Upwork Platform - Backend Assignment

Difficulty: (7/10) - no AI, feel free to use Google

Time Limit: 2hr

Evaluation: Automated test cases will be run against your backend
LINK : https://github.com/rahul-MyGit/upwork_contest_test

Tech Stack (REQUIRED)

- Node.js, Express, PostgreSQL, JWT (Authentication), bcrypt (Password hashing), Zod

Objective

Build a **Freelance Marketplace Platform** backend where:

- **Freelancers** can create service listings and manage proposals
- **Clients** can post projects, review proposals, and hire freelancers
- **Both roles** can manage contracts, milestones, and payments
- All APIs follow **strict contracts** so automated tests can validate them

User Roles

Role	Description
`client`	Posts projects and hires freelancers
`freelancer`	Creates service listings and submits proposals

Core Rules (Hard Rules)

1. One freelancer can have multiple service listings with different categories

2. Freelancers **cannot** hire other freelancers or post projects
3. Clients **cannot** create service listings or submit proposals
4. A project can receive multiple proposals but only **one** can be accepted
5. Once a proposal is accepted, a contract is automatically created
6. Contracts have milestones that must be completed sequentially
7. Payments are released only after milestone approval
8. Reviews can only be submitted after contract completion
9. JWT required for **all APIs except signup/login**
10. Responses must match the format **exactly**

CRITICAL: JavaScript `undefined` will cause tests to fail. Always use `null` in response

---

Database Schema (PostgreSQL)

Below is a suggested schema. Feel free to modify as needed, but ensure your responses match the required format.

users

```
CREATE TABLE users (
    id VARCHAR(255) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL CHECK (role IN ('client', 'free
lancer')),
    bio TEXT,
    skills JSONB DEFAULT '[]',
    hourly_rate DECIMAL(10,2),
    profile_image VARCHAR(500),
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

services

```
CREATE TABLE services (
    id VARCHAR(255) PRIMARY KEY,
    freelancer_id VARCHAR(255) REFERENCES users(id),
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    category VARCHAR(100) NOT NULL,
    pricing_type VARCHAR(50) NOT NULL CHECK (pricing_type IN
('fixed', 'hourly')),
    price DECIMAL(10,2) NOT NULL,
    delivery_days INTEGER NOT NULL,
    rating DECIMAL(2,1) DEFAULT 0.0,
    total_reviews INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

projects

```
CREATE TABLE projects (
    id VARCHAR(255) PRIMARY KEY,
    client_id VARCHAR(255) REFERENCES users(id),
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    category VARCHAR(100) NOT NULL,
    budget_min DECIMAL(10,2) NOT NULL,
    budget_max DECIMAL(10,2) NOT NULL,
    deadline DATE NOT NULL,
    status VARCHAR(50) DEFAULT 'open' CHECK (status IN ('ope
n', 'in_progress', 'completed', 'cancelled')),
    required_skills JSONB DEFAULT '[]',
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

proposals

```
CREATE TABLE proposals (
    id VARCHAR(255) PRIMARY KEY,
    project_id VARCHAR(255) REFERENCES projects(id) ON DELETE
CASCADE,
    freelancer_id VARCHAR(255) REFERENCES users(id),
    cover_letter TEXT NOT NULL,
    proposed_price DECIMAL(10,2) NOT NULL,
    estimated_duration INTEGER NOT NULL,
    status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('p
ending', 'accepted', 'rejected')),
    submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(project_id, freelancer_id)
);
```

contracts

```
CREATE TABLE contracts (
    id VARCHAR(255) PRIMARY KEY,
    project_id VARCHAR(255) REFERENCES projects(id),
    freelancer_id VARCHAR(255) REFERENCES users(id),
    client_id VARCHAR(255) REFERENCES users(id),
    total_amount DECIMAL(10,2) NOT NULL,
    status VARCHAR(50) DEFAULT 'active' CHECK (status IN ('ac
tive', 'completed', 'cancelled')),
    started_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP,
    UNIQUE(project_id)
);
```

milestones

```
CREATE TABLE milestones (
    id VARCHAR(255) PRIMARY KEY,
    contract_id VARCHAR(255) REFERENCES contracts(id) ON DELE
TE CASCADE,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    amount DECIMAL(10,2) NOT NULL,
    due_date DATE NOT NULL,
    order_index INTEGER NOT NULL,
    status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('p
ending', 'submitted', 'approved', 'rejected')),
    submitted_at TIMESTAMP,
    approved_at TIMESTAMP,
    UNIQUE(contract_id, order_index)
);
```

reviews

