SB Works

Full Implementation Guide

With Screen Mockups, Sample Data & API Reference

MERN STACK · SOCKET.IO · JWT AUTH · BOOTSTRAP 5

MongoDB · Express.js · React.js · Node.js

Pages	Screens	API Endpoints	User Roles
40+	12 Mockups	35+ Endpoints	3 Roles

Table of Contents

All screens, features and endpoints covered in this guide

PART 6 Complete API Reference

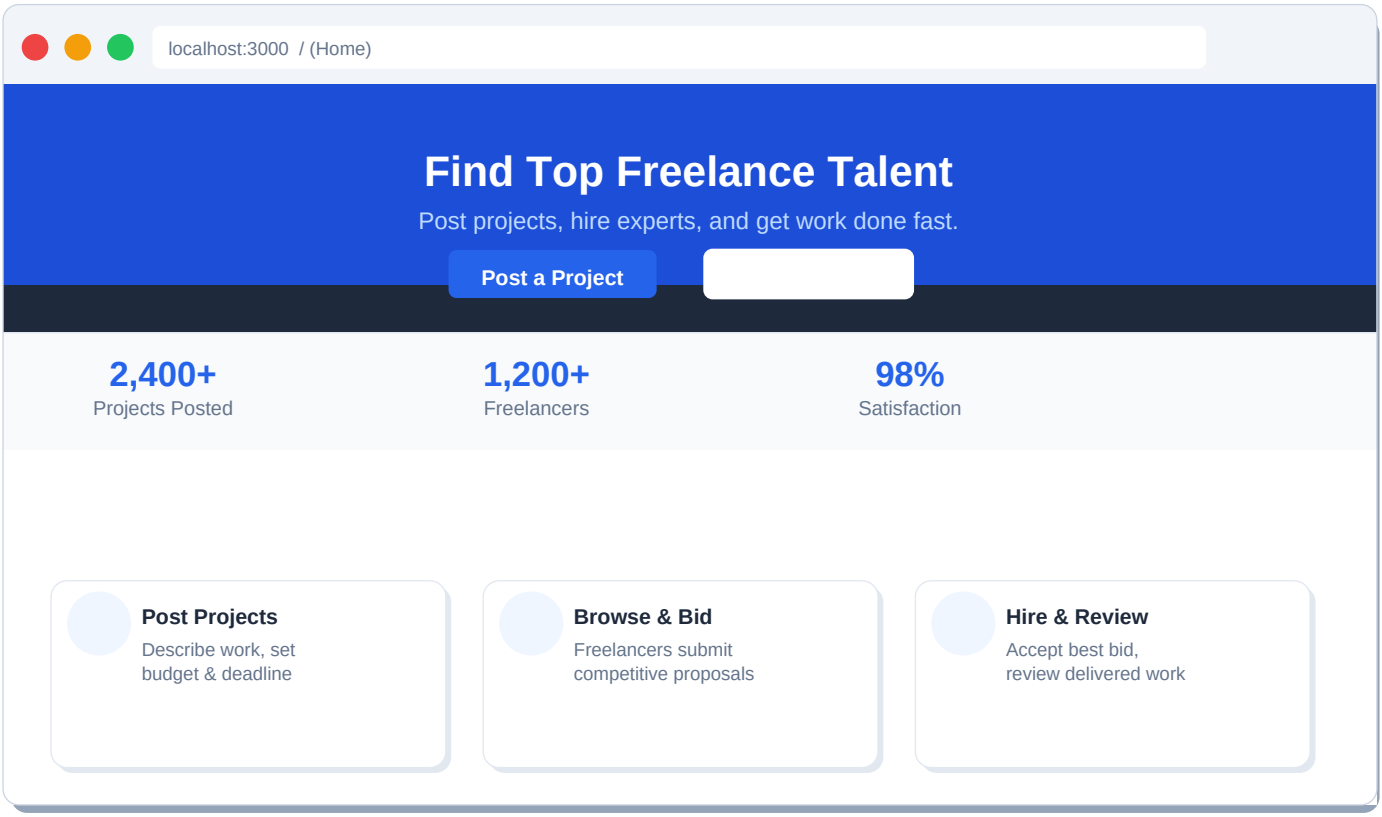
All 35+ endpoints with sample data

PART 1 — Authentication & Onboarding

Phone-based OTP login, profile completion, role selection

1.1 — Home Page / Landing Screen

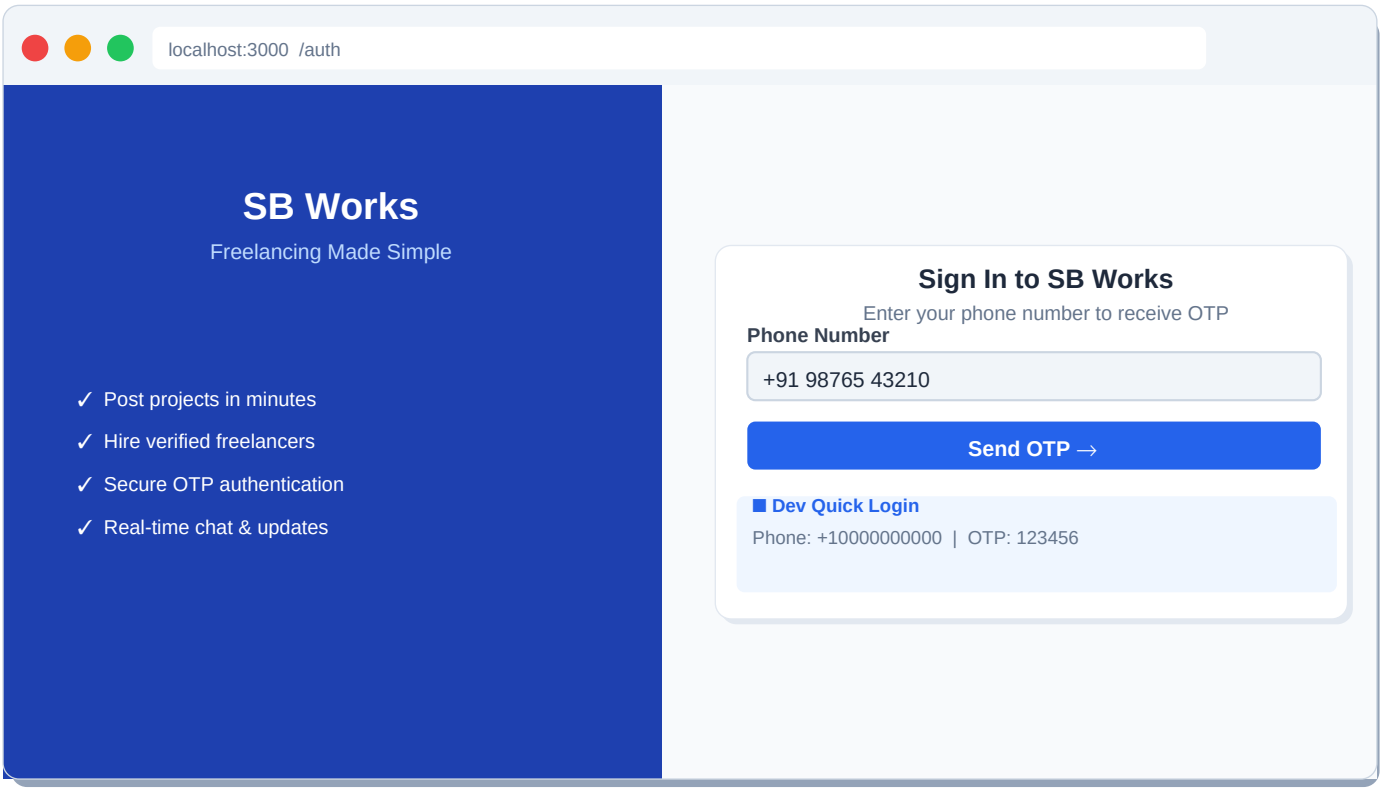
The landing page is fully public — no login required. It showcases the platform's value proposition with stats, feature cards, and calls to action for both clients and freelancers.



■ Route: src/pages/Home.jsx | No auth required | Links to /auth for login

1.2 — OTP Login: Step 1 — Enter Phone Number

The login page has two steps. Step 1: user enters phone number. Step 2: user enters OTP. In development mode, the OTP is printed directly to the backend terminal — no SMS needed.



POST `/api/user/get-otp` — Public

Accepts a phone number, generates a 6-digit OTP, saves it with 90-second expiry. In dev mode prints to console; in prod sends SMS via Twilio.

REQUEST BODY

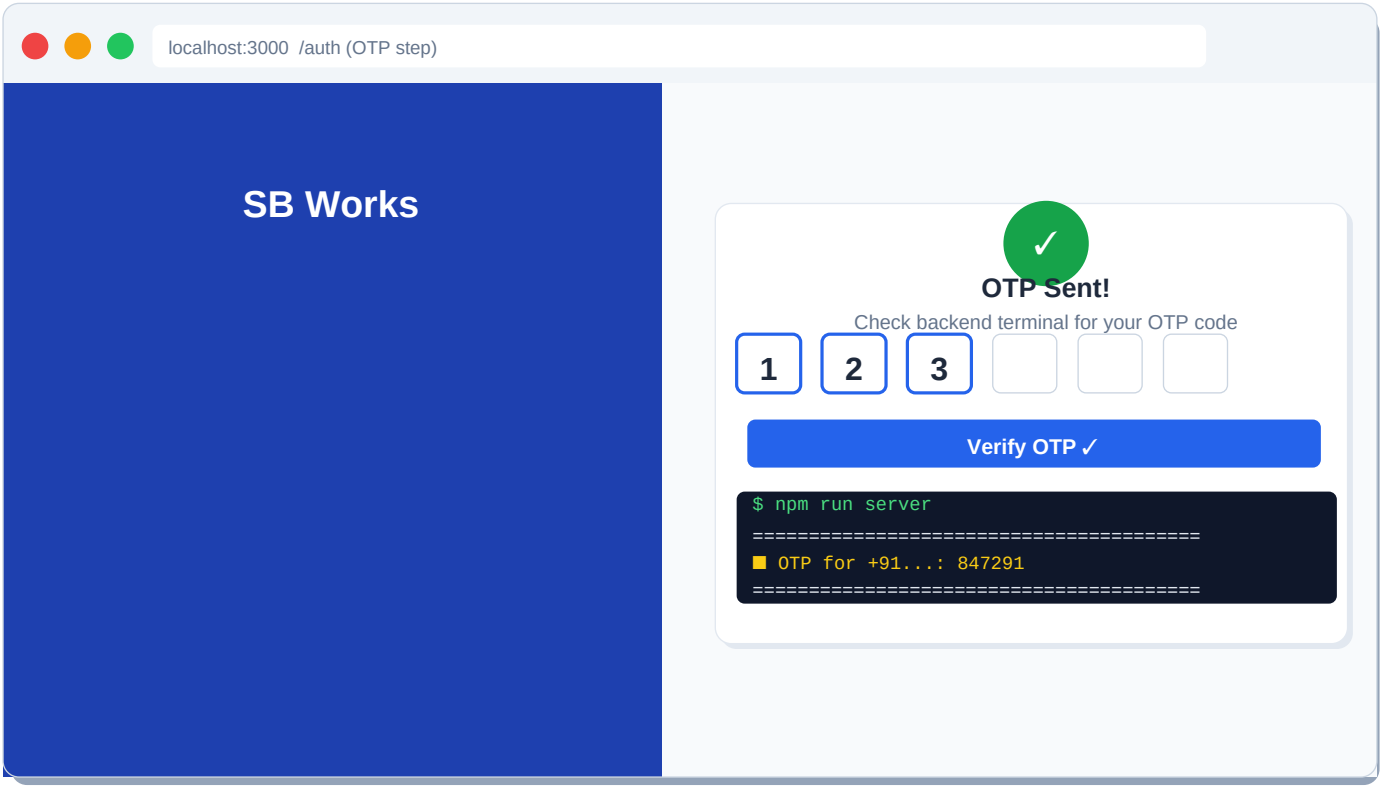
`phoneNumber`	String REQUIRED — e.g. "+919876543210"

RESPONSE (200 OK)

```
{ "message": "OTP sent successfully", "success": true }
```

1.3 — OTP Login: Step 2 — Verify OTP Code

After receiving the OTP (check backend terminal in dev mode), the user enters 6 digits. On success, JWT access + refresh tokens are set as HttpOnly cookies.



