1. Group4 - Phase 2

1.1. Background Info

1.1.1. Title

Khoury Bidding System - Freelancer Auction

1.1.2. Name

SkillMatchPro

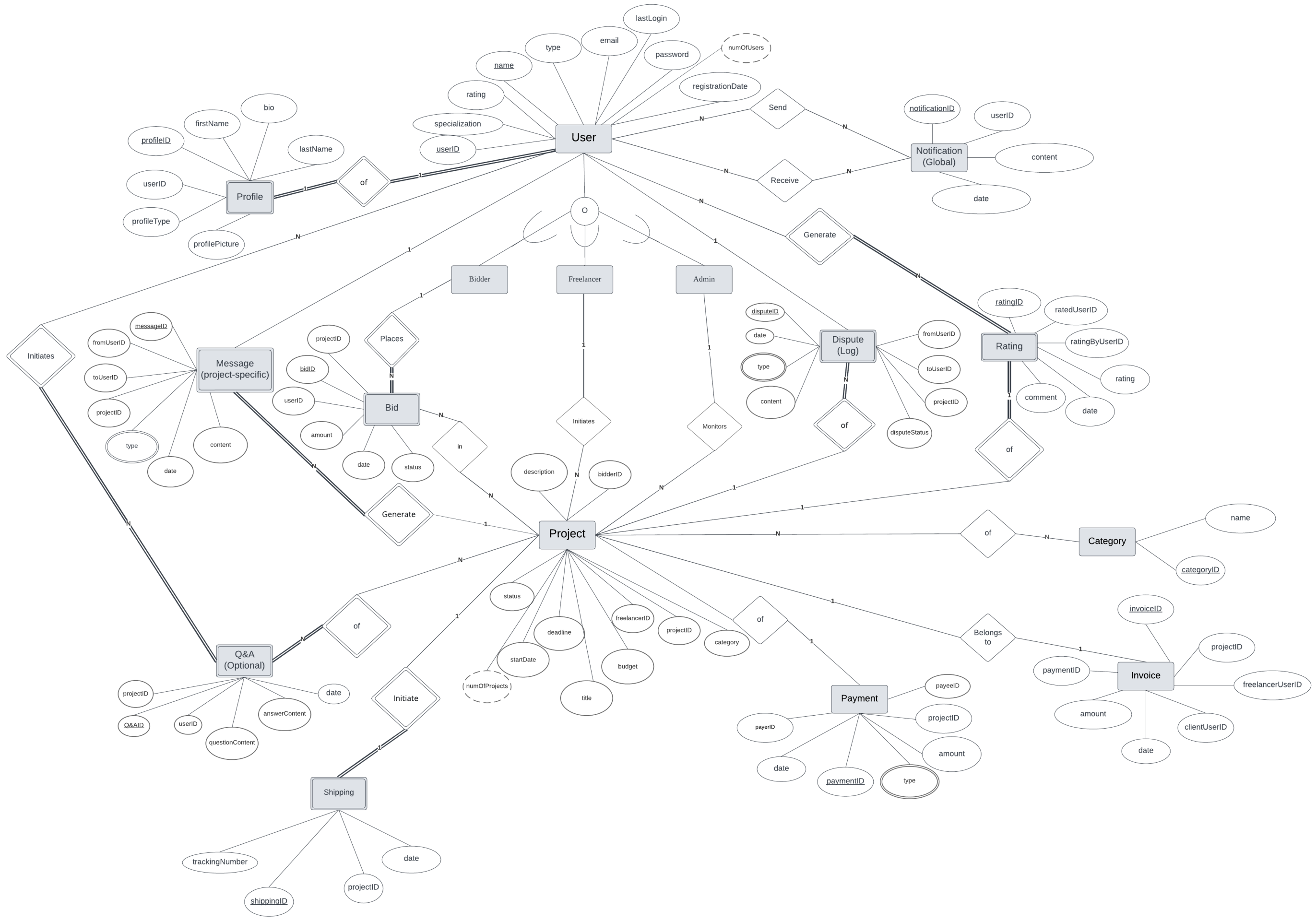
1.1.3. Member

Komal Upadhyay, Siying Lu, Yue Peng, Xinyao Chen

1.1.4. Abstraction (Brief Description)

This bidding system is a dynamic and user-friendly online platform designed to facilitate seamless interactions between clients and freelancers. It acts as a bridge connecting clients with specific project needs to talented freelancers who can fulfill those requirements. Built on a robust technology stack featuring Django with Python, MySQL hosted on Google Cloud Platform (GCP), our web-based application provides a secure and efficient environment for project bidding and management.