```
CREATE TABLE reviews (
    id VARCHAR(255) PRIMARY KEY,
    contract_id VARCHAR(255) REFERENCES contracts(id),
    reviewer_id VARCHAR(255) REFERENCES users(id),
    reviewee_id VARCHAR(255) REFERENCES users(id),
    rating INTEGER NOT NULL CHECK (rating >= 1 AND rating <=
5),
    comment TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(contract_id, reviewer_id)
);


```

```
```

```
# **Global API Rules**

### **Auth Header**
```
Authorization: Bearer <JWT_TOKEN>
```

### Response Format (STRICT)

**Success Response:**

```
{
  "success": true,
  "data": {},
  "error": null
}
```

Error Response:

```
{
  "success": false,
  "data": null,
  "error": "ERROR_CODE"
}
```

- **No extra keys allowed**
- **No nested error objects**
- **Error must be a string code, not an object**
- **ALWAYS RETURN NULL AND NOT UNDEFINED**

API Endpoints (12 Total)

1. POST /api/auth/signup
```

Register a new user (client or freelancer)

Request Body

```
{
  "name": "Rahul Kumar",
  "email": "rahul@example.com",
  "password": "rahul123",
  "role": "freelancer",
  "bio": "Full-stack developer with 5 years of experience",
  "skills": ["JavaScript", "React", "Node.js"],
  "hourlyRate": 2500
}
```

Success Response – `201 Created`

```
{
  "success": true,
  "data": {
    "id": "usr_1a2b3c4d5e",
    "name": "Rahul Kumar",
    "email": "rahul@example.com",
    "role": "freelancer",
    "bio": "Full-stack developer with 5 years of experience",
    "skills": ["JavaScript", "React", "Node.js"], // else []
    "hourlyRate": 2500
  },
  "error": null
}
```

Notes:

- If no role is given, default is `client`
- `bio`, `skills`, and `hourlyRate` are optional:
  - If not provided by user: can return null or omit from response

    - If provided: return the value
    - Empty arrays are acceptable for skills
- **IMPORTANT:** Never return `undefined` - use `null` for missing values

Error Responses

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

400 Bad Request – Email exists

```
{
  "success": false,
  "data": null,
  "error": "EMAIL_ALREADY_EXISTS"
}
```

---

2. POST /api/auth/login

Login and receive JWT token

Request Body

```
{
  "email": "rahul@example.com",
  "password": "rahul123"
}
```

Success Response – `200 OK`

```
{
  "success": true,
  "data": {
    "token": "JWT_TOKEN",
    "user": {
      "id": "usr_1a2b3c4d5e",
      "name": "Rahul Kumar",
      "email": "rahul@example.com",
      "role": "freelancer"
    }
  },
  "error": null
}
```

Error Responses

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

401 Unauthorized – Wrong credentials

```
{
  "success": false,
  "data": null,
  "error": "INVALID_CREDENTIALS"
}
```

---

3. POST /api/services – *(Freelancer Only)*

Create a new service listing

Headers: `Authorization: Bearer <token>`

Request Body

```
{
  "title": "Full-Stack Web Development",
  "description": "I will build a complete web application usi
ng MERN stack",
  "category": "Web Development",
  "pricingType": "fixed",
  "price": 50000,
  "deliveryDays": 14
}
```

Success Response – `201 Created`

```
{
  "success": true,
  "data": {
    "id": "srv_abc123",
    "freelancerId": "usr_1a2b3c4d5e",
    "title": "Full-Stack Web Development",
    "description": "I will build a complete web application u
sing MERN stack",
    "category": "Web Development",
    "pricingType": "fixed",
    "price": 50000,
    "deliveryDays": 14,
    "rating": 0.0,
    "totalReviews": 0
  },
  "error": null
}
```

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not a freelancer

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

---

4. POST /api/projects – *(Client Only)*

Post a new project

Headers: `Authorization: Bearer <token>`

Request Body

```
{
  "title": "E-commerce Website Development",
```

```
  "description": "Need a modern e-commerce platform with paym
ent integration",
  "category": "Web Development",
  "budgetMin": 80000,
  "budgetMax": 150000,
  "deadline": "2026-03-15",
  "requiredSkills": ["React", "Node.js", "MongoDB", "Payment
Gateway"] // or []
}
```

Success Response – `201 Created`