POST `/api/user/check-otp` — Public

Validates OTP code against stored value and expiry. On success, sets signed HttpOnly JWT cookies. Returns user data including isActive flag to determine redirect target.

REQUEST BODY

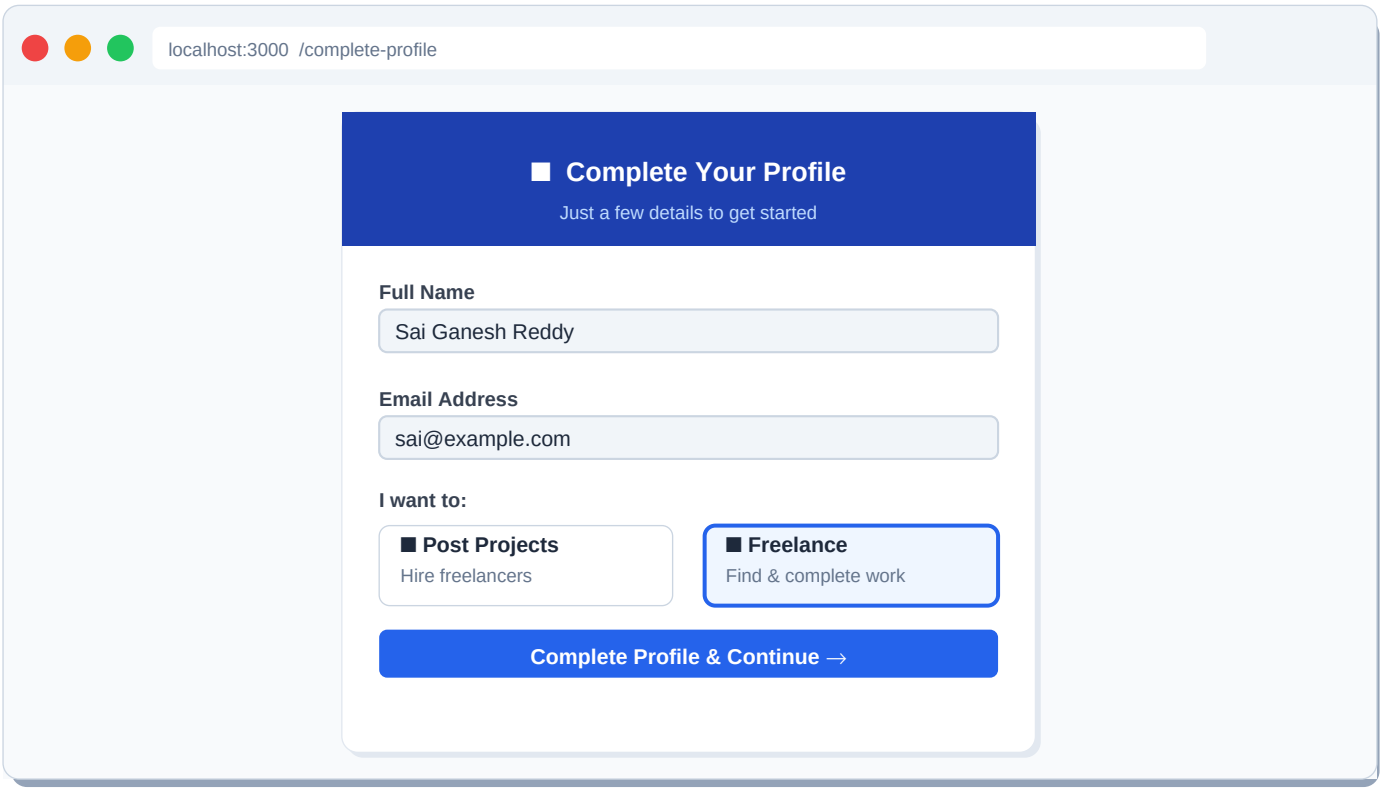
`phoneNumber`	String REQUIRED — same phone used in get-otp
`otp`	Number REQUIRED — 6-digit code from terminal/SMS

RESPONSE (200 OK)

```
{ "user": { "_id": "...", "phoneNumber": "+919876543210", "role": "USER", "isActive": false, "isVerifiedPhoneNumber": true }, "success": true }
```

1.4 — Complete Profile

New users (isActive=false) are redirected here after first login. They set their full name, email, and choose their role — OWNER (post projects) or FREELANCER (find work).



POST `/api/user/complete-profile` — Token

Sets name, email, and role for a newly registered user. Sets isActive=true so they are redirected to their role dashboard on next request.

REQUEST BODY

`name`	String REQUIRED — Full name, min 3 chars
`email`	String REQUIRED — Valid email address
`role`	String REQUIRED — "OWNER" or "FREELANCER"

RESPONSE (200 OK)

```
{ "user": { "name": "Rahul Kumar", "email": "rahul@example.com", "role": "FREELANCER",
"isActive": true }, "success": true }
```

GET `/api/user/profile` — Token

Returns the full profile of the currently authenticated user including role, status, skills, rating.

RESPONSE (200 OK)

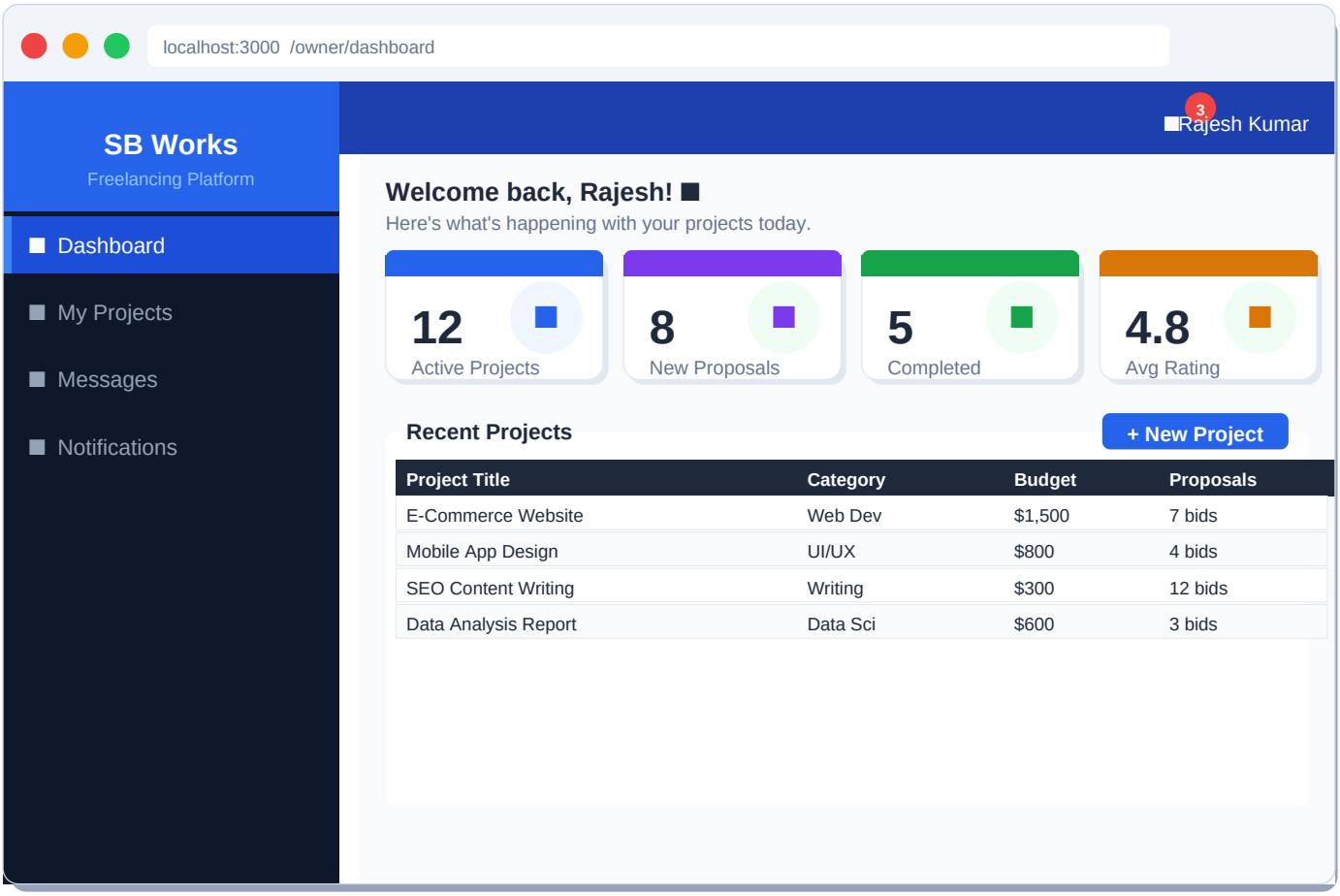
```
{ "user": { "_id": "64a3b...", "name": "Rahul Kumar", "email": "rahul@example.com", "role":
"FREELANCER", "status": 2, "rating": 4.8, "totalReviews": 12 } }
```

PART 2 — Client (Owner) Dashboard

Post projects, review proposals, hire freelancers, approve work, leave reviews

2.1 — Owner Dashboard Overview

The owner's home screen shows key stats (active projects, new proposals, completions, avg rating) and a table of recent projects. The sidebar provides access to all owner features.



■ Layout: src/layouts/OwnerLayout.jsx | Route guard: role must be OWNER or ADMIN | Stats are computed from live MongoDB queries — project count, proposal count, completions

2.2 — My Projects: List View

Shows all projects posted by the logged-in owner. Each project shows status (OPEN/CLOSED), category, budget, and proposal count. Owner can toggle status, edit, or delete projects.

