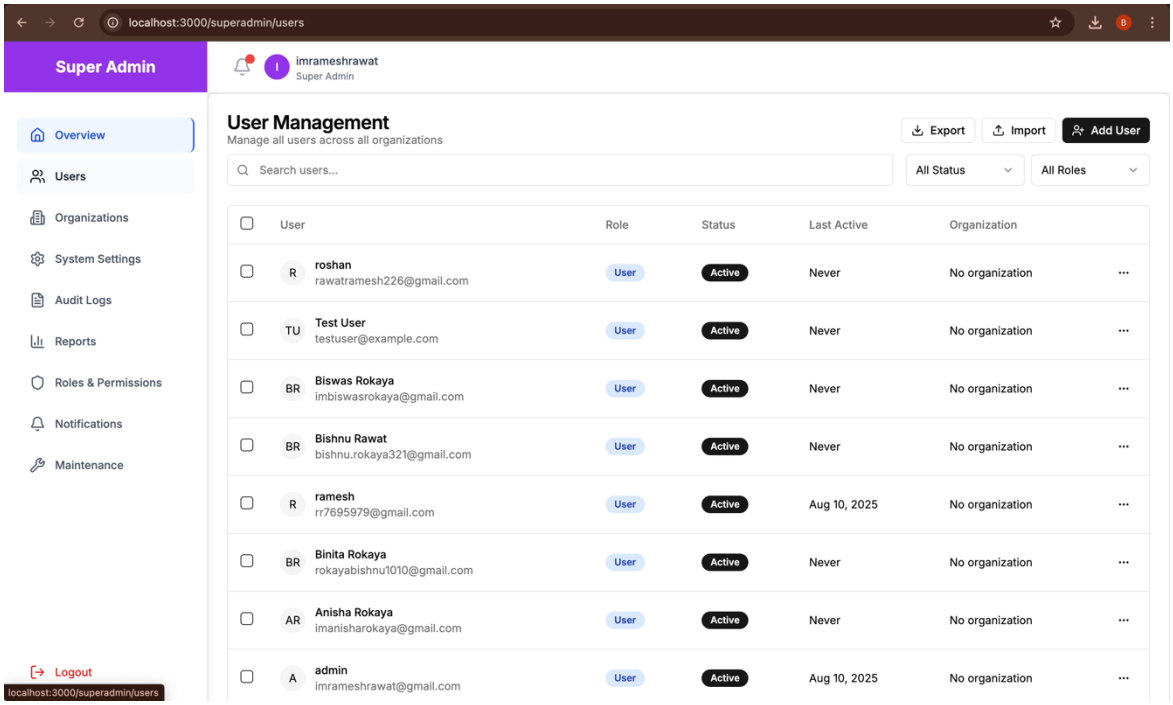
iv. Django admin panel



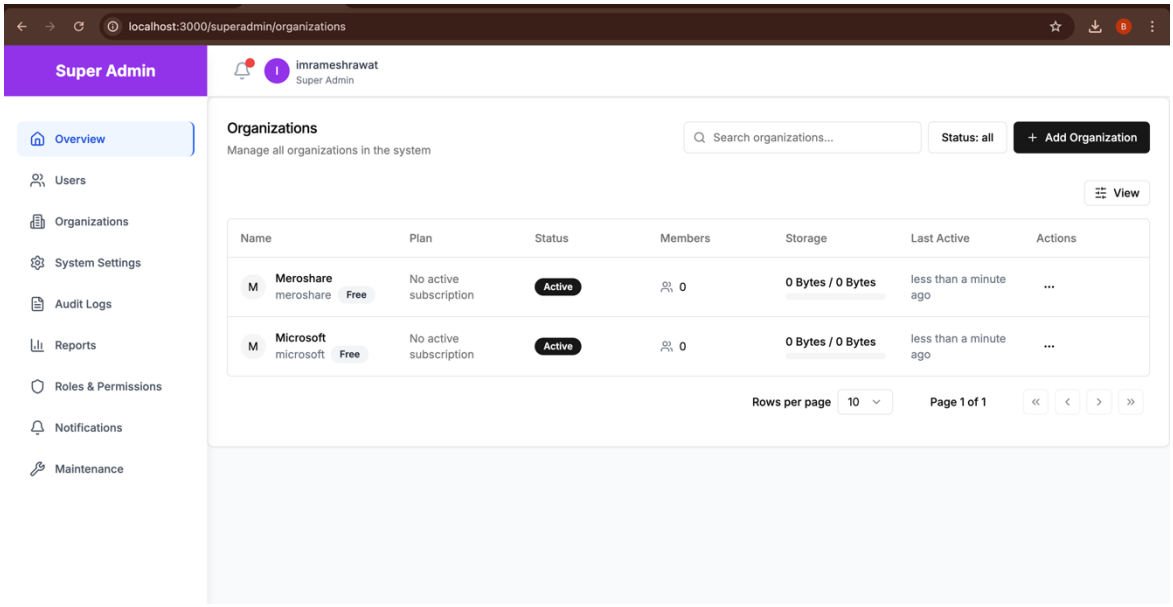