1.2. Step 5 EER Diagram



1.3. Step 6 Creating Relations

1.3.1. Database Tables

- **User Table**

> - **Primary Key**: userID
> - **Foreign Keys**:
>   - None

- **Project Table**

> - **Primary Key**: projectID
> - **Foreign Keys**:
>   - freelancerID (References User.userID)
>   - bidderID (References User.userID)
>   - categoryID (References Category.categoryID)

- **Bid Table**

> - **Primary Key**: bidID
> - **Foreign Keys**:
>   - userID (References User.userID)
>   - projectID (References Project.projectID)

- **Payment Table**

> - **Primary Key**: paymentID
> - **Foreign Keys**:
>   - payerID (References User.userID)
>   - payeeID (References User.userID)
>   - projectID (References Project.projectID)

- **Shipping Table**

> - **Primary Key**: shippingID
> - **Foreign Keys**:
>   - projectID (References Project.projectID)

- **Rating Table**

> - **Primary Key**: ratingID
> - **Foreign Keys**:
>   - ratedUserID (References User.userID)
>   - ratingByUserID (References User.userID)

- **Message Table**

> - **Primary Key**: messageID
> - **Foreign Keys**:
>   - fromUserID (References User.userID)
>   - toUserID (References User.userID)
>   - projectID (References Project.projectID)

- **Notification Table**

> - **Primary Key**: notificationID
> - **Foreign Keys**:
>   - userID (References User.userID)

Category Table

> - **Primary Key**: categoryID
> - **Foreign Keys**:
>   - None

Invoice Table

> - **Primary Key**: invoiceID
> - **Foreign Keys**:
>   - projectID (References Project.projectID)
>   - clientUserID (References User.userID)
>   - freelancerUserID (References User.userID)
>   - paymentID (References Payment.paymentID)

Profile Table

> - **Primary Key**: profileID
> - **Foreign Keys**:
>   - userID (References User.userID)

