Initial Project Setup / Proposal

CryptNote

1. Introduction

In this following proposal, the initial development phase of a note-sharing application to be called CryptNote is outlined and to be based on a similar idea to PrivNote.com. This application will enable the users to write messages that are encrypted and will automatically delete themselves once the contents have been viewed, optionally enforcing passcode security for viewing of the encrypted notes. The project will be created using Flask web framework and SQLite database with other features included being user authentication, user registration, user login, user profile and password protected notes.

2. What?

CryptNote is an online tool that can be used to exchange private information safely on the Internet. This way, users can create notes that are stored and remain accessible only via an individual link provided by the tool. The notes will be deleted once accessed to prevent invasion of privacy and security breach. The application will also include a user account system with a profile and the options to set up a passcode for the application.

3. Why?

With the issue of privacy being of immense value at the present times, CryptNote fulfills the requirement of safe, temporary exchange of information. With encryption and self-destructing notes, the application reduces the risks of data being stored permanently in these three areas.

4. List of Supported Features

- Note creation and sharing should only comprise of protected notes.
- Remember encryption and decryption
- Read-only access to notes which automatically delete upon opening• User identification (registration, login, and profile management)
- Locking notes using word or passcode

- REST API for core functionalities
- Note encryption and decryption
- Self-destructing notes after access
- User authentication (registration, login, profile management)
- Passcode protection for notes
- REST API for core functionalities
- Cross-browser compatibility (Chrome, Firefox, MS Edge)
- HTTPS support
- Logging and auditing capabilities

5. List of Not Supported Features

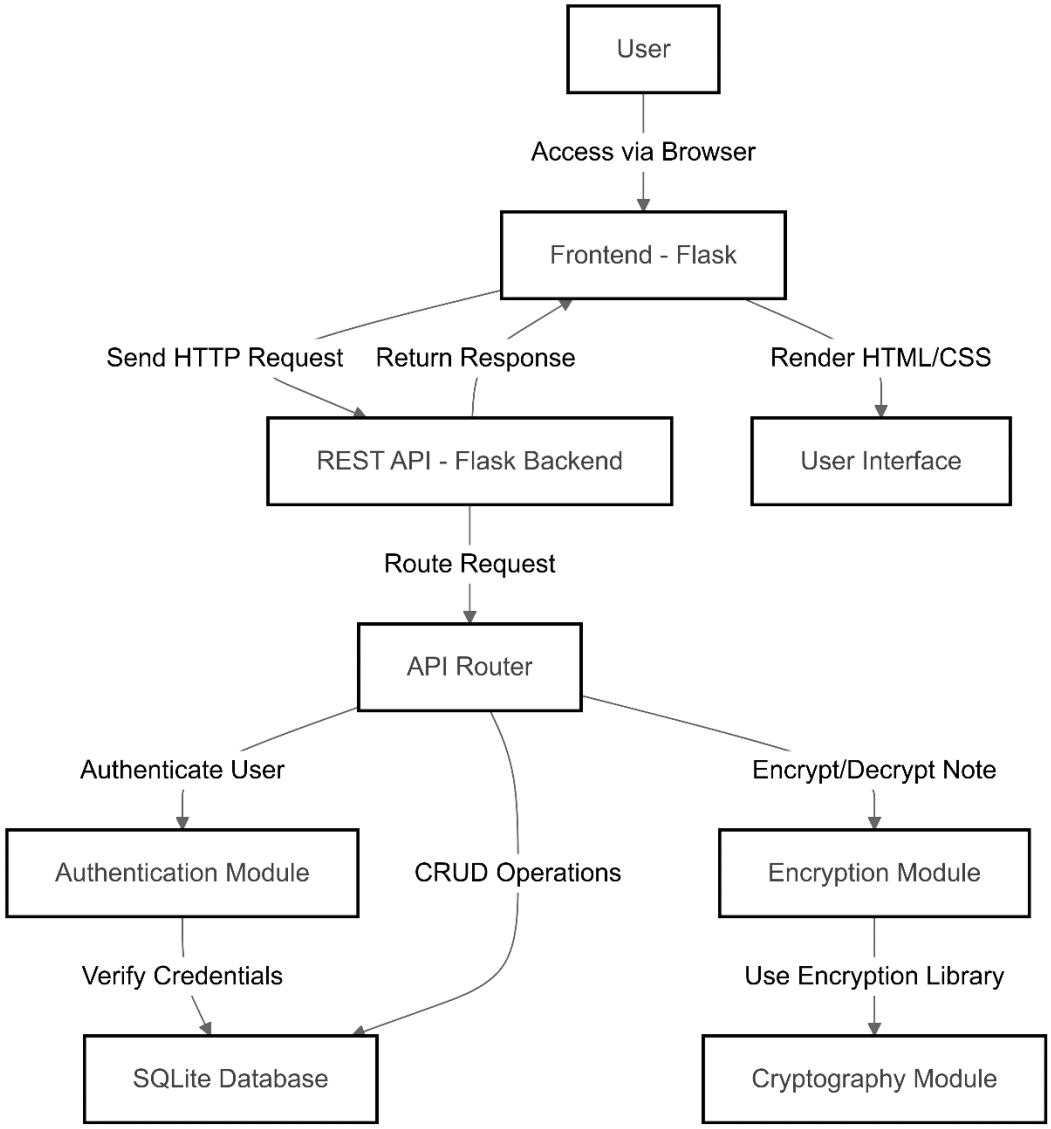
- Mobile application support (planned for future)
- Internationalization (planned for future)
- High availability and scaling for millions of users (planned for future)

