National University of Computer & Emerging Sciences
Department of Software Engineering
Course: Web Engineering - Spring 2025

Assignment 4

Total Marks: 55

Assignment Deadline	30 April, 2025

Instructions:

1. This assignment involves building the SkillSwap platform with real-world functionality connecting freelancers with clients. You must provide complete and functional React, CSS, JavaScript, Node.js, Express, and MongoDB code following the MERN stack architecture with microservices.
2. Submissions must be in a ZIP file containing your complete project folder structure including frontend (React), backend (Node.js/Express microservices), and database (MongoDB Atlas) components.
3. The assignments will be checked not through MOSS but through a custom built pledge detector tool which successfully detected 30+ pledges in the assignment 3 and 20+pledges in the assignment 2, so avoid plagiarism among each other.
4. Submit your assignment as a single ZIP file named 22F-XXXX_Web_Assignment_4.zip
5. Submissions not following the submission structure of the assignment will receive deductions.
6. Directly copying from peers/online sources/AI will result in 80% marks deductions in the assignment for both students.
7. The task is mainly of 50 marks. However, there are 5 bonus marks written in the end.
8. Submissions must follow the given file structure:

```
/skillswap
├── /client                 # React frontend
│   ├── /public             # Public assets
│   │   └── /images         # Static images
│   ├── /src                # Source files
│   │   ├── /components     # Reusable components
│   │   │   ├── /common     # Shared components
│   │   │   ├── /client     # Client-related components
│   │   │   ├── /freelancer # Freelancer-related components
│   │   │   └── /admin      # Admin-related components
│   │   ├── /pages          # Page components
│   │   │   ├── /client     # Client pages
│   │   │   ├── /freelancer # Freelancer pages
│   │   │   ├── /admin      # Admin pages
│   │   │   └── /auth       # Authentication pages
│   │   ├── /context        # Context API
│   │   ├── /hooks          # Custom hooks
│   │   ├── /utils          # Utility functions
│   │   └── /assets         # Local assets
│   │       ├── /css        # CSS files
│   │       └── /icons      # SVG icons
│   └── README.md           # Frontend documentation
├── /server                 # Backend Express application
│   ├── /controllers        # Request handlers
│   │   ├── /auth           # Authentication controllers
│   │   ├── /projects       # Project controllers
│   │   └── /notifications # Notification controllers
│   ├── /routes             # API routes
│   │   ├── /auth           # Authentication routes
│   │   ├── /projects       # Project routes
│   │   └── /notifications # Notification routes
│   ├── /models             # Database models
│   ├── /middleware         # Middleware functions
│   ├── /utils              # Utility functions
│   │   ├── hash.js         # Hashing utilities
│   │   ├── jwt.js          # JWT utilities
│   │   └── config.js       # Configuration
│   └── server.js           # Server entry point
├── README.md               # Project documentation
└── .gitignore              # Git ignore file
```

Problem Statement

In this assignment, you will build a core part of the SkillSwap platform that connects freelancers with clients seeking their services. Pakistan's freelance industry lacks a localized, secure, and scalable platform. SkillSwap will address this by:

1. Connecting freelancers (graphic designers, developers, writers) with potential clients
2. Offering a secure platform for project management and service delivery
3. Using modular architecture for better scalability and maintenance
4. Replacing local MongoDB with MongoDB Atlas (cloud-based database)
5. Implementing hashing for sensitive data (passwords, contracts, messages)

Features to Implement

Client Modules (10 Features)

1. **Role-Based Authentication (JWT + Hashing)**
    - Clients sign up with email/phone verification
    - Passwords must be hashed using bcrypt
    - JWT tokens for secure session management
    - Role-specific access control (client/freelancer/admin)
2. **Freelancer Search & Filtering System**
    - Allows students to find tutors based on multiple criteria
    - Use Tailwind CSS for responsive UI components
    - Real-time filter updates without page reloads
3. **Project Posting (CRUD)**
    - Post projects with detailed information (title, description, requirements, deadline)
    - Edit/delete projects using Express APIs
    - Form validation for all project fields
    - Display project status indicators
4. **Real-Time Bidding System**
    - Freelancers bid on projects
    - Clients see bid updates without page refresh (Socket.io mock)
    - Sort and filter incoming bids
    - Counter-offer functionality
5. **In-App Messaging**
    - Chat with freelancers in real-time
    - Messages stored in MongoDB Atlas with hashed metadata
    - Message read receipts
6. **Review & Rating System**
    - Rate freelancers post-project (1–5 stars + comments)
    - Display average ratings on freelancer profiles
    - Filter reviews by rating or date
    - Response option for freelancers
7. **Analytics Dashboard**
    - Track active projects and freelancer performance
    - Visualize data using Chart.js

