ASSIGNMENT - 5

DATABASE MANAGEMENT SYSTEM

SIDDARTH.P

192324264

Freelancer Marketplace System

Description

The **Freelancer Marketplace System** (FMS) is designed to manage freelancer profiles, client projects, bids, payments, and reviews. The database will handle key functionalities such as project posting, bid management, payment processing, and reviewing system. It ensures the tracking of deadlines, payment milestones, and review limits, along with automatic updates based on bid acceptance and payment completion. The system will also offer reporting capabilities for freelancer performance, popular skill sets, project success rates, and platform revenue.

Database Tables Design

1. Freelancers

Stores freelancer details including their skills and work history.

Column	Data Type	Description
freelancer_id	INT (PK)	Unique identifier for each freelancer
first_name	VARCHAR(50)	Freelancer's first name
last_name	VARCHAR(50)	Freelancer's last name
email	VARCHAR(100)	Freelancer's email address
phone_number	VARCHAR(15)	Freelancer's contact number
bio	TEXT	Freelancer's biography
hourly_rate	DECIMAL(10,2)	Freelancer's hourly rate
skills	VARCHAR(255)	Skills or expertise areas of the freelancer

2. Clients

Stores client details who post projects.

Column	Data Type	Description
client_id	INT (PK)	Unique identifier for each client
first_name	VARCHAR(50)	Client's first name
last_name	VARCHAR(50)	Client's last name
email	VARCHAR(100)	Client's email address
phone_number	VARCHAR(15)	Client's contact number

3. Projects

Tracks projects posted by clients.

Column	Data Type	Description
project_id	INT (PK)	Unique identifier for each project
client_id	INT (FK)	References Clients(client_id)
project_title	VARCHAR(100)	Title of the project
project_description	TEXT	Detailed description of the project
budget	DECIMAL(10,2)	Total budget for the project
deadline	DATETIME	Project submission deadline
status	VARCHAR(20)	Current status (e.g., Open, In Progress, Completed, Closed)

4. Bids

Records bids placed by freelancers for specific projects.

Column	Data Type	Description
bid_id	INT (PK)	Unique identifier for each bid
freelancer_id	INT (FK)	References Freelancers(freelancer_id)
project_id	INT (FK)	References Projects(project_id)
bid_amount	DECIMAL(10,2)	Amount bid by freelancer
bid_date	DATETIME	Date when the bid was placed
bid_status	VARCHAR(20)	Status of the bid (e.g., Pending, Accepted, Rejected)

5. Payments

Manages payments made to freelancers upon project completion or milestone achievement.

Column	Data Type	Description
payment_id	INT (PK)	Unique identifier for each payment
freelancer_id	INT (FK)	References Freelancers(freelancer_id)
project_id	INT (FK)	References Projects(project_id)
payment_date	DATETIME	Date when the payment was made
amount	DECIMAL(10,2)	Amount paid to the freelancer
payment_status	VARCHAR(20)	Payment status (e.g., Paid, Pending)

6. Reviews

Captures feedback provided by clients to freelancers for completed projects.

Column	Data Type	Description
review_id	INT (PK)	Unique identifier for each review
freelancer_id	INT (FK)	References Freelancers(freelancer_id)
client_id	INT (FK)	References Clients(client_id)
project_id	INT (FK)	References Projects(project_id)
rating	INT	Rating (e.g., 1-5 stars)
review_text	TEXT	Detailed review or feedback
review_date	DATETIME	Date when the review was posted

Constraints for Referential Integrity

- **Foreign Keys:**
    - client_id in **Projects** references **Clients(client_id)**.
    - freelancer_id in **Bids** references **Freelancers(freelancer_id)**.
    - project_id in **Bids** references **Projects(project_id)**.
    - freelancer_id in **Payments** references **Freelancers(freelancer_id)**.
    - project_id in **Payments** references **Projects(project_id)**.
    - freelancer_id in **Reviews** references **Freelancers(freelancer_id)**.
    - client_id in **Reviews** references **Clients(client_id)**.
    - project_id in **Reviews** references **Projects(project_id)**.
- **Primary Keys:**
    - Each table includes a primary key for unique record identification.
- **Check Constraints:**
    - Prevent bid amounts that exceed project budgets.
    - Ensure reviews are only posted for completed projects.

