SentiChat

A Project Report

Submitted by

Yarra Khyatisree – AP23110010215

Gadipudi Praneeth – AP231100100292

Chunduru Suhas – AP23110010953

Nandamuri Chetan Ram –

AP23110010066

Under the Supervision of

Dr. Boyapati Prasanthi
Assistant Professor
Department of Computer Science and Engineering
SRM University-AP

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRM UNIVERSITY-AP

NEERUKONDA

MANAGALAGIRI - 522503

ANDHRA PRADESH, INDIA

AUGUST - 2025

CERTIFICATE

This is to certify that the project work entitled **"SentiChat"** is a Bonafide record of project work carried out by the following students:

- **Yarra Khyatisree** (Roll No.: AP23110010292)
- **Gadipudi Praneeth** (Roll No.: AP23110010055)
- **Chunduru Suhas** (Roll No.: AP23110010019)
- **Nandamuri Chetan Ram** (Roll No.: AP23110010066)

from the **Department of Computer Science and Engineering, SRM University-AP.** The students conducted this project work under my supervision during the period **June 2025 to July 2025**. It is further certified that, to the best of my knowledge, this project has not previously formed the basis for the award of any degree or any similar title to this or any other candidate.

This is also to certify that the project work represents the **teamwork** of the candidates.

Dr. Boyapati Prasanthi

Assistant Professor
Department of Computer Science & Engineering
SRM University-AP
Andhra Pradesh.

TABLE OF CONTENTS

Introduction

The SentiChat project is a web-based application designed to provide users with a seamless registration and login system integrated with an interactive chatbot interface. The application allows users to create accounts, log in securely, and engage in conversations with a sentiment-aware chatbot that adapts its responses based on the user's emotional tone. The chatbot, named SentiChat, uses natural language processing to analyze user inputs and maintain a dynamic conversation environment. The project aims to combine user authentication with an engaging chat experience, making it suitable for users seeking both functionality and interactivity.

The application is built using PHP for backend processing, MySQL for database management, HTML/CSS for the frontend, and JavaScript for dynamic interactions. Additionally, it integrates an external API for chatbot responses, enhancing its conversational capabilities.

Problem Definition

In today's digital world, users expect applications that are secure, user-friendly, and interactive. Many existing platforms either focus solely on user authentication or provide basic chat functionalities without personalization. The lack of integration between secure user management and emotionally intelligent chat systems creates a gap in delivering a holistic user experience.

This project addresses the need for a system that:

- Provides secure user registration and login mechanisms.

- Offers an interactive chatbot that can detect and respond to user sentiments.

- Ensures a responsive and visually appealing interface for seamless interaction.

Problem Statement

The objective is to develop a web application that combines a robust user authentication system with a sentiment-aware chatbot. The system should allow users to register, log in, and access a homepage displaying their profile information. Upon login, users should be redirected to a chatbot interface (SentiChat) that analyzes the sentiment of their messages and adjusts its responses accordingly. The application must ensure data security, ease of use, and an engaging conversational experience.

Objectives:

The primary objectives of the SentiChat project are:

1. User Authentication: Implement a secure registration and login system to manage user accounts.

2. Database Management: Store user credentials securely in a MySQL database with encrypted passwords.

3. Chatbot Integration: Develop a sentiment-aware chatbot that responds dynamically to user inputs.

4. Sentiment Analysis: Enable the chatbot to analyze user messages and adjust its tone based on detected emotions (e.g., happy, sad, neutral).

5. User-Friendly Interface: Design an intuitive and responsive interface for both authentication and chatbot interactions.

6. Cross-Platform Compatibility: Ensure the application is accessible across various devices and browsers.

Software Requirements Specifications

The software requirements for the SentiChat project are divided into functional and non-functional requirements.

Functional Requirements

- **User Registration:** Users can create accounts by providing their first name, last name, email, and password.

- **User Login:** Users can log in using their email and password, with validation to prevent unauthorized access.

- **Session Management:** Maintain user sessions to ensure secure access to the homepage and chatbot interface.

- **Chatbot Interaction:** Users can send messages to the chatbot, which responds based on sentiment analysis.

- **Sentiment Detection:** The chatbot identifies positive, negative, or neutral sentiments in user messages.

- **Theme Switching:** Users can toggle between light and dark themes for the chatbot interface.

- **Chat History Management:** Users can clear the chat history to start a fresh conversation.

Non-Functional Requirements

- **Security:** Passwords are encrypted using MD5 hashing (though a stronger hashing algorithm like bcrypt is recommended for production).

- **Performance:** The application should respond to user inputs within 2 seconds under normal conditions.

- **Scalability:** The system should handle multiple users simultaneously.

- **Usability:** The interface should be intuitive, with clear navigation and responsive design.

