VOL BANK SYSTEM SPECIFICATIONS

SMITH-PETER-B00123456

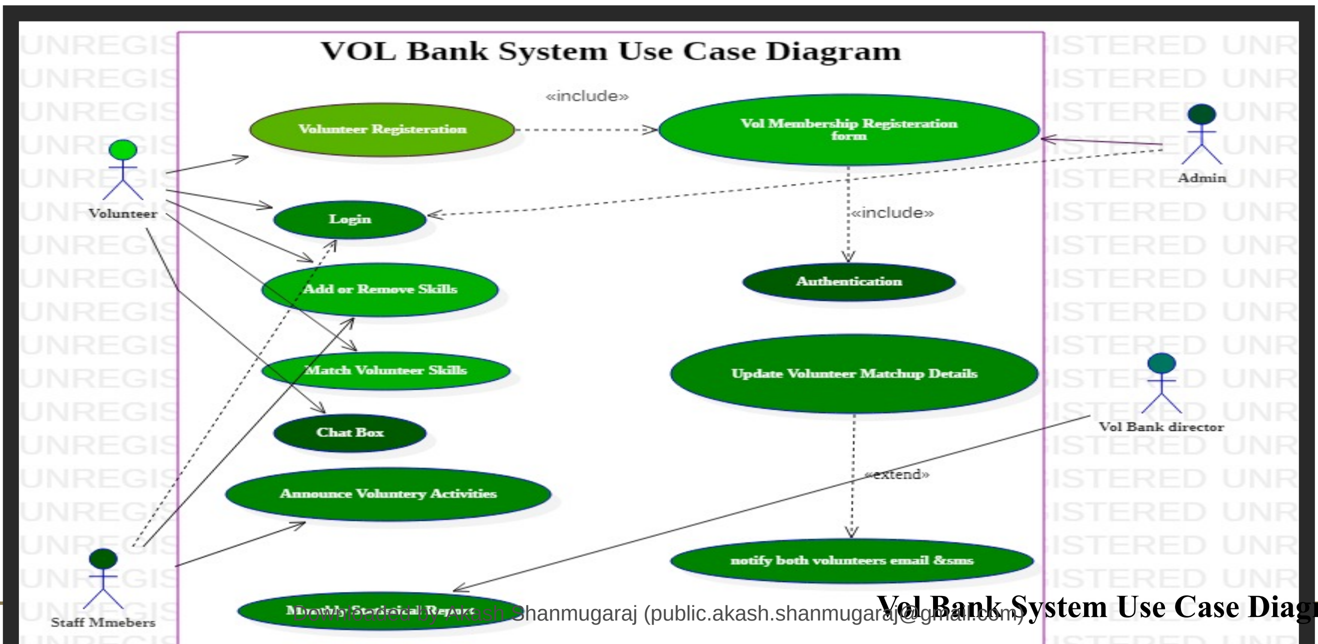
TEACHER NAME

USE CASE DIAGRAM

A use case is a technique used in software engineering to identify and define the interactions between actors (users or other systems) and a system or application. A use case represents a specific goal or task that a user wants to accomplish using the system, and it defines the steps required to achieve that goal.

Here's how use cases work:

1. Identify actors: The first step in creating a use case is to identify the actors that will be interacting with the system. Actors can be users, other systems, or external entities that interact with the system.
2. Define goals: For each actor, identify the goals they want to achieve using the system. A goal represents a specific task or function that the actor wants to perform.
3. Describe steps: Once the goals have been identified, the steps required to achieve those goals can be defined. These steps represent the sequence of interactions between the actor and the system.
4. Document alternate paths: In addition to the main path of interactions, it's important to document any alternate paths that the actor might take. For example, if an error occurs during a transaction, the actor may need to take a different path to achieve their goal.
5. Identify dependencies: Use cases can have dependencies on other use cases or external systems. It's important to identify these dependencies to ensure that the use cases are complete and accurate.
6. Validate with stakeholders: Finally, use cases should be validated with stakeholders to ensure that they accurately represent the system and meet the needs of the users.


VOL Bank System Use Case Diagram
«include»
Volunteer Registeration
Vol Membership Registeration form
Admin
Volunteer
Login
«include»
Add or Remove Skills
Authentication
Match Volunteer Skills
Update Volunteer Matchup Details
Vol Bank director
Chat Box
«extend»
Announce Voluntery Activities
notify both volunteers email &sms
Staff Mmebers
Monthly Statistical Report
UNREGISTERED

Vol Bank System Use Case Diagram

USE CASE DESCRIPTION

Scenario of the use case for "Register Membership"

