1. Introduction

The authentication module in PulsePing provides secure user access to the platform. It ensures that only verified users can manage monitored URLs, view uptime data, and receive notifications. The system follows a stateless, token-based architecture suitable for modern distributed applications.

---

2. Authentication Method

PulsePing adopts **JWT (JSON Web Token) based authentication**, enabling stateless and scalable user sessions. The server issues a token upon successful login, and the client includes this token in subsequent API requests. The backend validates the token before granting access to protected resources.

---

3. Password Security

User passwords are never stored in plain text.
Before persisting to the database, all passwords are processed using a secure one-way hashing algorithm (bcrypt). During login, the provided password is verified against the stored hash. This approach ensures compliance with standard security practices and mitigates credential exposure risks.

---

4. Required User Data

The authentication module relies on minimal user information to keep the flow simple and efficient.

Field	Description
name	Display name of the user
email	Unique identifier used for login
password	Used during signup/login; stored as a hashed value

These fields map directly to the **Users** table in the database schema.

---

5. Authentication Flows

5.1. User Registration (Signup)

- User submits name, email, and password.
- System validates uniqueness of the email.
- Password is hashed and stored along with user details.
- A new user entry is created in the database.

5.2. User Login

- User provides email and password.
- Credentials are validated against stored records.
- Upon successful verification, the system issues a JWT access token.
- The client stores this token and uses it for all authenticated operations.

---

6. JWT Token Specification

- The token contains essential claims such as user_id, issued-at timestamp, and expiration time.
- Tokens are signed using a secure server-side secret to prevent tampering.
- The backend validates token authenticity and validity before processing any request.
- Expired or invalid tokens result in immediate access denial.

---

7. Protected API Endpoints

All endpoints related to user-specific data require authentication. These include, but are not limited to:

- URL creation
- URL deletion
- URL list retrieval
- Access to logs
- Access to alert history

A valid JWT must be included in the Authorization header for these operations.

---

8. Database Integration

Authentication relies on the **Users** table as the primary data source.
All other entities—such as URLs, check logs, and alerts—are associated with users through foreign key relationships. This ensures proper ownership and isolation of user data across the system.

---

9. Design Rationale

JWT was selected due to its suitability for stateless APIs, compatibility with frontend frameworks like Next.js, reduced server overhead, and industry-wide adoption in SaaS platforms. This approach aligns with the lightweight and scalable design goals of PulsePing.