- **Compatibility:** The application should support modern browsers (Chrome, Firefox, Safari) and be responsive on mobile and desktop devices.

Hardware Requirements

- **Server:** A web server (e.g., Apache) with PHP and MySQL support.

- **Client:** Any device with a modern web browser.

- **Storage:** Minimal storage for the database (estimated < 100 MB for initial users).

Software Requirements

- **Operating System:** Any OS supporting a web server (e.g., Windows, Linux).

- **Backend:** PHP 7.4+, MySQL 5.7+.

- **Frontend:** HTML5, CSS3, JavaScript (ES6+).

- **External Libraries:**

  - Font Awesome for icons.

  - Marked.js for rendering Markdown in chatbot responses.

- **API:** OpenRouter API for chatbot responses.

- **Development Tools:** VS Code, XAMPP (for local testing), Git for version control.

Design (DFD Diagram)

The Data Flow Diagram (DFD) illustrates the flow of data within the SentiChat application. Below is a description of the Level-0 and Level-1 DFDs.

Level-0 DFD (Context Diagram)

- **Entities:** User, SentiChat System.

- **Data Flow:**

  - User → SentiChat System: Registration details (first name, last name, email, password), login credentials (email, password), chat messages.

  - SentiChat System → User: Registration confirmation, login status, homepage data, chatbot responses.

Level-1 DFD

- **PROCESSES:**

  - **Registration:** Validates user input, checks for duplicate emails, and stores data in the database.

  - **Login:** Verifies email and password against the database and initiates a session.

  - **Homepage Display:** Retrieves user data from the database and displays it.

  - **Chatbot Interaction:** Sends user messages to the API, processes responses, and updates the chat interface.

  - **Sentiment Analysis:** Analyzes user messages and adjusts chatbot behavior.

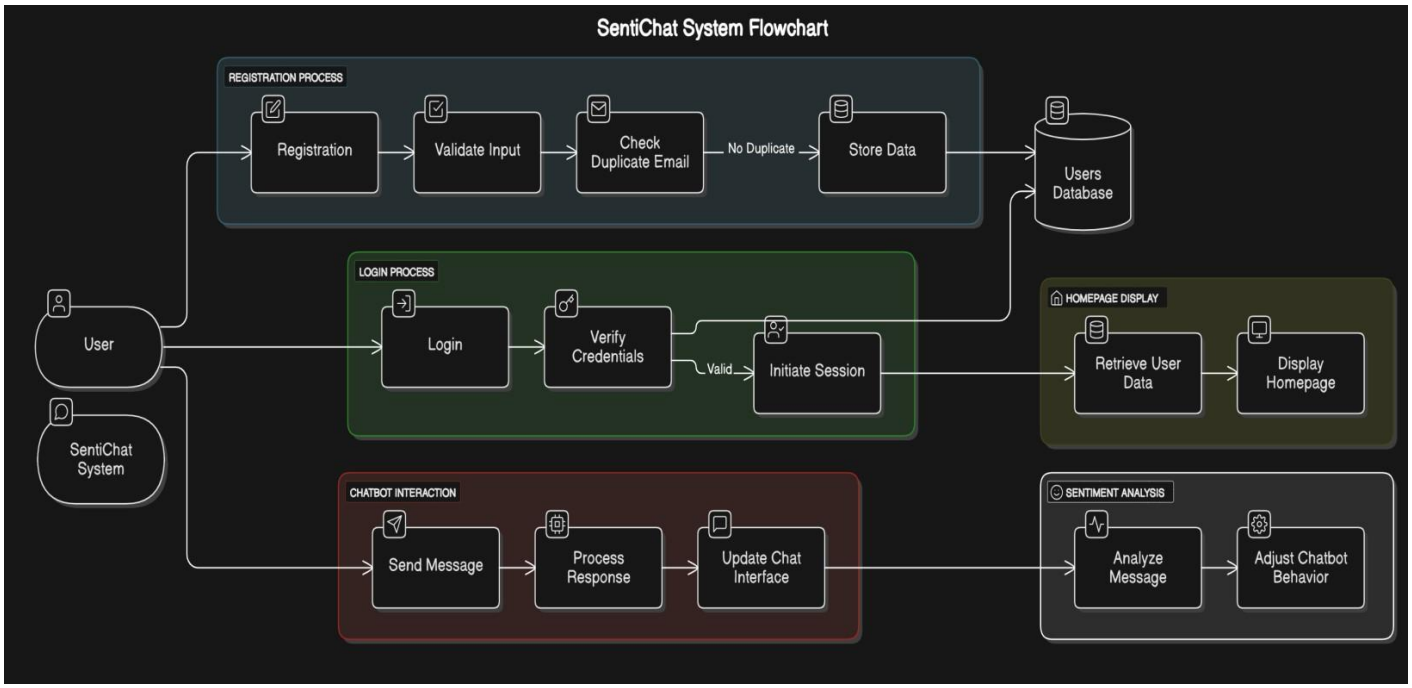
- **DATA STORES:**

  - **Users Database:** Stores user information (firstName, lastName, email, password).