Vol Bank system has registered with new users
Required to sign-up in both cases to login the system with email and password
After login can staff members add/remove skills (inherits scenario of use case need for **membership registration form)**
Volunteer member submission form **(refers to the included use case of Register & new members or volunteers sign-up)**

```
if volunteers form is valid
   then repeat the process
                        new volunteers can login and start using the system
                  until (volunteers and staff members can match up each other according to   email/password
                  for login) or
                        (no volunteers can match up with Vol bank and staff members)
            if a Vol Bank and staff members have already login account
                  then one can add/remove skills                         (normal case)
                        tell the admin about login and registration form submission and teachers to teach
                        Spanish too.
                  else inform admin to cancel the submission box          special case)
            endif
      else reject the matchup details and send message to volunteers      (error case)
end if
```

Scenario of the use case Vol Bank staff members

```
if Vol Bank staff members match up with volunteers according to their skills
      then matchup is valid
      else matchup is considered invalid
endif
```

Refinement of "Adding/removing skills"
Record volunteers / staff members skills
Record registered and new users details
Record Teachers language skills
Update Vol Bank registration system with matchup details and inform other volunteers

Pseudocode for Vol Bank “Registration

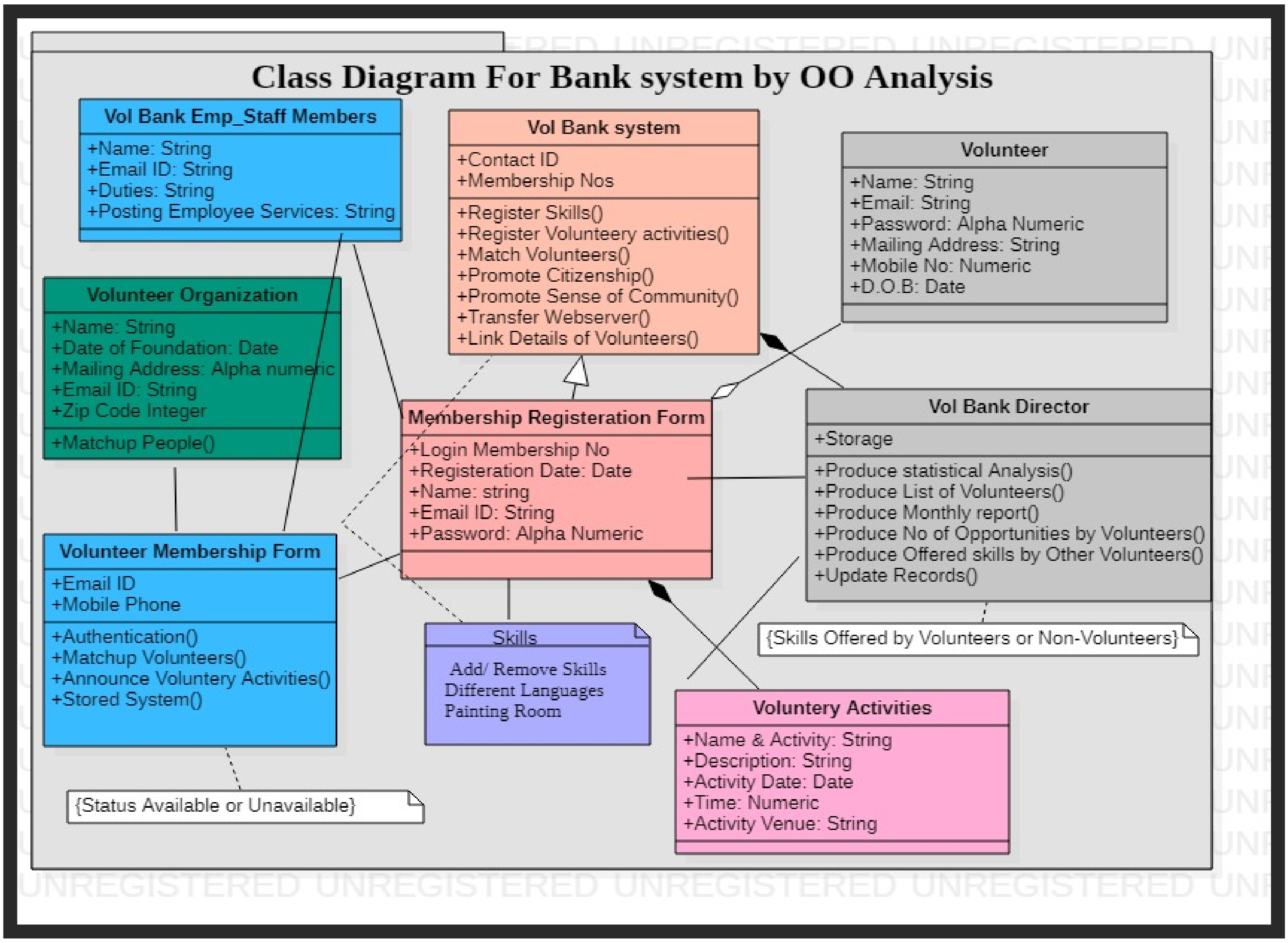
CLASS DIAGRAM

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the structure of a system by showing the classes, attributes, operations, and relationships between them. It provides a high-level view of the system's architecture and is used to help design and document software systems.

