Spam Detector

# **System Requirements**

## **System Requirements**

For the development and successful operation of the Spam Detection Web Application, the following system requirements are necessary:

**Hardware Requirements:**

* **Processor**: Minimum Dual Core 2.0 GHz
* **RAM**: At least 4 GB (8 GB or more recommended for smooth functioning)
* **Storage**: 200 MB of free space for project files, SQLite database, and system files.
* **Display**: Any modern display resolution (minimum 1024x768) that supports web browsers.

**Software Requirements:**

* **Operating System**: Windows, macOS, or Linux (Ubuntu 18.04 or higher recommended)
* **Programming Language**: Python 3.x
* **Web Framework**: Flask (for back-end)
* **Database**: SQLite (for user data storage and spam reports)
* **Frontend**: HTML, CSS, JavaScript (for UI design)
* **Browser**: Google Chrome, Firefox, or any modern web browser
* **Libraries/Dependencies**: Flask, SQLAlchemy, Flask-WTF, Flask-Login, Flask-Mail, requests, geopy
* **IDE/Code Editor**: VS Code, PyCharm, or any modern text editor
* **Hosting Platform**: Local or cloud-based server (Heroku, AWS, etc.)

## **Functional Requirements**

The application has the following functional requirements to ensure the smooth execution of its features:

1. **User Authentication**: Users must be able to register, log in, and log out of the system. Passwords should be securely hashed, and user sessions maintained.
2. **Spam Detection**: The system should allow users to input text directly or upload a file (such as .txt or .csv) for spam detection. The system should process the input and return whether the content is spam or not spam.
3. **Geolocation Functionality**: If the input is determined to be spam, the system should fetch and display the geographical location of the person or source who sent the spam.
4. **About Page**: This page must include sections like ‘About Us,’ ‘Our Mission,’ ‘Frequently Asked Questions (FAQ),’ and a feedback form for user comments or suggestions.
5. **Contact Page**: A contact page should be available to allow users to send emails to the application administrators.
6. **Feedback System**: A feedback form that allows users to submit their feedback on the application and its performance.
7. **File Upload**: Users should have the option to either type or upload a file containing potential spam text, and the system should process both forms of input.
8. **Data Storage**: Spam-related data should be stored in an SQLite database for future reference, analysis, or user feedback.
9. **Email Notifications**: For certain actions such as spam detection or contact forms, users should receive email notifications via the system’s mail functionality.

## **Non-Functional Requirements**

1. **Performance**: The system should efficiently process spam detection for both small text inputs and file uploads, responding in under 2 seconds for a single text entry and under 5 seconds for file uploads of moderate size.
2. **Scalability**: The application should be scalable, capable of handling multiple users and an increasing volume of spam detection tasks.
3. **Security**: User data, including login credentials and spam reports, must be encrypted. The application should protect against common web vulnerabilities (SQL injection, XSS, etc.).
4. **Usability**: The user interface should be intuitive and user-friendly, ensuring ease of navigation across the home, about, contact, and spam detection pages.
5. **Portability**: The system should run on all major web browsers and operating systems without additional configurations.
6. **Reliability**: The system should be robust and handle errors (such as file upload issues or server downtime) gracefully, providing appropriate messages to the user.
7. **Maintainability**: The codebase should be well-organized, allowing future developers to easily modify or update the system.
8. **Availability**: The system should have an uptime of 99%, ensuring it is always accessible to users.

# **2. System Analysis and Design**

This chapter provides the design and analysis of the Spam Detection Web Application. It includes diagrams and brief descriptions of the system's components and architecture.

