# Freelancer market place for college students

**✅ 1. Clear Strategy for Division**

* **Frontend Developer (Person A):** Focuses on UI/UX, React components, page navigation, API calls.
* **Backend Developer (Person B):** Focuses on REST API, database models, business logic, authentication, and hosting backend.

📌 **Rule of thumb:** If it's **seen on screen**, it's frontend. If it's **stored or processed**, it's backend.

| **Feature** | **Frontend (Person A)** | **Backend (Person B)** |
| --- | --- | --- |
| **Authentication** | Forms for login/register, error handling, routing to dashboards | JWT Auth API, password hashing, role-based access |
| **User Dashboard** | Show user info, active jobs, navigation UI | Fetch user data from MongoDB via /api/user/:id |
| **Post Project** | Form to enter title, description, budget, skills | Save project via POST /api/projects, validate input |
| **View Projects** | Grid/list of projects, filters by skill/budget | API: GET /api/projects, optional filters |
| **Project Details** | Show project info, button to apply | GET project by ID /api/projects/:id |
| **Submit Proposal** | Form (cover letter, price, days), submit button | POST /api/proposals, link to user/project |
| **View Proposals (Client)** | List proposals for a project | GET /api/projects/:id/proposals |
| **Hire Freelancer** | Click to hire → show modal → confirm | PUT /api/projects/:id/hire |
| **Chat System** | Chat UI, message input | Socket.IO, message storage API |
| **Review & Rating** | Review form after completion | POST /api/review, link to project & freelancer |
| **Notifications** | Show new proposals, hire confirmation | Optional: Send from backend via API or socket |
| **Profile Page** | View/edit profile, list of past work | GET & PUT /api/user/:id, file upload via Multer |

GET /api/projects # List projects

POST /api/projects # Create project

GET /api/projects/:id # View one project

POST /api/proposals # Submit a proposal

GET /api/user/:id # Get user profile

POST /api/auth/register # Register user

POST /api/auth/login # Login

PUT /api/projects/:id/hire # Hire freelancer

## 📐 4. ****Component Mapping: Who Builds What****

### 🧑‍🎨 Person A – Frontend (React + Tailwind)

* /login → LoginPage.jsx
* /register → RegisterPage.jsx
* /dashboard → Dashboard.jsx
* /project/:id → ProjectDetails.jsx
* /post-project → PostProjectForm.jsx
* /chat → ChatBox.jsx
* /profile/:id → ProfilePage.jsx
* /proposals → ProposalsList.jsx
* Setup React Router, State Management (Context API or Redux)
* Use axios to talk to backend APIs

### 🛠️ Person B – Backend (Node.js + Express + MongoDB)

* models/User.js, Project.js, Proposal.js
* routes/auth.js, routes/project.js, routes/proposal.js
* controllers/ for each route logic
* middleware/auth.js for protecting routes
* utils/socket.js for chat
* Connect MongoDB using mongoose.connect
* Setup nodemon, .env, CORS

| **Task** | **Who Leads** | **Collaborates** |
| --- | --- | --- |
| Plan API endpoints | Backend dev | Frontend dev (needs to know endpoints) |
| Define database schema | Backend dev | Frontend dev (needs to know expected JSON) |
| Authentication flow | Backend dev | Frontend dev (design forms, call login API) |
| Project form validation | Frontend dev | Backend dev (duplicate server-side) |
| Hosting / Deployment | Deploy separately | Deploy Netlify (frontend) and Render (backend) |
| GitHub Repo Structure | Shared | Define /client and /server subfolders |

## 🔄 6. ****Example Workflow for One Feature: “Post a Project”****

### Frontend

* Create React form: title, description, budget
* On submit → axios.post("/api/projects", data)
* If success → show toast & redirect

### Backend

* Route: POST /api/projects
* Controller: Validate data, save to DB
* Return status 201 Created + project JSON

