Full Technical Report

```text

# Mental Health Support Chatbot: Consolidated Technical Document

## 1. Introduction and Goals (Updated Intake)

This document provides a comprehensive overview of the design and technical specifications for a mental health support chatbot aimed at young adults (18–30) dealing with anxiety and stress. The chatbot will provide 24/7 support and resources. This document consolidates the User Story, Project Design Document (PDD), and System Design Document (SDD) to provide a single source of truth for the development team and stakeholders.

\*\*User Story:\*\* As a young adult user, I want to be able to create a secure account, log in, and manage my password, so that I can access personalized mental health support and resources while ensuring my data is protected.

### 1.1. Goals

* Reduce the average response time to user inquiries to under 5 minutes within the first 3 months of launch.
* Ensure user data privacy and security by achieving a score of 95% or higher on annual security audits.
* Offer personalized support based on user needs, measured by a 4.5-star or higher average user rating for recommendation relevance.
* Comply with relevant data protection regulations, with zero reported instances of non-compliance.

### 1.2. Stakeholders

* \*\*Users:\*\* Young adults (18-30) experiencing anxiety and stress.
* \*\*Project Manager:\*\* Responsible for overall project execution and delivery.
* \*\*Development Team:\*\* Responsible for building and maintaining the chatbot.
* \*\*Security Team:\*\* Responsible for ensuring the security of the chatbot and user data.
* \*\*Business Analyst:\*\* Responsible for gathering and documenting requirements.

## 2. User Story and Acceptance Criteria (Updated User Story)

* \*\*Title\*\*: Secure User Authentication and Authorization
* \*\*Epic\*\*: Mental Health Chatbot
* \*\*Feature\*\*: User Account Management
* \*\*Priority\*\*: High
* \*\*Story Points\*\*: 8
* \*\*User Story\*\*: As a young adult user, I want to be able to create a secure account, log in, and manage my password, so that I can access personalized mental health support and resources while ensuring my data is protected.
* \*\*🎯 Acceptance Criteria\*\*
* \*\*Account Creation:\*\*
* [ ] User can create an account using a valid email address and a strong password that meets complexity requirements (e.g., minimum length, special characters). (FR.1.1)
* [ ] The system sends a verification email to the user's registered email address upon account creation.
* [ ] User can verify their email address by clicking on the link in the verification email.
* [ ] Upon successful email verification, the user's account is activated.
* [ ] The system displays an error message if the email address is already registered.
* [ ] The system enforces a strong password policy, requiring a minimum length of 12 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character.
* [ ] The system provides real-time feedback to the user on password strength during account creation and password reset.
* [ ] The system checks for and disallows the use of commonly used passwords.
* \*\*Login:\*\*
* [ ] User can log in to their account using their registered email address and password. (FR.1.2)
* [ ] The system displays an error message for invalid login credentials.
* [ ] The system implements account lockout after 5 failed login attempts within 15 minutes for a duration of 30 minutes.
* [ ] The system implements rate limiting on login attempts to prevent brute-force attacks (e.g., no more than 5 attempts per minute per IP address).
* \*\*Password Reset:\*\*
* [ ] User can request a password reset via email. (FR.1.3)
* [ ] The system sends a password reset link to the user's registered email address.
* [ ] User can reset their password by clicking on the link in the password reset email and providing a new, strong password.
* [ ] Password reset links use strong, unpredictable tokens.
* [ ] Password reset links expire after 15 minutes.
* [ ] The system implements measures to prevent password reset link reuse.
* \*\*Security:\*\*
* [ ] User passwords are securely stored using encryption. (FR.1.4, SEC-1)
* [ ] The system offers multi-factor authentication (MFA) as an optional security enhancement. (FR.1.4)
* [ ] The system complies with relevant data privacy regulations (e.g., GDPR, CCPA). (FR.1.5)
* [ ] The system provides an option for anonymous usage, with limited feature availability. (FR.1.6)
* [ ] The system uses secure cookies (HTTPOnly, Secure attributes) for session management.
* [ ] The system implements session timeouts after 30 minutes of inactivity.
* [ ] The system regenerates session IDs after login and password reset.
* [ ] The system sanitizes all user inputs to prevent XSS attacks.
* [ ] The system implements a Content Security Policy (CSP) to mitigate XSS risks.
* [ ] User data is encrypted at rest using AES-256 encryption.
* [ ] Data in transit is encrypted using HTTPS.
* [ ] Regular data backups are performed.
* [ ] Access to sensitive data is restricted based on the principle of least privilege.
* [ ] All third-party libraries are regularly updated to the latest versions.
* [ ] A software composition analysis (SCA) tool is used to identify and address vulnerabilities in third-party dependencies.
* \*\*Anonymous Usage:\*\*
* [ ] User can access the chatbot anonymously without creating an account.
* [ ] Anonymous users have access to a limited set of features.
* [ ] The system clearly indicates which features are available to anonymous users.