6. List of Future Planned Features

- Mobile app development (Android and iOS)
- Internationalization and localization
- Vertical and horizontal scaling
- High availability and containerization

7. How

1. High-Level Diagram



8. List of Components/Modules

- **User**: Its purpose is to indirectly interact with the application, more specifically through a web browser.
- **Frontend - Flask:** First HTTP requests and serves the HTML part of the user interface. It sends on the requests to the backend API for further processing.

- **REST API - Flask Backend:** Receive the incoming request and send it to the API Router and he forwards it to the specific module.
- **API Router**: Sends queries to modules depending on what is expected of them to perform such as authentication, encrypting a note, and interacting with the database.
- **Authentication Module:** Responsible for log in, registrations and maintaining the session of the user. As for ID check, it authenticates users with the information recorded on the SQLite as the standard data database.
- **SQLite Database:** Stores user data, encrypted notes and other necessary information which are required for the functioning of the account. It is used for authentication and for the manipulation of data in the system, Create, Read, Update, Delete.
- **Encryption Module:** Authors the encoding and decoding of notes based on the security of data. In cryptographic function, it utilizes the Cryptography Module.
- **Cryptography Module:** Contains all the required components of the cryptographic method and functions needed for secure data processing.
- **User Interface:** converts the HTML and CSS to a form that can be rendered by the user for viewing the web pages with an interactive interface

9. Languages to be Used for Each Module

- **Frontend**:
    - HTML, CSS, JavaScript
- **Backend**:
    - Python (Flask)
- **Database**:
    - SQL (SQLite)

10. List of 3rd Party/Open Source Modules

- Flask (BSD License)
- SQLite (Public Domain)
- Jinja2 (BSD License)

- Werkzeug (BSD License)

11. Table of Licenses

Flask – BSD License

SQLite – Public Domain

Jinja2 – BSD

Werkzeug - BSD

12. List of Any 3rd Party Services/APIs

- None required initially. Future consideration for cloud hosting services.

13. REST API Endpoints with Payloads

Endpoint	Method	Description	Payload
/api/notes	POST	Create a new note	`{ "content": "string", "passcode": "string" }`
/api/notes/:id	GET	Retrieve a note	`{ "passcode": "string" }`
/api/auth/register	POST	Register a new user	`{ "username": "string", "password": "string" }`
/api/auth/login	POST	User login	`{ "username": "string", "password": "string" }`

14. Build Steps/Scripts

- Set up a virtual environment: python -m venv vEnv
- Install dependencies: pip install -r requirements.txt
- Run database migrations: flask db upgrade
- Navigate to the project directory: cd myprivnote

- Follow build steps to set up the environment and database

References

- Flask Documentation: Flask Docs
- SQLite Documentation: SQLite Docs
- PrivNote: PrivNote Website