# Software Requirements Specification (SRS) for FreeGency

## 1. Introduction

### 1.1 Purpose

FreeGency is an online platform designed to facilitate collaboration between clients and teams. The platform allows:

* **Team Leaders** to register and create/manage teams.
* **Normal Users** to register either as clients (to post projects) or as potential team members (to search for jobs and join teams).
* Once a normal user joins a team, they become a **Team Member** and gain access to a dedicated “My Teams” section for team interactions, including group chats and task management.

### 1.2 Scope

The project covers the core functionality for user registration, project posting, team formation, job searches, team chats, task assignments via Trello integration, and secure payment processing. Advanced features such as video calling and advanced analytics are out of scope for this phase.

### 1.3 Definitions and Abbreviations

* **Team Leader:** A user who registers with the intention of creating and managing a team.
* **Normal User:** A user who registers without the intent to lead a team. Normal users can act as clients (by posting projects) or later become team members if they join a team.
* **Team Member:** A normal user who has joined a team using a provided team code.
* **Trello Integration:** The connection with Trello to manage tasks, track progress, and synchronize task-related data.
* **Payment Gateway:** A third-party integration (e.g., Stripe, PayPal, or Razorpay) to securely handle transactions between clients and teams.

## 2. System Overview

FreeGency offers two primary user journeys:

1. **Team Leader Journey:** Registration as a team leader requires immediate team creation. Team leaders manage team composition, post jobs, assign tasks (via Trello), and apply for projects posted by clients.
2. **Normal User Journey:**
   * **As a Client:** They can post projects and hire teams based on proposals.
   * **As a Job Seeker:** They have access to a “Find a Job” section to view and apply for jobs posted by team leaders. If invited or upon applying, they may join an existing team via a team code, thereby becoming a team member.
3. **Team Member Functionality:** Normal users who have joined one or more teams (up to three teams maximum) gain access to a “My Teams” section that includes:
   * A list of teams they are part of.
   * A chat interface for real-time communication with other team members.
   * A Tasks section showing the assignments given by the team leader.

## 3. Stakeholders

1. **Normal Users (Clients/Job Seekers):**
   * **As Clients:** Can post projects, review team proposals, process payments, and rate teams.
   * **As Job Seekers:** Can search for job opportunities, apply for positions posted by team leaders, or join teams via a team code.
2. **Team Leaders:**
   * Must register as team leaders and are required to create a team immediately after registration.
   * Can invite normal users to join their teams using a generated 16-character team code.
   * Manage team membership, post jobs, assign tasks, and review project/payment statuses.
3. **Team Members:**
   * Normal users who join a team.
   * Gain access to a dedicated “My Teams” section with team chats and task tracking features.

## 4. Functional Requirements

### 4.1 Common Functionality (All Users)

* **Registration/Login:**
  + Users can choose to register as a **Team Leader** or as a **Normal User**.
  + All users can log in, and some features may be accessible for guests (where applicable).
  + Once registered as a normal user, if the user is later invited to a team (or chooses to join one), additional team member functionalities become available.

### 4.2 Normal User Functionality

#### 4.2.1 As a Client

* **Project Posting:**
  + Post project details including title, description, budget, and timeline.
* **Review Offers:**
  + View and evaluate proposals submitted by teams.
* **Payment Processing:**
  + Add and manage payment methods (credit card, bank transfer, etc.).
  + Release payments to the selected team upon project completion.
* **Review Teams:**
  + Leave reviews and ratings for teams after project completion.

#### 4.2.2 As a Job Seeker

* **Find a Job Section:**
  + Browse job listings and projects posted by team leaders.
  + Apply for jobs or express interest in joining a team.
* **Team Joining:**
  + Join an existing team by entering a team code provided by a team leader.

#### 4.2.3 Transition to Team Member

* **My Teams Section (Accessible After Joining a Team):**
  + **Team Listing:**
    - View all teams they are a member of (up to a maximum of 3 teams).
  + **Team Chat:**
    - Click on a team to open a chat interface for communication with other team members.
  + **Tasks Overview:**
    - Access tasks assigned by the team leader.
  + **Leave Team:**
    - Option to leave a team at any time.

### 4.3 Team Leader Functionality

* **Registration/Login:**
  + Register as a team leader and create a team immediately upon registration.
* **Team Creation:**
  + Create a team and automatically generate a 16-character team code for invitations.
* **Team Management:**
  + Invite normal users to join the team by sharing the team code.
  + Manage incoming join requests (approve or reject).
  + Remove team members when necessary.
* **Job Posting (Optional):**
  + Post jobs or positions that normal user (job seekers) can apply for.
* **Project Application:**
  + Apply to projects posted by normal users acting as clients.
* **Task Assignment:**
  + Assign tasks to team members using Trello integration.
* **Progress Tracking:**
  + Monitor task progress (Pending, In Progress, Completed) through Trello.
* **Payment Status Monitoring:**
  + View and track the status of payments released by clients.
* **Review Clients:**
  + After project completion, rate and review clients.