## **2.1. Entity Relationship Diagram (ERD)**

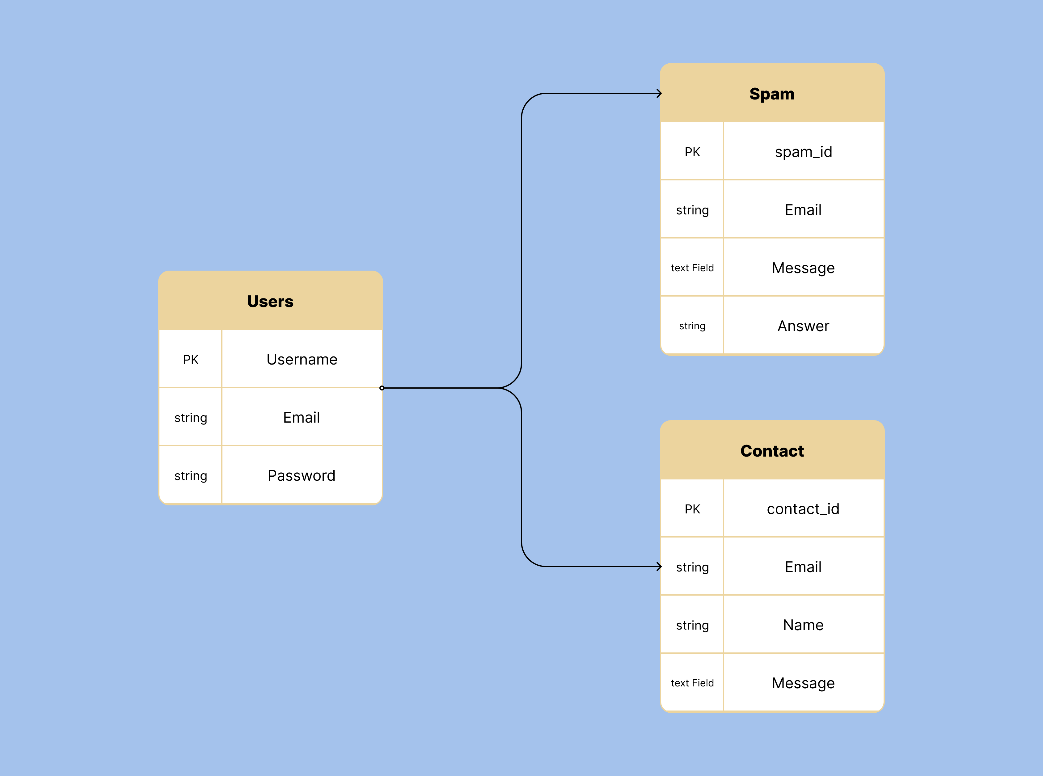


Figure 1: Entity Relationship Diagram (ERD)

The Entity Relationship Diagram illustrates the relationships between the different entities in the system. Major entities include:

* **User**: Stores information such as username, email, and hashed password.
* **Spam Report**: Records details of spam detections, including input text or file, spam status, and geolocation details.
* **Feedback**: Contains user feedback submitted through the about or contact page.

