**Project Name: FreeGency**

**Purpose** To create an online platform where:

* Clients can post projects and hire teams.
* Team Leaders can create teams, manage team members, assign tasks, and apply for projects.
* Team Members can join teams, search for jobs, and complete assigned tasks.

**Stakeholders**

1. **Clients:**
   * Post projects and hire teams.
2. **Team Leaders:**
   * Create teams, manage team members, assign tasks, and apply for projects.
3. **Team Members:**
   * Join teams, search for jobs, and complete assigned tasks.

**Current System**

* The current system is a freelance platform for individuals.
* **Limitations:**
  + No support for team collaboration.
  + Limited functionality for team formation and management.
  + No task assignment or tracking features.
  + No payment processing system for secure transactions.
  + No review system for teams or clients.

**Functional Requirements**

**For Clients**

1. **Register/Login:**
   * Clients can register and log in or continue as guests.
2. **Search for Teams:**
   * Clients can search for teams based on skills, ratings, or past projects.
3. **Post Projects:**
   * Clients can post project details (title, description, budget, timeline).
4. **Review Offers:**
   * Clients can review offers from teams and choose the best one.
5. **Payment Processing:**
   * Clients can add payment methods (credit card, bank transfer, etc.).
   * Clients can release payments to teams upon project completion.
6. **Review Teams:**
   * Clients can leave reviews and ratings for teams after project completion.

**For Team Leaders**

1. **Register/Login:**
   * Team leaders must register/login separately and create a team room.
2. **Create Team:**
   * Team leaders can create a team and generate a 16-character team code for inviting members.
3. **Invite Team Members:**
   * Team leaders can share the team code with potential members.
4. **Remove Team Members:**
   * Team leaders can remove members from the team.
5. **Manage Team Requests:**
   * Team leaders can approve or reject requests to join the team.
6. **Apply for Projects:**
   * Team leaders can apply for projects posted by clients.
7. **Assign Tasks:**
   * Team leaders can assign tasks to team members using Trello integration.
8. **Track Task Progress:**
   * Team leaders can view the status of tasks (Pending, In Progress, Completed) via Trello.
9. **Track Payment Status:**
   * Team leaders can view the status of payments from clients.
10. **Review Clients:**

* Team leaders can leave reviews and ratings for clients after project completion.

**For Team Members**

1. **Register/Login:**
   * Team members use the same registration/login flow as clients.
2. **Join Team:**
   * Team members can join a team using the team code shared by the team leader.
3. **Leave Team:**
   * Team members can leave the team they are on.
4. **Search for Jobs:**
   * Team members can search for jobs in other teams.
5. **Complete Tasks:**
   * Team members can mark tasks as completed in Trello.
6. **Your Teams Section:**
   * Once a team member joins a team, the "Your Teams" section appears in their interface.

**For Payment Gateway**

1. **Process Payment:**
   * Handles secure payment transactions.
2. **Send Payment Status:**
   * Updates the system with payment success/failure status.

**For Trello Integration**

1. **Create Trello Board:**
   * Automatically create a Trello board for each project.
2. **Sync Tasks:**
   * Sync tasks between FreeGency and Trello.
3. **Track Progress:**
   * Track task progress using Trello’s boards, lists, and cards.

**Non-Functional Requirements**

1. **Scalability:**
   * The system should handle 1,000 concurrent users.
2. **Security:**
   * Data should be encrypted for secure transactions (e.g., SSL/TLS for communication).
   * Payment data must comply with PCI DSS standards.
3. **Performance:**
   * The platform should respond within 2-3 seconds for most operations.
   * Payment transactions should be processed within 5 seconds.
4. **Usability:**
   * The interface should be intuitive and easy to use for both clients and teams.
5. **Reliability:**
   * The system should have minimal downtime (e.g., 99.9% uptime).

**Constraints**

1. **Budget:**
   * Use free tools (since it’s a graduation project).
2. **Timeline:**
   * 8 months to complete the project.
3. **Technology:**
   * Backend: .NET (for server-side logic and APIs).
   * Frontend: Flutter (for cross-platform mobile and web apps).
   * Database: SQL Server (for storing user, team, project, and payment data).
4. **Third-Party Integrations:**
   * Payment gateway (e.g., Stripe, PayPal, or Razorpay).
   * Trello for task management.
   * Email service (for sending notifications).

**Scope**

**In Scope**

1. **Core Functionality:**
   * User registration and authentication (clients, team leaders, team members).
   * Project posting and team application workflow.
   * Team creation and management.
   * Task assignment and tracking using Trello.
   * Search functionality for teams and projects.
2. **Payment Processing:**
   * Secure payment integration for clients to pay teams.
   * Payment tracking and status updates for teams.
3. **Review System:**
   * Clients can review teams.
   * Team leaders can review clients.
4. **Team Management:**
   * Team leaders can generate a 16-character team code for inviting members.
   * Team members can join teams using the team code.
   * Team leaders can remove members from the team.
   * Team members can leave the team.

**Out of Scope**

* Video call integration (will be added in future versions).
* Advanced analytics and reporting.