    - Filter analytics by date ranges
    - Export reports in CSV/PDF format

Freelancer Modules

1. **Profile Management (CRUD)**
    - Add skills, portfolio items, and profile information
    - MongoDB Atlas: freelancers collection with verified: boolean field
    - Profile completeness indicator
2. **Bid Management**
    - Submit/edit bids for projects
    - Track bid status (pending, accepted, rejected)
    - Dedicated API endpoints for bid analytics (e.g., avg. bid price)
3. **Project Management Tools**
    - Track active and completed projects
    - Manage multiple projects simultaneously
    - Organize projects by category or status
4. **Project Timeline**
    - Update progress (e.g., "50% completed") with React state
    - Deadline tracking with reminders
    - Time tracking for hourly projects
    - Milestone status updates

Admin Modules

1. **Freelancer Verification**
    - Approve/reject freelancers based on ID/docs (PDF/IMG)
    - Verification levels (Basic, Verified, Premium)
    - Document management system
2. **Platform Analytics**
    - Monitor transactions, user growth, and popular skills
    - MongoDB Aggregation for complex data analysis
    - Trend visualization and forecasting
    - Revenue tracking and projections
3. **Notification System**
    - Email/SMS for disputes, verifications, and updates (Twilio mock)
    - Template management for communication
    - Scheduled notifications
    - User preference settings for notification types

Technical Requirements

Frontend (React)

- **Components:** FreelancerCard, BidForm, ProjectTrackerModal, ContractViewer, MilestoneTracker

- **State Management:** Manage projects, bids, chat messages, and user sessions
- **Styling:** Tailwind CSS for responsive grids/modals/forms
- **Conditional Rendering:** Show features based on user role and verification status
- **Form Handling:** Validation, error messages, and submission handling

Backend (MERN Stack)

- **Authentication APIs**
  - `/api/auth/signup` (role-based: client/freelancer)
  - `/api/auth/login` (JWT token generation)
  - `/api/auth/verify` (email/phone verification)
  - `/api/auth/reset-password` (password reset flow)
- **Project APIs**
  - `/api/projects` (CRUD for projects)
  - `/api/projects/:id/bids` (bid management)
  - `/api/projects/:id/milestones` (milestone tracking)
  - `/api/projects/:id/contract` (contract management)
- **Notification APIs**
  - `/api/notify/email` (send email notifications)
  - `/api/notify/sms` (send SMS alerts)
  - `/api/notify/in-app` (in-app notifications)
  - `/api/notify/preferences` (manage notification settings)

Database (MongoDB Atlas(No local MongoDB))

- **Clients:** { name, email (hashed), password (hashed), projects: [projectIds] }
- **Freelancers:** { name, skills: ["React", "UI/UX"], portfolio: [...], verified: Boolean, bids: [bidIds] }
- **Projects:** { title, description, deadline, clientId, status, bids: [{freelancerId, message}] }
- **Contracts:** { projectId, clientId, freelancerId, terms, signatures, hash, versions: [...] }
- **Messages:** { senderId, receiverId, content, timestamp, readStatus, metadata (hashed) }

Advanced Topics to Implement

1. **Hashing Implementation**
   - Bcrypt for passwords with appropriate salt rounds
   - SHA-256 for contracts and sensitive documents
   - Hashed metadata for messages to protect privacy
2. **Modular API Architecture**
   - Use Express Router for modular endpoints
   - Implement clean separation of concerns
   - Create reusable middleware

    - Ensure proper error handling across modules
3. **Real-time Communication**
    - Implement WebSockets for instant messaging
    - Real-time notifications for project updates
    - Live collaboration tools for project management
    - Status indicators for user activity
4. **Cloud Database Integration**
    - Configure and connect to MongoDB Atlas
    - Implement proper indexing for performance
    - Set up data replication and backups
    - Handle connection pooling and error recovery

Bonus Challenges (5 marks)

1. **Real-time Collaboration Tools (2 marks)**
    - Implement a real-time dashboard showing project activity
    - Use WebSockets for live updates without page refresh
    - Create interactive project management tools
2. **Push Code to GitHub and Deploy Demo (3 marks)**
    - Push your complete project to a GitHub repository
    - Deploy a working demo to a hosting platform (Heroku, Netlify, Vercel etc.)
    - Document the deployment process in your README.md