### 4.4 Payment Gateway Integration

* **Secure Payment Processing:**
  + Handle secure transactions between clients and teams.
* **Status Updates:**
  + Update the system with the success or failure status of payment transactions.
* **Compliance:**
  + Ensure payment data complies with PCI DSS standards.

### 4.5 Trello Integration

* **Board Creation:**
  + Automatically create a Trello board for each new project or team task board.
* **Task Synchronization:**
  + Sync tasks between FreeGency and Trello, reflecting updates in real time.
* **Progress Tracking:**
  + Track task status using Trello’s boards, lists, and cards.

## 5. Non-Functional Requirements

* **Scalability:**
  + The system should support up to 1,000 concurrent users.
* **Security:**
  + Data encryption (SSL/TLS) must be implemented for secure communication.
  + Payment data must be stored and processed in compliance with PCI DSS.
* **Performance:**
  + The platform should respond within 2–3 seconds for most operations.
  + Payment transactions should be processed within 5 seconds.
* **Usability:**
  + The user interface should be intuitive and easy to navigate for all user types.
* **Reliability:**
  + The system must aim for 99.9% uptime with minimal downtime.

## 6. Constraints

* **Budget:**
  + Utilize free and open-source tools as much as possible (considering it is a graduation project).
* **Timeline:**
  + The project is to be completed within 8 months.
* **Technology Stack:**
  + **Backend:** NodeJS (for server-side logic and APIs)
  + **Frontend:** Flutter (for cross-platform mobile and web applications)
  + **Database:** MongoDB (a NoSQL, document-oriented database for flexible and scalable data storage)
* **Third-Party Integrations:**
  + Payment gateway (e.g., Stripe, PayPal, or Razorpay)
  + Trello for task management
  + Email service for sending notifications

## 7. Project Scope

### 7.1 In Scope

* **Core Functionality:**
  + User registration and authentication for team leaders and normal users.
  + Project posting and team application workflow.
  + Team creation, management, and joining functionalities.
  + “Find a Job” section for normal users to explore opportunities.
  + “My Teams” section for team members, including team chat and task management.
* **Payment Processing:**
  + Secure integration for handling client payments and tracking payment statuses.
* **Review System:**
  + Clients can review teams.
  + Team leaders can review clients.
* **Task Assignment & Tracking:**
  + Integration with Trello for task creation, assignment, synchronization, and progress tracking.

### 7.2 Out of Scope in this phase

* Video call integration (to be considered for future versions).
* Advanced analytics and reporting functionalities.

## 8. Deliverables

1. **Working Application:**
   * A fully functional platform featuring distinct interfaces for normal users (clients/job seekers), team leaders, and team members.
   * Integrated payment processing and Trello-based task management.
   * Team management features including team code generation, member invitations, and real-time chat within the “My Teams” section.
2. **Documentation:**
   * Software Requirements Specification (SRS)
   * System Design Specification (SDS)
   * User manuals for all user roles (normal users, team leaders, and team members)
3. **Testing Reports:**
   * Unit, integration, and user acceptance testing (UAT) reports.
   * Detailed test reports for payment gateway and Trello integrations.

## 9. Deliverables

## 1. Users Collection

### Purpose

Store all user-related information. Since a “normal user” can later become a team member or act as a client and a team leader is a specific type of user, you can use a role field or even a discriminator if needed.

### Possible Fields

* **\_id:** ObjectId (unique identifier)
* **username:** String
* **email:** String (with a unique index)
* **passwordHash:** String
* **role:** String (e.g., "normal", "teamLeader")
* **profileDetails:** Object (this can include additional info like bio, skills, etc.)
* **teamMemberships:** Array of team IDs (for users who are part of teams)
* **createdAt/updatedAt:** Date

### Considerations

* **Indexing:** Make sure to index fields like email and possibly username for faster lookup.
* **Embedding vs. Referencing:** You might choose to store minimal team membership info (just IDs) in the user document and then look up the detailed team data in the Teams collection.

## 2. Teams Collection

### Purpose

Represent teams created by team leaders. This collection will hold information about the team, its unique code, leader, and members.