GET	/api/project/owner-projects	■ Owner

Returns all projects owned by the authenticated user, populated with proposal count. Sorted by creation date descending (newest first).

RESPONSE (200 OK)

{ "projects": [{ "_id": "64a3b...", "title": "React E-Commerce Website", "status": "OPEN", "budget": 1500, "category": { "title": "Web Development" }, "proposals": [...], "freelancer": null, "createdAt": "2024-01-15T..." }], "success": true }

PATCH `/api/project/:id` ■ Owner

Toggles project status between OPEN and CLOSED. A closed project does not appear in the freelancer browse feed. Cannot close if a freelancer is assigned.

RESPONSE (200 OK)

{ "project": { "status": "CLOSED", ... }, "success": true }

DELETE `/api/project/:id` ■ Owner

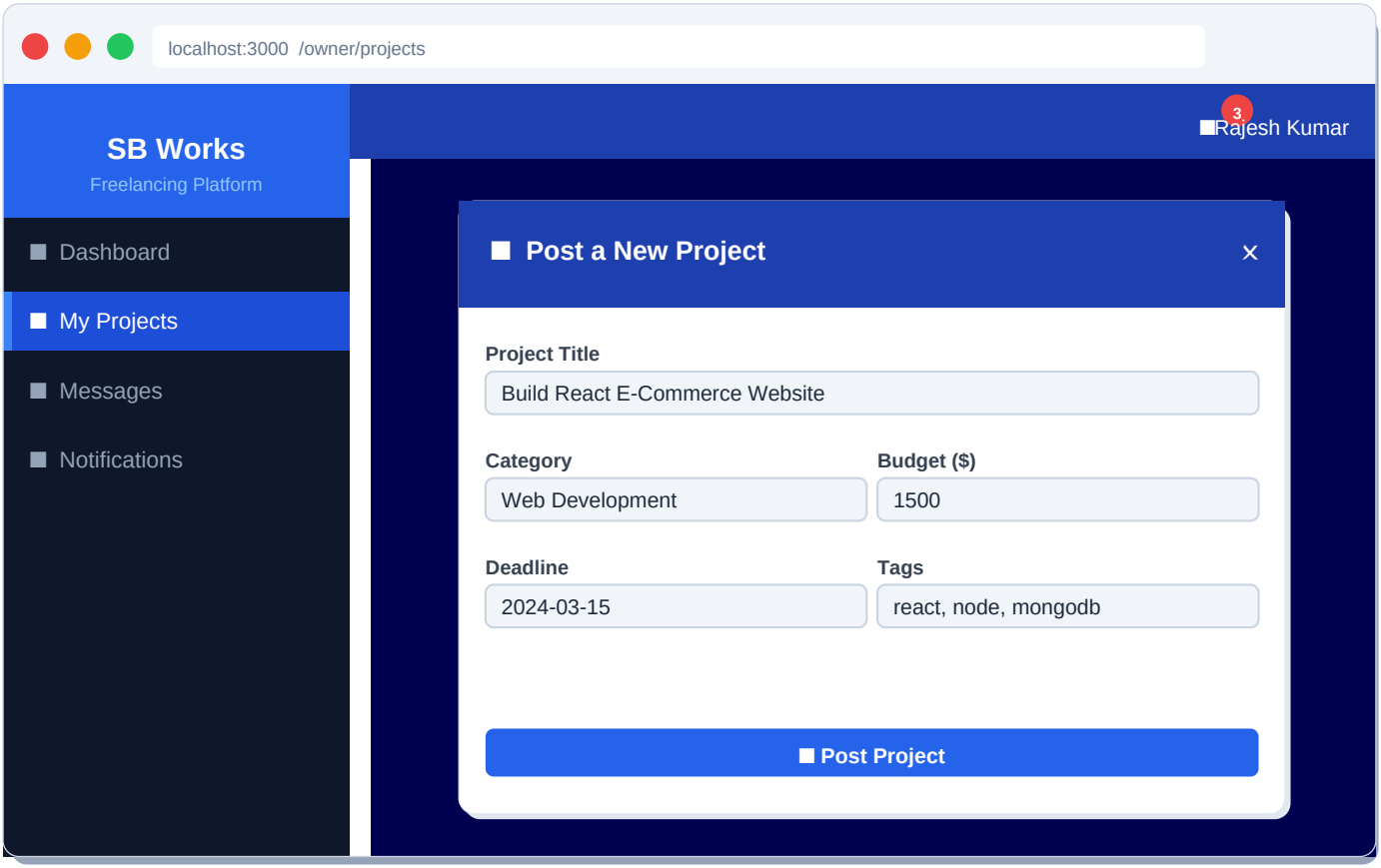
Deletes a project. Only allowed if no freelancer has been assigned yet. All associated proposals are also deleted.

RESPONSE (200 OK)

{ "message": "Project deleted successfully", "success": true }

2.3 — Create New Project

The "Post Project" button opens a modal form. The owner fills in all project details and submits. The project is immediately visible to freelancers in the browse feed.



POST `/api/project/add` ■ Owner

Creates a new project. The project status defaults to OPEN. The authenticated user is set as the owner automatically.

REQUEST BODY

`title`	String REQUIRED — min 10 chars, e.g. "Build React E-Commerce Website"
`description`	String REQUIRED — detailed project description
`category`	ObjectId REQUIRED — category ID from /api/category/list
`budget`	Number REQUIRED — budget in USD, e.g. 1500
`deadline`	Date REQUIRED — ISO date string e.g. "2024-03-15"
`tags`	Array[String] OPTIONAL — e.g. ["react", "node", "mongodb"]

RESPONSE (200 OK)

```
{ "project": { "_id": "64c5d...", "title": "Build React E-Commerce Website", "status": "OPEN", "budget": 1500, "owner": "64a3b...", "proposals": [] }, "success": true }
```

PATCH `/api/project/update/:id` ■ Owner

Updates any project field (title, description, budget, deadline, tags). Cannot change category or owner. Returns updated project.

REQUEST BODY

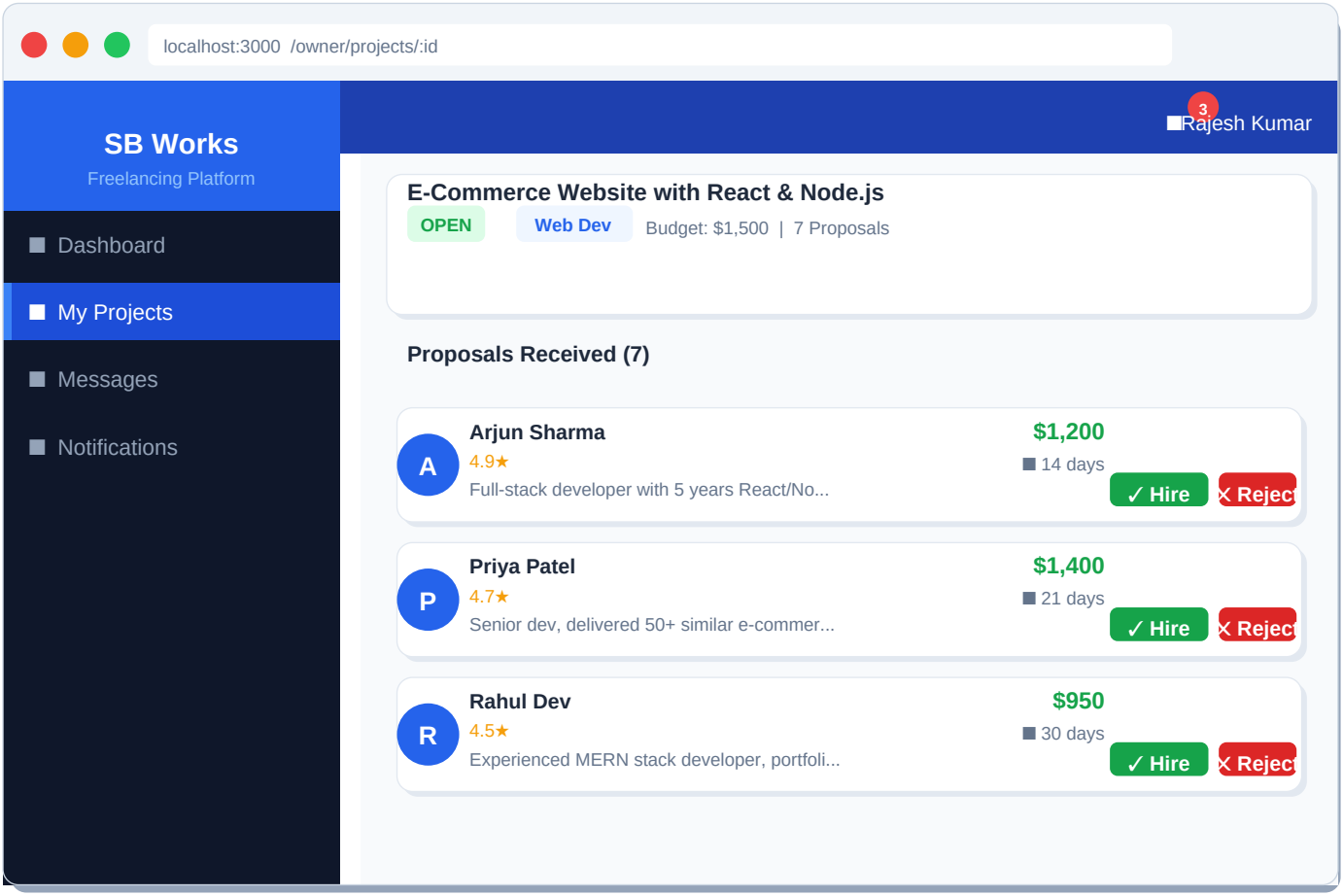
`title`	String OPTIONAL — updated title
`description`	String OPTIONAL — updated description
`budget`	Number OPTIONAL — updated budget
`deadline`	Date OPTIONAL — updated deadline
`tags`	Array OPTIONAL — updated tags array

RESPONSE (200 OK)

```
{ "project": { ...updated fields... }, "success": true }
```

2.4 — View Project Detail & Proposals

Clicking a project opens the detail view. The owner sees all submitted proposals with freelancer ratings, bid amounts, delivery times, and cover letters. Accept button hires the freelancer; reject dismisses the proposal.



GET `/api/project/:id` — Owner

Returns full project details including all proposals (populated with freelancer profiles, ratings, names), current submission (if any), and assigned freelancer.

RESPONSE (200 OK)

```
{ "project": { "_id": "64a3b...", "title": "React E-Commerce...", "proposals": [ { "_id": "64b2c...", "price": 1200, "duration": 14, "description": "Cover letter text...", "user": { "name": "Arjun Sharma", "rating": 4.9 }, "status": 1 } ], "freelancer": null, "status": "OPEN" } }
```

2.5 — Accept or Reject a Proposal

When owner accepts a proposal, the freelancer is assigned to the project, the project status changes to CLOSED (no more bids), and all other pending proposals are auto-rejected. Both parties receive real-time notifications.

