SE-I COURSE PROJECT (PHASES 1 & 2 COVER SHEET)

FOR TEAMS OF FIVE MEMBERS ONLY

Discussions Scheduled for Week 11 or 12 *(Specific dates TBA later).*

- Print 1 copy of this cover sheet and attach it to a printed copy of the documentation *(SRS, … etc.)*. You must submit softcopies of all your documents *(as PDFs)*; details will be announced later.
- Please write all your names in Arabic.
- Please make sure that your students' IDs are correct.
- Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
- Please attend the discussion on time *(announced separately)*, late teams will lose 3grades.

Project Name: Bug Tracking system project

Team Information ***(typed not handwritten, except for the attendance signature)*:**

	ID [Ordered by ID]	Full Name [In Arabic]	Attendance [Handwritten Signature]	Final Grade
1	Salma Ahmed Essa	201900344		
2	Doha Mostafa Abdelnaby	201900385		
3	Maram Mohammed	201900784		
4	Nuurhan Khaled Mostafa	201900913		
5	Yasmeen Mohy Eldeen	201900962		

Teaching-Assistant's Signature: _____

_____
2

Table of Content

Introduction

In this Project we analyze the software Requirements and design Object Oriented models for a web application, so the following two phases provides an over view for the bug tracking system.

System Requirements Validation:

Phase 1

2.1 Functional Requirements

Requirement Name	Login
Requirement Id	LN
Actor	Admin , Staff & Customer
priority	Must
ID	Description
LN1	System Must Allow Admin To Register and Log in with His Username & Password
LN2	System Must Allow Staff To Register and Log in with His Username & Password
LN3	System Must Allow Customer To Register and Log in with His Username & Password

Requirement Name	Register
Requirement Id	RS
Actor	Customer
Priority	Must
ID	Description
RS1	System must allow customer to add (User Name , Password , E-mail, ID)

Requirements	Add Staff
Requirement Id	AS
Actor	Admin
Priority	Should
ID	Description
SA1	System must allow admin to add Staff (user name- password-E-mail-ID-category)
SA	System must allow admin to update staff details
SA	System must allow the admin to delete staff

Requirements	Assign Staff
Requirement Id	AS
Actor	Admin & Staff
Priority	Must
ID	Description
AS1	The system must allow admin to choose the staff who will solve the bug
AS2	The system must allow staff to send the bug to another staff

Requirements	Create Project
Requirement Id	CP
Actor	Admin
Priority	Must
ID	Description
CP1	the system must allow the admin to add projects (name, description)
CP2	the system must allow the admin to Update projects
CP3	the system must allow the admin to delete projects
CP4	the system must allow the admin to View projects

Requirements	Send Bug
Requirement Id	SB
Actor	Customer
Priority	Should
ID	Description
SB1	The system must allow the User to enter bug details (project name – error category – error details)
SB2	The system must allow the User to upload print screen of the bug
SB 3	The system must allow the User to have a ticket number to flow the bug

Requirements	Flow Bugs
Requirement Id	FB
Actor	Admin , Customer & Staff
Priority	Should
ID	Description
FB1	The system must allow the admin to follow the status of the bug “Pending – In Progress – Delivered – Staff Change – Finished”)
FB2	The system must allow the staff to follow the status of the bug “Pending – In Progress – Delivered – Staff Change – Finished”)
FB3	The system must allow the user to follow the status of the bug “Pending – In Progress – Delivered – Staff Change – Finished”)