### Possible Fields

* **\_id:** ObjectId
* **teamName:** String
* **teamCode:** String (16-character unique code)
* **teamLeaderId:** ObjectId (reference to the Users collection)
* **members:** Array of user IDs (or even embedded objects if you need additional member info)
* **createdAt/updatedAt:** Date
* **additionalInfo:** Object (could include team description, etc.)

### Considerations

* **Relationships:** The team leader is a user, so you reference the user’s \_id. Members can be stored as an array of ObjectIds.
* **Data Duplication:** You might not need to embed full user documents here unless you always display user details with the team. Referencing is generally more flexible.

## 3. Projects Collection

### Purpose

Store projects posted by clients (normal users acting as clients). Projects will have details such as title, description, budget, timeline, and possibly an array of proposals.

### Possible Fields

* **\_id:** ObjectId
* **title:** String
* **description:** String
* **budget:** Number
* **timeline:** String or Object (depending on how you model time)
* **clientId:** ObjectId (reference to the Users collection)
* **status:** String (e.g., "open", "in progress", "completed")
* **acceptedTeamId:** ObjectId (if a team is chosen)
* **proposals:** Array (optional—more on this below)
* **createdAt/updatedAt:** Date

### Considerations

* **Proposals:**
  + **Embedded Approach:** If the number of proposals per project is small, you can embed them directly in the project document.
  + **Separate Collection:** If proposals are complex or expected to be numerous, create a separate **Proposals** collection (discussed next).

## 4. Proposals Collection (Optional)

### Purpose

If you decide not to embed proposals directly into the Projects document, a separate collection can hold each proposal submitted by teams for a project.

### Possible Fields

* **\_id:** ObjectId
* **projectId:** ObjectId (reference to the Projects collection)
* **teamId:** ObjectId (reference to the Teams collection)
* **proposalDetails:** Object (can include offered budget, timeline, etc.)
* **status:** String (e.g., "pending", "accepted", "rejected")
* **createdAt/updatedAt:** Date

### Considerations

* This approach is useful if you expect many proposals or need to query proposals independently (for instance, listing all proposals from a particular team).

## 5. Payments Collection

### Purpose

Keep track of all transactions between clients and teams.

### Possible Fields

* **\_id:** ObjectId
* **projectId:** ObjectId (reference to the Projects collection)
* **clientId:** ObjectId (reference to the Users collection)
* **teamId:** ObjectId (reference to the Teams collection)
* **amount:** Number
* **status:** String (e.g., "pending", "completed", "failed")
* **transactionDetails:** Object (could include payment method, gateway transaction ID, etc.)
* **createdAt/updatedAt:** Date

### Considerations

* **Indexing:** You might want to index projectId and status to quickly retrieve payment statuses.

## 6. Reviews Collection

### Purpose

Store reviews left by clients for teams and by team leaders for clients.

### Possible Fields

* **\_id:** ObjectId
* **reviewerId:** ObjectId (user leaving the review)
* **revieweeId:** ObjectId (user or team being reviewed)
* **entityType:** String (e.g., "team" or "client") to distinguish the review target
* **projectId:** ObjectId (if the review is tied to a specific project)
* **rating:** Number
* **comment:** String
* **createdAt:** Date

### Considerations

* Reviews could be stored in one collection with an attribute to indicate what type of entity is being reviewed or split into separate collections if the queries differ significantly.

## 7. Tasks Collection

### Purpose

Store task assignments (especially if you want to mirror some of the information from Trello). This can serve as a sync layer or a cache for tasks assigned by team leaders.

### Possible Fields

* **\_id:** ObjectId
* **teamId:** ObjectId (reference to the Teams collection)
* **projectId:** ObjectId (if tasks are project-specific)
* **assignedTo:** ObjectId (reference to the user)
* **description:** String
* **status:** String (e.g., "pending", "in progress", "completed")
* **trelloCardId:** String (if you need to link back to Trello)
* **createdAt/updatedAt:** Date

### Considerations

* Decide if tasks should be a stand-alone collection or if they can be embedded within a project or team document, based on how you plan to query and update them.

## 8. ChatMessages Collection

### Purpose

Store messages exchanged in team chats. Keeping these in a separate collection can make it easier to paginate and index chats for each team.

### Possible Fields

* **\_id:** ObjectId
* **teamId:** ObjectId (reference to the Teams collection)
* **senderId:** ObjectId (reference to the Users collection)
* **message:** String
* **attachments:** Array (optional, for file URLs or media links)
* **timestamp:** Date

### Considerations

* **Indexing:** Index on teamId and timestamp to quickly retrieve recent messages for a team chat.
* **Retention:** Consider if you need any data retention policies or archival mechanisms for older messages.