PATCH `/api/proposal/:id` — Owner

Updates proposal status. status=2 accepts the proposal (hires freelancer, closes project, rejects other proposals). status=0 rejects. Sends Socket.io notification to freelancer.

REQUEST BODY

`status`	Number REQUIRED — 2 = Accept (hire), 0 = Reject

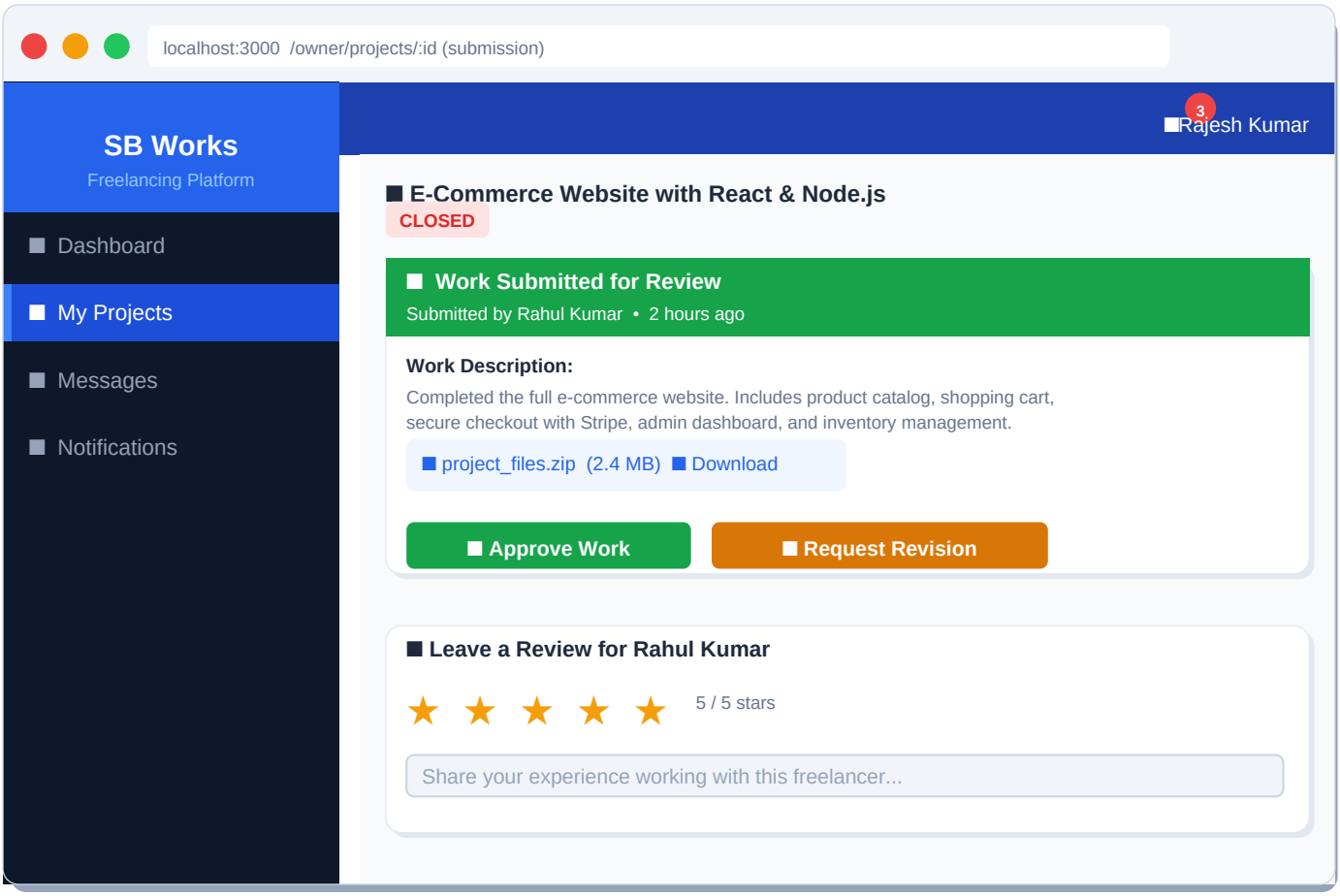
RESPONSE (200 OK)

```
// On Accept (status=2): { "proposal": { "status": 2 }, "project": { "status": "CLOSED", "freelancer": "64c3d..." }, "message": "Proposal accepted. Freelancer hired!", "success": true } // On Reject (status=0): { "proposal": { "status": 0 }, "message": "Proposal rejected", "success": true }
```

■ Socket.io events fired: receiveNotification → freelancer gets "Proposal Accepted!" or "Proposal Rejected" bell notification in real-time

2.6 — Review Submitted Work

Once the freelancer submits work, the owner sees a "Submitted Work" section on the project detail page. They can download the file, read the description, then either approve or request revision.



PATCH /api/submission/review/:id — Owner

Owner reviews submitted work. status="approved" closes the project and notifies freelancer. status="revision_requested" sends feedback and allows freelancer to resubmit.

REQUEST BODY

status	String REQUIRED — "approved" or "revision_requested"
clientFeedback	String REQUIRED if revision — feedback text for the freelancer

RESPONSE (200 OK)

```
// Approved: { "submission": { "status": "approved" }, "message": "Work approved!", "success": true } // Revision requested: { "submission": { "status": "revision_requested", "clientFeedback": "Please add mobile responsiveness" }, "success": true }
```

2.7 — Leave a Review for the Freelancer

After approving work, the owner can leave a star rating (1–5) and written review. This updates the freelancer's average rating and total review count on their profile.

POST `/api/review/add` ■ Owner

Creates a review for the freelancer. Updates the freelancer's profile with new average rating and increments totalReviews. Sends real-time notification to the freelancer.

REQUEST BODY

`project`	ObjectId REQUIRED — project ID the review is for
`reviewee`	ObjectId REQUIRED — freelancer's user ID
`rating`	Number REQUIRED — integer 1 to 5
`comment`	String REQUIRED — written review, min 10 chars

RESPONSE (200 OK)


```
{ "review": { "_id": "64d4e...", "rating": 5, "comment": "Exceptional work! Delivered ahead of schedule...", "reviewer": { "name": "Rajesh Kumar" }, "reviewee": { "name": "Rahul Kumar" } }, "success": true }
```


GET `/api/review/user/:userId` ■ Token

Returns all reviews for a specific user (freelancer). Includes reviewer info, rating, comment, and the project context for each review.

RESPONSE (200 OK)

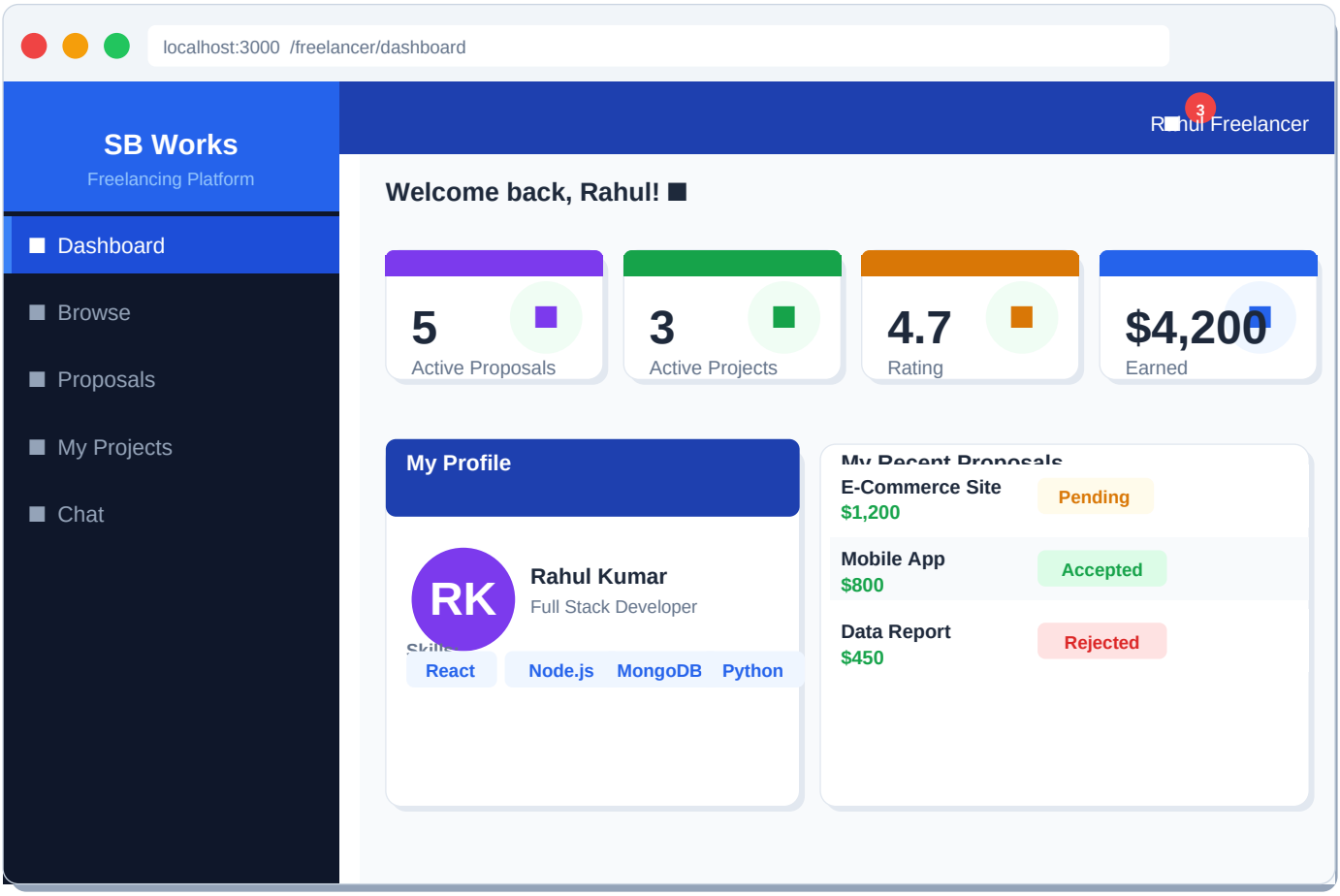

```
{ "reviews": [ { "rating": 5, "comment": "Excellent developer...", "reviewer": { "name": "Rajesh Kumar" }, "project": { "title": "E-Commerce..." } } ], "averageRating": 4.8, "totalReviews": 12, "success": true }
```

