**10. Entities:**

1. **Entities diagram:**

*The following figure shows the* ***Entity Relationship Diagram (ERD)*** *for the* ***SkyRocketing*** *platform. It depicts key entities like* ***User****,* ***Task****,* ***Project****,* ***Team****,* ***Client****, and* ***CRM****, along with their attributes and relationships. This diagram illustrates how the platform's data structure supports operations such as task management, project tracking, and client relationship management.*

*A diagram of a computer network

Description automatically generated with medium confidence*

**Entity Relationship Diagram (ERD) - Detailed Explanation**

The **Entity Relationship Diagram (ERD)** for **SkyRocketing** illustrates the database structure of the application and represents how different entities interact with each other. The SkyRocketing platform is an all-in-one tool designed to help small, fast-growing startups manage their business operations, communication, resources, and projects. The ERD serves as a blueprint for how the platform will store, retrieve, and manage data.

**1. Entities and Their Attributes**

**User Entity**

The **User** entity stores all the details related to the platform users. These users may include entrepreneurs, team members, or administrators who interact with the platform.

* **userID (PK)**: A unique identifier for each user. This is a primary key (PK), ensuring that every user in the system can be uniquely identified.
* **name**: The full name of the user (e.g., "Sarah" or "David").
* **email**: The email address of the user. This field can be used for communication and login purposes.
* **role**: The role of the user (e.g., Admin, Manager, Developer). This is important because it determines the user’s access rights and what actions they can perform in the system.
* **password**: A hashed version of the user’s password, used for authentication.

**Task Entity**

The **Task** entity represents the individual tasks that need to be completed as part of a project. These tasks are assigned to users, and each task has specific details such as its status, description, and deadline.

* **taskID (PK)**: A unique identifier for each task. It serves as the primary key.
* **title**: The title or name of the task, which gives a brief overview of what the task is about (e.g., "Complete Product Design").
* **description**: A more detailed description of the task, explaining the steps or objectives involved.
* **status**: The current status of the task (e.g., "In Progress", "Completed", "Pending"). This helps track task progress.
* **startDate**: The date when the task was initiated or started. This helps in tracking project timelines.
* **assigneeID (FK)**: Foreign Key (FK) referencing the **User** entity. It identifies which user the task is assigned to. A user can be assigned multiple tasks.
* **projectID (FK)**: Foreign Key (FK) referencing the **Project** entity. It indicates which project the task is a part of.

**Project Entity**

The **Project** entity captures the details of a project within the platform. A project can have multiple tasks and teams assigned to it, and it helps track the overall progress.

* **projectID (PK)**: A unique identifier for each project. This is a primary key.
* **name**: The name of the project (e.g., "Mobile App Development").
* **description**: A description of the project, explaining its scope, objectives, and key deliverables.
* **startDate**: The date the project was started. This helps track project timelines.
* **endDate**: The expected or actual end date of the project. This field can be nullable as some projects may not have an end date set immediately.

**Team Entity**

The **Team** entity stores information about the different teams that work on the various tasks within the project. Teams can include members with different roles, such as developers, designers, or marketing experts.

* **teamID (PK)**: A unique identifier for each team. This serves as the primary key.
* **teamName**: The name of the team (e.g., "Frontend Development Team").
* **members**: A count or a list of the number of members within the team. This can include the user IDs of the members associated with the team.

**Client Entity**

The **Client** entity stores information about the clients of the startup. These could be external businesses or individuals that the startup works with, either on projects or as a customer for the product or service being developed.

* **clientID (PK)**: A unique identifier for each client. This is the primary key.
* **name**: The name of the client organization or individual.
* **email**: The email address of the client, used for communication.
* **contactPerson**: The name of the contact person from the client’s organization (e.g., a project manager or business representative).

**CRM Entity**

The **CRM** (Customer Relationship Management) entity captures client-related interactions and notes. This entity helps the startup manage customer relationships and track the status of different client interactions.

* **crmID (PK)**: A unique identifier for each CRM record.
* **clientID (FK)**: Foreign Key referencing the **Client** entity. This helps link the CRM record to a specific client.
* **notes**: Notes related to the client, which could include customer feedback, project status updates, or communication logs.

**2. Relationships Between Entities**

The relationships between entities define how they are connected and how data flows between them.

**User to Task (One-to-Many Relationship)**

* **One user can create and be assigned many tasks**, but each task is assigned to only one user.
* The **User** entity is linked to the **Task** entity through a **one-to-many relationship**. This relationship indicates that a single user can have many tasks, but each task is assigned to one user.

**Explanation**: For instance, Sarah, as the entrepreneur, may create multiple tasks related to different aspects of the project, but each task can be assigned to a different team member.

**Task to Project (Many-to-One Relationship)**

* **Many tasks belong to one project**, but each task is part of only one project.
* The **Task** entity has a **many-to-one relationship** with the **Project** entity. This means a project can have many tasks, but each task belongs to only one project.

**Explanation**: For example, a project such as "Mobile App Development" will have many tasks (e.g., design, coding, testing), but each task belongs to this one project.

**Task to Team (Many-to-One Relationship)**

* **Many tasks can be managed by one team**, but each task is handled by only one team.
* The **Task** entity is linked to the **Team** entity with a **many-to-one relationship**. This relationship signifies that a team can work on multiple tasks, but each task is managed by only one team.

**Explanation**: A team (e.g., the frontend development team) will be responsible for managing multiple tasks, but each task can only be assigned to one team.

**Client to CRM (One-to-Many Relationship)**

* **One client can have multiple CRM records**.
* The **Client** entity has a **one-to-many relationship** with the **CRM** entity. This relationship shows that a single client can have several CRM records associated with them (e.g., different projects, feedback, or notes related to that client).

**Explanation**: For example, a client such as "Tech Innovators Inc." may have multiple CRM entries, tracking different aspects of their projects or their communications with the startup.

**Project to Team (One-to-Many Relationship)**

* **One project can involve multiple teams**, but each team can work on several projects.
* The **Project** entity is related to the **Team** entity through a **one-to-many relationship**. This shows that a project can have multiple teams working on different aspects of the project (e.g., design team, development team, testing team).

**Explanation**: For example, a project such as "Mobile App Development" will involve multiple teams working on various tasks related to the project.

**3. Nullability and Constraints**

* **Nullable Fields**: Some attributes like **endDate** in the **Project** entity or **notes** in the **CRM** entity can be nullable. Not every project will have an end date immediately, and not all CRM records will have notes at creation time.
* **Non-nullable Fields**: Attributes like **userID**, **taskID**, **projectID**, **teamID**, **clientID**, and **crmID** are non-nullable, as they are required to uniquely identify records within their respective entities.

**4. Summary of Cardinality**

* **One-to-Many (1:N)**: This relationship is used where one instance of an entity is associated with many instances of another entity. For example, a **User** can create and be assigned many **Tasks**, but each **Task** is only assigned to one **User**.
* **Many-to-One (N:1)**: This relationship signifies that many instances of one entity can be linked to one instance of another entity. For example, many **Tasks** can belong to one **Project**.