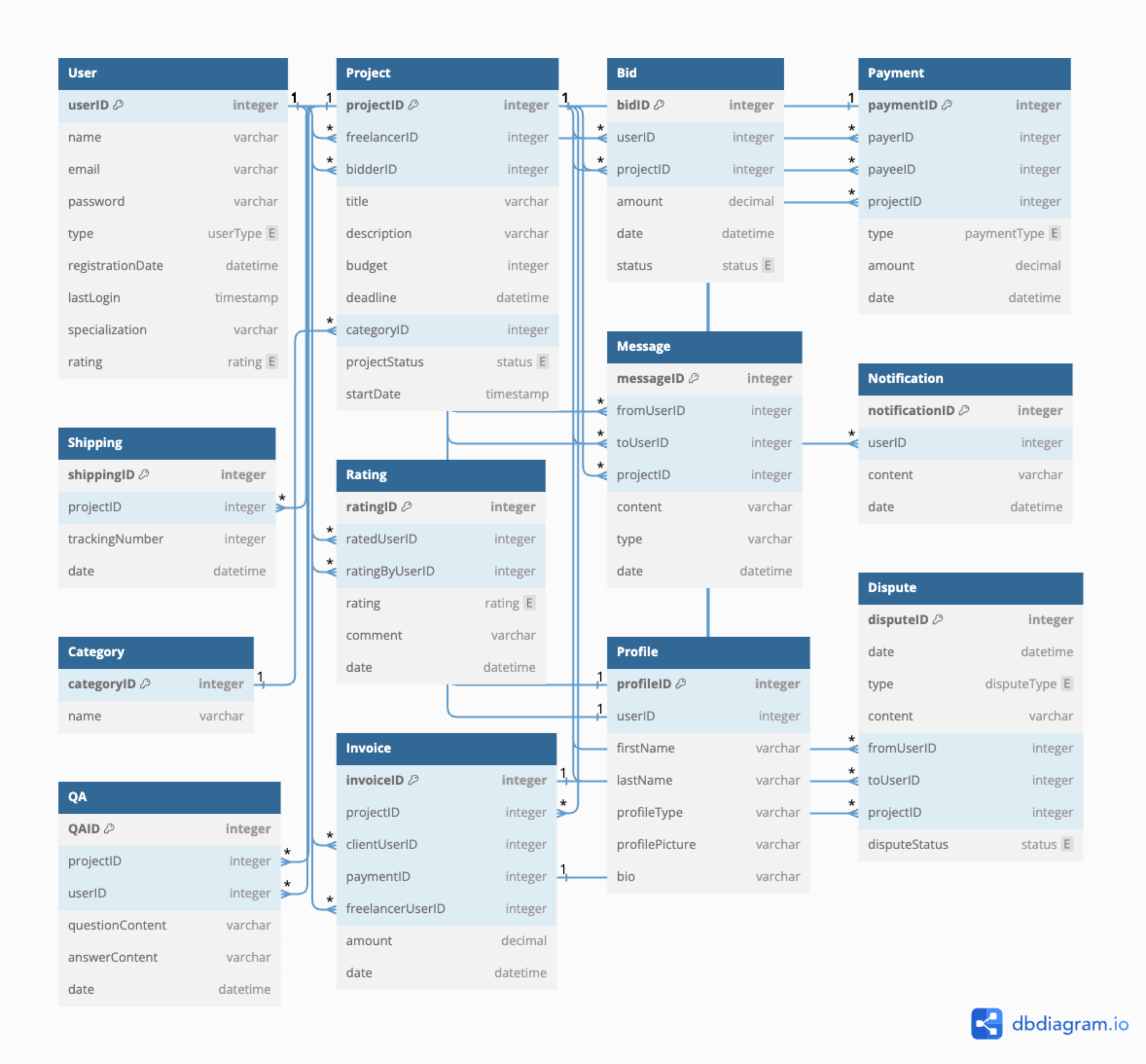
Dispute Table

> - **Primary Key**: disputeID
> - **Foreign Keys**:
>   - fromUserID (References User.userID)
>   - toUserID (References User.userID)
>   - projectID (References Project.projectID)

QA Table

> - **Primary Key**: QAID
> - **Foreign Keys**:
>   - projectID (References Project.projectID)
>   - userID (References User.userID)

1.3.2. Relational Diagram



User	
userID	**integer**
name	varchar
email	varchar
password	varchar
type	userType E
registrationDate	datetime
lastLogin	timestamp
specialization	varchar
rating	rating E

Project	
projectID	**integer**
freelancerID	integer
bidderID	integer
title	varchar
description	varchar
budget	integer
deadline	datetime
categoryID	integer
projectStatus	status E
startDate	timestamp

Bid	
bidID	**integer**
userID	integer
projectID	integer
amount	decimal
date	datetime
status	status E

Payment	
paymentID	**integer**
payerID	integer
payeeID	integer
projectID	integer
type	paymentType E
amount	decimal
date	datetime

Message	
messageID	**integer**
fromUserID	integer
toUserID	integer
projectID	integer
content	varchar
type	varchar
date	datetime

Notification	
notificationID	**integer**
userID	integer
content	varchar
date	datetime

Shipping	
shippingID	**integer**
projectID	integer
trackingNumber	integer
date	datetime

Rating	
ratingID	**integer**
ratedUserID	integer
ratingByUserID	integer
rating	rating E
comment	varchar
date	datetime

Dispute	
disputeID	**integer**
date	datetime
type	disputeType E
content	varchar
fromUserID	integer
toUserID	integer
projectID	integer
disputeStatus	status E