* [ ] Content filtering and moderation mechanisms are implemented to prevent misuse of the chatbot.
* [ ] The functionality available to anonymous users is limited to basic information retrieval and FAQs.
* [ ] Anonymous user activity is logged for potential investigation.
* \*\*Edge Cases:\*\*
* [ ] The system displays an informative error message if the user enters invalid characters in their password during account creation or password reset.
* [ ] The system displays an informative error message if the user enters an invalid email address during password reset.
* [ ] The system provides alternative account recovery options if email delivery fails.
* [ ] The system implements rate limiting on all API endpoints to prevent DoS attacks.
* [ ] The system is protected by a Web Application Firewall (WAF).
* [ ] The system displays an appropriate message if the email verification link is expired, already used, or invalid.
* [ ] The system invalidates concurrent login sessions from the same account.
* [ ] Inactive accounts are automatically archived after 12 months of inactivity.
* \*\*📘 Functional Requirements Mapping\*\*
* FR.1.1: User can create accounts using a valid email address and a secure password.
* FR.1.2: User can log in to their accounts using their registered email address and password.
* FR.1.3: User can reset their password via email verification.
* FR.1.4: User data is securely stored and protected; multi-factor authentication is offered.
* FR.1.5: Compliance with data privacy regulations (GDPR, CCPA).
* FR.1.6: Option for anonymous usage with limited feature availability.
* SEC-1: Secure password storage using encryption.
* \*\*✅ Definition of Done\*\*
* [ ] All acceptance criteria are met.
* [ ] Code is reviewed and approved.
* [ ] Unit tests are written and passing.
* [ ] Security vulnerabilities are addressed, including but not limited to XSS, SQL injection, and CSRF.
* [ ] Rate limiting and account lockout mechanisms are tested and verified to be effective.
* [ ] Password policy enforcement is tested and verified.
* [ ] Password reset link security is tested and verified.
* [ ] Session management security is tested and verified.
* [ ] XSS vulnerability testing is performed, and no vulnerabilities are found.
* [ ] CSP is configured and verified.
* [ ] Encryption mechanisms are tested and verified.
* [ ] Data backup and recovery procedures are tested.
* [ ] A security audit and penetration test have been conducted, and identified vulnerabilities have been addressed.
* [ ] Third-party libraries are scanned for vulnerabilities, and identified vulnerabilities have been addressed.
* [ ] Content filtering and moderation mechanisms are tested and verified.
* [ ] DoS attack mitigation mechanisms are tested and verified.
* [ ] Account recovery failure scenarios are tested and verified.
* [ ] Email verification failure scenarios are tested and verified.
* [ ] Concurrent login session handling is tested and verified.
* [ ] Inactive account handling is tested and verified.
* [ ] The feature is deployed to the staging environment for testing.
* [ ] User documentation is updated.

## 3. Project Design Document (PDD)

### 3.1. Functional Requirements

* \*\*Account Creation:\*\* Users can create accounts with a valid email address and a strong password. \*\*Email verification is mandatory.\*\*
* \*\*Login:\*\* Users can log in securely with their credentials.
* \*\*Password Management:\*\* Users can reset their passwords via email verification.
* \*\*Chat Interface:\*\* Users can interact with the chatbot through a conversational interface, supporting text-based input and responses, as well as the ability to share and receive multimedia content (e.g., images, videos, links).
* \*\*Resource Library:\*\* Users can access a library of mental health resources (articles, videos, etc.).
* \*\*Personalized Recommendations:\*\* The chatbot provides personalized recommendations based on user input and assessed needs, utilizing a recommendation engine that considers user history, preferences, and identified risk factors.
* \*\*Account Deletion:\*\* Users can delete their accounts.

### 3.2. Non-Functional Requirements

* \*\*Availability:\*\* The chatbot should be available 24/7 with a target uptime of 99.9%.
* \*\*Performance:\*\* The chatbot should respond to user input within 2 seconds.
* \*\*Scalability:\*\* The system should be able to handle a large number of concurrent users (up to 10,000) without performance degradation.
* \*\*Security:\*\* The system should be secure and protect user data, adhering to OWASP guidelines and industry best practices.
* \*\*Usability:\*\* The chatbot should be easy to use and understand, with a user-friendly interface and clear instructions.
* \*\*Accessibility:\*\* The chatbot should be accessible to users with disabilities (WCAG compliant).

### 3.3. System Architecture

The chatbot will be a multi-tiered application consisting of:

* \*\*Frontend:\*\* User interface for interacting with the chatbot (web and/or mobile app).
* \*\*Backend:\*\* API layer for handling user requests and business logic.
* \*\*Database:\*\* Stores user data, resources, and conversation history.
* \*\*NLP Engine:\*\* Processes user input and generates appropriate responses.
* \*\*Recommendation Engine:\*\* Provides personalized recommendations based on user input and assessed needs.
* \*\*Email Service:\*\* Handles sending email verification and password reset emails.

### 3.4. System Diagram