## **2.2. Use Case Diagram**

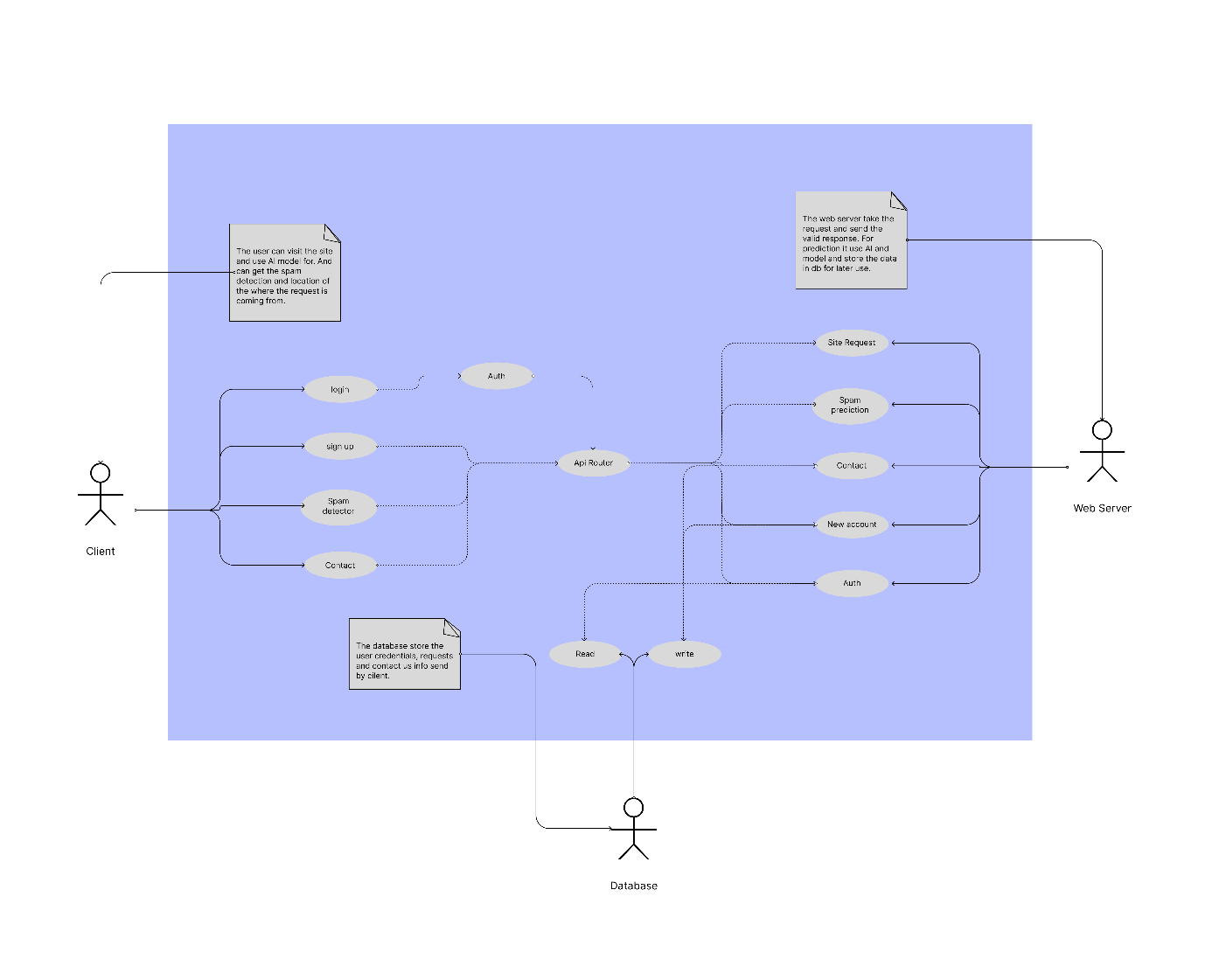


Figure 2: Use Case Diagram

The Use Case Diagram highlights the interaction between users and the system. Key use cases include:

* **Login/Sign-Up**
* **Spam Detection**
* **Submit Feedback**
* **Contact Admins**

