COIMBATORE INSTITUTE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University)

DEPARTMENT OF COMPUTING
M.Sc. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19MAM33-DATABASE MANAGEMENT SYSTEM

NAME: NANDA.S
REG NO: 7176 23 34 025
BATCH: 2023-28
BRANCH: MSc.AI&ML

FREELANCER HUB

1. Introduction

freelancer hub is an online marketplace through which freelancers and clients connect for work on several projects. The site enables clients to post jobs, make proposals, sign contracts, and send/receive payments with freelancers. This case study examines the database that supports real-time operation of freelancer hub.

2.Abstract:

the "Freelance Work Arena" is a comprehensive platform that connects freelancers with clients, facilitating job posting, proposal submissions, contract management, and communication. The system provides distinct interfaces for freelancers and clients, allowing for efficient job management, proposal handling, and payment tracking. By streamlining these processes, the platform aims to enhance the freelancing experience, offering secure and transparent workflows for both parties.

3. Application Overview

freelancer hub makes it possible for the following operation:

- **Account management:** Registration, profile development and account management.
- **Project posting :** Clients post jobs, and freelancers submit proposals.
- **Contract management:** Clients and freelancers offer or create service contracts.
- **Payments services:** freelancer hub manages payments and ensured correct payouts to freelancers via custodial accounts
- **Real-time Communication:** Messages and alerts between clients and freelancers can be carried out in real-time.

4. Database Type

Relational Database (SQL):

used for structured data like

- **User Management:** Tables for users (clients and freelancers), where structured data like name, email, ratings, and skills are stored.
- **Job Listings & Contracts:** Tables to store job details, proposals, and contracts with clear relationships between clients and freelancers.
- **Payment Transactions:** Handling payments,payouts and disbursements and reliable transactional support.

NoSQL Database:

Applied in the case of unstructured or semi-structured data which would include real-time chat messages, notices, or logs.

- **Messaging System:** A NoSQL database could handle chat messages and notifications in a scalable, real-time manner.
- **Freelancer Skills and Preferences:** A document-based NoSQL database like MongoDB could store unstructured data such as freelancer profiles, where new fields or attributes are frequently added or modified
- **Activity Logs & Analytics:** NoSQL databases are ideal for storing large volumes of activity logs or behavioral data, which can later be processed for insights and analysis.

5. Database Structure

The database schema for freelancer hub would likely be normalized and consist of the following key tables:

Users:

- Information about clients and freelancers.

- user_id, name, email, password, user_type, profile_picture, location, join_date, bio, contact_info

Jobs:

- Details of job postings.
- job_id, client_id, title, description, category_id, budget, status, posted_date

Proposals:

- Offers made by freelancers for jobs.
- proposal_id, job_id, freelancer_id, proposal_date, cover_letter, bid_amount, status

Contracts:

- Agreements between clients and freelancers.
- contract_id, job_id, client_id, freelancer_id, start_date, end_date, payment_terms, contract_status

Payments:

- Records of money transactions.
- payment_id, contract_id, amount, payment_date, payment_method, status

Messages:

- Communication between users.
- message_id, sender_id, receiver_id, message_text, sent_date, is_read

Ratings and Reviews:

- Feedback given by users.
- review_id, contract_id, reviewer_id, rating, review_text, review_date

Skills:

- ❖ List of skills freelancers can showcase.
- ❖ skill_id, skill_name, description

Categories:

- ❖ Different types of jobs or services.
- ❖ category_id, category_name, description

Notifications:

- ❖ Alerts and updates for users.
- ❖ notification_id, user_id, message, created_date, is_read

System Design:

Database Design:

Users Table: Contains details about freelancers and clients (username, password, email, phone, user type).

Jobs Table: Stores job details such as job ID, client ID, category, title, description, budget, status.

Proposals Table: Holds proposal details submitted by freelancers (job ID, freelancer ID, bid amount, cover letter, status).

Contracts Table: Stores contract data between clients and freelancers (contract ID, job ID, freelancer ID, client ID, status).

Messages Table: Handles communication between clients and freelancers.

Payments Table: Records payment details related to contracts.

Reviews Table: Captures reviews shared between clients and freelancers upon contract completion.

Technologies Used:

Frontend: Java Swing for user interfaces (client/freelancer home pages, job posting forms, etc.).

Backend: Java with MySQL for handling server-side logic and database interactions.

Database: MySQL database named freelance_work_arena.

Workflow:

User Login/Registration:

Users either log in or register (as a freelancer or client) and are directed to their respective home pages.

Job Posting (Client):

The client posts jobs by entering job details and setting a budget.

Jobs are displayed in the job list for freelancers to browse and apply.

Proposal Submission (Freelancer):

Freelancers view available jobs, select one, and submit proposals specifying bid amount and cover letter.

Proposal Management (Client):

The client reviews proposals and accepts/rejects them. Upon acceptance, a contract is created.

Contract Management:

Freelancers update the contract status and submit work when complete.

Clients approve the work and release payments accordingly.

Payments:

Freelancers request payment after completing a contract.

Clients approve and process payments through the system.

Messaging:

Both parties can communicate via the messaging feature throughout the job/contract process.

