Freelance Platform

Database Design, Table Structures, Constraints, and API Endpoints

System Design Document

November 18, 2025

Contents

1 Overview

This document presents the database design for the Freelance Platform, including:

- Conceptual ER model
- Full table structures (Style A)
- Constraints (PK, FK, Unique, Check)
- Many-to-many relationship tables
- API endpoints
- UML diagrams (ER + Negotiation flow)

2 Actor Tables

2.1 Users Table (Parent)

Column	Type	Description
id	UUID	Primary Key
first_name	VARCHAR(150)	User first name / username
last_name	VARCHAR(150)	User last name / username
email	VARCHAR(254)	Unique email
password	TEXT	Hashed password
role	ENUM	admin / client / freelancer / company
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

Constraints: PK(id), UNIQUE(email), CHECK(role IN ('admin','client','freelancer','company'))

2.2 Admins Table (Child)

user_id	UUID	PK, FK → users.id
profile_photo	VARCHAR(255)	Profile image URL
phone_number	VARCHAR(50)	Phone contact

2.3 Clients Table (Child)

user_id	UUID	PK, FK → users.id
profile_photo	VARCHAR(255)	Profile image URL
phone_number	VARCHAR(50)	Phone contact
city	VARCHAR(120)	City
wilaya	VARCHAR(120)	Wilaya

2.4 Freelancers Table (Child)

user_id	UUID	PK, FK → users.id
profile_photo	VARCHAR(255)	Profile image URL
description	TEXT	Freelancer description for profile/card
categories	JSONB	Categories of work
skills	JSONB	Skill list
city	VARCHAR(120)	City
wilaya	VARCHAR(120)	Wilaya
phone_number	VARCHAR(50)	Phone
years_experience	INT	Experience years
national_id_number	VARCHAR(50)	Optional
social_links	JSONB	LinkedIn / GitHub / Behance
rate	DECIMAL(10,2)	Hourly / project rate
education	JSONB	Optional education
ccp_account	VARCHAR(50)	CCP account number
baridi_account	VARCHAR(50)	BaridiMob account number
cv_attachments	VARCHAR(255)	Optional CV file path / URL

2.5 Companies Table (Child)

user_id	UUID	PK, FK → users.id
registered_name	VARCHAR(255)	Company name
registration_number	VARCHAR(120)	CNRC
tax_id	VARCHAR(120)	NIF
registered_address	TEXT	Address
business_email	VARCHAR(254)	Login email
logo	VARCHAR(255)	Logo URL
description	TEXT	Company description
industry	VARCHAR(150)	Sector
is_verified	BOOLEAN	Verification status

3 Skills and Categories

3.1 Skills Table

id	UUID	Primary Key
name	VARCHAR(120)	Unique skill

Constraints: PK(id), UNIQUE(name)

3.2 Categories Table

id	UUID	Primary Key
name	VARCHAR(120)	Unique category

Constraints: PK(id), UNIQUE(name)

4 FAQ Table

id	UUID	Primary key
question	TEXT	FAQ question
answer	TEXT	FAQ answer
created_at	TIMESTAMP	Timestamp

5 Requests and Projects

5.1 Requests Table (Client submissions)

id	UUID	Primary key
client_id	UUID	FK → clients.user_id
title	VARCHAR(255)	Request title
description	TEXT	Detailed description
attachments	JSONB	Files uploaded
category	VARCHAR(120)	Optional category
budget_min	DECIMAL	Minimum budget
budget_max	DECIMAL	Maximum budget
status	ENUM	open / closed / cancelled
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(client_id → clients.user_id)