PART 3 — Freelancer Dashboard

Browse projects, submit proposals, track status, submit work

3.1 — Freelancer Dashboard Overview

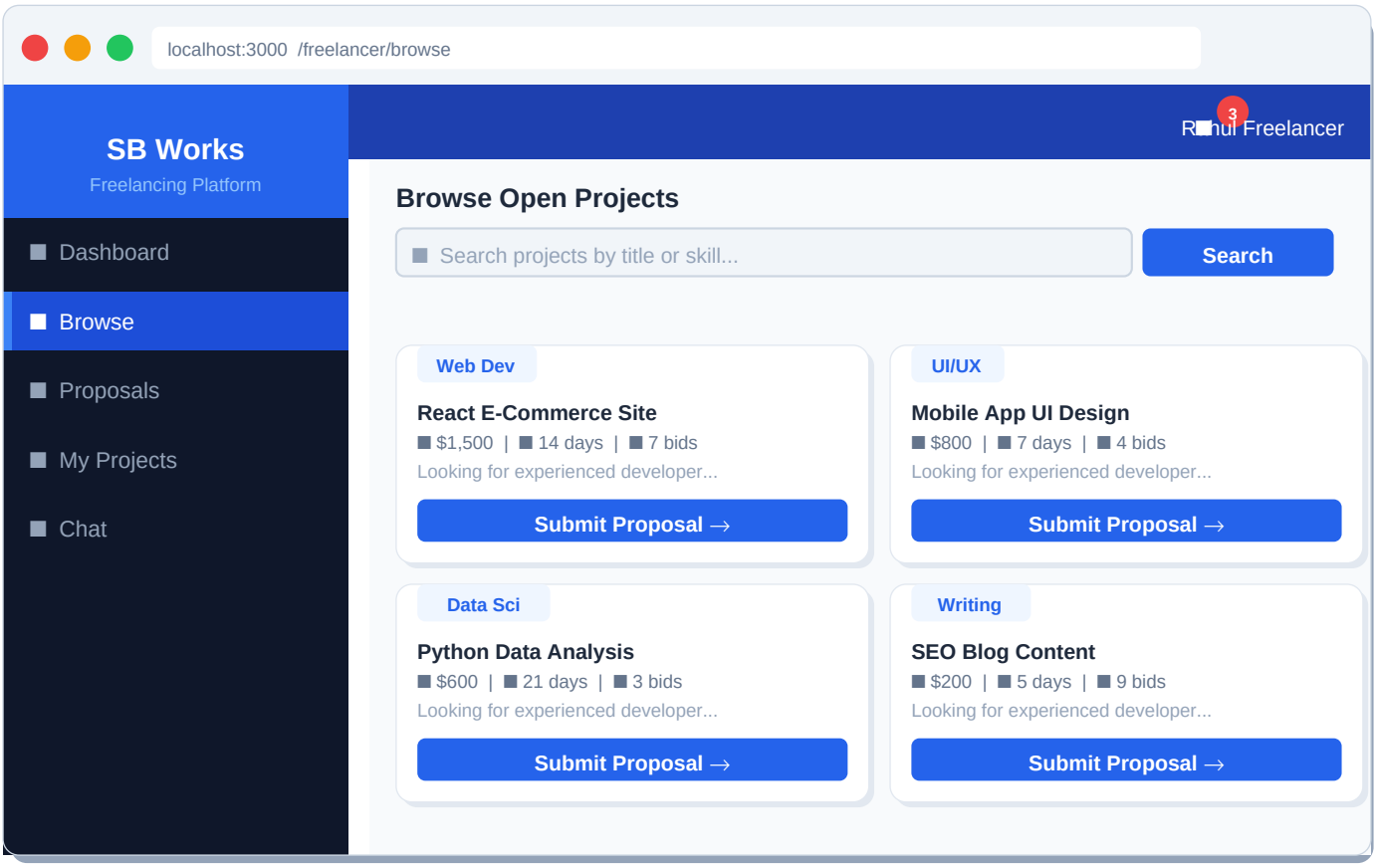
The freelancer's home shows their active proposals, current projects, rating, and total earnings. A profile card shows their skills and rating. Recent proposals are listed inline.



■ Layout: src/layouts/FreelancerLayout.jsx | Sidebar: Dashboard, Browse, My Proposals, My Projects, Chat | Stats computed from live DB queries

3.2 — Browse Open Projects

Freelancers see all OPEN projects in a card grid. Each card shows category, title, budget, deadline, and proposal count. A search bar allows filtering by keyword.



GET `/api/project/list` Token

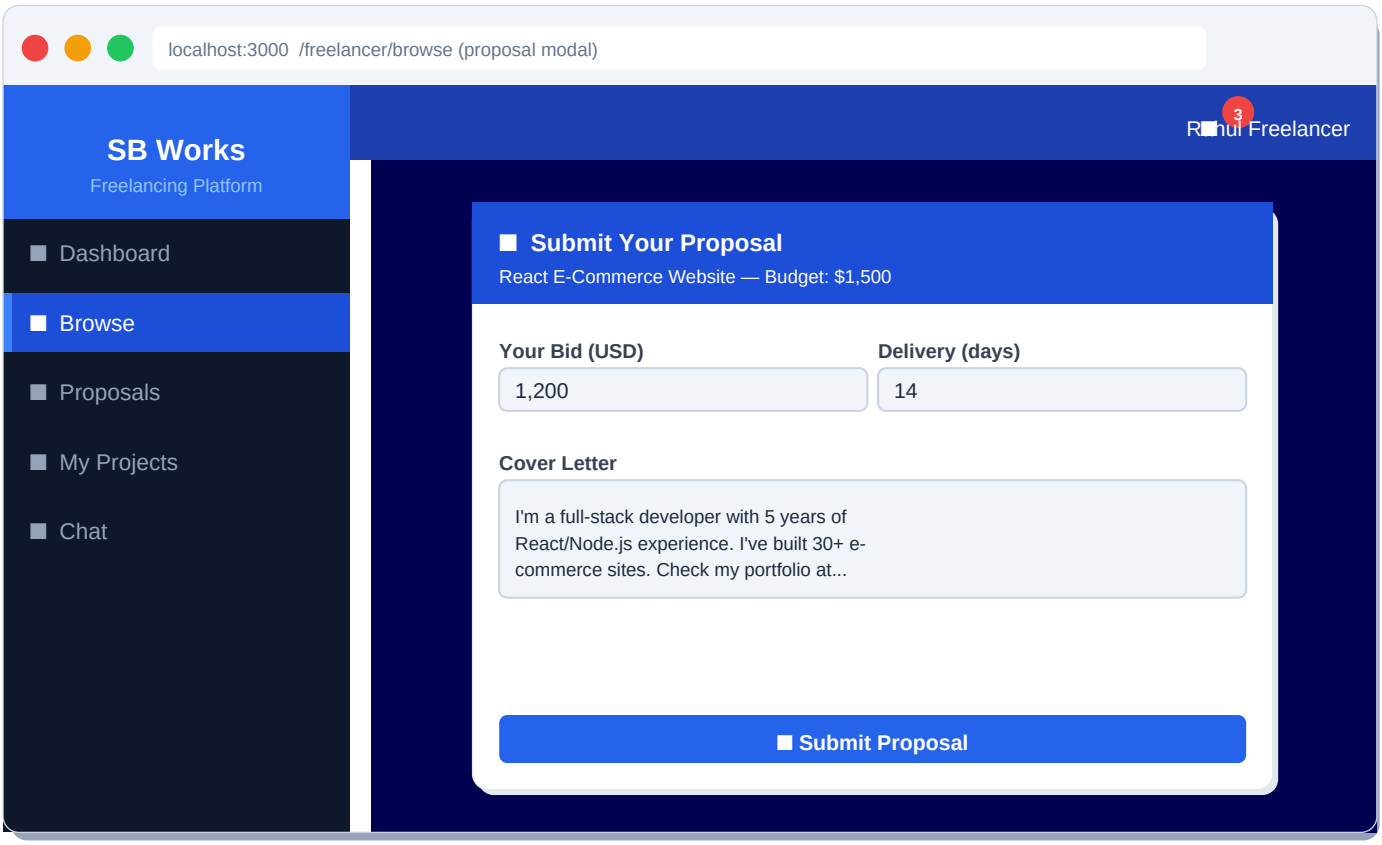
Returns all projects with status=OPEN. Populated with category and owner info. Supports query param ?search=keyword to filter by title. Does not return projects where the current user has already submitted a proposal.

RESPONSE (200 OK)

```
{ "projects": [ { "_id": "64a3b...", "title": "React E-Commerce Website", "status": "OPEN",
"budget": 1500, "deadline": "2024-03-15", "category": { "title": "Web Development" }, "owner":
{ "name": "Rajesh Kumar" }, "proposals": [ ... ], "_proposalCount": 7 } ], "success": true }
```

3.3 — Submit a Proposal

Clicking "Submit Proposal" on a project card opens a modal. Freelancer enters bid amount, delivery time, and a cover letter. Submission notifies the project owner in real-time.



POST `/api/proposal/add` — Freelancer

Creates a new proposal. Validates that user has not already submitted a proposal for this project (one per user per project). Notifies project owner via Socket.io.

REQUEST BODY

`projectId`	ObjectId REQUIRED — ID of the project being bid on
`price`	Number REQUIRED — bid amount in USD
`duration`	Number REQUIRED — delivery time in days
`description`	String REQUIRED — cover letter text, min 20 chars

RESPONSE (200 OK)

```
{ "proposal": { "_id": "64b2c...", "price": 1200, "duration": 14, "status": 1, "user": "64c3d...", "project": "64a3b..." }, "message": "Proposal submitted successfully", "success": true }
```

■■ Duplicate prevention: if you submit a proposal for a project you already bid on, the API returns a 400 error: "You have already submitted a proposal for this project"

3.4 — My Proposals — Track Status

Freelancers can track all their submitted proposals in a table showing project title, bid amount, delivery time, and current status (Pending / Accepted / Rejected).

GET `/api/proposal/list` — Freelancer

Returns all proposals submitted by the authenticated freelancer. Populated with project title, owner name, and category. Status: 0=Rejected, 1=Pending, 2=Accepted.