SCHEMA:

```
create table users (
    user_id int primary key auto_increment,
    name varchar(255) not null,
    email varchar(255) unique not null,
    password varchar(255) not null,
    user_type enum('client', 'freelancer') not null,
    profile_picture varchar(255),
    location varchar(255),
    join_date date not null,
    bio text,
    contact_info varchar(255)
);

create table jobs (
    job_id int primary key auto_increment,
    client_id int not null,
    title varchar(255) not null,
    description text,
    category_id int,
    budget decimal(10, 2),
    status enum('open', 'closed') not null,
    posted_date date not null,
    foreign key (client_id) references users(user_id),
    foreign key (category_id) references categories(category_id)
);

create table proposals (
    proposal_id int primary key auto_increment,
    job_id int not null,
    freelancer_id int not null,
    proposal_date date not null,
    cover_letter text,
    bid_amount decimal(10, 2) not null,
    status enum('pending', 'accepted', 'rejected') not null,
    foreign key (job_id) references jobs(job_id),
    foreign key (freelancer_id) references users(user_id)
);

create table contracts (
    contract_id int primary key auto_increment,
    job  id int not null,
```

```
client_id int not null,
    freelancer_id int not null,
    start_date date not null,
    end_date date,
    payment_terms text,
    contract_status enum('active', 'completed', 'terminated') not null,
    foreign key (job_id) references jobs(job_id),
    foreign key (client_id) references users(user_id),
    foreign key (freelancer_id) references users(user_id)
);

create table payments (
    payment_id int primary key auto_increment,
    contract_id int not null,
    amount decimal(10, 2) not null,
    payment_date date not null,
    payment_method varchar(255),
    status enum('pending', 'completed') not null,
    foreign key (contract_id) references contracts(contract_id)
);

create table messages (
    message_id int primary key auto_increment,
    sender_id int not null,
    receiver_id int not null,
    message_text text not null,
    sent_date date not null,
    is_read boolean,
    foreign key (sender_id) references users(user_id),
    foreign key (receiver_id) references users(user_id)
);

create table ratings_reviews (
    review_id int primary key auto_increment,
    contract_id int not null,
    reviewer_id int not null,
    rating int check (rating between 1 and 5),
    review_text text,
    review_date date not null,
    foreign key (contract_id) references contracts(contract_id),
    foreign key (reviewer_id) references users(user_id)
);
```

```
create table skills (
    skill_id int primary key auto_increment,
    skill_name varchar(255) not null,
    description text
);

create table categories (
    category_id int primary key auto_increment,
    category_name varchar(255) not null,
    description text
);

create table notifications (
    notification_id int primary key auto_increment,
    user_id int not null,
    message text not null,
    created_date date not null,
    is_read boolean,
    foreign key (user_id) references users(user_id)
);
```

Joins:

```
-- Freelancer Job Applications View
CREATE VIEW freelancer_job_applications AS
SELECT
   u.username AS freelancer_name,
   j.title AS job_title,
   p.status AS proposal_status,
   IFNULL(c.status, 'No Contract') AS contract_status
FROM
   users u
JOIN proposals p ON u.user_id = p.freelancer_id
JOIN jobs j ON p.job_id = j.job_id
LEFT JOIN contracts c ON j.job_id = c.job_id
WHERE u.user_type = 'freelancer';

-- Load Proposals for Client
SELECT p.proposal_id, j.title, u.username, p.bid_amount, p.status
FROM proposals p
JOIN jobs j ON p.job_id = j.job_id
JOIN users u ON p.freelancer_id = u.user_id
WHERE j.client_id = (SELECT user_id FROM users WHERE username
= ?);

-- Fetching payment history for a specific user (client or freelancer)
SELECT p.payment_id, p.contract_id, c.client_id, c.freelancer_id,
p.amount, p.payment_date, p.status
FROM payments p
JOIN contracts c ON p.contract_id = c.contract_id
JOIN users u1 ON c.client_id = u1.user_id
JOIN users u2 ON c.freelancer_id = u2.user_id
WHERE u1.username = ? OR u2.username = ?;

-- Example query to get job details for a freelancer
SELECT j.job_id, j.title, j.description, j.budget, j.skills_required,
j.category_id, j.client_id
```

```
FROM jobs j
JOIN categories c ON j.category_id = c.category_id
WHERE c.name = ? AND j.status = 'open';

-- Fetching contracts between a freelancer and client
SELECT c.contract_id, c.job_id, c.client_id, c.freelancer_id,
c.start_date, c.end_date, c.status
FROM contracts c
JOIN users u1 ON c.client_id = u1.user_id
JOIN users u2 ON c.freelancer_id = u2.user_id
WHERE u1.username = ? OR u2.username = ?;

-- Query for fetching feedback and ratings given to freelancers by
clients
SELECT r.review_id, r.contract_id, r.client_id, r.freelancer_id, r.rating,
r.feedback
FROM reviews r
WHERE r.freelancer_id = ?;
```

View:

```
-- Freelancer Job Applications View
CREATE VIEW freelancer_job_applications AS
SELECT
   u.username AS freelancer_name,
   j.title AS job_title,
   p.status AS proposal_status,
   IFNULL(c.status, 'No Contract') AS contract_status
FROM
   users u
JOIN proposals p ON u.user_id = p.freelancer_id
JOIN jobs j ON p.job_id = j.job_id
LEFT JOIN contracts c ON j.job_id = c.job_id
WHERE u.user_type = 'freelancer';

-- Update Password Query
```

```
CREATE VIEW FreelancerJobView AS
SELECT j.job_id, j.title, j.description, j.budget, c.category_name
FROM jobs j
JOIN categories c ON j.category_id = c.category_id;
```

Triggers:

```
-- Trigger to update job status after contract creation
DELIMITER //
CREATE TRIGGER update_job_status_after_contract
AFTER INSERT ON contracts
FOR EACH ROW
BEGIN
   UPDATE jobs
   SET status = 'in progress'
   WHERE job_id = NEW.job_id;
END;
//
DELIMITER ;
```

Insertions:

```
-- Insert New User Query
INSERT INTO users (username, email, phone_no, password,
user_type) VALUES (?, ?, ?, ?, ?);

-- Insert Contract Query
INSERT INTO contracts (job_id, freelancer_id, client_id, status)
VALUES (?, ?, (SELECT user_id FROM users WHERE username = ?), ?);

-- Insert New Job Query
INSERT INTO jobs (client_id, category_id, title, description, budget,
posted_date)
VALUES (?, ?, ?, ?, ?, CURDATE());

-- Insert Proposal Query
```

```
INSERT INTO proposals (job_id, freelancer_id, cover_letter,
bid_amount, status)
VALUES (?, ?, ?, ?, ?);
```