```
{
  "success": true,
  "data": {
    "id": "proj_xyz789",
    "title": "E-commerce Website Development",
    "description": "Need a modern e-commerce platform with pa
yment integration",
    "category": "Web Development",
    "budgetMin": 80000,
    "budgetMax": 150000,
    "deadline": "2026-03-15",
    "status": "open",
    "requiredSkills": ["React", "Node.js", "MongoDB", "Paymen
t Gateway"],
    "createdAt": "2026-01-29T10:00:00Z"
  },
  "error": null
}
```

Notes:

- `clientId` field should be included in response (not shown in example above)
- All timestamp fields should be ISO 8601 format

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not a client

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Invalid schema or deadline in past

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

```
Example Error:
if (budgetMin > budgetMax) {
  return res.status(400).json({
    success: false,
    data: null,
    error: "INVALID_REQUEST"
  });
}
```

5. GET /api/projects

Search and filter projects

Headers: `Authorization: Bearer <token>`

Query Parameters (all optional, if none provided return all open projects):

- `category` – Filter by category (case-insensitive)
- `minBudget` – Minimum budget
- `maxBudget` – Maximum budget
- `status` – Filter by status (open/in_progress/completed/cancelled)
- `skills` – Comma-separated skills to match

Example:

- `/api/projects?category=Web Development&minBudget=50000`
- `/api/projects?status=open&skills=React,Node.js`

Success Response – `200 OK`


```
{
  "success": true,
  "data": [
    {
      "id": "proj_xyz789",
      "clientId": "usr_client123",
      "clientName": "Amit Sharma",
      "title": "E-commerce Website Development",
      "description": "Need a modern e-commerce platform with
payment integration",
      "category": "Web Development",
      "budgetMin": 80000,
      "budgetMax": 150000,
      "deadline": "2026-03-15",
      "status": "open",
      "requiredSkills": ["React", "Node.js", "MongoDB", "Paym
```

```
ent Gateway"],
      "createdAt": "2026-01-29T10:00:00Z",
      "proposalCount": 5
    }
  ],
  "error": null
}
```

Field Calculations:

- `proposalCount` = total number of proposals submitted for this project
- `clientName` = name from users table

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

---

6. POST /api/projects/:projectId/proposals – *(Freelancer Only)*

Submit a proposal for a project

Headers: `Authorization: Bearer <token>`

Request Body

```
{
  "coverLetter": "I have 5 years of experience building e-com
merce platforms...",
  "proposedPrice": 120000,
```

```
  "estimatedDuration": 30
}
```

Notes:

- `estimatedDuration` is in days

Success Response – `201 Created`

```
{
"success": true,
"data": {
"id": "prop_abc123",
"projectId": "proj_xyz789",
"freelancerId": "usr_1a2b3c4d5e",
"coverLetter": "I have 5 years of experience building e-comme
rce platforms...",
"proposedPrice": 120000,
"estimatedDuration": 30,
"status": "pending",
"submittedAt": "2026-01-29T11:00:00Z"
},
"error": null
}
```

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not a freelancer

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Already submitted proposal

```
{
  "success": false,
  "data": null,
  "error": "PROPOSAL_ALREADY_EXISTS"
}
```

400 Bad Request – Project not open

```
{
  "success": false,
  "data": null,
  "error": "PROJECT_NOT_OPEN"
}
```

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
```

```
  "error": "PROJECT_NOT_FOUND"
}
```

---

7. GET /api/projects/:projectId/proposals – *(Client Only - Own Projects)*

Get all proposals for a project

Headers: `Authorization: Bearer <token>`

Success Response – `200 OK`

```
{
  "success": true,
  "data": [
    {
      "id": "prop_abc123",
      "freelancerId": "usr_1a2b3c4d5e",
      "freelancerName": "Rahul Kumar",
      "freelancerSkills": ["JavaScript", "React", "Node.js"],
      "coverLetter": "I have 5 years of experience building e
-commerce platforms...",
      "proposedPrice": 120000,
      "estimatedDuration": 30,
      "status": "pending",
      "submittedAt": "2026-01-29T11:00:00Z"
    }
  ],
  "error": null
}
```

