# API & Architecture Documentation

## Internal Application Routes

### Authentication

| **Method** | **Endpoint** | **Description** | **Access Control** |
| --- | --- | --- | --- |
| GET/POST | /login | Handles user login & 2FA verification. | Public |
| GET/POST | /register | Creates new user, hashes password, generates 2FA secret. | Public |
| GET | /logout | Terminates user session. | Authenticated |
| GET | /setup\_2fa/<id> | Generates QR code for TOTP setup. | Public (Registration flow) |

### Dashboard & Features

| **Method** | **Endpoint** | **Description** | **Access Control** |
| --- | --- | --- | --- |
| GET | /dashboard | User overview and navigation hub. | Authenticated |
| GET/POST | /checker | Password strength analysis & HIBP check. | Authenticated |

## External API Integration

### Have I Been Pwned (HIBP) - k-Anonymity Implementation

This application utilizes the HIBP API v3 to check for compromised passwords without violating user privacy.

**Workflow:**

1. **Input:** User enters password P.
2. **Hashing:** System calculates SHA-1(P).
   * *Example:* 5BAA61E4C9B93F3F0682250B6CF8331B7EE68FD8
3. **Truncation:** System splits hash into **Prefix** (first 5 chars) and **Suffix** (remaining chars).
   * *Prefix:* 5BAA6
   * *Suffix:* 1E4C9...
4. **Request:** System sends GET request to https://api.pwnedpasswords.com/range/5BAA6.
5. **Response:** API returns a list of hundreds of suffixes that match that prefix, along with their breach counts.
6. **Comparison:** System iterates through the response list locally to see if our *Suffix* exists.
   * *Match:* Password is pwned.
   * *No Match:* Password is safe.

**Security Benefit:** The full hash is never sent over the network. The API server never sees the full hash, preserving the user's password privacy.