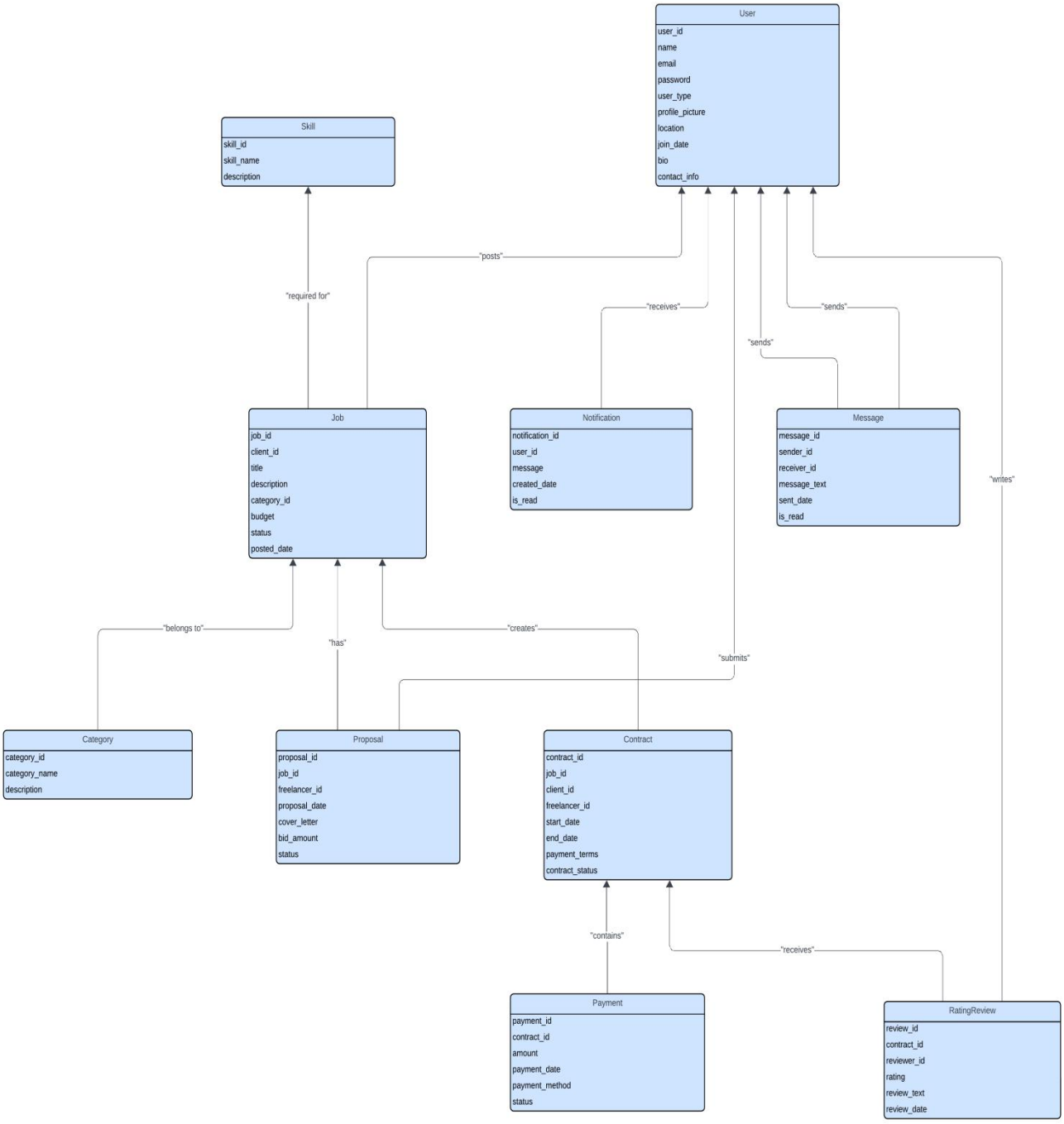
Update Queries:

```
-- Update Job Details Query
UPDATE jobs SET title = ?, description = ?, budget = ? WHERE job_id
= ?;

-- Update Proposal Status Query
UPDATE proposals SET status = 'accepted' WHERE proposal_id = ?;

-- Reject Proposal Query
UPDATE proposals SET status = 'Rejected' WHERE proposal_id = ?;
```

Delete Queries:

```
-- Delete Proposal Query
DELETE FROM proposals WHERE proposal_id = ?;

-- Query to check if a specific job exists by job_id
SELECT job_id FROM jobs WHERE job_id = ?;
```

ER Diagram:


User
user_id
name
email
password
user_type
profile_picture
location
join_date
bio
contact_info
Skill
skill_id
skill_name
description
"posts"
"required for"
"receives"
"sends"
"sends"
"writes"
Job
job_id
client_id
title
description
category_id
budget
status
posted_date
Notification
notification_id
user_id
message
created_date
is_read
Message
message_id
sender_id
receiver_id
message_text
sent_date
is_read
"belongs to"
"has"
"creates"
"submits"
Category
category_id
category_name
description
Proposal
proposal_id
job_id
freelancer_id
proposal_date
cover_letter
bid_amount
status
Contract
contract_id
job_id
client_id
freelancer_id
start_date
end_date
payment_terms
contract_status
"contains"
"receives"
Payment
payment_id
contract_id
amount
payment_date
payment_method
status
RatingReview
review_id
contract_id
reviewer_id
rating
review_text
review_date

Relationships

One-to-One :

- has_notification -> User to Notification
- has_payment -> Contract to Payment

One-to-Many :

- posts -> User (Client) to Job -> Client creates a job.
- submits -> User (Freelancer) to Proposal -> Freelancer submits a proposal for a job.
- receives -> Job to Proposal -> Job receives proposals from freelancers.
- has_contracts -> Job to Contract -> Job can have one or more contracts.
- receives -> Contract to RatingReview -> Contract receives ratings and reviews.
- belongs_to -> Category to Job -> Job belongs to a specific category.
- sends -> User to Message -> User sends messages to other users.