## **2.3. Sequence Diagram**

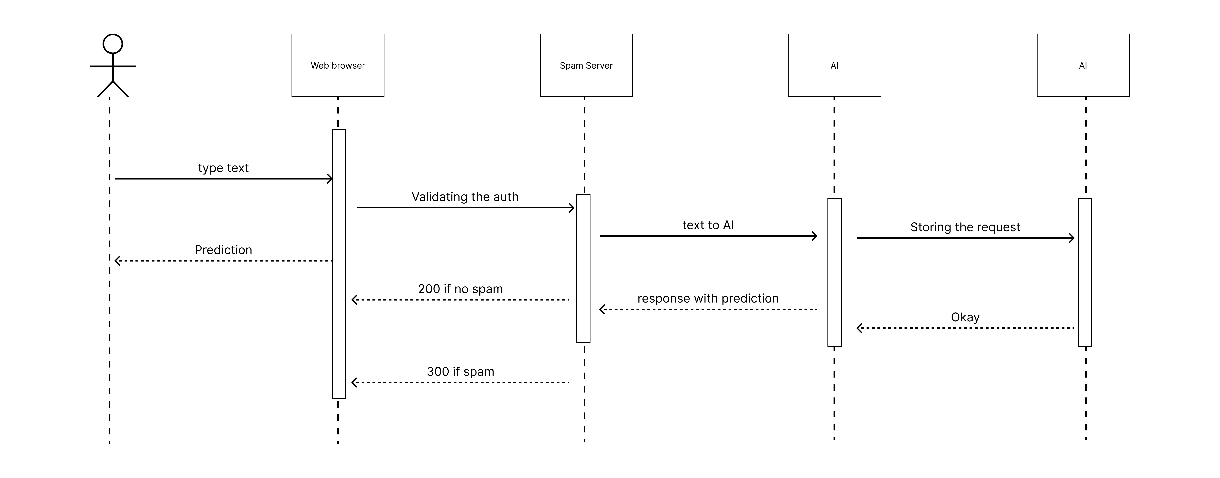


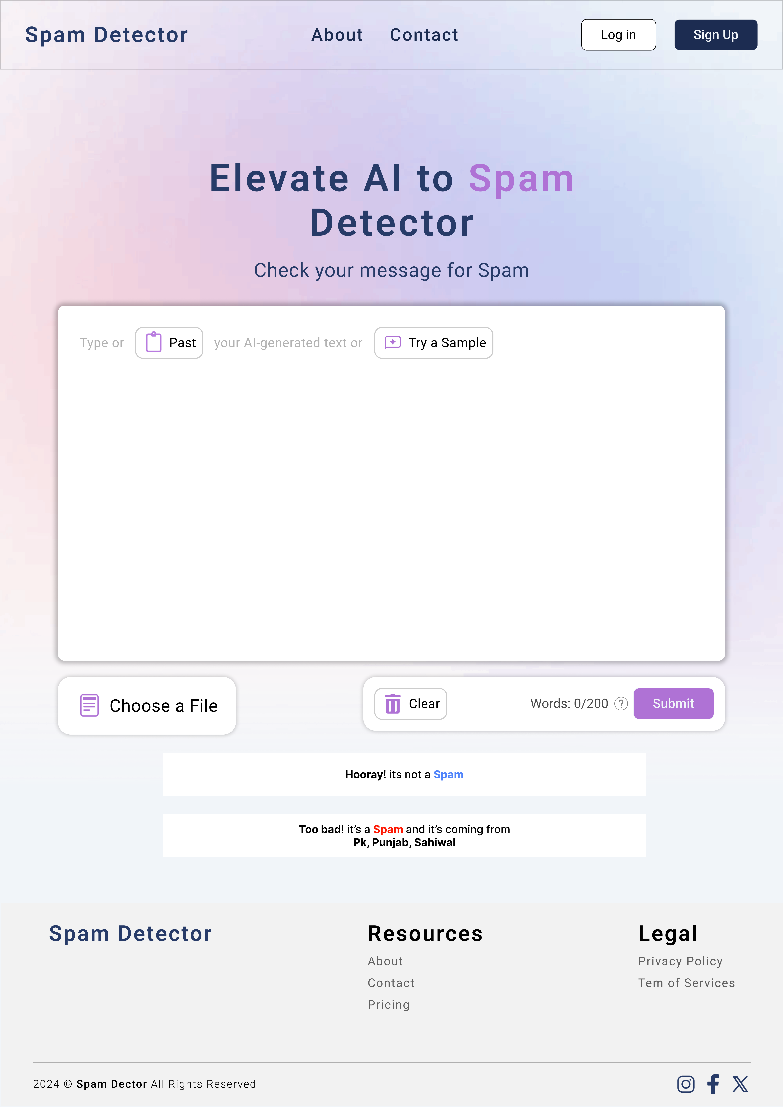
Figure 3: Sequence Diagram

This Sequence Diagram details the step-by-step flow of actions for a typical spam detection task, from user input (text or file) to the final spam report.

