**ChatterCloud**

**Real-Time Chat Application with MongoDB and AWS Integration**

**Abstract:**

The rapid growth of internet usage and cloud technologies has led to a significant demand for real-time communication platforms. ChatterCloud is a real-time chat application that leverages modern technologies such as MongoDB for database management and Amazon Web Services (AWS) for scalable cloud-based storage, designed to provide users with seamless, instant messaging and media-sharing capabilities. This project aims to build a highly efficient and scalable chat system using a microservices architecture with Flask as the back-end framework, Flask-SocketIO for WebSocket-based communication, and AWS S3 for handling media uploads.

MongoDB is chosen as the primary database due to its flexibility in storing both structured and unstructured data, making it ideal for real-time applications that handle dynamic user-generated content such as text, images, and multimedia files. AWS S3, with its durability, scalability, and security, serves as the media storage platform for the application. Users can upload and share files such as images, videos, and documents, which will be stored in the AWS S3 bucket and served back to the user via secure links.

The system architecture incorporates WebSockets to enable full-duplex communication channels between the client and the server, ensuring real-time message transmission. Flask-SocketIO allows the server to push updates to clients in real-time, eliminating the need for constant polling and reducing bandwidth usage. This ensures a highly responsive and dynamic user experience, where messages, notifications, and shared files are instantly visible to users in various chat rooms.

The development of this chat application will follow the Software Development Life Cycle (SDLC) methodology, ensuring that every aspect, from planning to deployment, is carefully managed and executed. The project will be broken down into well-defined stages, starting with requirement gathering, design, and system setup, followed by coding and development, testing, and finally, deployment.

**Tools Required:**

* Visual Studio
* MongoDb
* AWS

**7-Week Schedule:**

Week 1: Project Planning and Setup

* Define project scope and features.
* Set up the development environment.
* Install necessary tools
* Set up a MongoDB instance
* Create an AWS S3 bucket for media storage.

Week 2: MongoDB Integration and Flask Setup

* Design the database schema for user and message storage.
* Set up a basic Flask application.
* Integrate MongoDB with Flask
* Test basic data flow from Flask to MongoDB

Week 3: Real-Time Messaging with WebSockets

* Implement message broadcasting functionality across rooms.
* Create basic HTML templates
* Test WebSocket connections to ensure real-time communication works smoothly.

Week 4: AWS S3 Integration for File Uploads

* Integrate AWS S3 with the Flask app.
* Allow users to upload files (images, videos, etc.) in chat.
* Store file metadata in MongoDB along with messages.
* Ensure files are securely uploaded to and accessible from AWS S3.

Week 5: User Authentication and Security Enhancements

* Implement user authentication
* Secure WebSocket connections to allow only authenticated users.
* Store user session data and chat history in MongoDB.

Week 6: Frontend Enhancements and UI Improvements

* Improve the front-end UI using basic HTML
* Enhance the chat room interface to display user lists, timestamps for messages, and file previews.
* Perform user experience (UX) testing and gather feedback for improvements.

Week 7: Final Testing, Deployment, and Documentation

* Conduct final testing to ensure all functionalities work seamlessly.
* Debug and resolve any outstanding issues.