Many-to-Many

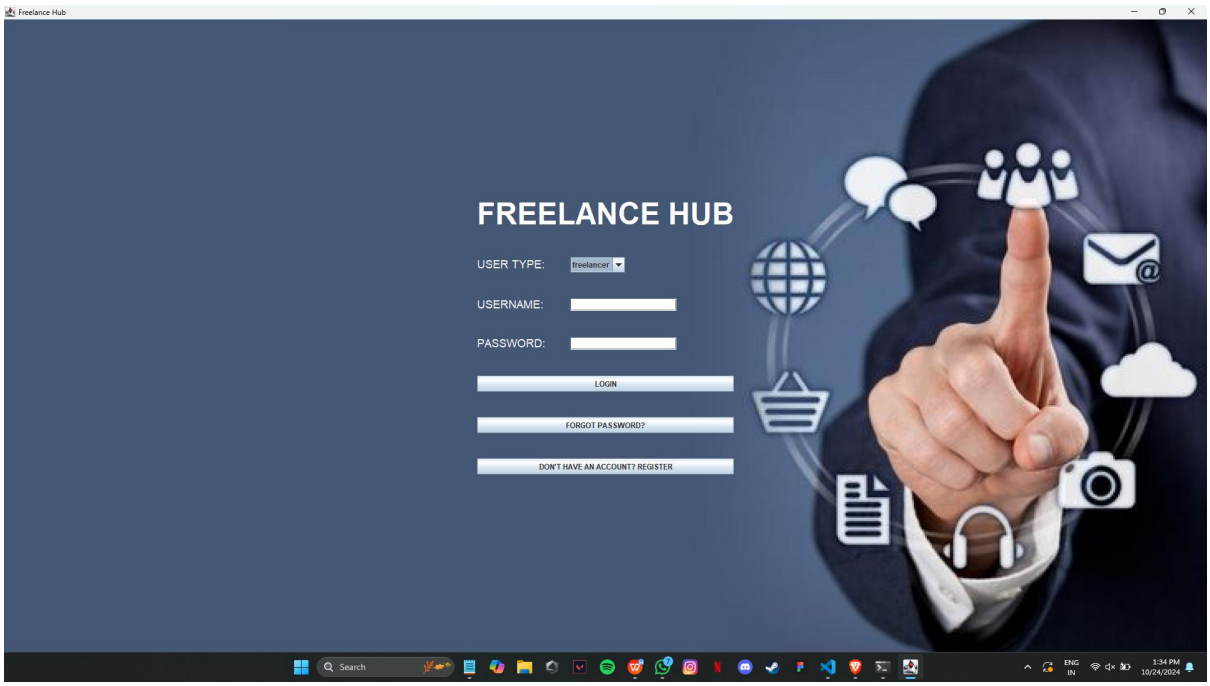
- exchanges -> User to Message -> Users exchange messages with each other.
- applies_for -> Proposal to Job -> Proposal applies for a job.
- requires -> Skills to Jobs/Proposals
- creates -> Client to Contract
- contains -> Contract to Payment
- writes -> User to RatingReview