Category	
categoryID	**integer**
name	varchar

Profile	
profileID	**integer**
userID	integer
firstName	varchar
lastName	varchar
profileType	varchar
profilePicture	varchar
bio	varchar

QA	
QAID	**integer**
projectID	integer
userID	integer
questionContent	varchar
answerContent	varchar
date	datetime

Invoice	
invoiceID	**integer**
projectID	integer
clientUserID	integer
paymentID	integer
freelancerUserID	integer
amount	decimal
date	datetime

1.3.3. Enums Used

userType:

> - admin
> - freelancer
> - bidder

status:

> - active
> - in_progress
> - completed
> - rejected
> - awaiting_bids

paymentType:

> - offline

> - online

rating:

> - 1
> - 2
> - 3
> - 4
> - 5

disputeType:

> - payment_issue
> - service_not_provided
> - low

1.4. Step 7 Basic Queries

1.4.1. Retrieve all active projects currently available on the platform.

· **Relational Algebra:**

$$\pi_*(\sigma_{\text{projectStatus} = \text{'awaiting\_bids'}}(\text{Project})) \tag{1}$$

· **SQL:**

```sql
SELECT *
FROM Project
WHERE projectStatus = 'awaiting_bids';
```

1.4.2. Find the total number of projects in each category.

· **Relational Algebra:**

$$\pi_{category\mathcal{F}(\text{COUNT(*)})}(Project) \tag{2}$$

· **SQL:**

```sql
SELECT category, COUNT(*)
FROM Project
GROUP BY category;
```

1.4.3. List all projects that have a budget greater than $1,000.

· **Relational Algebra:**

$$\pi_*(\sigma_{\text{budget} > 1000}(\text{Project})) \tag{3}$$

· **SQL:**

```sql
SELECT *
FROM Project
WHERE budget > 1000;
```

1.4.4. Show the TOP 5 projects that have received the highest number of bids.

· **Relational Algebra:**

$$\begin{gathered}\pi_{\text{projectID, title, NumberOfBids}}(\\ \text{Order}_{\text{NumberOfBids DESC, limit 5}}(\\ \mathcal{F}_{\text{projectID, title, COUNT(b.bidId) AS NumberOfBids}}(\\ \text{Project}(\bowtie_{\text{projectID = b.projectID}} \text{Bid})))\end{gathered} \tag{4}$$

· **SQL:**

```sql
SELECT p.projectID, p.title, COUNT(b.bidId) AS NumberOfBids
FROM Project p
LEFT JOIN Bid b ON p.projectID = b.projectID
GROUP BY p.projectID, p.title
ORDER BY NumberOfBids DESC
LIMIT 5;
```

1.4.5. Identify the freelancers with the highest average project completion ratings.

· **Relational Algebra:**

$$\pi_{\text{userId,name,AverageRating}}\left(\sigma_{\text{userType}='freelancer'}\left(\mathcal{F}_{\rho_{\text{userId}\to u.userId,\text{name}\to u.name}}(u) \bowtie (u.userId = r.ratedUserId)\rho_{\text{userId}\to u.userId}(r)\right)\right) \tag{5}$$

· **SQL:**

```sql
SELECT u.userID, u.name, AVG(r.rating) AS AverageRating
FROM User u
INNER JOIN Rating r ON u.userID = r.ratedUserID
WHERE u.Type = 'freelancer'
GROUP BY u.userID, u.name
ORDER BY AverageRating DESC
LIMIT 1;
```

1.4.6. Calculate the average budget for projects in the Specific 'Web Development' category.

· **Relational Algebra:**

$$\begin{gathered}\pi_{\text{category, average\_budget}}(\\ (\mathcal{F}_{\text{category, AVG(budget) AS average\_budget}}(\\ (\sigma_{\text{category = 'specific\_category'}}(\text{Project}))))\end{gathered} \tag{6}$$

· **SQL:**

```sql
SELECT category, AVG(budget) AS average_budget
FROM Project
WHERE category = 'specific_category'
GROUP BY category;
```