RESPONSE (200 OK)


```
{ "proposals": [ { "_id": "64b2c...", "price": 1200, "duration": 14, "status": 1, "project": { "title": "React E-Commerce Website", "budget": 1500, "owner": { "name": "Rajesh Kumar" } } }, { "_id": "64b3d...", "price": 800, "duration": 7, "status": 2, "project": { "title": "Mobile App UI Design" } } ], "success": true }
```

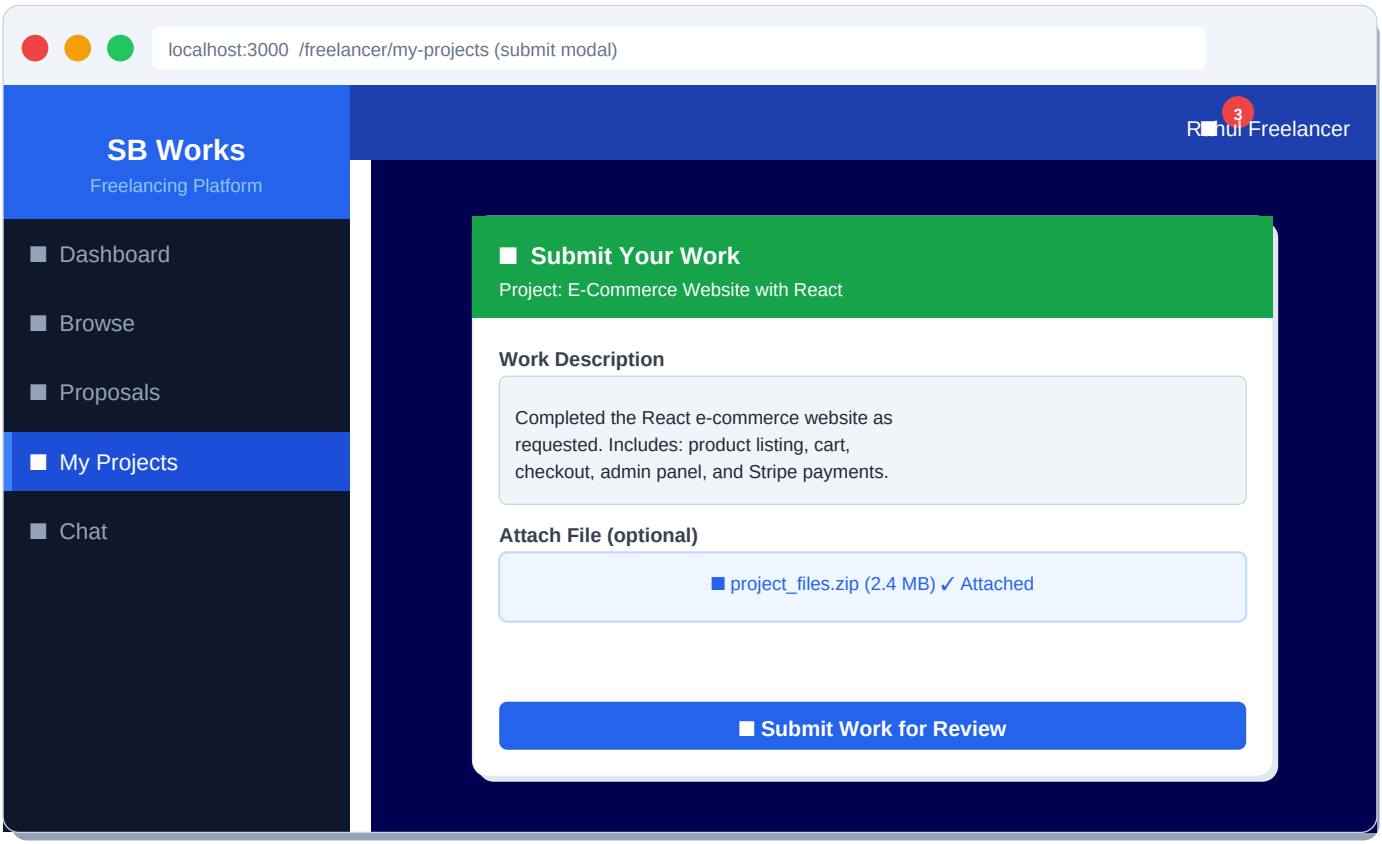

3.5 — My Active Projects

Shows projects where the freelancer's proposal was accepted. Each project card has a "Submit Work" button to deliver completed work to the client.

■ Logic: Query all proposals with status=2 (accepted) for current user, then populate project details. Route: GET /api/proposal/list → filter by status=2 on frontend, or use the accepted-proposals endpoint.

3.6 — Submit Completed Work

The submit work modal lets the freelancer write a description and optionally attach a file (zip, pdf, images). The file is uploaded via multipart/form-data to the /uploads/ folder.



POST /api/submission/submit ■ Freelancer

Uploads work submission. Content-Type must be multipart/form-data. File is saved to /uploads/ folder with a unique name. Notifies client via Socket.io.

REQUEST BODY

`projectId`	String REQUIRED — project ID (form field)
`description`	String REQUIRED — work description (form field)
`file`	File OPTIONAL — any file type, saved to /uploads/

RESPONSE (200 OK)

```
{ "submission": { "_id": "64e5f...", "project": "64a3b...", "freelancer": "64c3d...",
"description": "Completed the website...", "fileUrl":
"/uploads/1705123456789-project_files.zip", "fileName": "project_files.zip", "status":
"submitted" }, "success": true }
```

GET `/api/submission/project/:projectId` ■ Token

Returns the submission for a specific project. Used by both client (to review work) and freelancer (to check submission status and client feedback).

RESPONSE (200 OK)

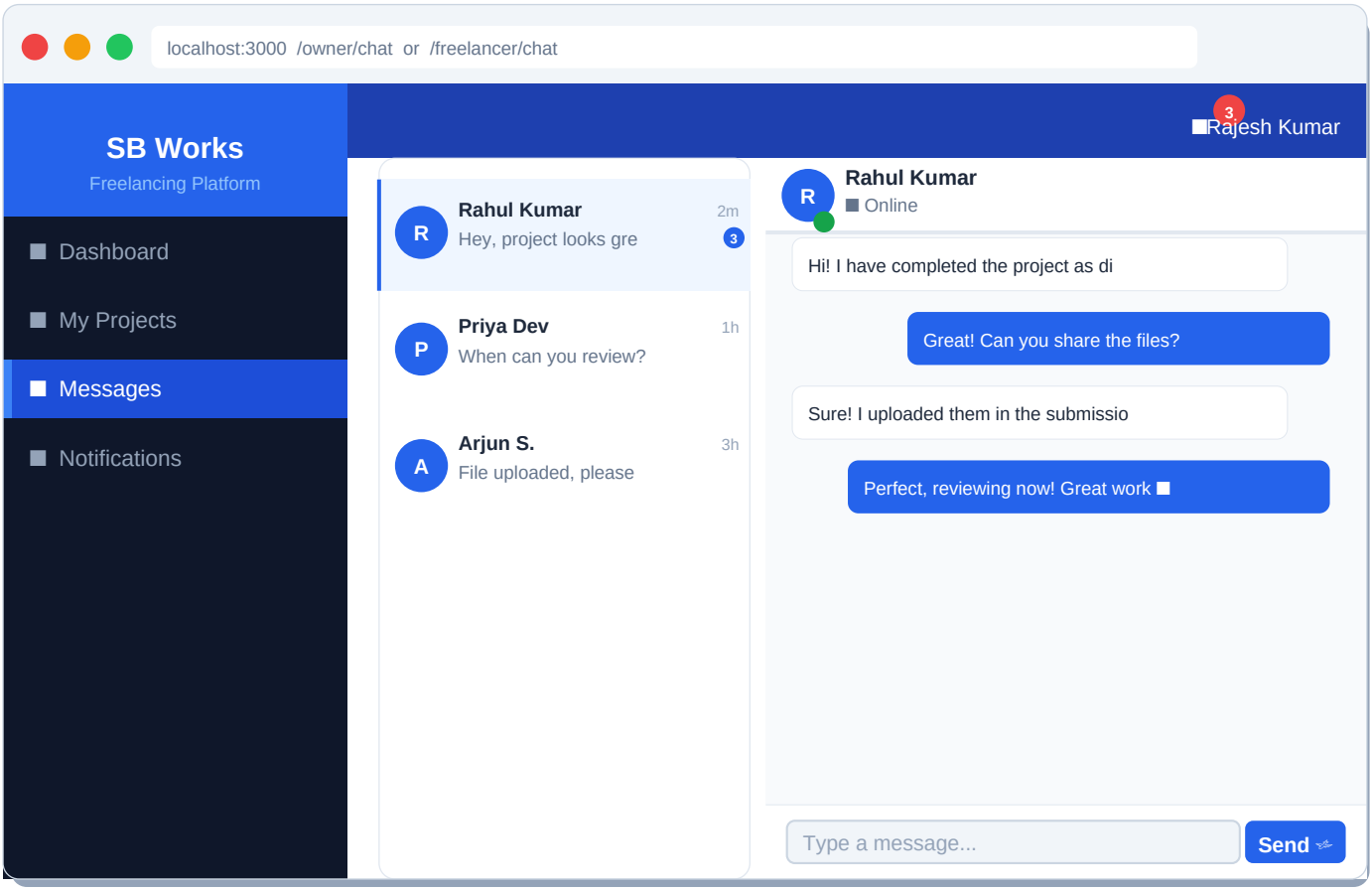
```
{ "submission": { "status": "revision_requested", "clientFeedback": "Please add mobile
responsiveness and dark mode", "fileUrl": "/uploads/...", "fileName": "project_files.zip",
"freelancer": { "name": "Rahul Kumar" } }, "success": true }
```

PART 4 — Real-time Chat & Notifications

Socket.io powered messaging and notification system

4.1 — Chat Interface

The chat system is fully real-time via Socket.io. The left panel shows all conversations with unread counts. The right panel shows the message thread with the selected user. Messages are persisted in MongoDB so history is preserved across sessions.



POST `/api/message/send` Token

Saves message to MongoDB. Emits "receiveMessage" event to receiver via Socket.io if they are online. Message appears instantly in the chat window.

REQUEST BODY

`receiverId`	ObjectId REQUIRED — user ID of recipient
`content`	String REQUIRED — message text
`projectId`	ObjectId OPTIONAL — link message to a project context

RESPONSE (200 OK)

```
{ "message": { "_id": "64f6g...", "sender": "64a3b...", "receiver": "64c3d...", "content": "Hi! I have uploaded the files, please review.", "isRead": false, "createdAt": "2024-01-15T10:30:00Z" }, "success": true }
```

GET `/api/message/conversation/:userId` ■ Token

Returns all messages between the authenticated user and the specified userId. Sorted oldest to newest for display in chat window.

RESPONSE (200 OK)

```
{ "messages": [ { "sender": "64a3b...", "receiver": "64c3d...", "content": "Hi! Interested in your project", "isRead": true }, { "sender": "64c3d...", "receiver": "64a3b...", "content": "Great! Please submit a proposal", "isRead": true } ], "success": true }
```

GET `/api/message/conversations` ■ Token

Returns all unique conversation partners with their latest message and unread count. Used to populate the left sidebar conversation list.