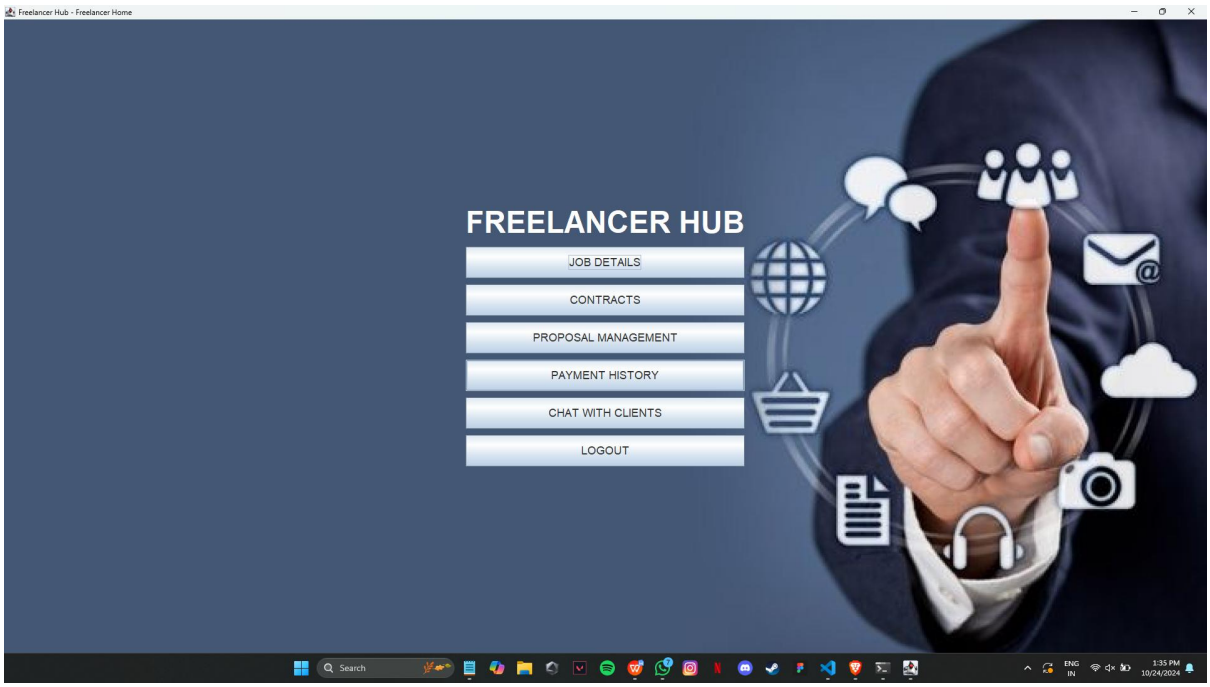
6. Data Volume

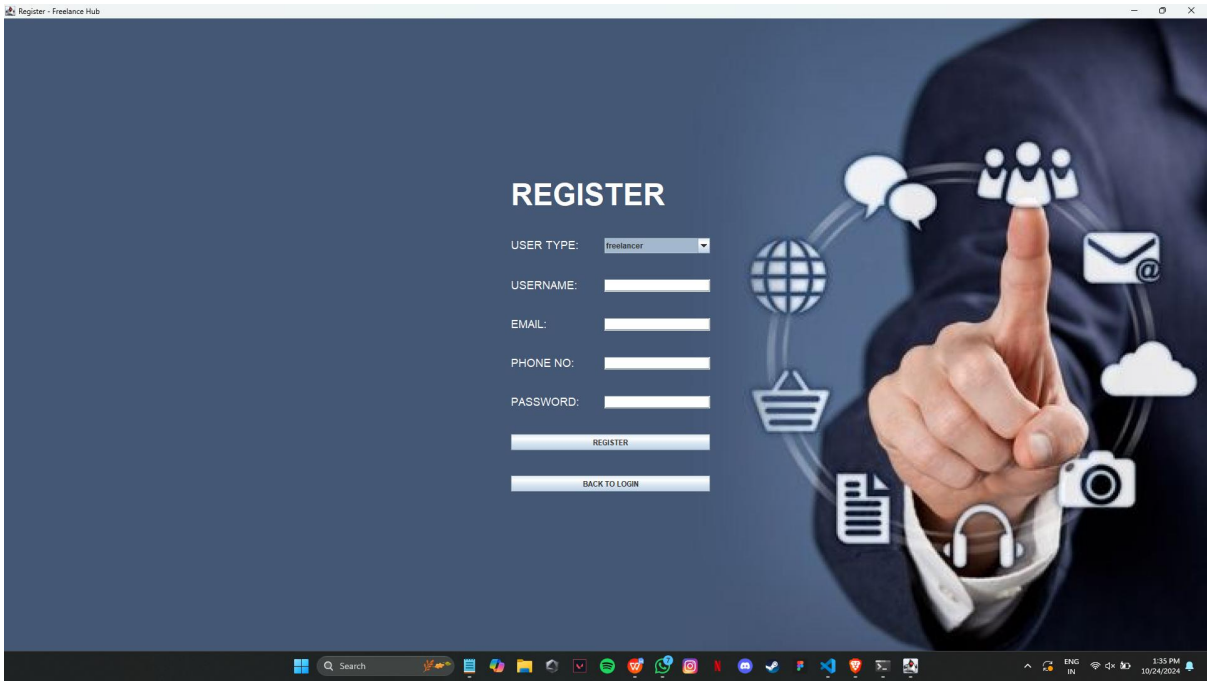
freelancer hub handles millions of users and contracts daily. Based on statistics, the platform serves around 20 million freelancers and 5 million clients, with 3 million jobs posted annually. Given this volume,

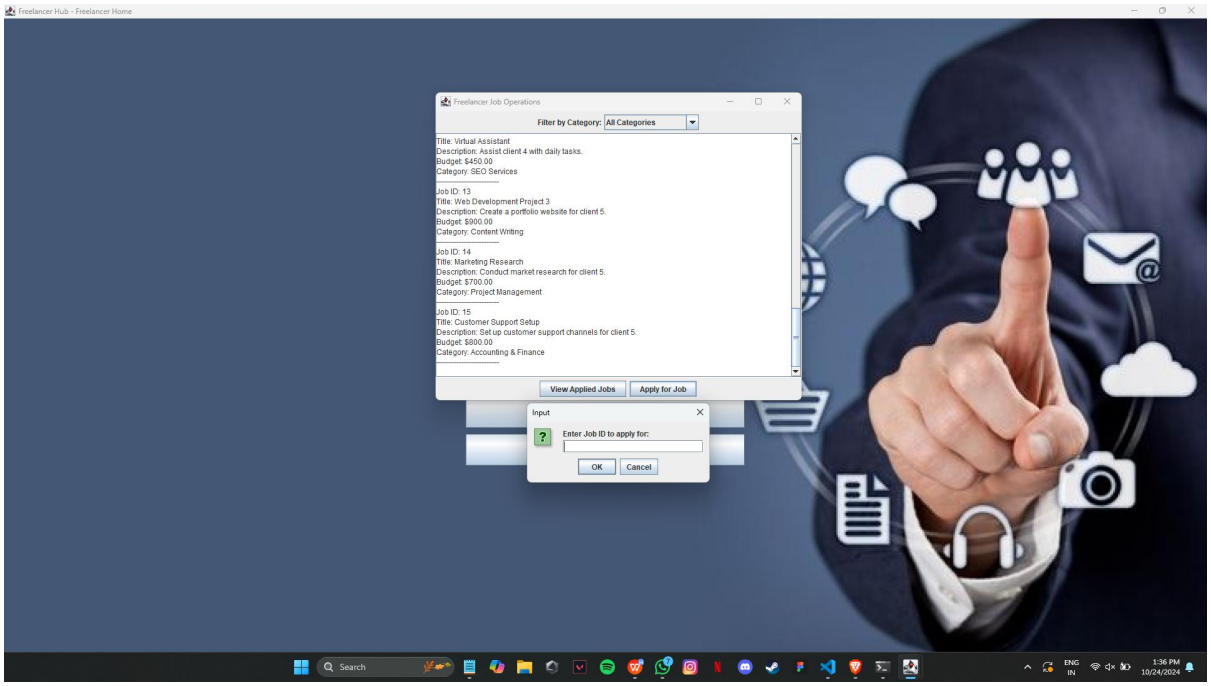
the database must be capable of storing a vast amount of structured data as well as unstructured data

