PROJECT REPORT ABSTRACT

# 1. Title of the Project:

Django Real-Time Chat Application

# 2. Abstract:

The Django Chat Application provides a real-time communication platform for users to chat seamlessly and securely over the web. The system is built using the Django framework and Django Channels to support WebSocket-based messaging. This application is intended for use within institutions, companies, or any organization that requires private and secure internal communication.  
  
The system allows users to register, log in, send and receive instant messages, manage contacts, and view message history. Its lightweight interface and real-time updates offer an efficient alternative to traditional communication tools. The application is designed with security, scalability, and user-friendliness in mind.

## 1.2 Purpose of the Project

Effective internal communication is critical for any organization. Many messaging platforms are external or third-party tools that lack privacy and customizability. This project addresses that gap by offering a customizable, secure, and real-time messaging platform built specifically for internal use.

The purpose of the system is:  
- To reduce communication delays in workplaces or educational environments  
- To improve message management, history tracking, and team collaboration  
- To offer an easily maintainable and extendable platform using Django

## 1.3 Problems in the Existing System

• No Internal Privacy: Most existing chat apps are hosted externally  
• Delay in Communication: Email and manual notifications are slow  
• Data Redundancy: No centralized control over stored communication  
• Security Issues: Sensitive data may be exposed in public apps

## 1.4 Solution to These Problems

• Real-Time Chat: Implemented using WebSockets for instant messaging  
• User Authentication: Secure access using Django's auth system  
• Database-Driven: Stores messages, contacts, and logs efficiently  
• Highly Maintainable: Built with modular Django structure  
• Scalable & Secure: Easily expandable for new features like file sharing, group chat, etc.

## System Workflow Example

[User] → [Login] → [Chat Room List] → [Join/Create Room] → [Send/Receive Messages] → [Logout]

## System Analysis – DFD (Level 0)

[User]  
 |  
 |---> [Login/Registration System] ---> [User Database]  
 |  
 |---> [Chat System]  
 |  
 |---> [Message Handler] ---> [Message Database]  
 |  
 |<--- [WebSocket Server for Real-Time Updates]

# Hardware & Software Requirements

Hardware:  
- HDD: 20 GB minimum  
- RAM: 2 GB minimum  
- Processor: Core i3 or above

Software:  
- OS: Windows/Linux  
- Backend: Django 5.x, Django Channels  
- Frontend: HTML, CSS, JavaScript  
- Database: SQLite or MySQL/PostgreSQL  
- Browser: Chrome / Firefox