RESPONSE (200 OK)

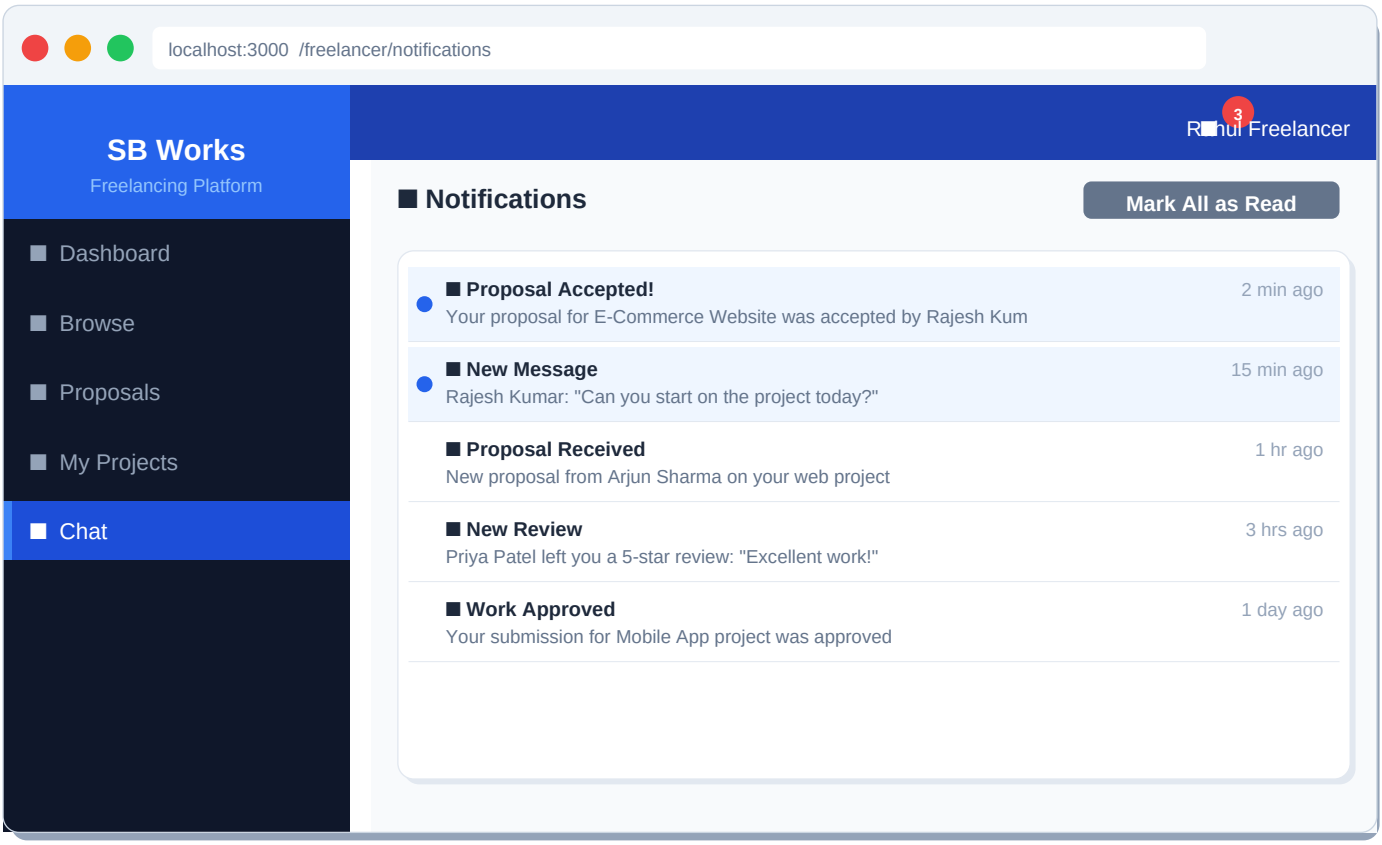
```
{ "conversations": [ { "user": { "name": "Rahul Kumar", "_id": "64c3d..." }, "lastMessage": "Files are uploaded!", "unreadCount": 3 }, { "user": { "name": "Priya Patel" }, "lastMessage": "When can you review?", "unreadCount": 0 } ], "success": true }
```

Socket.io Events Reference

Event	Direction	Payload	When Triggered
join	Client → Server	{ userId: "64a3b..." }	After login — joins personal notification room
sendMessage	Client → Server	{ receiverId, content, projectId? }	User sends a chat message
receiveMessage	Server → Client	Full message object	New message arrives for the user
sendNotification	Client → Server	{ receiverId, notification }	Manual notification push
receiveNotification	Server → Client	{ title, message, type, link }	Any platform notification
disconnect	Automatic	—	User closes browser / loses connection

4.2 — Notification Center

Bell icon in topbar shows unread count badge. Clicking opens notification center. Unread items are highlighted in blue. Users can mark individual or all notifications as read.



GET `/api/notification` — Token

Returns all notifications for the authenticated user, sorted newest first. Unread notifications have isRead=false.

RESPONSE (200 OK)

```
{ "notifications": [ { "_id": "64g7h...", "title": "Proposal Accepted!", "message": "Your proposal was accepted by Rajesh Kumar", "type": "bid", "isRead": false, "link": "/freelancer/my-projects", "createdAt": "2024-01-15T10:25:00Z" } ], "unreadCount": 3, "success": true }
```

PATCH `/api/notification/read-all` — Token

Marks all notifications for the current user as read (isRead=true). Bell badge count drops to 0.

RESPONSE (200 OK)

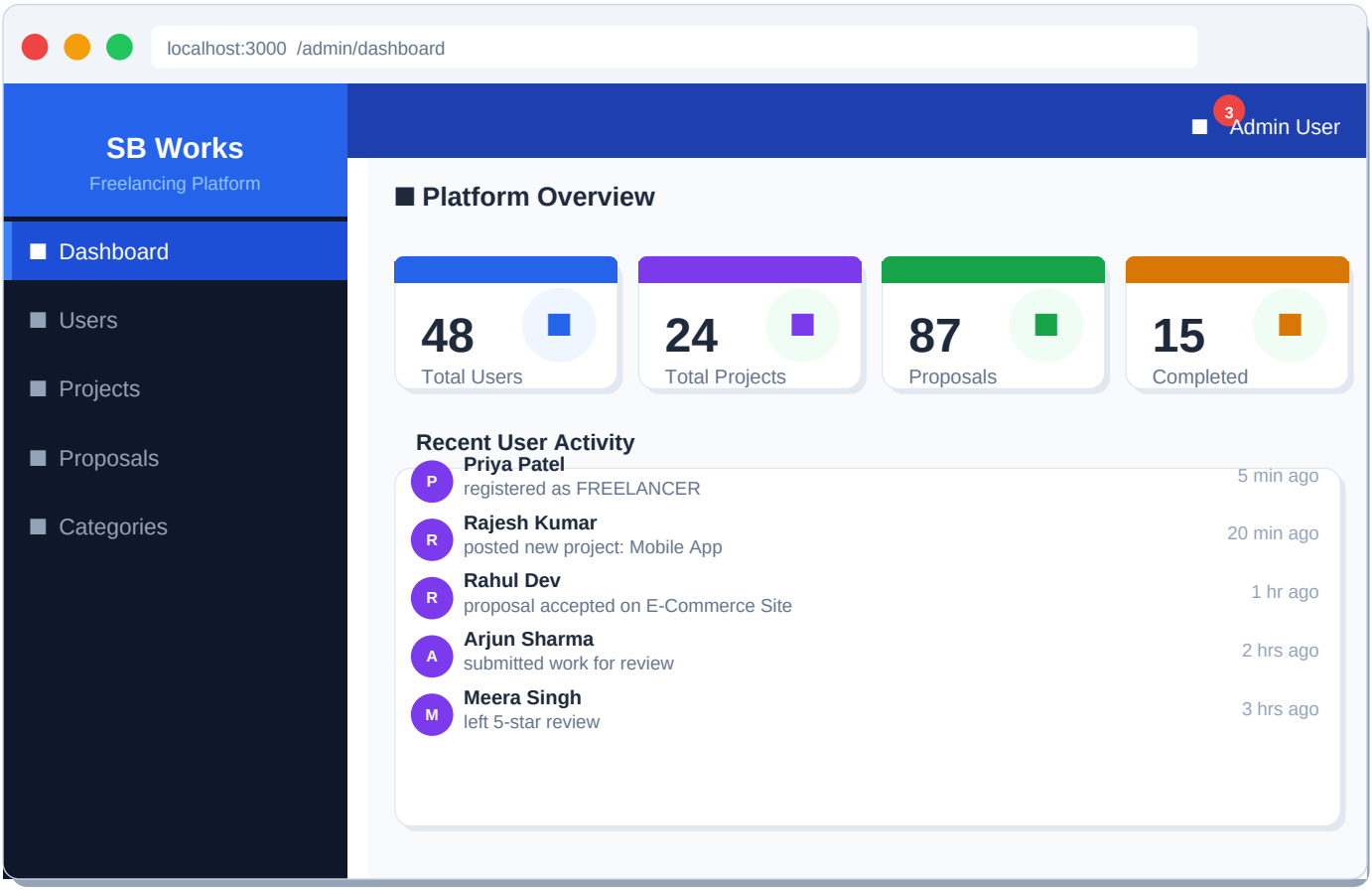
```
{ "message": "All notifications marked as read", "success": true }
```

PART 5 — Admin Panel

User management, platform oversight, category management

5.1 — Admin Dashboard

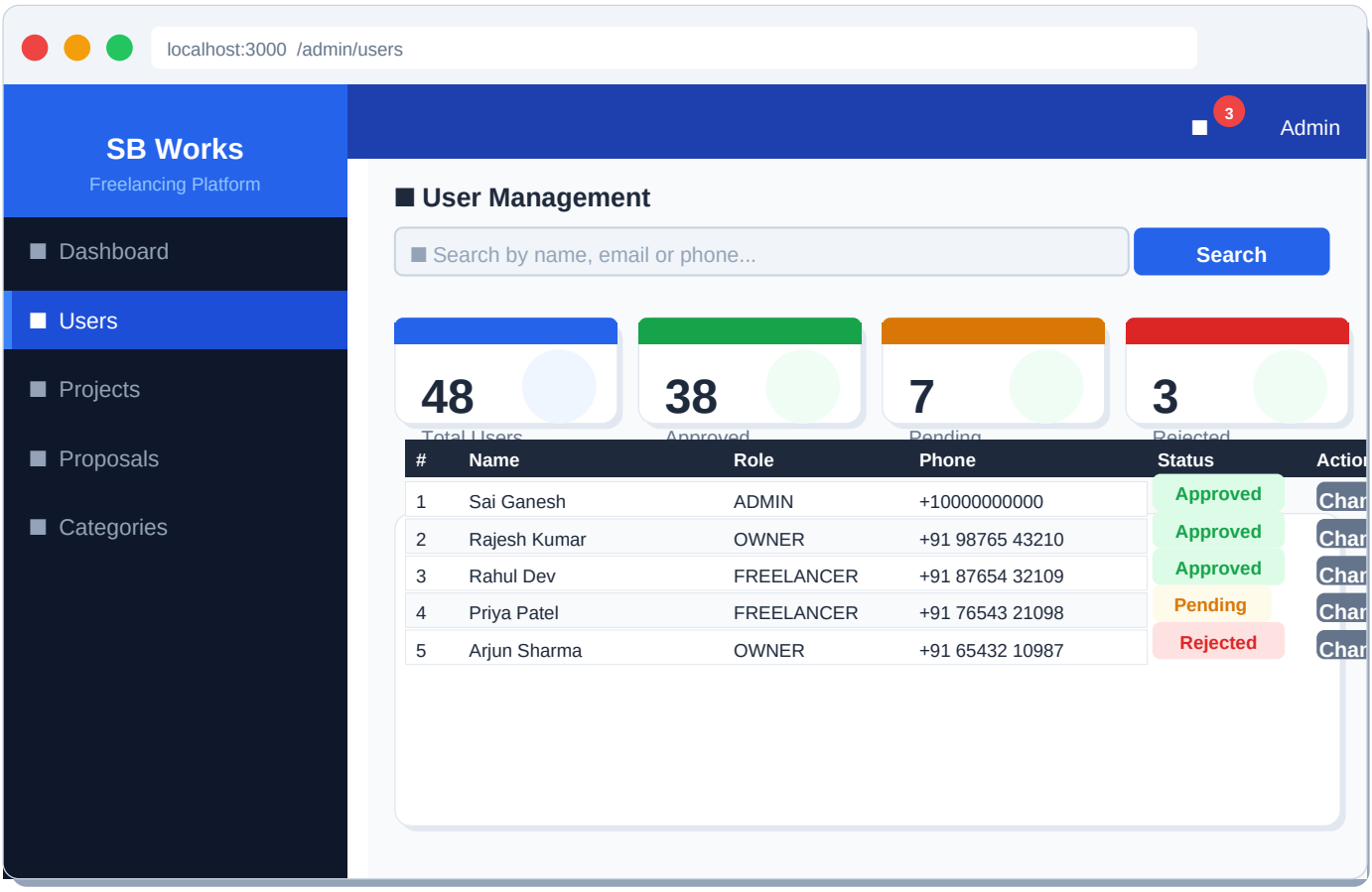
The admin has a full overview of platform activity — total users, projects, proposals, and completions. Recent activity feed shows all user actions in real-time.