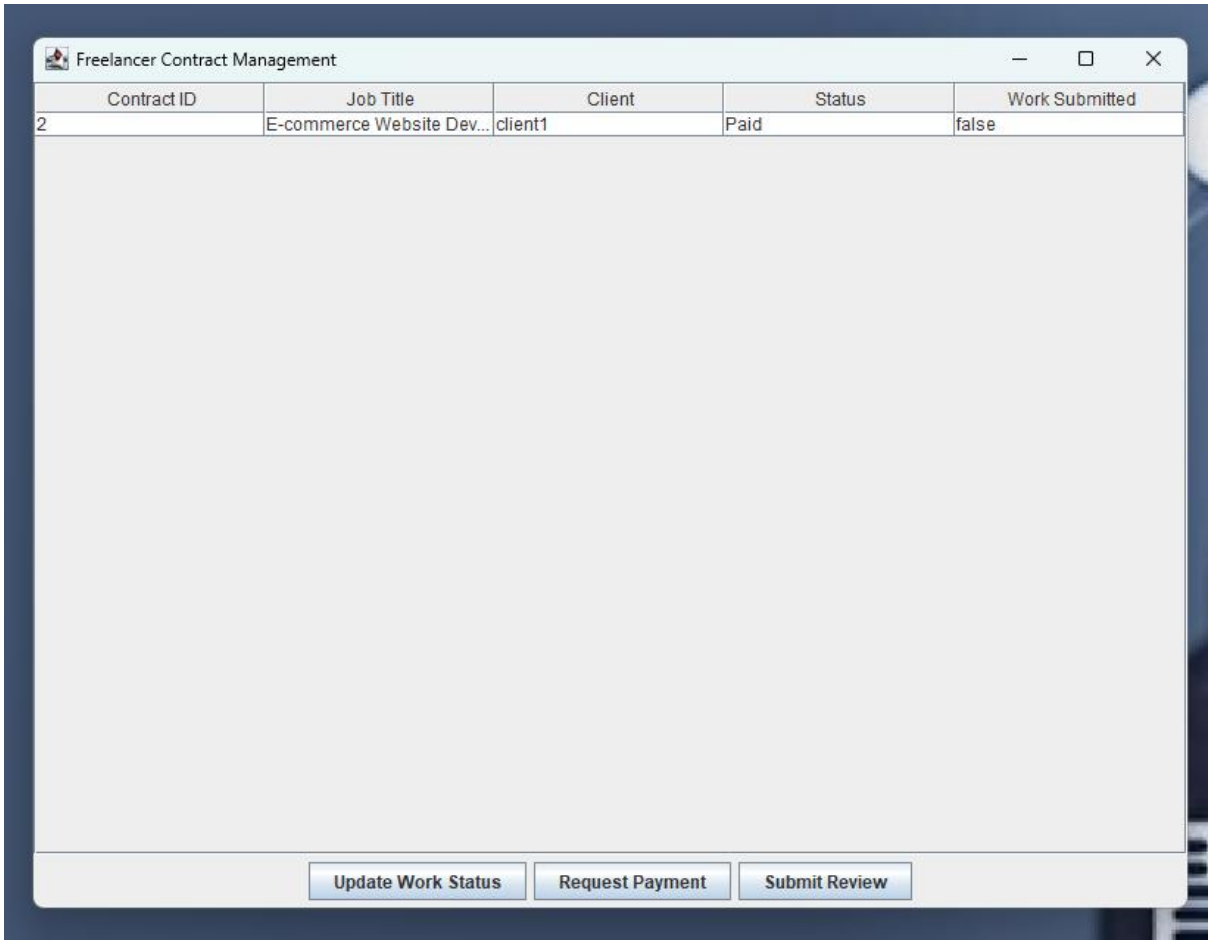
- ❖ Estimated data storage: Hundreds of terabytes.
- ❖ Growth rate: Given the increasing number of users and transactions, the database likely grows at a rate of several gigabytes per day.






Register - Freelance Hub
REGISTER
USER TYPE:
freelancer
USERNAME:
EMAIL:
PHONE NO:
PASSWORD:
REGISTER
BACK TO LOGIN
Search
ENG
IN
1:35 PM
10/24/2024

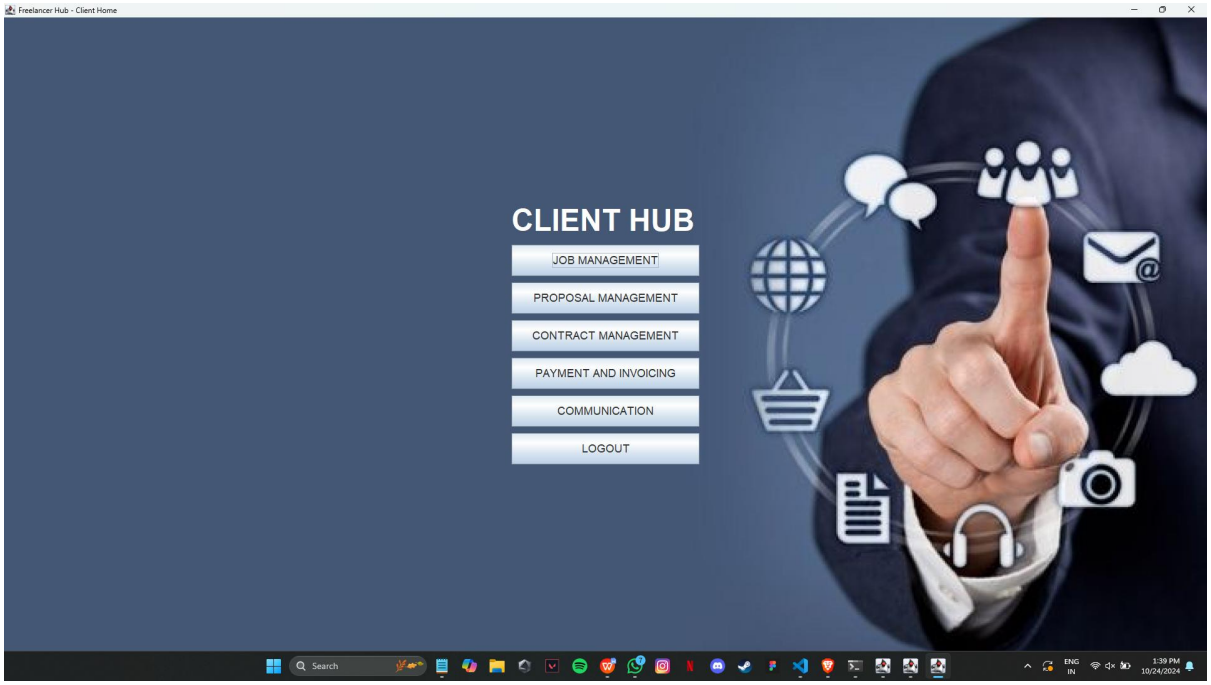

Freelancer Hub - Freelancer Home
Freelancer Job Operations
Filter by Category:
All Categories
Title: Virtual Assistant
Description: Assist client 4 with daily tasks.
Budget: $450.00
Category: SEO Services
Job ID: 13
Title: Web Development Project 3
Description: Create a portfolio website for client 5.
Budget: $900.00
Category: Content Writing
Job ID: 14
Title: Marketing Research
Description: Conduct market research for client 5.
Budget: $700.00
Category: Project Management
Job ID: 15
Title: Customer Support Setup
Description: Set up customer support channels for client 5.
Budget: $800.00
Category: Accounting & Finance
View Applied Jobs
Apply for Job
Input
Enter Job ID to apply for:
OK
Cancel
Search
ENG
IN
1:36 PM
10/24/2024