```mermaid

graph LR

subgraph User

A[User]

end

subgraph Frontend

B[Login Page]

C[Signup Page]

D[Chat Interface]

E[Password Reset Page]

F[Account Settings Page]

end

subgraph Backend

G[Authentication Service]

H[Chat Service]

I[User Management Service]

J[Password Reset Service]

K[Account Deletion Service]

L[API Gateway]

end

subgraph Database

M[User Database]

N[Chat History Database]

end

subgraph NLP Engine

O[NLP Engine]

end

subgraph Recommendation Engine

P[Recommendation Engine]

end

%% Account Creation Flow

A -- "1. Enters signup details" --> C

C -- "2. Sends signup request" --> L

L -- "3. Routes to User Management Service" --> I

I -- "4. Validates user data" --> I

I -- "5. Creates user account" --> M

M -- "6. Account created confirmation" --> I

I -- "7. Returns success message" --> L

L -- "8. Returns success to Frontend" --> C

C -- "9. Account created notification" --> A

%% Login Flow

A -- "1. Enters login credentials" --> B

B -- "2. Sends login request" --> L

L -- "3. Routes to Authentication Service" --> G

G -- "4. Authenticates user" --> M

M -- "5. Returns user details" --> G

G -- "6. Generates authentication token" --> L

L -- "7. Returns token to Frontend" --> B

B -- "8. Stores token and redirects to Chat Interface" --> D

D -- "9. User logged in" --> A

%% Chat Interaction Flow

A -- "1. Sends message" --> D

D -- "2. Sends message to Backend" --> L

L -- "3. Routes to Chat Service" --> H

H -- "4. Stores message in Chat History" --> N

H -- "5. Sends message to NLP Engine" --> O

O -- "6. Analyzes message and identifies intent" --> H

H -- "7. Sends message and intent to Recommendation Engine" --> P

P -- "8. Generates relevant recommendations" --> H

H -- "9. Formats response with recommendations" --> L

L -- "10. Returns response to Frontend" --> D

D -- "11. Displays response to User" --> A

%% Password Reset Flow

A -- "1. Requests password reset" --> E

E -- "2. Sends email request" --> L

L -- "3. Routes to Password Reset Service" --> J

J -- "4. Verifies user email" --> M

M -- "5. Generates reset token" --> J

J -- "6. Sends reset link to user email" --> A

A -- "7. Clicks reset link and enters new password" --> E

E -- "8. Sends new password and token" --> L

L -- "9. Routes to Password Reset Service" --> J

J -- "10. Validates token and updates password" --> M

M -- "11. Password updated confirmation" --> J

J -- "12. Returns success message" --> L

L -- "13. Returns success to Frontend" --> E

E -- "14. Password reset confirmation" --> A

%% Account Deletion Flow

A -- "1. Requests account deletion" --> F

F -- "2. Sends deletion request" --> L

L -- "3. Routes to Account Deletion Service" --> K

K -- "4. Authenticates user" --> M

M -- "5. Verifies user account" --> K

K -- "6. Deletes user account and data" --> M

N -- "7. Deletes user chat history" --> K

K -- "8. Account deletion confirmation" --> L

L -- "9. Returns success message" --> F

F -- "10. Account deletion confirmation" --> A

```

### 3.5. User Flows

#### 3.5.1 Account Creation

1. User navigates to the registration page.

2. User enters their email address and password.

3. System validates the email format and \*\*password strength (see section 3.6)\*\*.

4. System creates a new user account.

5. System sends a verification email to the user's email address.

6. User clicks the verification link in the email.

7. System verifies the user's email address and activates the account.

8. User is redirected to the login page.

\*\*Error Handling:\*\*

* \*\*Email address already registered:\*\* Display an error message prompting the user to log in or reset their password.
* \*\*Invalid email format:\*\* Display an error message prompting the user to enter a valid email address.
* \*\*Weak password:\*\* Display an error message prompting the user to enter a stronger password, along with a password strength meter.
* \*\*Verification link expired:\*\* Display an error message prompting the user to request a new verification email.

#### 3.5.2 Login

1. User navigates to the login page.

2. User enters their email address and password.

3. System authenticates the user's credentials.

4. System creates a new session for the user.

5. User is redirected to the chatbot interface.

\*\*Error Handling:\*\*

* \*\*Invalid credentials:\*\* Display an error message prompting the user to try again or reset their password.
* \*\*Account not verified:\*\* Display an error message prompting the user to verify their email address.

#### 3.5.3 Password Reset

1. User navigates to the password reset page.

2. User enters their email address.

3. System validates the email format.

4. System sends a password reset email to the user's email address.

5. User clicks the password reset link in the email.

6. User is redirected to a page where they can enter a new password.

7. System validates the new password strength.

8. User submits the new password.

9. System updates the user's password.

10. User is redirected to the login page.

\*\*Error Handling:\*\*

* \*\*Invalid email format:\*\* Display an error message prompting the user to enter a valid email address.
* \*\*Email address not found:\*\* Display an error message indicating that the email address is not registered.
* \*\*Password reset token expired:\*\* Display an error message prompting the user to request a new password reset email.
