FreeGency Project: RESTful API Endpoints

---

1. User Management

1.1 Register User

- **Endpoint**: POST /api/users/register
- **Request Body**:

```
{
    "UserName": "john_doe",
    "UserPassword": "password123",
    "Email": "john@example.com",
    "Role": "Client" // or "Team Leader"
}
```

- **Process Logic**:
    1. Validate the input fields (e.g., email format, password strength).
    2. Check if the email is already registered.
    3. Hash the password using bcrypt.
    4. Insert the user into the Users table.
    5. Return a success message and the UserID.
- **Response**:

```
{
    "UserID": 1,
    "Message": "User registered successfully"
}
```

1.2 Login User

- **Endpoint**: POST /api/users/login
- **Request Body**:

```
{
    "Email": "john@example.com",
    "UserPassword": "password123"
}
```

- **Process Logic**:
    1. Validate the email and password.
    2. Check if the email exists in the Users table.
    3. Verify the password against the hashed password in the database.
    4. Generate a JWT (JSON Web Token) for authentication.
    5. Return the JWT and user details.
- **Response**:

```
{
    "Token": "jwt_token",
    "UserID": 1,
```

```
    "Role": "Client"
}
```

---

2. Team Management

2.1 Create Team (Team Leader Only)

- **Endpoint**: POST /api/teams
- **Request Body**:

```
{
    "TeamName": "Dev Team",
    "LeaderID": 1,
    "Description": "A team of developers"
}
```

- **Process Logic**:
    1. Validate the input fields.
    2. Generate a unique 16-character team code.
    3. Insert the team into the Teams table.
    4. Return the team code and success message.
- **Response**:

```
{
    "TeamID": 1,
    "TeamCode": "ABCD1234EFGH5678",
    "Message": "Team created successfully"
}
```

2.2 Join Team (Team Member)

- **Endpoint**: POST /api/teams/join
- **Request Body**:

```
{
    "TeamCode": "ABCD1234EFGH5678",
    "UserID": 2
}
```

- **Process Logic**:
    1. Validate the team code and user ID.
    2. Check if the user has already joined 3 teams (TeamCount >= 3).
    3. Insert the user into the TeamMembers table.
    4. Increment the TeamCount in the Users table.
    5. Return a success message.
- **Response**:

```
{
    "TeamID": 1,
    "Message": "User joined the team successfully"
}
```

2.3 Leave Team (Team Member)

- **Endpoint**: DELETE /api/teams/leave
- **Request Body**:

```
{
    "TeamID": 1,
    "UserID": 2
}
```

- **Process Logic**:
    1. Validate the team ID and user ID.
    2. Remove the user from the TeamMembers table.
    3. Decrement the TeamCount in the Users table.
    4. Return a success message.
- **Response**:

```
{
    "Message": "User left the team successfully"
}
```

---

3. Project Management

3.1 Post Project (Client)

- **Endpoint**: POST /api/projects
- **Request Body**:

```
{
    "ClientID": 1,
    "Title": "Website Development",
    "Description": "Build a website",
    "Budget": 1000
}
```

- **Process Logic**:
    1. Validate the input fields.
    2. Insert the project into the Projects table.
    3. Create a Trello board for the project.
    4. Return the project ID and Trello board ID.
- **Response**:

```
{
    "ProjectID": 1,
    "TrelloBoardID": "BOARD_ID",
    "Message": "Project posted successfully"
}
```

3.2 Apply for Project (Team Leader)

- **Endpoint**: POST /api/projects/apply
- **Request Body**:

```
{
    "TeamID": 1,
    "ProjectID": 1
}
```

- **Process Logic**:
    1. Validate the team ID and project ID.
    2. Insert the application into the Applications table.
    3. Return a success message.
- **Response**:

```
{
    "ApplicationID": 1,
    "Message": "Application submitted successfully"
}
```

---

4. Task Management (Trello Integration)

4.1 Create Trello Board

- **Endpoint**: POST /api/projects/{ProjectID}/create-trello-board
- **Process Logic**:
    1. Call the Trello API to create a new board.
    2. Store the Trello board ID in the Projects table.
    3. Return the Trello board ID.
- **Response**:

```
{
    "TrelloBoardID": "BOARD_ID",
    "Message": "Trello board created successfully"
}
```

4.2 Sync Tasks

- **Endpoint**: POST /api/projects/{ProjectID}/sync-tasks
- **Process Logic**:
    1. Fetch tasks from the Trello board using the Trello API.
    2. Sync tasks with the FreeGency database.
    3. Return a success message.
- **Response**:

```
{
    "Message": "Tasks synced with Trello successfully"
}
```

---

5. Payment Management

5.1 Make Payment (Client)

- **Endpoint**: POST /api/payments

- **Request Body**:

```
{
    "ClientID": 1,
    "TeamID": 1,
    "Amount": 500
}
```

- **Process Logic**:
    1. Validate the client ID, team ID, and amount.
    2. Call the payment gateway API to process the payment.
    3. Insert the payment into the Payments table.
    4. Return a success message.
- **Response**:

```
{
    "PaymentID": 1,
    "Message": "Payment initiated successfully"
}
```

5.2 Track Payment Status

- **Endpoint**: GET /api/payments/{PaymentID}
- **Process Logic**:
    1. Fetch the payment status from the Payments table.
    2. Return the payment details.
- **Response**:

```
{
    "PaymentID": 1,
    "Status": "Completed",
    "Timestamp": "2023-10-01T12:00:00Z"
}
```

---

6. Review Management

6.1 Leave Review

- **Endpoint**: POST /api/reviews
- **Request Body**:

```
{
    "ReviewerID": 1,
    "TeamID": 1,
    "Rating": 5,
    "Comment": "Great team!"
}
```

- **Process Logic**:
    1. Validate the reviewer ID, team ID, and rating.
    2. Insert the review into the Reviews table.
    3. Return a success message.

- **Response**:

```
{
    "ReviewID": 1,
    "Message": "Review submitted successfully"
}
```

6.2 Get Reviews

- **Endpoint**: GET /api/reviews/{TeamID}
- **Process Logic**:
    1. Fetch all reviews for the specified team from the Reviews table.
    2. Return the list of reviews.
- **Response**:

```
[
    {
        "ReviewID": 1,
        "ReviewerID": 1,
        "Rating": 5,
        "Comment": "Great team!",
        "Created_At": "2023-10-01T12:00:00Z"
    }
]
```