Skill Exchange Project Documentation

# 1. Functional Requirements

1. User Registration and Login (JWT-based authentication)
2. Profile Management (Add/Edit skills and skills to learn)
3. Skill Matching (Find users who can teach what you want to learn and vice versa)
4. Real-time Chat between matched users
5. Session Management (Token validation on app load, auto-logout if invalid)

# 2. Database Design

## User Collection

* username: String, required, unique
* email: String, required, unique
* password: String, required (hashed)
* skills: [String]
* wantsToLearn: [String]

## Chat Collection

* users: [ObjectId] (references User)
* messages: [ { sender: ObjectId (User), content: String, timestamp: Date } ]

# 3. Project Structure & REST API Design

## Backend Structure

* controllers/: Business logic for authentication, chat, skills, and matching
* routes/: Express route definitions for each feature
* models/: Mongoose schemas for User and Chat
* middleware/: JWT authentication middleware
* index.js: Main server file, sets up Express, Socket.io, and MongoDB connection

## Frontend Structure

* src/: React components (App.jsx, Login.jsx, Register.jsx, Skills.jsx)
* public/: Static assets and images

## Main REST API Endpoints

* POST /api/auth/register - Register a new user
* POST /api/auth/login - Login and receive JWT token
* GET /api/auth/check - Validate JWT token
* GET /api/skills - Get user skills and wants
* POST /api/skills - Update user skills and wants
* GET /api/match - Find matching users
* GET /api/chats/:otherUserId - Get chat history with another user
* POST /api/chats/:otherUserId - Send a message to another user

# 4. UI Design

1. Login/Register screen: Allows user to register or login
2. Profile screen: Shows and allows editing of skills and skills to learn
3. Match screen: Shows list of matching users and allows starting a chat
4. Chat window: Real-time chat interface with matched users
5. Logout button: Allows user to log out and clear session