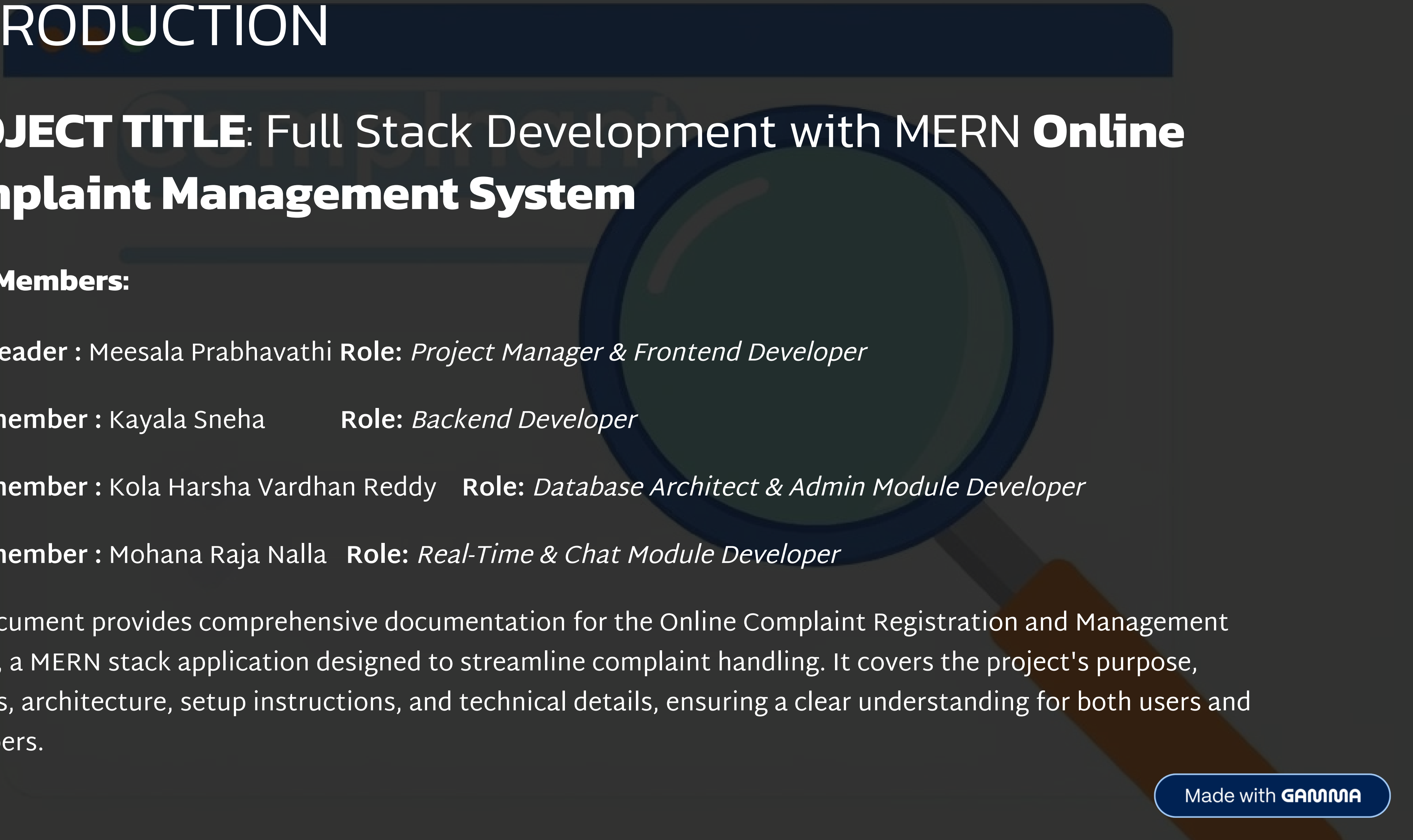


INTRODUCTION

PROJECT TITLE: Full Stack Development with MERN **Online Complaint Management System**

Team Members:

Team Leader : Meesala Prabhavathi **Role:** *Project Manager & Frontend Developer*

Team member : Kayala Sneha **Role:** *Backend Developer*

Team member : Kola Harsha Vardhan Reddy **Role:** *Database Architect & Admin Module Developer*

Team member : Mohana Raja Nalla **Role:** *Real-Time & Chat Module Developer*

This document provides comprehensive documentation for the Online Complaint Registration and Management System, a MERN stack application designed to streamline complaint handling. It covers the project's purpose, features, architecture, setup instructions, and technical details, ensuring a clear understanding for both users and developers.