* \*\*Weak password:\*\* Display an error message prompting the user to enter a stronger password.

#### 3.5.4 Account Deletion

1. User navigates to the account settings page.

2. User clicks the "Delete Account" button.

3. System prompts the user to confirm their decision.

4. User confirms the account deletion.

5. System initiates the account deletion process, which includes:

* \*\*Data anonymization or deletion:\*\* User data will be anonymized or deleted according to the \*\*Data Retention Policy (see section 3.11)\*\* and relevant regulations (GDPR, CCPA).
* \*\*Session invalidation:\*\* All active sessions for the user will be invalidated.

6. System displays a confirmation message to the user.

\*\*Error Handling:\*\*

* \*\*Unexpected error during deletion:\*\* Display an error message and prompt the user to try again later.

### 3.6. Password Strength Requirements

Passwords must meet the following requirements:

* Minimum length: 12 characters
* Must contain at least one uppercase letter
* Must contain at least one lowercase letter
* Must contain at least one number
* Must contain at least one special character (e.g., !@#$%^&\*)

A password strength meter will be implemented to provide users with real-time feedback on the strength of their password.

### 3.7. Security Best Practices

The application will adhere to OWASP (Open Web Application Security Project) guidelines to ensure security. This includes:

* \*\*Input validation:\*\* All user input will be validated to prevent injection attacks.
* \*\*Output encoding:\*\* All output will be encoded to prevent cross-site scripting (XSS) attacks.
* \*\*Authentication and authorization:\*\* Strong authentication and authorization mechanisms will be implemented to protect user data.
* \*\*Regular security audits:\*\* Regular security audits will be conducted to identify and address potential vulnerabilities.
* \*\*Encryption:\*\* Sensitive data will be encrypted both in transit and at rest. HTTPS will be enforced for all communication.

### 3.8. Session Management

Session management will be implemented using JSON Web Tokens (JWT).

* \*\*Session duration:\*\* Sessions will expire after 30 minutes of inactivity.
* \*\*Session invalidation:\*\* Sessions can be invalidated by the user (e.g., by logging out) or by the system (e.g., due to inactivity or security reasons).
* \*\*Storage:\*\* JWTs will be stored in HTTP-only cookies to prevent XSS attacks.

### 3.9. Data Retention Policy

User data will be retained for as long as the user has an active account. Upon account deletion, user data will be anonymized or deleted within 30 days, in accordance with GDPR and CCPA regulations. Certain data, such as aggregated and anonymized usage statistics, may be retained for longer periods for analytical purposes.

### 3.10. Compliance

The application will comply with the following regulations:

* \*\*GDPR (General Data Protection Regulation):\*\* The application will comply with GDPR requirements for data privacy and security, including obtaining user consent for data collection, providing users with access to their data, and allowing users to delete their data.
* \*\*CCPA (California Consumer Privacy Act):\*\* The application will comply with CCPA requirements for data privacy and security for California residents, including providing users with the right to know what personal information is collected about them, the right to delete their personal information, and the right to opt-out of the sale of their personal information.
* \*\*HIPAA (Health Insurance Portability and Accountability Act):\*\* While the chatbot is not intended to provide medical advice or treatment, we will implement safeguards to protect the privacy and security of any health-related information that users may share.

### 3.11. Monitoring and Logging

Comprehensive monitoring and logging mechanisms will be implemented to track user activity and identify potential security issues.

* \*\*User activity logging:\*\* All user activity, including logins, logouts, password resets, and chatbot interactions, will be logged.
* \*\*Error logging:\*\* All errors will be logged, including the error message, timestamp, and user ID (if available).
* \*\*Security event logging:\*\* Security events, such as failed login attempts and suspicious activity, will be logged.
* \*\*Monitoring dashboards:\*\* Monitoring dashboards will be created to visualize key metrics, such as user activity, error rates, and security events.
* \*\*Alerting:\*\* Alerts will be configured to notify administrators of critical events, such as security breaches and system outages.
* \*\*Log Retention:\*\* Logs will be retained for a period of 12 months.

### 3.12. Data Design

* \*\*3.12.1 Data Model:\*\* The chatbot system will utilize a relational database (PostgreSQL) to store user data, resource metadata, and conversation history. Key entities include Users, Resources, Conversations, and Messages.
* \*\*3.12.2 Data Storage:\*\* User data will be stored in encrypted format within the PostgreSQL database. Resource files (e.g., articles, videos) will be stored in a secure cloud storage service (e.g., AWS S3) with appropriate access controls.
* \*\*3.12.3 Data Security:\*\* Data at rest will be encrypted using AES-256 encryption. Access to the database and cloud storage will be restricted to authorized personnel only. Regular backups will be performed to prevent data loss.