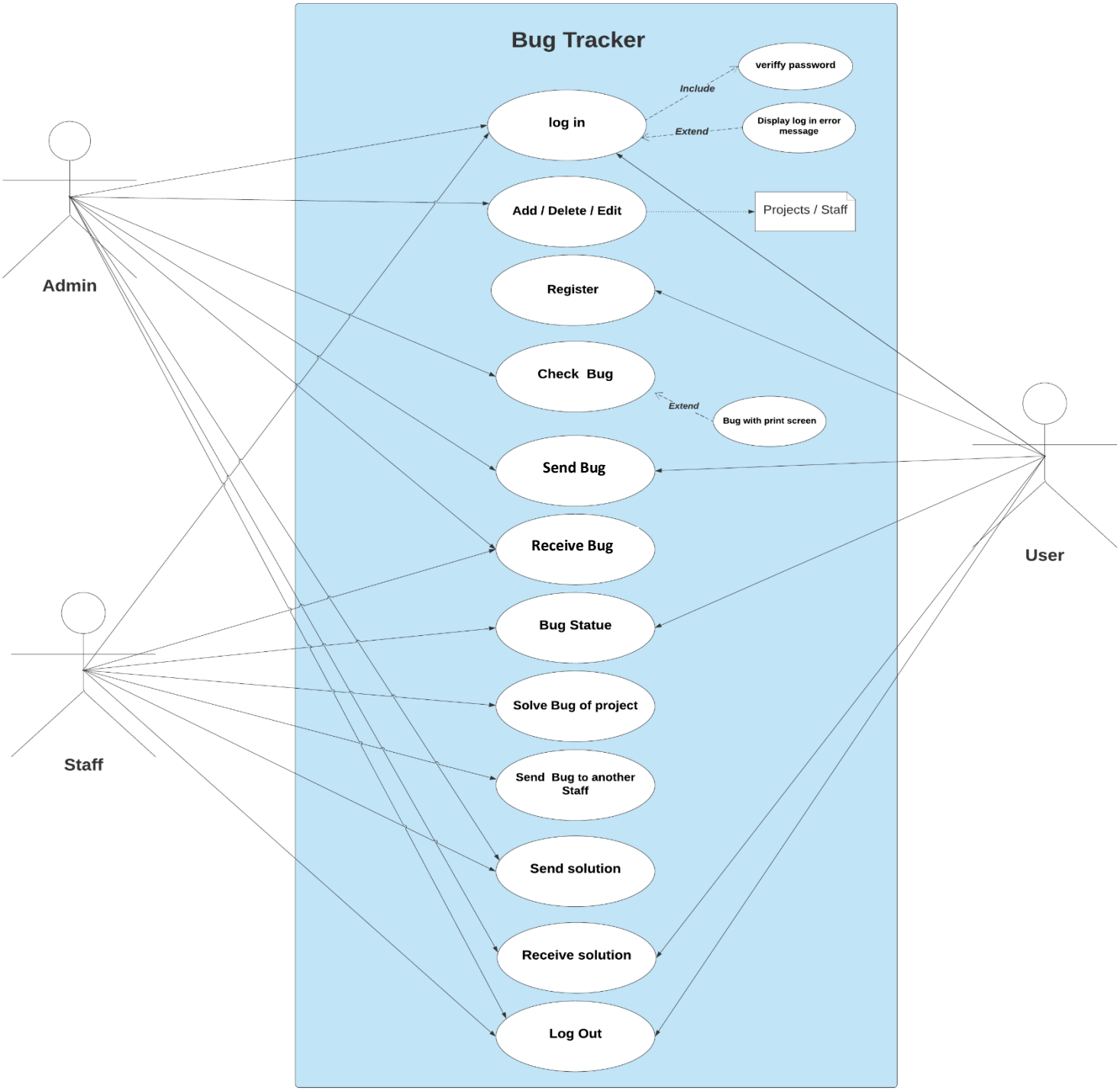
Requirements	View Bugs
Requirement Id	VB
Actor	Admin& Staff
Priority	Must
ID	Description
VB1	The system must allow the admin to view all the bugs which sent from the customer
VB2	The system must allow the Staff to see all the bugs assign to them

Requirements	Solution Message
Requirement Id	SM
Actor	Admin, Staff & Customer
Priority	must
ID	Description
SM1	The system must allow admin to see the solution message
SM2	The system must allow admin to Accept or Refuse the Solution
SM3	The system must allow the customer to see the solution messages from staff or admin