Project Overview

The Online Complaint Registration and Management System is a user-friendly software solution designed to streamline the process of submitting, tracking, and resolving complaints or issues encountered by individuals or organizations. It provides a centralized platform for efficient complaint management, allowing users to securely register complaints, track their progress in real-time, and interact with assigned agents for issue resolution.

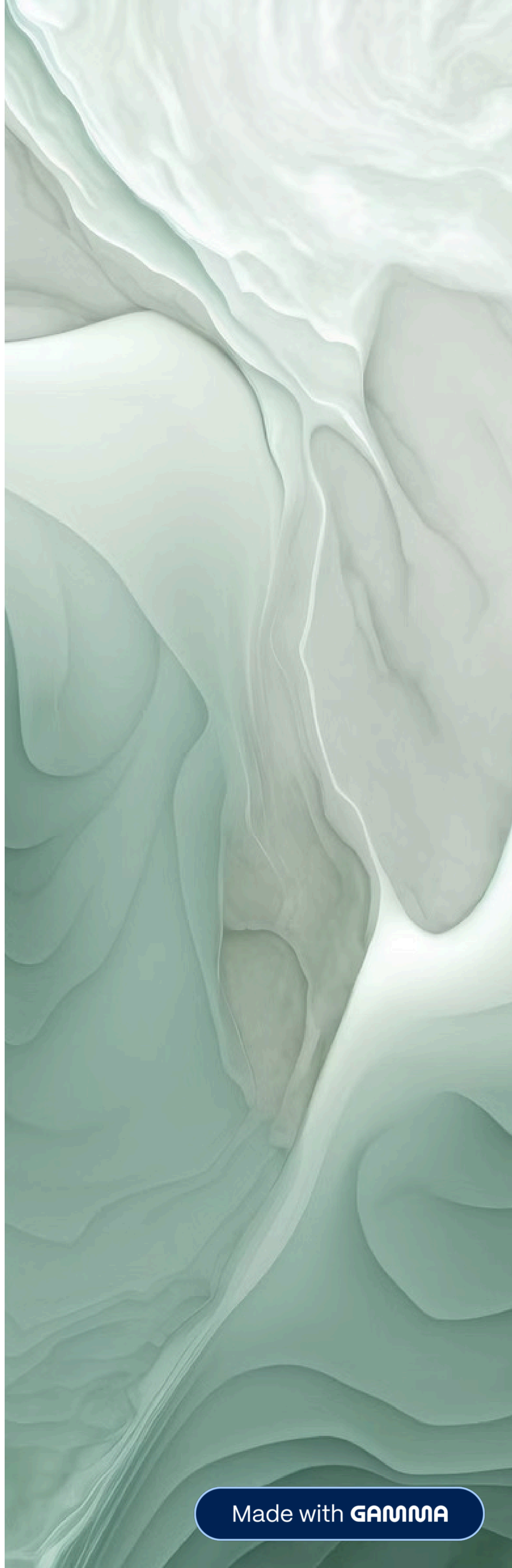
Key features include user registration, detailed complaint submission, real-time tracking with notifications, intelligent complaint routing, and robust security measures. This system ensures timely and effective handling of complaints while prioritising user data security and confidentiality.

Purpose

To optimise the complaint handling process and empower organisations to efficiently resolve customer complaints, aligning with industry guidelines and regulatory compliance obligations.

Features

- User registration and account management.
- Detailed complaint submission with attachments.
- Real-time tracking and notifications (email/SMS).
- Agent interaction via built-in messaging.
- Intelligent routing of complaints to appropriate personnel.
- Robust security and confidentiality measures.