v. Super-admin Dashboard



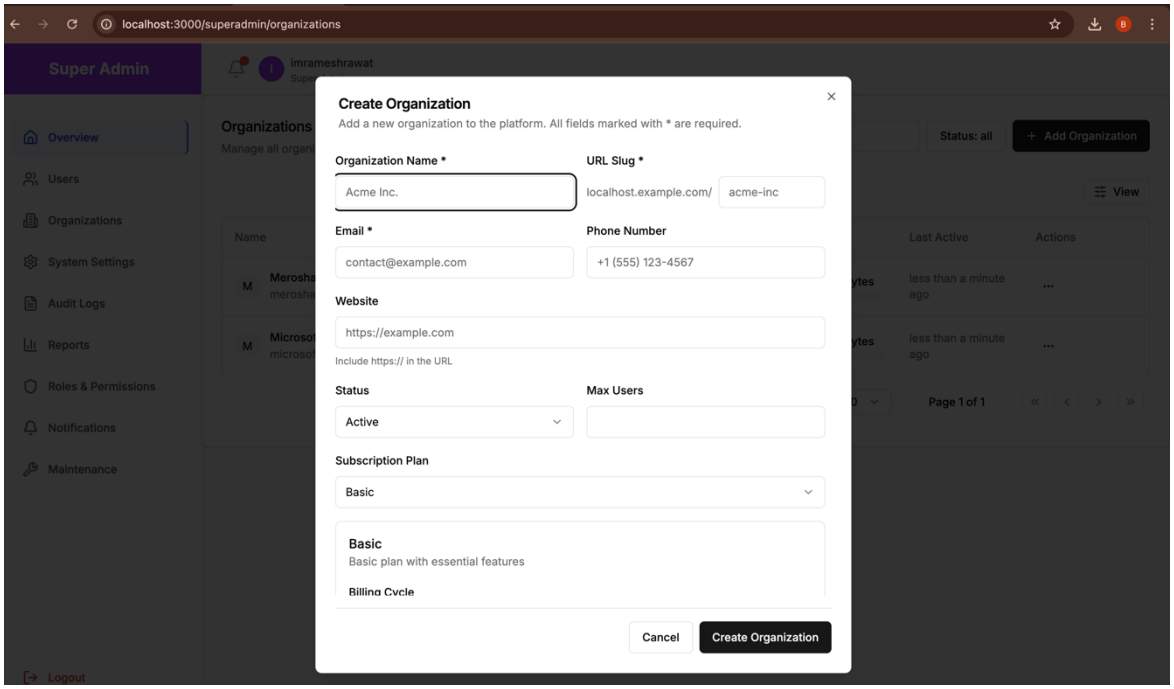
vi. Super admin User management