Stored Procedures

a. Post Project

Allows a client to post a project with specific details.

```
CREATE PROCEDURE PostProject(
   IN clientId INT,
   IN projectTitle VARCHAR(100),
   IN projectDescription TEXT,
   IN budget DECIMAL(10,2),
   IN deadline DATETIME
)
BEGIN
   INSERT INTO Projects (client_id, project_title, project_description, budget, deadline, status)
   VALUES (clientId, projectTitle, projectDescription, budget, deadline, 'Open');
END;
```

b. Place Bid

Allows a freelancer to place a bid on a project.

```
CREATE PROCEDURE PlaceBid(
   IN freelancerId INT,
   IN projectId INT,
   IN bidAmount DECIMAL(10,2)
)
BEGIN
   DECLARE projectBudget DECIMAL(10,2);

   -- Get project budget to check if the bid is within the budget
```

```
    SELECT budget INTO projectBudget FROM Projects WHERE project_id = projectId;

    IF bidAmount <= projectBudget THEN
        INSERT INTO Bids (freelancer_id, project_id, bid_amount, bid_date, bid_status)
        VALUES (freelancerId, projectId, bidAmount, NOW(), 'Pending');
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Bid exceeds project budget';
    END IF;
END;
```

c. Process Payment

Allows for processing payments to freelancers when the project is completed.

```
CREATE PROCEDURE ProcessPayment(
    IN freelancerId INT,
    IN projectId INT,
    IN amount DECIMAL(10,2)
)
BEGIN
    -- Insert payment record
    INSERT INTO Payments (freelancer_id, project_id, payment_date, amount, payment_status)
    VALUES (freelancerId, projectId, NOW(), amount, 'Paid');

    -- Update project status to 'Completed'
    UPDATE Projects
    SET status = 'Completed'
    WHERE project_id = projectId;
END;
```

<u>Triggers</u>

a. Update Project Status on Bid Acceptance

Automatically updates the project status to "In Progress" when a bid is accepted.

```
CREATE PROCEDURE PlaceBid(
  IN freelancerId INT,
  IN projectId INT,
  IN bidAmount DECIMAL(10,2)
)
BEGIN
  DECLARE projectBudget DECIMAL(10,2);

  -- Get project budget to check if the bid is within the budget
  SELECT budget INTO projectBudget FROM Projects WHERE project_id = projectId;

  IF bidAmount <= projectBudget THEN
    INSERT INTO Bids (freelancer_id, project_id, bid_amount, bid_date, bid_status)
    VALUES (freelancerId, projectId, bidAmount, NOW(), 'Pending');
  ELSE
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Bid exceeds project budget';
  END IF;
END;
```

b. Update Payment Status on Payment Completion

Automatically updates the payment status to "Paid" once the payment is processed.

```
CREATE TRIGGER AfterPaymentInsert
AFTER INSERT ON Payments
```

```
FOR EACH ROW
BEGIN
  UPDATE Projects
  SET status = 'Completed'
  WHERE project_id = NEW.project_id;
END;
```

<u>**SQL Queries for Reports**</u>

a. Freelancer Performance Report

Generates a report of freelancer ratings and the number of completed projects.

```
SELECT f.freelancer_id, f.first_name, f.last_name, COUNT(p.project_id) AS completed_projects,
AVG(r.rating) AS average_rating
FROM Freelancers f
JOIN Payments pay ON f.freelancer_id = pay.freelancer_id
JOIN Projects p ON pay.project_id = p.project_id
JOIN Reviews r ON p.project_id = r.project_id
WHERE p.status = 'Completed'
GROUP BY f.freelancer_id;
```

b. Popular Skill Sets

Identifies the most frequently mentioned skills among freelancers.

```
SELECT skills, COUNT(*) AS skill_count
FROM Freelancers
GROUP BY skills
ORDER BY skill_count DESC;
```

c. Project Success Rate

Calculates the success rate of projects (those marked as 'Completed').

```
SELECT COUNT(project_id) AS total_projects,
    SUM(CASE WHEN status = 'Completed' THEN 1 ELSE 0 END) AS completed_projects,
    (SUM(CASE WHEN status = 'Completed' THEN 1 ELSE 0 END) / COUNT(project_id)) * 100 AS
success_rate
FROM Projects;
```

d. Platform Revenue

Generates a report on the total revenue of the platform from freelancer payments.

```
SELECT SUM(amount) AS platform_revenue FROM Payments;
```

Conclusion

The **Freelancer Marketplace System** database effectively supports the management of freelancer profiles, client projects, bids, payments, and reviews. Through constraints, stored procedures, and triggers, it ensures smooth operations by managing bid deadlines, payment milestones, and project status updates. The system provides detailed reports on freelancer performance, popular skills, project success rates, and platform revenue, enabling platform administrators to optimize operations and enhance the user experience. This design promotes efficiency and transparency in the freelancing ecosystem, benefiting both freelancers and clients.