**User Front-end Requirements:**

1. **User Registration and Authentication:**
   * User-friendly registration and login pages.
   * Password reset functionality.
   * Social media login integration (optional).
2. **User Profile Management:**
   * Users should be able to create and edit their profiles.
   * Include fields for skills, location, availability, and a brief bio.
   * Profile picture upload functionality.
3. **Skill Listings:**
   * Users should be able to create and browse listings for skills they offer and skills they are looking for.
   * Filter and search options to find relevant skill listings.
   * Ability to view detailed information about a skill listing, including the user offering it.
4. **Messaging System:**
   * In-app messaging to facilitate communication between users.
   * Real-time chat or messaging notifications.
   * Message history and file attachments.
5. **Notification System:**
   * Alerts for new messages, skill requests, and matches.
   * Email notifications (optional).
6. **Matching and Requests:**
   * Users can send skill swap requests to others.
   * A system to approve, reject, or accept requests.
   * EXTRA AFTER DONE >> A matching algorithm to suggest potential skill swap partners.
7. **Reviews and Ratings:**
   * Users can leave feedback and ratings for each other after a skill swap.
   * Publicly visible user ratings and reviews.
8. **Settings and Preferences:**
   * Account settings, including email preferences and notification settings.
   * Privacy settings for profile visibility.
9. **User Dashboard:**
   * A personalized dashboard to view matches, pending requests,and recent messages.

**Admin Back-end Requirements:**

1. **Database:**
2. **User Authentication:**
   * Implement secure user authentication
   * Store hashed passwords.
3. **Skill Listings Management:**
   * APIs to create, read, update, and delete skill listings.
   * Implement user authorization checks to ensure only the owner can edit their listings.
4. **Messaging System:**
   * Develop a real-time messaging system

using WebSocket chat API.

**EXTRA IF I COULD: Matching Algorithm:**

* + Implement a matching algorithm to suggest potential skill swap partners based on user skills and preferences.

1. **Notifications and Email Integration:**
   * Set up a notification system to send real-time alerts to users.
   * Configure email notifications (if required).
2. **Request Handling:**
   * Develop APIs to send, accept, reject, and manage skill swap requests.
3. **Review and Rating System:**
   * Create APIs for users to leave reviews and ratings for each other.
   * Calculate and display average user ratings.

**EXTRA NOT REQUIRED: Security:**

* + Implement security best practices to protect user data and prevent common web vulnerabilities, such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).

1. **Scalability and Performance:**
   * Design the backend to be scalable to handle a growing user base.
   * Optimize queries and code for performance.
2. **API Documentation:**
   * Create clear and comprehensive API documentation for internal and external use.
3. **Deployment:**
   * Deploy the backend on a reliable hosting environment (e.g., AWS, Azure, Heroku) and ensure high availability.
4. **Monitoring and Logging:**
   * Set up monitoring and logging to track system health and user activity.

**Admin Front-end Requirements:**

1. **Dashboard:**
   * An overview of key platform statistics, such as the number of users, active listings, and recent activities.
   * Search for specific user
2. **User Management:**
   * The ability to view and delete accounts.
   * Search and filter options for user management.
3. **Skill Listing Management:**
   * View and moderate skill listings
   * Ability to add or remove skill categories.
4. **Message Monitoring:**
   * Access to the messaging system to investigate reported issues or inappropriate content.
5. **Review and Rating Management:**
   * Ability to view, or delete user reviews and ratings.
   * Investigate and address reported issues with reviews or ratings.
6. **Report Handling:**
   * A section for handling user-generated reports, including the ability to take action based on reported incidents.
7. **Notification and Alerts:**
   * Real-time alerts or notifications for admin-related activities, such as new user reports or flagged content.
8. **Settings and Configuration:**
   * Admin panel settings for customization and configuration of

**Admin Back-end Requirements:**

1. **Access Control:**
   * Implement role-based access control to restrict admin actions based on their permissions.
2. **User Management:**
   * Create back-end APIs to manage user accounts, including suspend, restore, or delete functionality.
   * Log admin actions and user interactions for auditing purposes.
3. **Content Moderation:**
   * Implement content moderation tools and APIs for removing or flagging inappropriate content, such as listings, messages, and reviews.
4. **Report Handling:**
   * Set up a system to receive and manage user-generated reports.
   * Implement workflows for resolving reported issues.
5. **Analytics and Reporting:**
   * Collect and provide reports on platform usage and activities for decision-making and monitoring.
6. **Security and Compliance:**
   * Ensure the admin panel is secure and compliant with data protection regulations.
   * Implement features like two-factor authentication for admin accounts.
7. **Notification System:**
   * Enable real-time notifications for admin actions and user reports.
   * Configure email notifications for critical incidents.
8. **Documentation and Training:**
   * Create documentation for admin tools and workflows.
   * Provide training and support to admin staff.
9. **Audit Trail:**
   * Maintain a comprehensive audit trail to track admin actions and changes to the platform.
10. **Data Export and Backup:**
    * Allow admin export and backup options for user data and platform content.