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not the project owner

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
  "error": "PROJECT_NOT_FOUND"
}
```

---

8. PUT /api/proposals/:proposalId/accept – *(Client Only)*

Accept a proposal and create a contract

Headers: `Authorization: Bearer <token>`

Request Body

```
{
  "milestones": [
    {
      "title": "Design & Planning",
```

```
      "description": "Complete UI/UX design and architecture
planning",
      "amount": 40000,
      "dueDate": "2026-02-10"
    },
    {
      "title": "Development Phase 1",
      "description": "Frontend development and API integratio
n",
      "amount": 50000,
      "dueDate": "2026-02-25"
    },
    {
      "title": "Final Delivery",
      "description": "Testing, deployment and documentation",
      "amount": 30000,
      "dueDate": "2026-03-10"
    }
  ]
}
```

Notes:

- Total of milestone amounts must equal the proposed price
- Milestones are auto-ordered by the order they appear in array
- First milestone gets `order_index = 1`, second gets `2`, etc.

Notes for description:

- `description` is optional in milestone objects
- If not provided: can return null or empty string
- Milestone amounts must sum exactly to proposedPrice

Success Response – `200 OK`

```
{
  "success": true,
  "data": {
    "proposal": {
      "id": "prop_abc123",
      "status": "accepted"
    },
    "contract": {
      "id": "contract_xyz123",
      "projectId": "proj_xyz789",
      "freelancerId": "usr_1a2b3c4d5e",
      "clientId": "usr_client123",
      "totalAmount": 120000,
      "status": "active",
      "startedAt": "2026-01-29T12:00:00Z"
    },
    "milestones": [
      {
        "id": "mile_001",
        "contractId": "contract_xyz123",
        "title": "Design & Planning",
        "description": "Complete UI/UX design and architectur
e planning",
        "amount": 40000,
        "dueDate": "2026-02-10",
        "orderIndex": 1,
        "status": "pending"
      },
      {
        "id": "mile_002",
        "contractId": "contract_xyz123",
        "title": "Development Phase 1",
        "description": "Frontend development and API integrat
ion",
        "amount": 50000,
```

```
        "dueDate": "2026-02-25",
        "orderIndex": 2,
        "status": "pending"
      },
      {
        "id": "mile_003",
        "contractId": "contract_xyz123",
        "title": "Final Delivery",
        "description": "Testing, deployment and documentatio
n",
        "amount": 30000,
        "dueDate": "2026-03-10",
        "orderIndex": 3,
        "status": "pending"
      }
    ]
  },
  "error": null
}
```


Side Effects:

- Proposal status changes to 'accepted'
- All other proposals for the same project are automatically rejected
- Project status changes to 'in_progress'
- Contract is created
- Milestones are created

Error Responses

401 Unauthorized


```
{
  "success": false,
  "data": null,
```

```
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not the project owner

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Proposal already accepted/rejected

```
{
  "success": false,
  "data": null,
  "error": "PROPOSAL_ALREADY_PROCESSED"
}
```

400 Bad Request – Milestone amounts don't match proposed price

```
{
  "success": false,
  "data": null,
  "error": "INVALID_MILESTONE_AMOUNTS"
}
```

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
  "error": "PROPOSAL_NOT_FOUND"
}
```

---

9. GET /api/contracts

Get all contracts for current user (both as client and freelancer)

Headers: `Authorization: Bearer <token>`

Query Parameters (optional):

- `status` – Filter by status (active/completed/cancelled)
- `role` – Filter by user's role in contract (client/freelancer)

Success Response – `200 OK`

```
{
  "success": true,
  "data": [
    {
      "id": "contract_xyz123",
      "projectId": "proj_xyz789",
      "projectTitle": "E-commerce Website Development",
      "freelancerId": "usr_1a2b3c4d5e",
      "freelancerName": "Rahul Kumar",
      "clientId": "usr_client123",
      "clientName": "Amit Sharma",
      "totalAmount": 120000,
      "status": "active",
      "startedAt": "2026-01-29T12:00:00Z",
      "completedAt": null,
      "currentMilestone": {
        "id": "mile_001",
```

```
        "title": "Design & Planning",
        "status": "pending",
        "dueDate": "2026-02-10"
      }
    }
  ],
  "error": null
}
```


Field Calculations:

- `currentMilestone` = the first milestone that is not 'approved' (pending/submitted/rejected), or null if all approved

Error Responses

401 Unauthorized