### 3.13. Integration Design

* \*\*3.13.1 External Interfaces:\*\* The chatbot will integrate with external services for email verification (e.g., SendGrid, Mailgun) and potentially for future integration with mental health APIs for resource recommendations.
* \*\*3.13.2 API Specifications:\*\* The chatbot will expose a RESTful API for communication between the frontend and backend. API endpoints will be secured using JWT authentication.
* \*\*3.13.3 Data Mapping:\*\* Data mapping will be defined for each integration to ensure seamless data exchange between the chatbot and external systems.

## 4. Component Mapping

### 4.1. Frontend

* \*\*Technology:\*\* React, JavaScript, HTML, CSS (React was chosen for its component-based architecture, ease of use, and large community support, facilitating rapid development and maintainability.)
* \*\*Responsibilities:\*\*
* Provides the user interface for interacting with the chatbot.
* Handles user input and displays chatbot responses.
* Manages user authentication and authorization.
* Displays mental health resources and personalized recommendations.
* \*\*Communication:\*\* Communicates with the backend API via HTTP requests.

### 4.2. Backend

* \*\*Technology:\*\* Python (Flask/Django), Node.js (Express) (Python with Flask/Django is selected for its robust libraries for NLP and machine learning, while Node.js with Express offers scalability and performance for handling API requests.)
* \*\*Responsibilities:\*\*
* Handles user requests and business logic.
* Authenticates and authorizes users.
* Interacts with the database to store and retrieve data.
* Communicates with the NLP engine to process user input.
* Generates personalized recommendations.
* \*\*Communication:\*\*
* Receives HTTP requests from the frontend.
* Interacts with the database using an ORM (e.g., SQLAlchemy or Mongoose).
* Communicates with the NLP engine via API calls.
* Communicates with the Recommendation Engine via API calls.

### 4.3. Database

* \*\*Technology:\*\* PostgreSQL, MongoDB (PostgreSQL is chosen for its reliability, data integrity, and support for complex queries, while MongoDB offers flexibility for storing unstructured data like conversation logs.)
* \*\*Responsibilities:\*\*
* Stores user data, mental health resources, and conversation history.
* Provides data persistence and retrieval.
* \*\*Communication:\*\* Interacted with by the backend API using an ORM.

### 4.4. NLP Engine

* \*\*Technology:\*\* Dialogflow, Rasa (Dialogflow and Rasa are selected for their ability to understand and process natural language, enabling the chatbot to engage in meaningful conversations with users.)
* \*\*Responsibilities:\*\*
* Processes user input and understands the user's intent.
* Generates appropriate responses.
* \*\*Communication:\*\* Receives API calls from the backend and returns processed responses.

### 4.5. Recommendation Engine

* \*\*Technology:\*\* Python, scikit-learn, TensorFlow, PyTorch
* \*\*Responsibilities:\*\*
* Analyzes user data and identifies relevant patterns.
* Matches users with appropriate mental health resources.
* Generates personalized recommendations based on user preferences and risk factors.
* \*\*Communication:\*\* Receives API calls from the backend and returns personalized recommendations.

### 4.6. Email Service

* \*\*Technology:\*\* SendGrid, Mailgun, or AWS SES
* \*\*Responsibilities:\*\*
* Sends email verification and password reset emails.
* \*\*Communication:\*\* Called by the backend API to send emails.

## 5. System Design Document (SDD)

### 5.1. Data Design

* \*\*5.1.1 Data Model:\*\* The chatbot system will utilize a relational database (PostgreSQL) to store user data, resource metadata, and conversation history. Key entities include Users, Resources, Conversations, and Messages.
* \*\*5.1.2 Data Storage:\*\* User data will be stored in encrypted format within the PostgreSQL database. Resource files (e.g., articles, videos) will be stored in a secure cloud storage service (e.g., AWS S3) with appropriate access controls.
* \*\*5.1.3 Data Security:\*\* Data at rest will be encrypted using AES-256 encryption. Access to the database and cloud storage will be restricted to authorized personnel only. Regular backups will be performed to prevent data loss.

### 5.2. Integration Design

* \*\*5.2.1 External Interfaces:\*\* The chatbot will integrate with external services for email verification (e.g., SendGrid, Mailgun) and potentially for future integration with mental health APIs for resource recommendations.
* \*\*5.2.2 API Specifications:\*\* The chatbot will expose a RESTful API for communication between the frontend and backend. API endpoints will be secured using JWT authentication.
* \*\*5.2.3 Data Mapping:\*\* Data mapping will be defined for each integration to ensure seamless data exchange between the chatbot and external systems.

### 5.3. API Specifications

The chatbot will expose a RESTful API for communication between the frontend and backend. API endpoints will be secured using JWT authentication.

#### 5.3.1. General Error Handling

All API endpoints will return standard HTTP status codes to indicate the success or failure of a request. In case of an error, the response body will contain a JSON object with the following structure:

```json

{

"success": false,

"error": {

"code": "ERROR\_CODE",

"message": "Error message describing the issue."

}

}

```

Common error codes include:

* `AUTH\_FAILED`: Authentication failed (invalid credentials, missing token).
* `VALIDATION\_ERROR`: Request body validation failed.
* `RESOURCE\_NOT\_FOUND`: The requested resource was not found.
* `INTERNAL\_SERVER\_ERROR`: An unexpected error occurred on the server.
* `RATE\_LIMIT\_EXCEEDED`: The rate limit for the endpoint has been exceeded.
* `EMAIL\_ALREADY\_EXISTS`: The email is already registered.
* `INVALID\_TOKEN`: The provided token is invalid or expired.

#### 5.3.2. Rate Limiting

API endpoints will be rate-limited to prevent abuse and ensure availability. The following headers will be included in API responses to indicate the current rate limit status:

* `X-RateLimit-Limit`: The maximum number of requests allowed within the rate limit window.
* `X-RateLimit-Remaining`: The number of requests remaining in the current rate limit window.
* `X-RateLimit-Reset`: The time at which the rate limit window will reset (in UTC seconds since epoch).

#### 5.3.3. Authentication API

##### 5.3.3.1. `POST /api/auth/register`

* \*\*Description:\*\* Registers a new user.
* \*\*Request Body:\*\*