- **DATA FLOW:**

  - User → Registration: Input registration details.

- Registration → Users Database: Store user data.
- User → Login: Input email and password.
- Login → Users Database: Verify credentials.
- Login → Homepage: Redirect with session data.
- Homepage → Users Database: Fetch user details.
- User → Chatbot: Send messages.
- Chatbot → API: Request responses.
- API → Chatbot: Return responses.
- Chatbot → User: Display responses with sentiment indicators.



Implementation

The SentiChat application is implemented using a combination of frontend and backend technologies. Below is an overview of the implementation details based on the provided files.

Frontend Implementation

- **index.php:** Contains the HTML structure for the registration and login forms. It uses Font Awesome for icons and links to `*style.css*` for styling and `*script.js*`for dynamic form switching.

  - Registration form collects first name, last name, email, and password.
  - Login form collects email and password.
  - Buttons toggle between forms using JavaScript.

- **style.css:** Defines a responsive design with a gradient background, animated form transitions, and hover effects. Key features include:

  - Floating labels for input fields.
  - Custom button styles with gradients and shadows.
  - Icon animations for social login options.

- **script.js:** Handles form visibility by toggling between the signup and signin containers.

- **homepage.php:** Displays a personalized greeting using session data retrieved from the database.

- **mood.html:** Implements the SentiChat interface with a sidebar for persona selection, a chat area, and an input field.

  - Uses CSS variables for light/dark themes.
  - Includes animations for messages and typing indicators.
  - Integrates Marked.js for rendering Markdown responses.
  - JavaScript handles chat interactions, sentiment analysis, and API calls.

Backend Implementation

- **connect.php:** Establishes a connection to the MySQL database using the `mysqli` extension.

  - Host: localhost
  - User: root
  - Password: (empty)
  - Database: login

- **register.php:** Handles registration and login logic.

  - **Registration:**

    - Collects form data (firstName, lastName, email, password).
    - Encrypts password using MD5 (Note: MD5 is outdated; bcrypt or Argon2 is recommended).
    - Checks for existing emails to prevent duplicates.
    - Inserts new user data into the `users` table.
    - Starts a session and redirects to `index.php`.

- **Login:**

  - Verifies email and password against the database.
  - Starts a session and redirects to `mood.html` on success.
  - Displays an error for incorrect credentials.

- **homepage.php:** Queries the `users` table to fetch the logged-in user's first and last names for display.

Chatbot Implementation

- **mood.html:**

- **Persona Selection:** Users can choose from predefined personas (Alex, Emma, Sam) with distinct tones.

- **Sentiment Analysis:** JavaScript analyzes messages for positive/negative words and emojis, updating the mood indicator (Happy, Sad, Neutral).

- **API Integration:** Sends user messages to the OpenRouter API (`deepseek-r1` model) and streams responses.

- **Dynamic UI:** Updates the chat area with incoming/outgoing messages, avatars, and timestamps.

- **Theme Toggle:** Switches between light and dark themes, storing the preference in `localStorage`.

- **Accessibility:** Includes ARIA attributes for screen reader support.

Security Considerations

- Passwords are hashed using MD5, which is vulnerable to attacks. A stronger algorithm should be used in production.

- Input validation is minimal; sanitization is needed to prevent SQL injection.

- Session management is basic; additional security measures (e.g., CSRF tokens) are recommended.

Database Design

The SentiChat application uses a MySQL database named `login` with a single table, `users`.

Table: users			
Field	**Type**	**Description**	**Constraints**
firstName	VARCHAR(50)	User's first name	NOT NULL
lastName	VARCHAR(50)	User's last name	NOT NULL
email	VARCHAR(100)	User's email address	NOT NULL, UNIQUE
password	VARCHAR(32)	Hashed password (MD5)	NOT NULL

Schema Creation Query

sql

```sql
CREATE DATABASE login;

USE login;

CREATE TABLE users (
   firstName VARCHAR(50) NoT NULL,
   lastName VARCHAR(50) NoT NULL,
   email VARCHAR(100) NoT NULL UNIQUE,
   password VARCHAR(32) NoT NULL
);
```

Notes

- The `email` field is unique to prevent duplicate accounts.

- The `password` field stores MD5 hashes, but a stronger hashing method is advisable.

- No primary key is explicitly defined in the code; adding an `id` field as a primary key is recommended.

Github Link: Click on the link

Output Screenshots