# Application Instruction Manual

## 1. Introduction

My app is a client-server application designed to connect users based on shared interests and age preferences. It features a Java backend that manages data persistence via a PostgreSQL database and a React frontend that provides distinct user experiences through role-based scenes (User vs. Admin).

## 2. System Requirements

- **Java Development Kit (JDK):** Version 11 or higher.
- **Node.js & npm:** For running the frontend.
- **PostgreSQL:** Database server running on port 5432.
- **IDE:** IntelliJ IDEA (Backend) and VS Code (Frontend) recommended.

## 3. Setup

### Database Setup

1. Open pgAdmin or your terminal.
2. Ensure a database named postgres exists.
3. Open the Query Tool and run the Database Initialization Script provided at the end of this document.
4. This script will automatically create the required tables (profiles, match_preferences) and configure the guest user with the correct permissions.

### Backend Setup

1. Open the project folder in IntelliJ IDEA.
2. Navigate to src/main/java/org/example/App.java.
3. Run the main method.

### Frontend Setup

1. Open a terminal in the project's root folder.
2. Run the command: npm start
3. The application will automatically open in your web browser

## 4. User Guide: Navigating the Application

The application is divided into three distinct scenes.

### Scene 1: Login & Role Selection

Upon launching the application, you are presented with the **Welcome Screen**.

- **Enter as User:** Logs you into the system with Guest privileges (Standard User).

- **Admin Dashboard:** Logs you into the system with Admin privileges (System Manager).

**Scene 2: User Dashboard (Client View)**

This scene allows general users to interact with the community. It is divided into three tabs:

1. **Browse Profiles:**
    - View a list of all current users.
    - Use the **Search Bar** to filter by username or interest.
    - **Edit/Rename:** Click the "Edit" (pencil) icon next to your name to update your username.
2. **Join (Create Profile):**
    - Fill in the form with a unique **Username**, **Age**, and **Primary Interest**.
    - Click **"Join Now"** to save your profile to the database.
3. **Match:**
    - Enter your own username to identify yourself.
    - Set **Min Age** and **Max Age** preferences.
    - The system will display a list of compatible friends who match your age criteria.

**Scene 3: Admin Dashboard (Manager View)**

This scene is for system administrators to monitor and manage the application state.

- **System Statistics:** View real-time counts of total profiles and database connection status.
- **Role Indicator:** Confirms that the backend is connected via the secure admin database role.
- **User Management Table:**
    - Displays a detailed table of all registered users.
    - **Force Delete:** Admins have a dedicated **"Delete"** button to permanently remove any user from the database.

```
CREATE TABLE IF NOT EXISTS profiles (
    username VARCHAR(50) PRIMARY KEY,
    age INT,
    primary_interest VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS match_preferences (
    id SERIAL PRIMARY KEY,
    profile_username VARCHAR(50) REFERENCES profiles(username)
ON DELETE CASCADE,
    min_age INT,
    max_age INT
);

DO
$do$
BEGIN
   IF NOT EXISTS (
      SELECT FROM pg_catalog.pg_roles
      WHERE  rolname = 'guest') THEN
      CREATE USER guest WITH PASSWORD 'guest123';
   END IF;
END
$do$;

GRANT CONNECT ON DATABASE postgres TO guest;
GRANT USAGE ON SCHEMA public TO guest;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA
public TO guest;
GRANT SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO guest;
```