```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```


---

10. PUT /api/milestones/:milestoneId/submit – *(Freelancer Only)*

Submit a milestone for review

Headers: `Authorization: Bearer <token>`

Note: Milestones must be completed sequentially (order_index)

Success Response – `200 OK`


```
{
  "success": true,
  "data": {
```

```
    "id": "mile_001",
    "contractId": "contract_xyz123",
    "title": "Design & Planning",
    "description": "Complete UI/UX design and architecture pl
anning",
    "amount": 40000,
    "dueDate": "2026-02-10",
    "orderIndex": 1,
    "status": "submitted",
    "submittedAt": "2026-02-09T15:00:00Z"
  },
  "error": null
}
```

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not the assigned freelancer

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Previous milestone not approved

```
{
  "success": false,
```

```
  "data": null,
  "error": "PREVIOUS_MILESTONE_INCOMPLETE"
}
```

400 Bad Request – Already submitted

```
{
  "success": false,
  "data": null,
  "error": "MILESTONE_ALREADY_SUBMITTED"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
  "error": "MILESTONE_NOT_FOUND"
}
```

---

11. PUT /api/milestones/:milestoneId/approve – *(Client Only)*

Approve a submitted milestone

Headers: `Authorization: Bearer <token>`

Success Response – `200 OK`

```
{
  "success": true,
  "data": {
    "id": "mile_001",
    "contractId": "contract_xyz123",
    "title": "Design & Planning",
    "status": "approved",
```

```
    "approvedAt": "2026-02-10T09:00:00Z"
  },
  "error": null
}
```

Side Effects:

- If this is the last milestone and it's approved:
  - Contract status changes to 'completed'
  - Contract `completed_at` is set
  - Project status changes to 'completed'

Error Responses

401 Unauthorized

```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```

403 Forbidden – Not the contract client

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Milestone not submitted

```
{
  "success": false,
  "data": null,
```

```
  "error": "MILESTONE_NOT_SUBMITTED"
}
```

400 Bad Request – Already approved

```
{
  "success": false,
  "data": null,
  "error": "MILESTONE_ALREADY_APPROVED"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
  "error": "MILESTONE_NOT_FOUND"
}
```

---

12. POST /api/reviews – *(Both Client and Freelancer)*

Submit a review after contract completion

Headers: `Authorization: Bearer <token>`

Request Body

```
{
  "contractId": "contract_xyz123",
  "rating": 5,
  "comment": "Excellent work! Delivered on time with great qu
ality."
}
```

Success Response – `201 Created`

```
{
  "success": true,
  "data": {
    "id": "review_abc123",
    "contractId": "contract_xyz123",
    "reviewerId": "usr_client123",
    "revieweeId": "usr_1a2b3c4d5e",
    "rating": 5,
    "comment": "Excellent work! Delivered on time with great
quality.",
    "createdAt": "2026-03-11T10:00:00Z"
  },
  "error": null
}
```


Notes:

- Can only review after contract is completed
- Both client and freelancer can review each other
- If reviewing a freelancer who has services, update the service rating:
  - `newRating = ((oldRating * totalReviews) + newRating) / (totalReviews + 1)`
  - `totalReviews = totalReviews + 1`

Error Responses

401 Unauthorized


```
{
  "success": false,
  "data": null,
  "error": "UNAUTHORIZED"
}
```


403 Forbidden – Not part of this contract

```
{
  "success": false,
  "data": null,
  "error": "FORBIDDEN"
}
```

400 Bad Request – Already reviewed

```
{
  "success": false,
  "data": null,
  "error": "ALREADY_REVIEWED"
}
```

400 Bad Request – Contract not completed

```
{
  "success": false,
  "data": null,
  "error": "CONTRACT_NOT_COMPLETED"
}
```

400 Bad Request – Invalid schema

```
{
  "success": false,
  "data": null,
  "error": "INVALID_REQUEST"
}
```

404 Not Found

```
{
  "success": false,
  "data": null,
```

```
  "error": "CONTRACT_NOT_FOUND"
}
```

Implementation Helper Guidelines

ID Generation

Use any method to generate unique string IDs:

- UUID v4
- nanoid
- Custom prefix + random string (e.g., `usr_`, `proj_`, `prop_`, `contract_`, `mile_`, `review_`)

Date and Time Validation Rules

IMPORTANT: All date/time comparisons should use JavaScript's `new Date()` directly. No timezone conversions needed.

1. Project Deadline Must Be in Future