5.2 Negotiation Table

id	UUID	Primary key
origin_type	ENUM	request / direct_hire
request_id	UUID (NULL)	FK → requests.id (NULL when origin = direct_hire)
client_id	UUID	FK → clients.user_id
freelancer_id	UUID	FK → freelancers.user_id
client_description	TEXT	Client full description of project
client_attachments	JSONB	Attachments by client
client_agreed	BOOLEAN	True when client agrees to proceed
freelancer_agreed	BOOLEAN	True when freelancer agrees to proceed
declined_by	UUID (NULL)	User ID of who declined (client or freelancer)
decline_reason	TEXT (NULL)	Explanation provided by the user who declined
status	ENUM	ongoing / agreed / declined
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(request_id → requests.id), CHECK((origin_type = 'direct_hire' AND request_id IS NULL) OR (origin_type = 'request' AND request_id IS NOT NULL))

5.3 Negotiation Floating Comments Table

id	UUID	Primary key
negotiation_id	UUID	FK → negotiation.id
user_id	UUID	FK → users.id
comment	TEXT	Comment content
parent_id	UUID (nullable)	For replies (threaded comments)
x_position	FLOAT	X coordinate on page
y_position	FLOAT	Y coordinate on page
status	ENUM	pending / resolved
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(negotiation_id → negotiation.id)

5.4 Negotiation Phases Table

id	UUID	Primary key
negotiation_id	UUID	FK → negotiation.id
title	VARCHAR(255)	Phase title
description	TEXT	Phase description
budget	DECIMAL	Phase budget
deadline	DATE	Phase deadline
deliverable	TEXT	Deliverable description
status	ENUM	pending / in_progress / completed / approved / revision_required
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(negotiation_id → negotiation.id)

5.5 Projects Table (Active Projects)

id	UUID	Primary key
negotiation_id	UUID	FK → negotiation.id
status	ENUM	in_progress / done
start_date	DATE	Start date
end_date	DATE	End date
created_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(negotiation_id → negotiation.id)

5.6 Project Phases Table

id	UUID	PK
project_id	UUID	FK → projects.id
title	VARCHAR(255)	Phase title
description	TEXT	Phase description
budget	DECIMAL	Phase budget

estimated_duration	INTEGER	Estimated duration in days
deliverable	TEXT	Final deliverable description
status	ENUM	pending / in_progress / waiting_for_client_review / done
created_at	TIMESTAMP	Timestamp
updated_at	TIMESTAMP	Timestamp

5.7 Deliverables Table

id	UUID	PK
phase_id	UUID	FK → project_phases.id
[redacted]		
attachments	VARCHAR(255)	Optional file path (NULL allowed)
text_content	TEXT	Optional text content / notes (NULL allowed)
submitted_at	TIMESTAMP	Submission timestamp
updated_at	TIMESTAMP	Timestamp

6 Reviews Table

id	UUID	PK
client_id	UUID	FK → clients.user_id
freelancer_id	UUID	FK → freelancers.user_id
rating	INT	1–5
feedback	TEXT	Review text
created_at	TIMESTAMP	Timestamp

7 Community System not implemented

7.1 Community Posts Table

id	UUID	PK
owner_id	UUID	FK → users.id
description	TEXT	Post content
attachments	JSONB	Files
updated_at	TIMESTAMP	Timestamp
created_at	TIMESTAMP	Timestamp

7.2 Community Comments Table

id	UUID	PK
post_id	UUID	FK → community_posts.id
user_id	UUID	FK → users.id
parent_id	UUID (NULL)	FK → community_comments.id (NULL = top-level comment)
comment	TEXT	Comment text
updated_at	TIMESTAMP	Timestamp

created_at	TIMESTAMP	Timestamp

7.3 Community Likes Table

id	UUID	PK
post_id	UUID	FK → community_posts.id
user_id	UUID	FK → users.id
created_at	TIMESTAMP	Timestamp

8 Job / Internship Offers Table

id	UUID	PK
company_id	UUID	FK → companies.id
title	VARCHAR(255)	Job title
type	ENUM	job / internship
requirements	TEXT	Requirements
duration	VARCHAR(120)	Duration
what_needed	TEXT	Needs
attachments	JSONB	Attachments (files, images, PDFs)
created_at	TIMESTAMP	Timestamp