Technical Architecture

The Online Complaint Registration and Management System follows a client-server model, leveraging the MERN stack for its robust and scalable architecture. The frontend serves as the client, providing the user interface, while the backend acts as the server, handling logic and data management.

Frontend

Built with React, the frontend incorporates the Axios library for seamless communication with backend RESTful APIs. Bootstrap and Material UI libraries are utilised to create a responsive and intuitive user interface, ensuring a superior experience for all users, including ordinary users, agents, and administrators.

Backend

The backend is powered by Node.js and the Express.js framework, managing server-side logic and API communication. It handles data storage and retrieval, ensuring efficient and scalable operations. The integration of Socket.io and WebRTC API facilitates real-time communication within the system.

Database

MongoDB is employed for data storage and retrieval, offering efficient and scalable management of user profiles, complaint registrations, and other essential data. This NoSQL database ensures reliable and quick access to information, crucial for a responsive complaint management system.

API Documentation & Authentication

This section details the RESTful API endpoints exposed by the backend and explains the authentication and authorisation mechanisms implemented in the system.

API Documentation

The backend exposes various endpoints for managing users and complaints. Each endpoint specifies the request method, required parameters, and expected response formats. Below are examples of common endpoints:

- **POST /api/auth/register**: User registration. Parameters: name, email, password. Response: user object, token.
- **POST /api/auth/login**: User login. Parameters: email, password. Response: user object, token.
- **POST /api/complaints**: Submit a new complaint. Parameters: title, description, category, userId. Response: new complaint object.
- **GET /api/complaints/:id**: Retrieve a specific complaint. Response: complaint object.
- **PUT /api/complaints/:id/status**: Update complaint status. Parameters: status. Response: updated complaint object.

Authentication

Authentication and authorisation are handled using JSON Web Tokens (JWTs). Upon successful login or registration, a JWT is issued to the user. This token must be included in the Authorization header of subsequent requests to protected routes.

The backend validates the JWT to verify the user's identity and permissions. Middleware functions are used to protect routes, ensuring that only authenticated and authorised users can access specific resources or perform certain actions.

User Interface & Testing

The user interface (UI) is designed for intuitive navigation and a seamless user experience. The system also incorporates a robust testing strategy to ensure reliability and functionality.

User Interface

The UI provides distinct dashboards and functionalities for different user roles: regular users, agents, and administrators. Key UI features include:

- **User Dashboard**: Allows users to submit new complaints, view their complaint history, track status updates, and interact with assigned agents via a built-in chat.
- **Agent Dashboard**: Enables agents to view assigned complaints, update their status, and communicate with users.
- **Admin Dashboard**: Provides administrators with an overview of all complaints, user management tools, and the ability to assign complaints to agents.

Screenshots or a link to a demo can be provided to showcase the application's visual aspects and functionality.

Testing

A comprehensive testing strategy is employed to ensure the system's stability and correctness. This includes:

- **Unit Testing**: Individual components and functions are tested in isolation to ensure they perform as expected.
- **Integration Testing**: Tests the interaction between different modules and services, such as frontend-backend communication and database interactions.
- **End-to-End Testing**: Simulates real user scenarios to verify the entire application flow, from user input to backend processing and UI updates.

Tools such as Jest and React Testing Library are used for frontend testing, while Mocha and Chai are utilised for backend API testing.

Known Issues & Future Enhancements

This section outlines any identified bugs or limitations within the current system and proposes potential improvements for future development.

Known Issues

While the system is designed for robust performance, some minor issues or limitations may exist:

- Occasional delays in real-time notifications under heavy load.
- Limited customisation options for email/SMS templates.
- File upload size restrictions for complaint attachments.
- No offline mode support for users.

These issues are being monitored and will be addressed in future updates.

Future Enhancements

Several features and improvements are planned to enhance the system's capabilities and user experience:

- Integration with third-party communication platforms (e.g., Slack, Microsoft Teams) for agent notifications.
- Advanced reporting and analytics for administrators to gain insights into complaint trends.
- Multi-language support for a broader user base.
- AI-powered intelligent routing for more precise complaint assignment.
- Mobile application development for on-the-go complaint management.
- Enhanced security features, including two-factor authentication (2FA).