■ Admin login credentials (seeded by npm run seed): Phone: +10000000000 | OTP: 123456 | These are permanent test credentials set in seed.js

5.2 — User Management

Admin sees all registered users in a searchable table. Each user has a status badge (Approved / Pending / Rejected) and a "Change Status" button to approve, pend, or reject users.



GET `/api/admin/user/list` ■ Admin

Returns all registered users with search support. Query param ?search= filters by name, email, or phone number.

RESPONSE (200 OK)

```
{ "users": [ { "_id": "64a3b...", "name": "Rahul Kumar", "email": "rahul@example.com",
"phoneNumber": "+919876543210", "role": "FREELANCER", "status": 2, "rating": 4.8,
"totalReviews": 12, "isActive": true } ], "total": 48, "success": true }
```

PATCH `/api/admin/user/verify/:userId` ■ Admin

Changes a user's account status. This is the primary way admin approves new signups. Status 2=Approved, 1=Pending, 0=Rejected. Rejected users cannot access the platform.

REQUEST BODY

`status`	Number REQUIRED — 0=Rejected, 1=Pending, 2=Approved

RESPONSE (200 OK)

```
{ "user": { "_id": "64c3d...", "name": "Priya Patel", "status": 2 }, "message": "User status
updated successfully", "success": true }
```

GET `/api/admin/user/profile/:userId` ■ Admin

Returns detailed profile of any user including their projects, proposals, submissions, and review history. Used for admin investigation of user activity.

RESPONSE (200 OK)

```
{ "user": { "name": "Rahul Kumar", "role": "FREELANCER", "status": 2, "proposals": [...], "submissions": [...], "reviews": [...] }, "success": true }
```

5.3 — Category Management

Admin manages the project categories that appear in the project creation form. The seed script creates 10 default categories. Admin can add, edit, or delete categories.

POST `/api/admin/category/add` ■ Admin

Creates a new project category. The title must be unique. Categories are immediately available in the project creation form.

REQUEST BODY

`title`	String REQUIRED — unique category name, e.g. "Blockchain Development"
`description`	String OPTIONAL — short description
`type`	String OPTIONAL — defaults to "project"

RESPONSE (200 OK)

```
{ "category": { "_id": "64h8i...", "title": "Blockchain Development", "description": "Solidity, Web3, DeFi projects", "type": "project" }, "success": true }
```

PATCH `/api/admin/category/update/:id` ■ Admin

Updates a category's title or description. All existing projects in this category automatically reflect the updated name.

REQUEST BODY

`title`	String OPTIONAL — updated category name
`description`	String OPTIONAL — updated description

RESPONSE (200 OK)

```
{ "category": { "title": "Blockchain & Web3 Dev", ... }, "success": true }
```

DELETE `/api/admin/category/remove/:id` ■ Admin

Deletes a category. Only possible if no projects are currently using this category. Returns error if category is in use.

RESPONSE (200 OK)

```
{ "message": "Category deleted successfully", "success": true }
```

PART 6 — Complete API Reference

All 35+ endpoints — method, path, authentication, and purpose

Base URL: http://localhost:5000/api | Auth: Signed HttpOnly JWT Cookie

Group	Method	Endpoint	Auth	Description
AUTH	POST	/user/get-otp	Public	Send OTP to phone number
AUTH	POST	/user/check-otp	Public	Verify OTP → set JWT cookies
AUTH	POST	/user/complete-profile	Token	Set name, email, role for new user
AUTH	GET	/user/profile	Token	Get current authenticated user
AUTH	GET	/user/refresh-token	Cookie	Refresh expired access token
AUTH	POST	/user/logout	None	Clear auth cookies
PROJECT	GET	/project/list	Token	All OPEN projects (freelancer browse)
PROJECT	GET	/project/owner-projects	Owner	My posted projects with proposals
PROJECT	POST	/project/add	Owner	Create a new project
PROJECT	GET	/project/:id	Token	Single project detail + proposals + submission
PROJECT	PATCH	/project/update/:id	Owner	Edit project fields
PROJECT	PATCH	/project/:id	Owner	Toggle OPEN ↔ CLOSED status
PROJECT	DELETE	/project/:id	Owner	Delete project (if no freelancer)
PROPOSAL	GET	/proposal/list	Freelancer	My submitted proposals
PROPOSAL	POST	/proposal/add	Freelancer	Submit proposal to a project
PROPOSAL	GET	/proposal/:id	Token	Get single proposal details
PROPOSAL	PATCH	/proposal/:id	Owner	Accept (2) or Reject (0) proposal
MESSAGE	POST	/message/send	Token	Send a chat message
MESSAGE	GET	/message/conversation/:userId	Token	Get chat history with a user

MESSAGE	GET	/message/conversations	Token	All conversations with last message
MESSAGE	PATCH	/message/read/:senderId	Token	Mark messages from sender as read
SUBMISSION	POST	/submission/submit	Freelancer	Upload work files + description
SUBMISSION	GET	/submission/project/:projectId	Token	Get submission for a project
SUBMISSION	GET	/submission/my	Freelancer	All my submissions + statuses
SUBMISSION	PATCH	/submission/review/:id	Owner	Approve or request revision
REVIEW	POST	/review/add	Owner	Submit star rating + comment
REVIEW	GET	/review/user/:userId	Token	All reviews for a user
REVIEW	GET	/review/project/:projectId	Token	All reviews for a project
NOTIF	GET	/notification	Token	My notifications list
NOTIF	PATCH	/notification/read/:id	Token	Mark one notification as read
NOTIF	PATCH	/notification/read-all	Token	Mark all notifications as read
NOTIF	DELETE	/notification/:id	Token	Delete a notification
CATEGORY	GET	/category/list	Token	All project categories
ADMIN	GET	/admin/user/list	Admin	All users with search
ADMIN	PATCH	/admin/user/verify/:userId	Admin	Change user status 0/1/2
ADMIN	GET	/admin/user/profile/:userId	Admin	User profile + all activity
ADMIN	POST	/admin/category/add	Admin	Add project category
ADMIN	PATCH	/admin/category/update/:id	Admin	Edit category
ADMIN	DELETE	/admin/category/remove/:id	Admin	Delete category

Sample Data Reference

Test accounts, sample projects, and expected responses

Seed Data Created by: npm run seed

■ ■ Admin Account — Phone: +10000000000 | OTP: 123456 (permanent) | Status: Approved | Role: ADMIN | isActive: true

10 Project Categories Created by Seed:

1. Web Development	2. Mobile Development	3. UI/UX Design	4. Content Writing	5. Digital Marketing
6. Video & Animation	7. Data Science	8. DevOps & Cloud	9. Other	10. Graphic Design

Sample API Request/Response — Create Project

Sample: Create Project

```
// POST /api/project/add
// Headers: Cookie: accessToken=<jwt_token>
// Content-Type: application/json
REQUEST:
{
"title": "Build a Full Stack E-Commerce Website with React & Node.js",
"description": "Need an experienced developer to build a complete e-commerce platform with product catalog, shopping cart, Stripe payments, admin dashboard, and inventory management.",
"category": "64a1b2c3d4e5f6g7h8i9j0k1",
"budget": 1500,
"deadline": "2024-03-15T00:00:00.000Z",
"tags": ["react", "nodejs", "mongodb", "stripe", "redux"]
}
RESPONSE (201 Created):
{
"project": {
"_id": "64c3d4e5f6g7h8i9j0k1l2m3",
"title": "Build a Full Stack E-Commerce Website with React & Node.js",
"status": "OPEN",
"budget": 1500,
"deadline": "2024-03-15T00:00:00.000Z",
"owner": { "_id": "64a3b4c5d6...", "name": "Rajesh Kumar" },
"category": { "_id": "64a1b2c3...", "title": "Web Development" },
"tags": ["react", "nodejs", "mongodb", "stripe", "redux"],
"proposals": [],
"freelancer": null,
"createdAt": "2024-01-15T10:00:00.000Z"
},
"success": true
}
```

Sample API Request/Response — Submit Proposal

Sample: Submit Proposal

```
// POST /api/proposal/add
// Headers: Cookie: accessToken=<jwt_token>
REQUEST:
{
"projectId": "64c3d4e5f6g7h8i9j0k1l2m3",
"price": 1200,
"duration": 14,
"description": "I am a full-stack developer with 5 years of React and Node.js experience. I have built 30+ e-commerce platforms and can deliver this project within 2 weeks. Please check my portfolio at github.com/rahuldev."
}
RESPONSE (201 Created):
{
"proposal": {
"_id": "64d4e5f6g7h8i9j0k1l2m3n4",
"price": 1200,
"duration": 14,
"status": 1,
"user": { "_id": "64c3d...", "name": "Rahul Kumar", "rating": 4.8 },
"project": { "_id": "64c3d4...", "title": "Build a Full Stack E-Commerce..." }
},
"message": "Proposal submitted successfully",
"success": true
}
// Socket.io event fired simultaneously:
// Server emits to project owner's room:
{ "title": "New Proposal!", "message": "Rahul Kumar submitted a proposal for your project",
"type": "bid", "link": "/owner/projects/64c3d4e5f6g7h8i9j0k1l2m3" }
```

Error Response Examples

Common Error Responses

```
// 401 Unauthorized — No token or expired token:
{ "message": "Authentication required. Please login.", "success": false }
// 403 Forbidden — Wrong role:
{ "message": "Access denied. This action requires OWNER role.", "success": false }
// 403 Forbidden — Account not approved:
{ "message": "Your account is pending admin approval.", "success": false }
// 400 Bad Request — Duplicate proposal:
{ "message": "You have already submitted a proposal for this project.", "success": false }
// 400 Bad Request — Validation error:
{ "message": "title must be at least 10 characters", "success": false }
// 404 Not Found:
{ "message": "Project not found", "success": false }
// 410 OTP Expired:
{ "message": "OTP has expired. Please request a new one.", "success": false }
```