9 Reports Table

Column	**Type**	**Description**
id	UUID	PK
reporter_id	UUID	FK → users.id
type	ENUM	client / post / request / freelancer / comment
target_id	UUID	ID of the target entity
text	TEXT	Report description
created_at	TIMESTAMP	Timestamp

10 Notifications Table

id	UUID	PK
receiver_id	UUID	FK → users.id
content	TEXT	Notification text
seen	BOOLEAN	Read status
created_at	TIMESTAMP	Timestamp

11 Help Table

id	UUID	PK
user_id	UUID	FK → users.id
problem_desc	TEXT	Description of the problem

status	ENUM	pending / resolved
created_at	TIMESTAMP	Timestamp

12 Media / Files Table

id	UUID	Primary key
owner_id	UUID	FK → users.id
entity_type	ENUM	request / negotiation / phaseDeliverable / CV / Post / company
entity_id	UUID	ID of related entity
file_url	TEXT	File URL
file_type	VARCHAR(50)	e.g., pdf, image
created_at	TIMESTAMP	Timestamp

Constraints: PK(id), FK(owner_id → users.id)

13 API Endpoints (Detailed)

13.1 Authentication / Users

Method	**Endpoint**	**Description**
POST	/auth/register/freelancer	Register a freelancer user account
POST	/auth/register/client	Register a new client account
POST	/auth/register/company	Register a new company account
POST	/auth/login	Authenticate user and get token
POST	/auth/logout	Logout user, invalidate session
GET	/auth/verify-email/:token	Verify user email using token
POST	/auth/forgot-password	Request password reset email
POST	/auth/reset-password	Reset password using token
GET	/freelancers/:id	Get freelancer profile information
PUT	/freelancers/:id	Update freelancer profile information
PUT	/freelancers/:id/password	Update freelancer password securely
GET	/clients/:id	Get client profile information
PUT	/clients/:id	Update client profile information
PUT	/clients/:id/password	Update client password securely
GET	/companies/:id	Get company profile information
PUT	/companies/:id	Update company profile information
PUT	/companies/:id/password	Update company password securely
DELETE	/users/:id	Soft delete user account (all types)

13.2 Requests

Method	**Endpoint**	**Description**
GET	/requests	List all user requests
GET	/requests/:client_id	List all client requests
POST	/requests	Create a new request

GET	/requests/:id	Get single request details
PUT	/requests/:id	Update request information
DELETE	/requests/:id	Soft delete request

13.3 Negotiations

Method	Endpoint	Description
POST	/negotiations/direct-hire/:freelancerId	Create negotiation via direct hiring
POST	/negotiations/:requestId/create	Create negotiation linked to request
GET	/negotiations/:id	Get negotiation + phases
POST	/negotiations/:id/phases	Add phase to negotiation
PUT	/negotiations/phases/:phaseId	Update negotiation phase
DELETE	/negotiations/phases/:phaseId	Delete negotiation phase
POST	/negotiations/:id/agree	Current user agrees to negotiation (client or freelancer)
POST	/negotiations/:id/decline	Current user declines negotiation + provides reason
DELETE	/negotiations/:id	Soft delete negotiation

13.4 Projects

Method	Endpoint	Description
GET	/projects/:user_id	List all projects for user (client or freelancer)
GET	/projects/:id/phases	Get all phases of a project
POST	/projects/:id/phases	Add new phase to a project
PUT	/projects/:id/phases/:phaseId	Update project phase (title, description, duration, budget)
DELETE	/projects/:id/phases/:phaseId	Soft delete a project phase
Phase Workflow APIs		
POST	/projects/phases/:phaseId/start	Mark phase as in_progress (freelancer)
POST	/projects/phases/:phaseId/submit	Submit deliverable → status = waiting_for_client_review
POST	/projects/phases/:phaseId/approve	Client approves deliverable → status = done
POST	/projects/phases/:phaseId/next	Move next phase to in_progress (system or client action)
Get	/projects/<int:id>	get projects details