**Deliverables**

1. **Working Application:**
   * A fully functional platform with client, team leader, and team member interfaces.
   * Payment processing functionality.
   * Task assignment and tracking functionality using Trello.
   * Review system for clients and teams.
   * Team management features (team code, remove members, leave team).
2. **Documentation:**
   * Software Requirements Specification (SRS).
   * System Design Specification (SDS).
   * User manuals for clients, team leaders, and team members.
3. **Testing Reports:**
   * Unit testing, integration testing, and user acceptance testing (UAT) results.
   * Payment gateway integration testing.
   * Trello integration testing.

**Database Schema**

-- Create the FreeGency Database

CREATE DATABASE FreeGency;

GO

-- Use the FreeGency Database

USE FreeGency;

GO

-- Create Users Table

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY(1,1),

UserName NVARCHAR(100) UNIQUE NOT NULL,

UserPassword NVARCHAR(255) NOT NULL,

Email NVARCHAR(100) UNIQUE NOT NULL,

Role NVARCHAR(50) NOT NULL, -- Client, Team Leader, Team Member

Created\_At DATETIME NOT NULL DEFAULT GETDATE(),

ProfileImageURL NVARCHAR(255),

TeamCount INT NOT NULL DEFAULT 0 -- Tracks number of teams joined

);

GO

-- Create Teams Table

CREATE TABLE Teams (

TeamID INT PRIMARY KEY IDENTITY(1,1),

TeamName NVARCHAR(100) NOT NULL,

LeaderID INT NOT NULL,

TeamCode NVARCHAR(16) UNIQUE NOT NULL, -- 16-character team code

Description NVARCHAR(255),

Created\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (LeaderID) REFERENCES Users(UserID)

);

GO

-- Create TeamMembers Table

CREATE TABLE TeamMembers (

TeamID INT NOT NULL,

UserID INT NOT NULL,

Status NVARCHAR(50) NOT NULL DEFAULT 'Active', -- Active, Removed, Left

PRIMARY KEY (TeamID, UserID),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID),

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

GO

-- Create Projects Table

CREATE TABLE Projects (

ProjectID INT PRIMARY KEY IDENTITY(1,1),

ClientID INT NOT NULL,

Title NVARCHAR(100) NOT NULL,

Description NVARCHAR(255),

Budget DECIMAL(10, 2),

Status NVARCHAR(50) NOT NULL, -- Open, In Progress, Completed

TrelloBoardID NVARCHAR(100), -- Trello Board ID for task management

Created\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (ClientID) REFERENCES Users(UserID)

);

GO

-- Create Applications Table

CREATE TABLE Applications (

ApplicationID INT PRIMARY KEY IDENTITY(1,1),

TeamID INT NOT NULL,

ProjectID INT NOT NULL,

Status NVARCHAR(50) NOT NULL, -- Pending, Accepted, Rejected

Applied\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID),

FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)

);

GO

-- Create JobAnnouncements Table

CREATE TABLE JobAnnouncements (

JobID INT PRIMARY KEY IDENTITY(1,1),

TeamID INT NOT NULL,

Title NVARCHAR(100) NOT NULL,

Description NVARCHAR(255),

Posted\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)

);

GO

-- Create JobApplications Table

CREATE TABLE JobApplications (

JobApplicationID INT PRIMARY KEY IDENTITY(1,1),

JobID INT NOT NULL,

UserID INT NOT NULL,

CV\_URL NVARCHAR(255) NOT NULL,

Applied\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (JobID) REFERENCES JobAnnouncements(JobID),

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

GO

-- Create Payments Table

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY IDENTITY(1,1),

ClientID INT NOT NULL,

TeamID INT NOT NULL,

Amount DECIMAL(10, 2) NOT NULL,

Status NVARCHAR(50) NOT NULL, -- Pending, Completed, Failed

Timestamp DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (ClientID) REFERENCES Users(UserID),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)

);

GO

-- Create Transactions Table

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY IDENTITY(1,1),

PaymentID INT NOT NULL,

Type NVARCHAR(50) NOT NULL, -- Credit, Debit

Amount DECIMAL(10, 2) NOT NULL,

Status NVARCHAR(50) NOT NULL, -- Success, Failed

Timestamp DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (PaymentID) REFERENCES Payments(PaymentID)

);

GO

-- Create Reviews Table

CREATE TABLE Reviews (

ReviewID INT PRIMARY KEY IDENTITY(1,1),

ReviewerID INT NOT NULL, -- UserID of the reviewer (Client or Team Leader)

TeamID INT NOT NULL, -- Team being reviewed

Rating INT NOT NULL CHECK (Rating BETWEEN 1 AND 5),

Comment NVARCHAR(255),

Created\_At DATETIME NOT NULL DEFAULT GETDATE(),

FOREIGN KEY (ReviewerID) REFERENCES Users(UserID),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)

);

GO

-- Create Indexes for Faster Queries

CREATE INDEX idx\_Users\_Email ON Users(Email);

CREATE INDEX idx\_Teams\_TeamCode ON Teams(TeamCode);

CREATE INDEX idx\_Projects\_ClientID ON Projects(ClientID);

CREATE INDEX idx\_Applications\_TeamID ON Applications(TeamID);

CREATE INDEX idx\_JobApplications\_UserID ON JobApplications(UserID);

CREATE INDEX idx\_Payments\_ClientID ON Payments(ClientID);

CREATE INDEX idx\_Reviews\_TeamID ON Reviews(TeamID);

GO

-- Print Success Message

PRINT 'FreeGency database and tables created successfully!';

GO