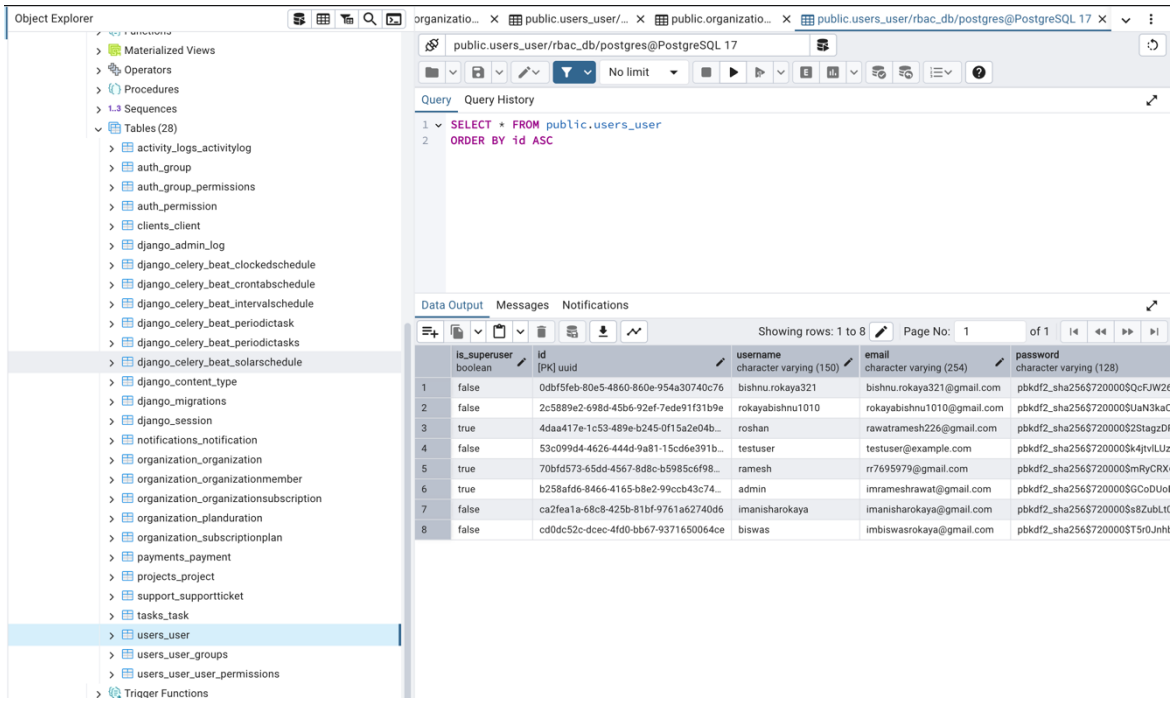
vii. Super admin organizations management