13.5 Comments

Method	Endpoint	Description
POST	/negotiations/:id/comments	Add comment to negotiation
[redacted]		
GET	/negotiations/:id/comments	Get all comments for negotiation

PUT	/comments/:id	Update comment (negotiation or project)
DELETE	/comments/:id	Soft delete comment
POST	/comments/:id/resolve	Mark comment as resolved
POST	/comments/:id/reply	Add reply to a comment (nested comment)

13.6 Media / Files

Method	Endpoint	Description
POST	/media/upload	Upload media file to entity
GET	/media/:entityType/:entityId	Get media files for entity
DELETE	/media/:id	Soft delete media file

13.7 Admin

Method	Endpoint	Description
GET	/admin/stats/users	Get number of freelancers, clients, companies
GET	/admin/stats/posts	Get number of posted jobs
GET	/admin/stats/requests	Get number of active requests
GET	/admin/stats/negotiations	Get active & declined negotiations
GET	/admin/stats/projects	Get active & declined projects
POST	/admin/skills	Add new skill
POST	/admin/categories	Add new category
GET	/skills	Get all skills
GET	/categories	Get all categories
GET	/admin/reports	Get all reports (pending and resolved)
POST	/admin/reports/:id/resolve	Mark report as resolved

13.8 FAQ

Method	Endpoint	Description
GET	/faq	Get all FAQs (public)
POST	/admin/faq	Add new FAQ (admin only)
PUT	/admin/faq/{id}	Update FAQ by ID (admin only)
DELETE	/admin/faq/{id}	Delete FAQ by ID (admin only)

13.9 Reviews

Method	Endpoint	Description
POST	/reviews	Add a review (client → freelancer)
GET	/reviews/freelancer/{freelancerId}	Get all reviews for a freelancer
PUT	/reviews/{id}	Update a review (owner only)
DELETE	/reviews/{id}	Soft delete a review (owner or admin)

13.10 Community APIs not implemented

Method	Endpoint	Description
	— Posts —	
POST	/community/posts	Create new post
GET	/community/posts	Get all posts (paginated)
GET	/community/posts/{id}	Get single post (with comments + likes count)
PUT	/community/posts/{id}	Update post (owner only)
DELETE	/community/posts/{id}	Soft delete post
	— Comments —	
POST	/community/posts/{postId}/comments	Add comment to post
POST	/community/comments/{id}/reply	Reply to a comment
GET	/community/posts/{postId}/comments	Get all comments for post (nested)
PUT	/community/comments/{id}	Update comment
DELETE	/community/comments/{id}	Soft delete comment
	— Likes —	
POST	/community/posts/{postId}/like	Like a post
DELETE	/community/posts/{postId}/unlike	Remove like from post
GET	/community/posts/{postId}/likes	Get list of users who liked post

13.11 Offer Management not implemented

Method	Endpoint	Description
POST	/offers	Create a new job or internship offer (company only)
PUT	/offers/{offer_id}	Update an existing job or internship offer (company only)
DELETE	/offers/{offer_id}	Delete a job or internship offer (company only)
GET	/offers	Get all job and internship offers
GET	/offers/company/{company_id}	Get all job and internship offers posted by a specific company

13.12 Reports Management not implemented

Method	Endpoint	Description
POST	/reports	Create a new report
DELETE	/reports/{report_id}	Delete a report (owner or admin)
GET	/reports/my	Get all reports submitted by the authenticated user

13.13 Notifications Management not implemented

Method	Endpoint	Description
POST	/notifications	Create a notification for a user
GET	/notifications/my	Get all notifications of the authenticated user

PUT	/notifications/my/mark-all-seen	Mark all notifications of the authenticated user as seen

13.14 Help Requests

Method	**Endpoint**	**Description**
POST	/help	Submit a help request
GET	/help/my	Get all help requests submitted by the authenticated user
PUT	/help/:id/resolve	Admin: mark a help request as resolved