SkillSync – Freelance Collaboration Platform

Objective

Build a full-stack web application using **React (Basic + Advanced)** on the frontend, **NestJS** for the backend, and **MySQL** as the database

- Authentication
- RESTful APIs
- Database relationships
- File handling
- Role-based access control
- Structured messaging

Project Scope

Build a freelance collaboration platform where:

- Clients can post projects and assign them to freelancers.
- Freelancers can browse projects, submit bids, and communicate with clients.
- Both clients and freelancers can send messages, share files, manage milestones, and track invoices.

Technical Requirements

Backend (NestJS)

- Modular architecture (services, controllers, DTOs, guards)
- Role-based access: client, freelancer
- JWT-based authentication (access & refresh tokens)
- File upload handling
- Middleware for request logging
- API to manage:
    - Users & profiles
    - Projects & bids
    - Messages (non-realtime, window-style)
    - Milestones & invoices
    - File/image uploads

Frontend (React)

- Responsive design with protected routes
- Role-based dashboards
- API integration via Axios
- Form validation (Formik, Yup, or similar)
- Messaging window per project
- Image/file upload with preview

Database (PostgreSQL + TypeORM)

Use relational design with at least these tables:

- users
- projects
- bids
- milestones
- messages
- files
- invoices
- skills

Deliverables

1. Backend Codebase (NestJS)
2. Frontend Codebase (React)
3. ER Diagram + TypeORM entities or SQL schema
4. Fully functional modules:
    - Authentication & Roles
    - Project & Bid Management
    - Messaging System
    - Milestone & Invoice Management
    - File Uploading
5. GitHub Repository with:

  - Complete code
  - README file
  - Screenshots (optional)

User Journey (Step-by-Step)

Common Steps (for All Users)

1. **Visit Website**
   - Land on homepage
   - View features, proceed to register/login
2. **Register Account**
   - Choose role (Client/Freelancer)
   - Fill in name, email, and password
3. **Login**
   - Submit credentials
   - Receive JWT token
4. **Complete Profile**
   - Add/edit bio, skills, and upload profile image
   - Dashboard varies by role
5. **Send & Receive Messages**
   - Threaded messages under projects
6. **Logout**
   - End session securely

Client Journey

7. **Create Project**
    - Title, category, description, budget, deadline
8. **View Bids**
    - See all freelancer bids
9. **Assign Project**
    - Choose a freelancer and assign
10. **Upload Files**
    - Add briefs and assets to the project
11. **Add Milestones**
    - Break into parts with deadlines
12. **Generate Invoices**
    - One per milestone; mark as paid

Freelancer Journey

7. **Add Skills**
    - From a predefined list
8. **Browse Projects**
    - Search/filter open jobs
9. **Submit Bid**

    - Enter amount, duration, message

10. **View Assigned Projects**
    - See accepted work
11. **Upload Deliverables**
    - Submit work files
12. **Track Milestones**
    - View progress and deadlines
13. **View/Download Invoices**
    - Check payments per milestone

UI Screen Details

Each screen includes purpose, role, fields, components, and actions.

---

1. Login / Register

- **Purpose**: Authenticate users and assign roles
- **Role**: Common
- **Fields**: Email, Password, Confirm Password (register), Role
- **Components**: Input fields, role toggle, login/register buttons, forgot password link
- **Actions**: Submit form, redirect to dashboard

---

2. Dashboard

- **Purpose**: Display an overview
- **Role**: Client / Freelancer
- **Info**: Welcome message, project stats
- **Components**: Sidebar nav, cards/table, stats
- **Actions**: Navigate, view details

---

3. Profile Management

- **Purpose**: Manage personal info and skills
- **Role**: Both
- **Fields**: Name, Bio, Skills, Profile Image
- **Components**: Form, file uploader, tag selector
- **Actions**: Save, upload image

---

4. Project Creation (Client Only)

- **Purpose**: Post new project
- **Role**: Client
- **Fields**: Title, Category, Description, Budget, Deadline
- **Components**: Form, submit button
- **Actions**: Save project, redirect

---

5. Project Listing/Search (Freelancer Only)

- **Purpose**: Show all active projects
- **Role**: Freelancer
- **Filters**: Category, Budget, Deadline
- **Components**: Filter panel, project cards
- **Actions**: View details, bid

---

6. Project Detail View

- **Purpose**: Project info and bidding
- **Role**: Freelancer (view/bid), Client (assign)
- **Info**: Description, bid list
- **Components**: Text view, bid form/table
- **Actions**: Submit bid, assign freelancer

---

7. Messaging Window

- **Purpose**: Project-related communication
- **Role**: Both
- **Fields**: Message text, attachments
- **Components**: Message bubbles, input box, scrollable thread
- **Actions**: Send, mark read

---

8. File Uploads

- **Purpose**: Share files
- **Role**: Both
- **Fields**: File (PDF, DOCX, JPG, PNG)
- **Components**: Upload area, file list
- **Actions**: Upload, download

---

9. Milestones & Invoices

- **Purpose**: Track progress and payments
- **Role**: Client (create), Freelancer (track)
- **Fields**: Title, Due Date, Amount, Status
- **Components**: Table, invoice card
- **Actions**: Create milestone, toggle paid

NOTE: You may use any free HTML UI template for layout inspiration.

Assignment Guidelines

Frontend (React.js):

- **DRY Principle**: Avoid repetitive code.
- **State Management**: Use **Redux** or **Context API**.
- **Form Validation**: Ensure proper validation and error handling.
- **RBAC**: Implement role-based access control.
- **Protected Routes**: Used to restrict access to authorized users.

Backend (NestJS):

- **SOLID Principles**: Follow best coding practices.
- **Middleware & Guards**: Handle auth, logging, and access control.
- **Rate Limiting**: Prevent API abuse.
- **Validation**: Validate incoming data with **class-validator**.
- **Migrations**: Manage database migrations.
- **API Docs**:

GitHub Repo:

- **README**: Include project overview, setup instructions, and environment variables.
- **Environment Details**: Mention all required environment variables.