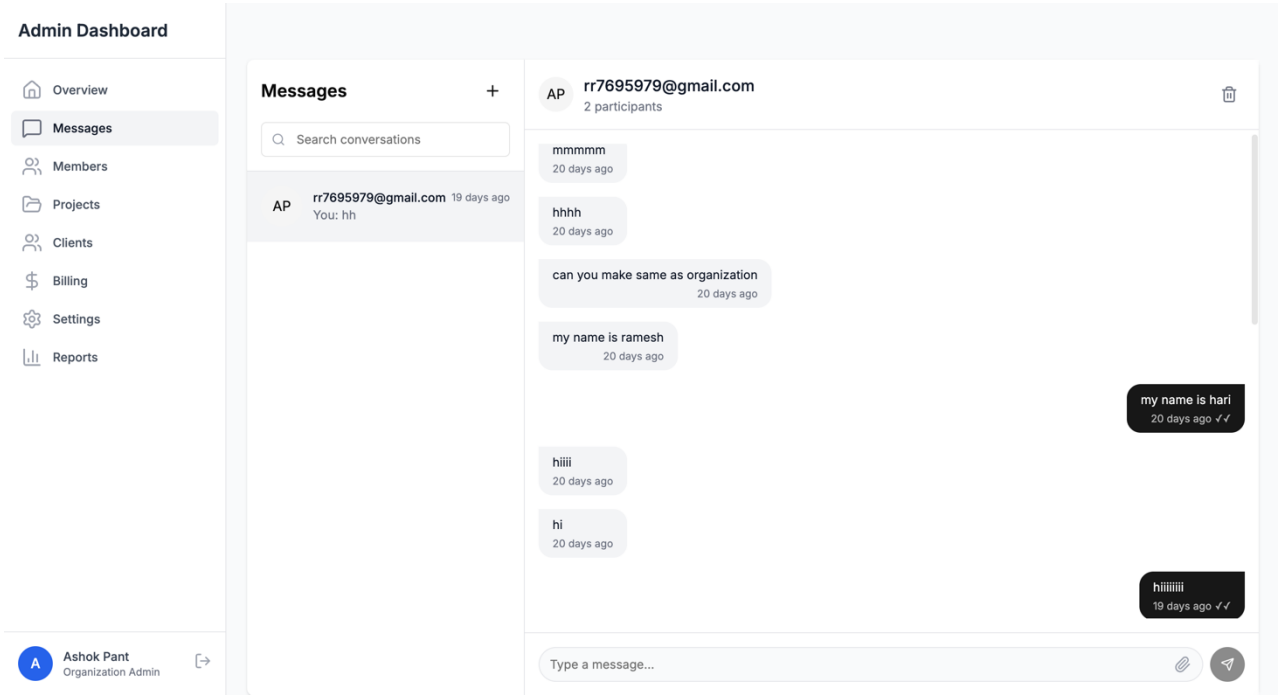
viii. Super admin creates organization



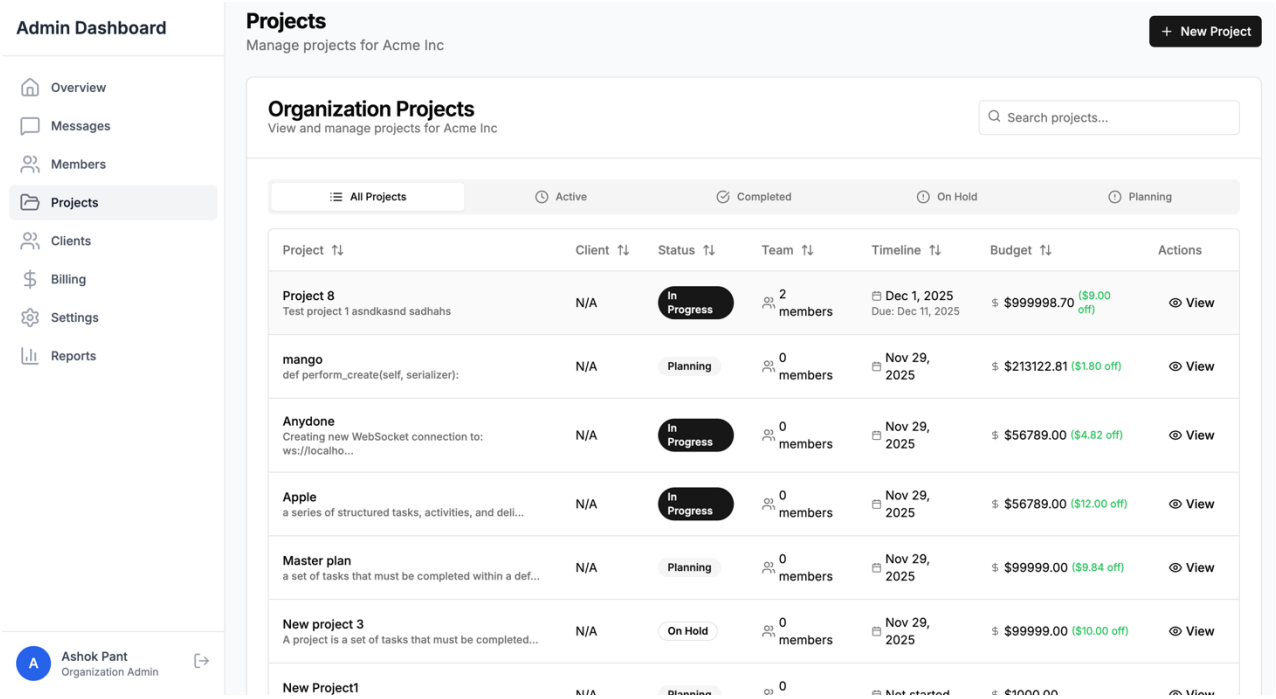
ix. Database



x. Messaging



xi. Project Page



xii. Client Page