## 7. ****Tips to Work Together****

* Use **GitHub** with branches:
  + frontend/feature-name
  + backend/feature-name
* Sync daily on:
  + New APIs added
  + Schema changes
  + Testing frontend-backend connection
* Use tools:
  + [Postman](https://www.postman.com/) (backend test)
  + [Swagger](https://swagger.io/) (document APIs)
  + [Figma](https://www.figma.com/) (optional UI planning)

## 🧾 Final Sorted Checklist

| **Priority** | **Feature** | **Status** | **Files to Work On** |
| --- | --- | --- | --- |
| ✅ | User Auth (JWT + SQL) | ✅ Done | authController.js, userModel.js |
| ✅ | Post Job | ✅ Done | jobController.js, jobsRoutes.js |
| ✅ | Profile Basic Info | ✅ Done | profileController.js, profileRoutes.js |
| 🟩 | Submit Proposal (Bid) | 🟩 Next | ✅ Bid.js, 🚀 bidController.js, bidRoutes.js |
| 🟨 | Hire Freelancer | 🟨 After | jobController.js, SQL UPDATE jobs |
| 🟨 | Freelancer Dashboard | 🟨 After | New route → GET /api/users/:id/bids |
| 🟨 | Client Dashboard | 🟨 After | New route → GET /api/users/:id/jobs |
| 🟦 | Review System | 🟦 Later | Create reviews table + routes |
| 🟦 | Chat System | 🟦 Optional | Socket.IO + messages table |
| 🟦 | File Upload | 🟦 Optional | Multer + uploads folder |
| 🟥 | Frontend API Integration | 🟥 After backend | React + axios setup |
| 🟥 | Deploy Project | 🟥 Final step | Render / Railway + Vercel / Netlify |

## ✅✅✅ COMPLETED MODULES

### ✅ 1. ****User Authentication****

* Login / Signup with SQL DB
* JWT token
* Password hashing

### ✅ 2. ****Job Posting (Client Side)****

* Client can post jobs (projects)
* Stored in jobs table
* Route: POST /api/jobs
* Fetch jobs: GET /api/jobs

### ✅ 3. ****Profile Module (Basic Info)****

* User profiles created
* Likely handled in profileController.js

## 🟩 IN PROGRESS OR TO START NEXT

### 🟩 4. ****Proposal / Bid System (Freelancer Side)**** → ****NEXT TO BUILD****

* Create SQL table: bids
* Route: POST /api/bids
* Each bid includes: cover letter, price, days
* View bids for a job: GET /api/jobs/:id/bids
* Mark status as pending, accepted, etc.

### 🟨 5. ****Hiring System (Client Side)****

* Update job with hired freelancer
* Route: PUT /api/jobs/:id/hire
* Also update bid status to accepted
* Add field hired\_freelancer\_id in jobs table

### 🟨 6. ****Freelancer Dashboard****

* Route: GET /api/users/:id/bids
* See all jobs the freelancer applied to

### 🟨 7. ****Client Dashboard****

* Route: GET /api/users/:id/jobs
* See all jobs posted
* Route: GET /api/users/:id/hired
* See hired freelancers

### 🟦 8. ****Review and Rating System****

* Table: reviews
* Route: POST /api/reviews
* Allow client to rate freelancer
* Route: GET /api/users/:id/reviews

### 🟦 9. ****Chat System (Advanced – Optional)****

* Table: messages
* Route: POST and GET chat messages by job or users
* Use Socket.IO if possible

### 🟦 10. ****File Upload System (Optional)****

* Resume, document upload (via Multer)
* Store file paths in DB

### 🟥 11. ****Frontend Integration****

* Connect all APIs with React (or frontend of your choice)
* Use axios for:
  + Auth
  + Post job
  + Submit proposal
  + Hire
  + View dashboards

### 🟥 12. ****Deployment****

* Backend on Render / Railway / VPS
* SQL DB on PlanetScale / ClearDB / Hosted MySQL
* Frontend on Vercel / Netlify