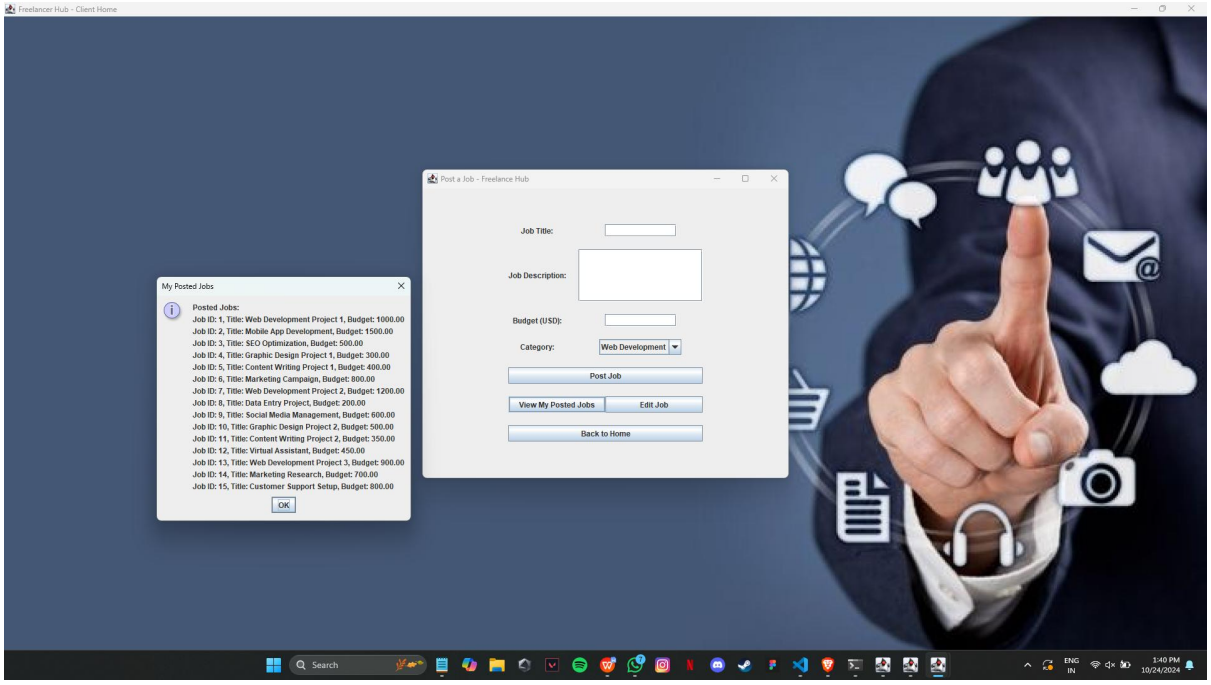


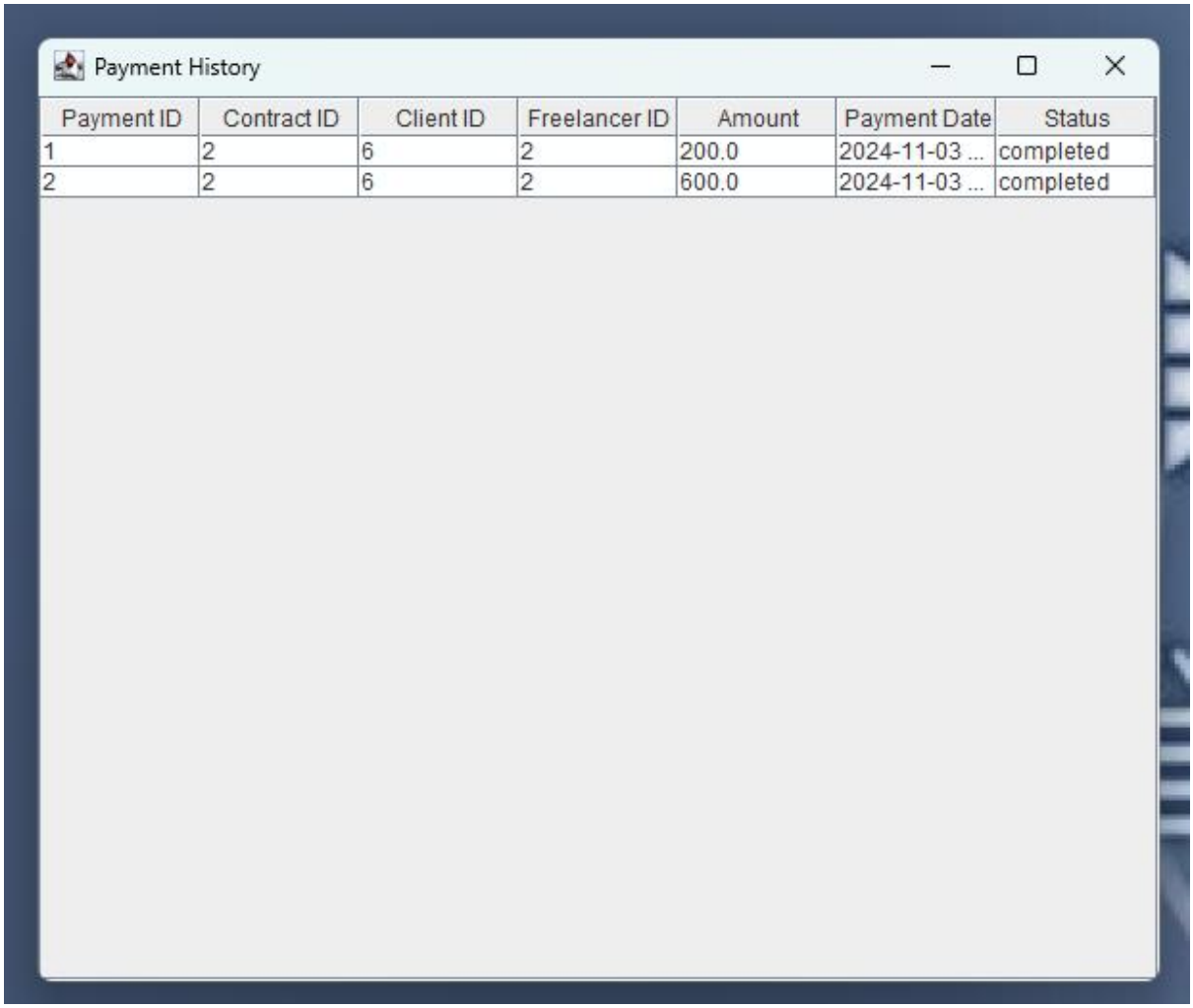
Client Proposal Management

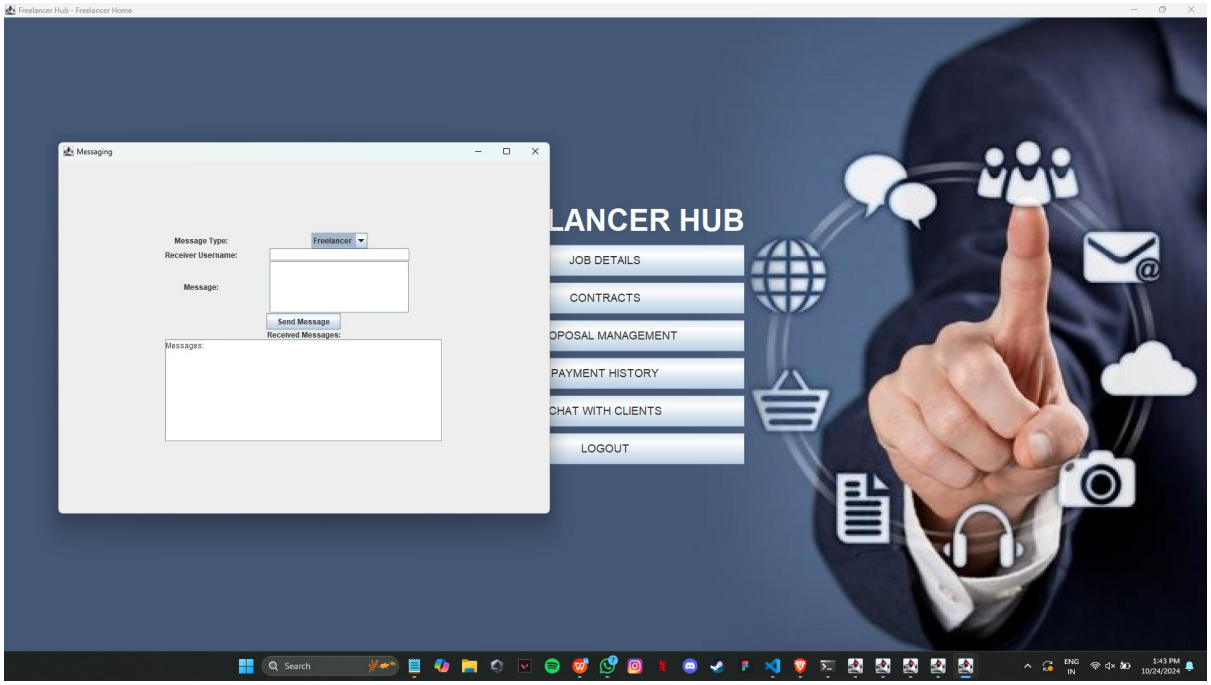
Proposal ID	Job ID	Freelancer ID	Bid Amount	Status
1	1	1	25.0	accepted
3	2	1	25.0	rejected
4	1	1	34.0	accepted
5	1	1	234.0	accepted
6	1	1	56.0	accepted
7	3	1	234.0	accepted
8	3	1	23.0	accepted
9	1	1	24.0	accepted

Accept Proposal | Reject Proposal


Freelancer Hub - Client Home
CLIENT HUB
JOB MANAGEMENT
PROPOSAL MANAGEMENT
CONTRACT MANAGEMENT
PAYMENT AND INVOICING
COMMUNICATION
LOGOUT
Search
ENG
IN
1:39 PM
10/24/2024


Freelancer Hub - Client Home
My Posted Jobs
Posted Jobs:
Job ID: 1, Title: Web Development Project 1, Budget: 1000.00
Job ID: 2, Title: Mobile App Development, Budget: 1500.00
Job ID: 3, Title: SEO Optimization, Budget: 500.00
Job ID: 4, Title: Graphic Design Project 1, Budget: 300.00
Job ID: 5, Title: Content Writing Project 1, Budget: 400.00
Job ID: 6, Title: Marketing Campaign, Budget: 800.00
Job ID: 7, Title: Web Development Project 2, Budget: 1200.00
Job ID: 8, Title: Data Entry Project, Budget: 200.00
Job ID: 9, Title: Social Media Management, Budget: 600.00
Job ID: 10, Title: Graphic Design Project 2, Budget: 500.00
Job ID: 11, Title: Content Writing Project 2, Budget: 350.00
Job ID: 12, Title: Virtual Assistant, Budget: 450.00
