# User Registration and Login system

**1. Front-End (User Registration and Login Page)**

**Registration Page**

* **Fields Required**:
  + **Way1**:
    - **Email Address**: For communication and verification purposes. A unique identifier for each user.
    - **Password**: Ensures account security.
    - **Confirm Password**: Minimizes entry mistakes.
    - **Terms and Conditions Checkbox**: Ensures users agree to the platform’s rules.
    - **Location (Auto-detected)**: Based on IP, display the approximate location and give the option for manual entry if needed. (don’t show in User Profile, Just save in database and give field in admin panel about user.
  + **Way2**:
    - **Use Gmail account to logging**
    - **Terms and Conditions Checkbox**: Ensures users agree to the platform’s rules.
    - **Location (Auto-detected)**: Based on IP, display the approximate location and give the option for manual entry if needed.(don’t show for user. Just save in database and give field in admin panel about user
* **UI/UX Features**:
  + Real-time validation (e.g., email availability, email format).
  + Password strength meter.
  + A user-friendly message indicating if any field is invalid.
  + Error/success messages.

**Login Page**

* **Fields Required**:
  + **Email**: To allow flexibility in login options.
  + **Password**: For authentication.
* **Additional Elements**:
  + "Remember Me" checkbox for persistent login.
  + "Forgot Password" link for account recovery.
  + **Captcha or reCAPTCHA**: Prevents automated login attempts.

**2. Back-End (Django with PostgreSQL)**

**Registration Process**

1. **Data Validation**:
   * Validate that email are unique.
   * Validate password strength based on predefined criteria (e.g., minimum length, special characters).
2. **User Creation and Encryption**:
   * **Hash Password**: Use Django’s make\_password function to hash and salt the password securely.
   * **IP and Location Logging**: Capture and store the IP and location derived from the IP address.
   * **Email Verification Token**:
     + Generate a unique token for each user.
     + Store it in a verification table or the user profile.
3. **Send Verification Email**:
   * Use Django’s email backend to send an email with a link containing the verification token.
   * Include a link with a secure one-time-use token.
4. **Save Data to Database**:
   * Only after the email is verified, set the account status to active.
   * Use PostgreSQL to store user details and hashed passwords.

**Login Process**

1. **Authenticate User**:
   * Check if the email and password combination matches.
   * Ensure that the account is verified and active.
2. **Session Management**:
   * Use Django’s session framework to create a session for authenticated users.
   * For security, set session expiration based on “Remember Me” (e.g., 60 minutes if unchecked, 7 days if checked).
3. **IP and Location Check** (Optional):
   * Track login IPs and locations to detect unusual login behavior.
   * Store login timestamps and locations in an audit table.

**3. Security Considerations**

1. **Password Handling**:
   * **Hashing**: Ensure passwords are hashed with a strong algorithm (Django uses PBKDF2 by default, which is secure).
   * **Minimum Length and Complexity**: Enforce a policy for strong passwords.
   * **Rate Limiting**: Set rate limits on login attempts to prevent brute-force attacks.
2. **Email Verification**:
   * Protect against fake accounts by enforcing email verification.
   * Store tokens securely, ensuring they expire after a limited time (e.g., 24 hours).
3. **Captcha for Bots**:
   * Use Google reCAPTCHA or similar solutions to prevent bot registration.
4. **Encryption and Security Protocols**:
   * Use HTTPS to encrypt all data in transit.
   * Store sensitive information (e.g., LinkedIn login credentials) in an encrypted format if required.
5. **Session Security**:
   * Set secure and HttpOnly flags on session cookies to prevent client-side scripts from accessing session information.
   * Enable CSRF protection for form submissions, particularly for the login form.
6. **Audit Logging**:
   * Log important actions like registration, login, password changes, and location-based access for monitoring.

**Example Workflow for Registration and Login**

**Registration Flow**

1. User fills in registration details and submits the form.
2. Backend validates and saves the user with a pending status.
3. Backend generates a unique token, sends an email with a verification link.
4. User clicks the link, backend verifies the token, and activates the account.

**Login Flow**

1. User submits login credentials.
2. Backend authenticates user, checks session and account status.
3. If successful, the user is redirected to the dashboard; otherwise, show an error.