Here's a step-by-step explanation of how to create a class diagram:

1. Identify the classes: Start by identifying the classes that make up the system. A class is a template or blueprint for creating objects that share similar properties and behaviors. Classes can represent entities, such as people or objects, or concepts, such as processes or systems.
2. Define the attributes: For each class, define the attributes or properties that describe its state. Attributes can be simple data types, such as integers or strings, or more complex types, such as objects or arrays. Attributes can be public or private, depending on their visibility.
3. Define the operations: For each class, define the operations or methods that define its behavior. Operations can be public or private, and can take parameters and return values. Operations can also be abstract, which means they are defined in the class but implemented in the subclasses.
4. Specify the relationships: Relationships represent the connections between classes and are important for understanding the structure of the system. There are several types of relationships, including inheritance, association, aggregation, and composition. Inheritance represents the relationship between a superclass and its subclasses, while association represents a relationship between two classes. Aggregation and composition represent different types of association, where aggregation is a weaker relationship and composition is a stronger relationship.

5. Indicate multiplicities: Multiplicities indicate how many objects can participate in a relationship. For example, a one-to-many relationship between two classes indicates that one object of the first class is related to many objects of the second class.
6. Add annotations: Finally, you can add annotations to provide additional information or clarify the diagram. Annotations can include comments, constraints, or other details that are relevant to the system being modeled.



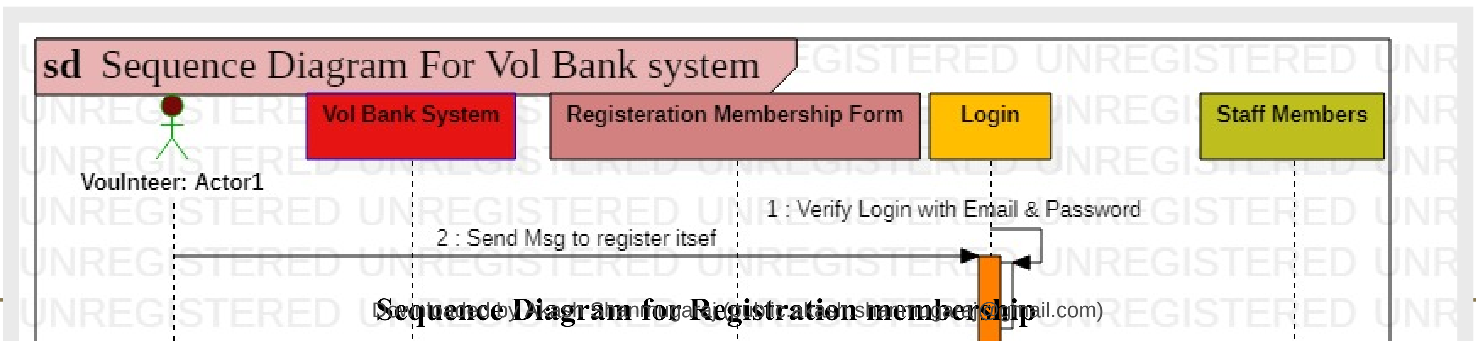
3.1 Class Diagram OO analysis for Vol Bank system

SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modeling Language) diagram that shows interactions between objects or components in a system over time. It depicts the sequence of messages exchanged between the different objects or components involved in a particular scenario or use case.

Here's a step-by-step explanation of how to create a sequence diagram:

1. Identify the actors and objects involved: Start by identifying the different actors and objects that participate in the scenario you want to depict. Actors are external entities that interact with the system, while objects are the internal components of the system.
2. Define the lifelines: Once you have identified the actors and objects, you need to create a vertical line for each of them to represent their lifeline. A lifeline represents the existence of an object during the execution of the scenario.
3. Add messages: Messages represent the interactions between the different objects or components in the system. You can add messages by drawing arrows between the lifelines, with the arrowhead pointing in the direction of the message flow.
4. Specify message types: There are several types of messages that can be used in a sequence diagram, including synchronous, asynchronous, return, and exception messages. It's important to specify the type of each message to accurately represent the interactions between the objects.
5. Indicate time sequence: To show the sequence of events, you can use numbers or letters along the timeline to indicate the order in which the messages are exchanged. You can also use dashed lines to represent delays or pauses in the message flow.
6. Add conditions and loops: You can use conditional and looping constructs in a sequence diagram to represent different scenarios or variations of the main scenario. For example, you can use an if-else construct to show alternative paths or a loop construct to show a repetitive process.
7. Add annotations: Finally, you can add annotations to provide additional information or clarify the diagram. Annotations can include comments, constraints, or other details that are relevant to the scenario being depicted.


sd Sequence Diagram For Vol Bank system
Voulnteer: Actor1
Vol Bank System
Registeration Membership Form
Login
Staff Members
1 : Verify Login with Email & Password
2 : Send Msg to register itsef
UNREGISTERED

Sequence Diagram for Registration membership