```
const today = new Date();
today.setHours(0, 0, 0, 0); // Reset to start of day

const deadline = new Date(deadlineDate); // e.g., "2026-03-1
5"

if (deadline <= today) {
  // Return error: INVALID_REQUEST
}
```

2. Project Status Rules

```
// Project can only receive proposals if status is 'open'
if (project.status !== 'open') {
  // Return error: PROJECT_NOT_OPEN
}
```

3. Sequential Milestone Submission

```
// Before submitting milestone with order_index N
// All milestones with order_index < N must have status 'appr
oved'

const previousMilestones = await db.query(
  'SELECT * FROM milestones WHERE contract_id = $1 AND order_
index < $2',
  [contractId, currentMilestone.order_index]
);

const allPreviousApproved = previousMilestones.every(m => m.s
tatus === 'approved');

if (!allPreviousApproved) {
  // Return error: PREVIOUS_MILESTONE_INCOMPLETE
}
```

4. Review Eligibility

A contract is eligible for review if:

1. The contract status is 'completed'
2. The user is either the client or freelancer of the contract
3. The user hasn't already reviewed this contract

```
const isParticipant =
  contract.client_id === userId ||
```

```
  contract.freelancer_id === userId;

if (!isParticipant) {
  // Return error: FORBIDDEN
}

if (contract.status !== 'completed') {
  // Return error: CONTRACT_NOT_COMPLETED
}

const existingReview = await db.query(
  'SELECT * FROM reviews WHERE contract_id = $1 AND reviewer_
id = $2',
  [contractId, userId]
);

if (existingReview) {
  // Return error: ALREADY_REVIEWED
}
```

Contract and Proposal Logic

Accepting a Proposal

When a proposal is accepted, perform these operations **atomically** (in a transaction):

```
// 1. Update proposal status
UPDATE proposals SET status = 'accepted' WHERE id = proposalI
d;

// 2. Reject all other proposals for the same project
UPDATE proposals
SET status = 'rejected'
WHERE project_id = projectId AND id != proposalId;
```

```
// 3. Update project status
UPDATE projects SET status = 'in_progress' WHERE id = project
Id;

// 4. Create contract
INSERT INTO contracts (id, project_id, freelancer_id, client_
id, total_amount, status)
VALUES (...);

// 5. Create milestones
INSERT INTO milestones (id, contract_id, title, description,
amount, due_date, order_index)
VALUES (...);
```

Completing a Contract

When the last milestone is approved:

```
// Check if this is the last milestone
const totalMilestones = await db.query(
  'SELECT COUNT(*) FROM milestones WHERE contract_id = $1',
  [contractId]
);

const approvedMilestones = await db.query(
  'SELECT COUNT(*) FROM milestones WHERE contract_id = $1 AND
status = $2',
  [contractId, 'approved']
);

if (approvedMilestones.count === totalMilestones.count) {
  // Update contract
  UPDATE contracts
  SET status = 'completed', completed_at = CURRENT_TIMESTAMP
  WHERE id = contractId;
```

```
  // Update project
  UPDATE projects
  SET status = 'completed'
  WHERE id = projectId;
}
```

Service Rating Update

When a review is submitted for a freelancer:

```
// Find all services by this freelancer
const services = await db.query(
  'SELECT * FROM services WHERE freelancer_id = $1',
  [freelancerId]
);

// Update each service's rating
for (const service of services) {
  const newRating =
    ((service.rating * service.total_reviews) + review.ratin
g) /
    (service.total_reviews + 1);

  await db.query(
    'UPDATE services SET rating = $1, total_reviews = total_r
eviews + 1 WHERE id = $2',
    [newRating, service.id]
  );
}
```

Authentication & Authorization

JWT Token Structure

```
{
  "userId": "usr_1a2b3c4d5e",
  "role": "freelancer",
  "email": "rahul@example.com"
}
```

Role-Based Access Control

```
// Freelancer-only endpoints
if (user.role !== 'freelancer') {
  return res.status(403).json({
    success: false,
    data: null,
    error: 'FORBIDDEN'
  });
}

// Client-only endpoints
if (user.role !== 'client') {
  return res.status(403).json({
    success: false,
    data: null,
    error: 'FORBIDDEN'
  });
}
```