Job ID: 13, Title: Web Development Project 3, Budget: 900.00
Job ID: 14, Title: Marketing Research, Budget: 700.00
Job ID: 15, Title: Customer Support Setup, Budget: 800.00
OK
Post a Job - Freelance Hub
Job Title:
Job Description:
Budget (USD):
Category:
Web Development
Post Job
View My Posted Jobs
Edit Job
Back to Home
Search
ENG
IN
1:40 PM
10/24/2024


Payment History
Payment ID
Contract ID
Client ID
Freelancer ID
Amount
Payment Date
Status
1
2
6
2
200.0
2024-11-03 ...
completed
2
2
6
2
600.0
2024-11-03 ...
completed


Freelancer Hub - Freelancer Home
Messaging
Message Type:
Freelancer
Receiver Username:
Message:
Send Message
Received Messages:
Messages:
LANCER HUB
JOB DETAILS
CONTRACTS
OPOSAL MANAGEMENT
PAYMENT HISTORY
CHAT WITH CLIENTS
LOGOUT
Search
ENG
IN
1:43 PM
10/24/2024

10.Conclusion:

The freelancer hub platform works with databases, capable of supporting real-time interaction, high-volume transactions, and global operations. A combination of relational and NoSQL databases enables the platform to deal both with structured and unstructured data. The proper use of caching, replication, and load balancing ensure that the performance and scalability demands of the users are met. As the site expands in size, the database will continue to require upgrading so that it can handle massive volumes of data as well as real-time interactions of users.