1.4.7. Find projects that are due to be completed within the next seven days.

· **Relational Algebra:**

$$\pi_*(\sigma_{\text{deadline <= DATE\_ADD(NOW(), INTERVAL 7 DAY)}}(\text{Project})) \tag{7}$$

- **SQL:**

```sql
SELECT *
FROM Project
WHERE deadline <= DATE_ADD(NOW(), INTERVAL 7 DAY);
```

1.4.8. Display all projects posted by a specific client.

- **Relational Algebra:**

$$\pi_*(\sigma_{\text{p.freelancerID} = \text{'specific\_Client\_ID'}}(\text{Project})) \tag{8}$$

- **SQL:**

```sql
SELECT *
FROM Project AS p
WHERE p.freelancerID = 'specific_client_id';
```

1.4.9. List the projects that are currently in progress.

- **Relational Algebra:**

$$\pi_*(\sigma_{\text{projectStatus} = \text{'in\_progress'}}(\text{Project})) \tag{9}$$

- **SQL:**

```sql
SELECT *
FROM Project
WHERE projectStatus = 'in_progress';
```

1.4.10. Find freelancers who specialize in a particular category (e.g., web development)

- **Relational Algebra:**

$$\sigma_{\text{specialization} = \text{'specific\_category'}}(\text{User}) \tag{10}$$

- **SQL:**

```sql
SELECT *
FROM User
WHERE specialization = 'specific_category';
```

1.4.11. Retrieve the total amount paid by a specific client for completed projects.

- **Relational Algebra:**

$$\begin{gathered}\mathcal{F}_{\text{SUM(payment.amount)}}(\\ (\sigma_{\text{Payment.payerID} = \text{'specific\_client\_id' AND Project.status} = \text{'completed'}} \\ ((\text{Payment} \bowtie_{\text{Payment.projectID} = \text{Project.projectID}} \text{Project})))\end{gathered} \tag{11}$$

- **SQL:**

```sql
SELECT SUM(payment.amount)
FROM Payment
INNER JOIN Project ON Payment.projectID = Project.projectID
WHERE Payment.payerID = 'specific_client_id'
AND Project.status = 'completed';
```

1.4.12. Show all projects that have not received any bids.

- **Relational Algebra:**

$$\pi_{*}(\sigma_{\text{NOT EXISTS}(\pi_{(\sigma_{\text{projectID} = \text{p.projectID}}(\text{Bid})))}}(\text{Project as p})) \tag{12}$$

- **SQL:**

```sql
SELECT *
FROM Project AS p
WHERE NOT EXISTS (
    SELECT 1
    FROM Bid AS b
    WHERE b.projectID = p.projectID
);
```

1.4.13. Identify the clients with the highest number of completed projects.

- **Relational Algebra:**

$$\begin{gathered}\pi_{\text{userID,completed\_projects\_count}}\\((\text{Order}_{\text{completed\_projects\_count DESC}}\\((\mathcal{F}_{\text{userID, COUNT(projectID) AS completed\_projects\_count}}\\((\sigma_{\text{status} = \text{'completed'}}(\text{Project}))))\end{gathered} \tag{13}$$

- **SQL:**

```sql
SELECT userID, COUNT(projectID) AS completed_projects_count
FROM Project
WHERE status = 'completed'
GROUP BY userID
ORDER BY completed_projects_count DESC;
```

1.4.14. Retrieve all messages related to a specific project.

- **Relational Algebra:**

$$\pi_{*}(\sigma_{\text{projectID} = \text{'specific\_project\_id'}}(\text{Message})) \tag{14}$$

- **SQL:**

```sql
SELECT *
FROM Message
WHERE projectID = 'specific_project_id';
```

1.4.15. List all users who have not logged in for the past month

- **Relational Algebra:**

$$\sigma_{\text{lastLogin} \leq \text{NOW()} - \text{INTERVAL 1 MONTH}}(\text{User}) \tag{15}$$

- **SQL:**

```sql
SELECT *
FROM User
WHERE lastLogin <= NOW() - INTERVAL 1 MONTH;
```