# **3. User Interface Design**

This section outlines the design and layout of the main pages within the Spam Detection Web Application.

## **3.1. Home**



The home page consists of:

* **Text Input Box**: A text box where users can type or paste content for spam detection.
* **File Upload Section**: An option to upload text or .csv files for processing.
* **Spam Detection Button**: A button to trigger the detection process.
* **Results Area**: Displays the spam status and geolocation

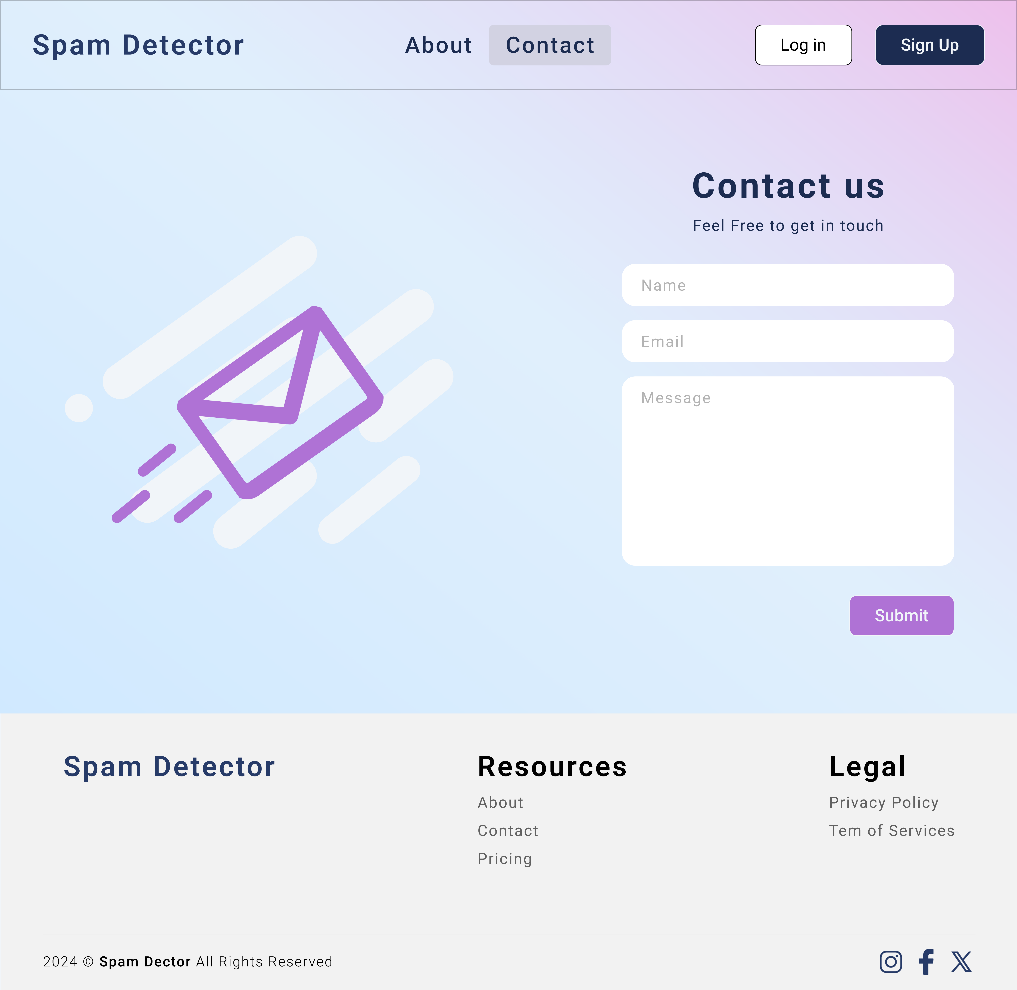
## **3.2. About page**



The About page includes:

* **About Us Section**: An introduction to the purpose and mission of the application.
* **Mission Statement**: A statement describing the goal of the system, particularly in combating spam.
* **Frequently Asked Questions (FAQ)**: Common questions related to spam detection, file upload issues, and geolocation accuracy.
* **Feedback Form**: A form where users can submit feedback or suggestions.