```json

{

"email": "user@example.com",

"password": "password123"

}

```

* \*\*Request Validation:\*\*
* `email`: Required, must be a valid email address.
* `password`: Required, minimum length of 8 characters.
* \*\*Response:\*\*

```json

{

"success": true,

"message": "User registered successfully. Please verify your email."

}

```

* \*\*Error Handling:\*\*
* `400 VALIDATION\_ERROR`: If the request body is invalid.
* `409 EMAIL\_ALREADY\_EXISTS`: If the email is already registered.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

##### 5.3.3.2. `POST /api/auth/login`

* \*\*Description:\*\* Logs in an existing user.
* \*\*Request Body:\*\*

```json

{

"email": "user@example.com",

"password": "password123"

}

```

* \*\*Request Validation:\*\*
* `email`: Required, must be a valid email address.
* `password`: Required, minimum length of 8 characters.
* \*\*Response:\*\*

```json

{

"success": true,

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

```

* \*\*Error Handling:\*\*
* `400 VALIDATION\_ERROR`: If the request body is invalid.
* `401 AUTH\_FAILED`: If the credentials are invalid.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

##### 5.3.3.3. `POST /api/auth/reset-password`

* \*\*Description:\*\* Sends a password reset email to the user.
* \*\*Request Body:\*\*

```json

{

"email": "user@example.com"

}

```

* \*\*Request Validation:\*\*
* `email`: Required, must be a valid email address.
* \*\*Response:\*\*

```json

{

"success": true,

"message": "Password reset email sent successfully."

}

```

* \*\*Error Handling:\*\*
* `400 VALIDATION\_ERROR`: If the request body is invalid.
* `404 RESOURCE\_NOT\_FOUND`: If the email is not found.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

##### 5.3.3.4. `POST /api/auth/verify-reset-token`

* \*\*Description:\*\* Verifies the password reset token.
* \*\*Request Body:\*\*

```json

{

"token": "reset\_token"

}

```

* \*\*Request Validation:\*\*
* `token`: Required, must be a valid reset token string.
* \*\*Response:\*\*

```json

{

"success": true,

"message": "Token is valid"

}

```

* \*\*Error Handling:\*\*
* `400 VALIDATION\_ERROR`: If the request body is invalid.
* `400 INVALID\_TOKEN`: If the token is invalid or expired.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

##### 5.3.3.5. `POST /api/auth/update-password`

* \*\*Description:\*\* Updates the user's password.
* \*\*Request Body:\*\*

```json

{

"token": "reset\_token",

"password": "new\_password"

}

```

* \*\*Request Validation:\*\*
* `token`: Required, must be a valid reset token string.
* `password`: Required, minimum length of 8 characters.
* \*\*Response:\*\*

```json

{

"success": true,

"message": "Password updated successfully."

}

```

* \*\*Error Handling:\*\*
* `400 VALIDATION\_ERROR`: If the request body is invalid.
* `400 INVALID\_TOKEN`: If the token is invalid or expired.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

#### 5.3.4. User API

##### 5.3.4.1. `GET /api/user/profile`

* \*\*Description:\*\* Retrieves the user's profile.
* \*\*Authentication:\*\* Requires a valid JWT token in the `Authorization` header.
* \*\*Response:\*\*

```json

{

"id": 1,

"email": "user@example.com"

}

```

* \*\*Error Handling:\*\*
* `401 AUTH\_FAILED`: If the JWT token is missing or invalid.
* `404 RESOURCE\_NOT\_FOUND`: If the user is not found.
* `500 INTERNAL\_SERVER\_ERROR`: If an unexpected error occurs.
* \*\*Rate Limiting Headers:\*\* Included in the response.

##### 5.3.4.2. `DELETE /api/user/delete`

* \*\*Description:\*\* Deletes the user's account.
* \*\*Authentication:\*\* Requires a valid JWT