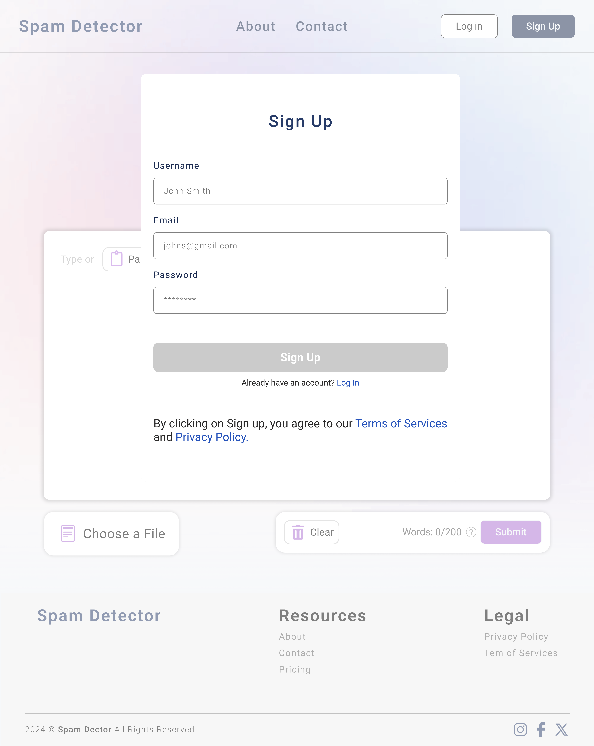
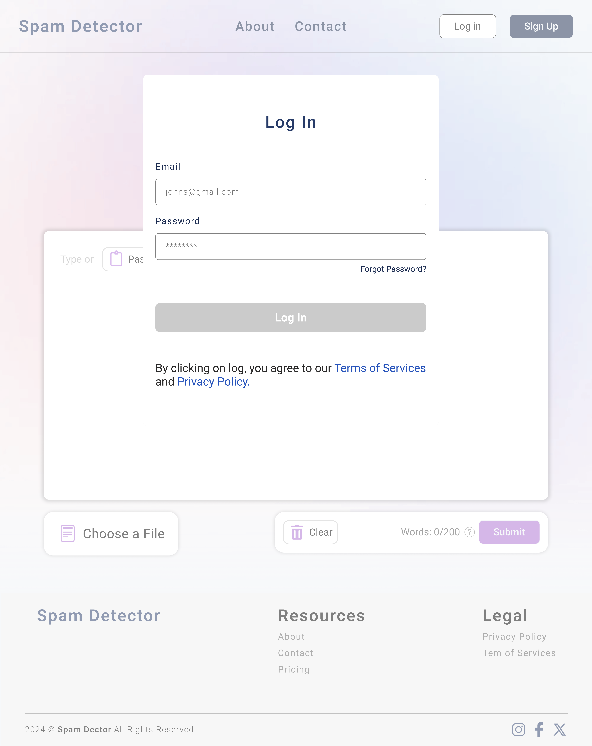
## **3.3. Contact Page**



The Contact page provides:

* **Email Form**: A simple form allowing users to contact the administrators directly with any questions, issues, or support needs.
* **Submit Button**: A button to send the message via email.

## **3.4. Login/Signup Page**



* **Signup Form**: A form where new users can register by entering their details such as username, email, and password.
* **Login Form**: A simple form where registered users can log in using their email and password.
* **Forgot Password Option**: A link allowing users to reset their password if they forget it.