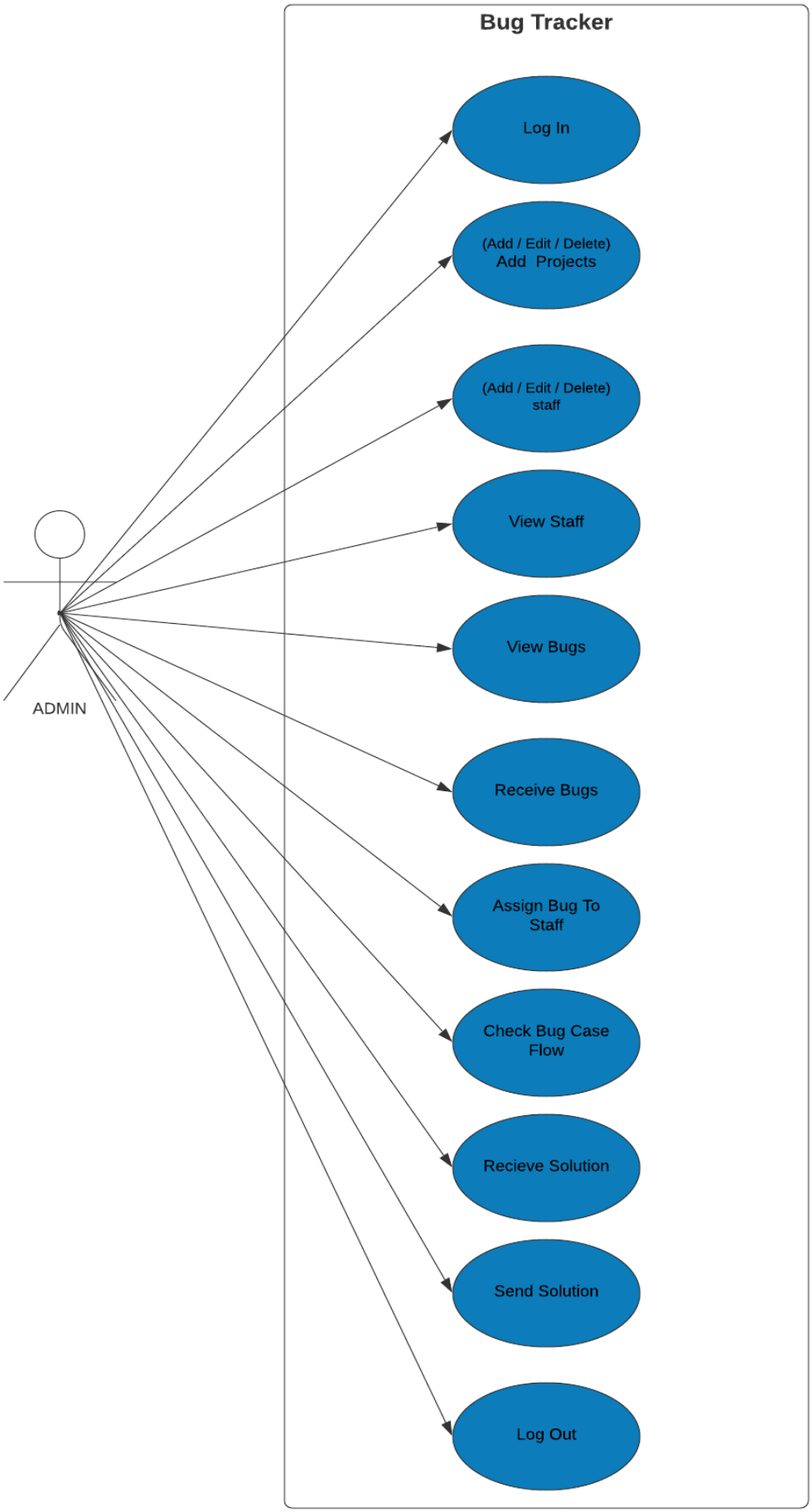
2.2 Non-functional requirement

User Requirements	System Requirements
1)Look-and-feel REQs	• The system should use a lot of animation. • The product shall comply with motif user interface guidelines
2) usability & humanity REQs	• The system shall be easy to use on the first attempt by a member of the public without training. • 90% of the general population should be able to send the details of the bug to solve it from web interface within 1 day
3) performance REQs	• Admin should response to request within 1 hour. • The system shall operate without failure for 25 days. • Staff should solve the bug within an hour from time of seeing the bug. • The system shall handle up to 10 users.
4) Operational & Environmental REQs	• The product will be used in a standard office environment expect that high level of background noise may occur.
5) Maintainability & Support REQs.	• The product shall be able to be modified to cope with a new class of users.
6) Cultural REQs	• The language used in the interface should be formal and polite. • The product should not use in appropriate symbol or languages.
7) Legal REQs	• The product prevents any copies from personal data due to copyrights
8) Security REQs	• Access; uninterrupted/continual access to data & functionality by authorized users. • Privacy: protection of data from unauthorized access/disclosure. • Integrity: prevention of unauthorized modification/deletion of data (data consistency). • Immunity; protection against threats and attacks.

2.3 Use case diagram(s):



2.3.A: “ Admin Use case”



Use Case for Admin:

- The admin logs in to the system.
- Then, he starts to add projects to the system.
- He can view all bugs on system and all staff with their categories.
- The admin could Add/Edit/Delete staff with their categories.
- When customer catch bug he sends it to system, the admin receives the bug and assign it to staff.
- The admin could check the bug case flow.
- The admin receive the solution of bug from staff and send it to customer.

2.3.B: " Staff Use case"

Use Case For Staff:

- The staff can log in system.
- The system assign bugs to staff so when staff log in he could see the assigned bugs.
- The staff works in categories, so each category solve kind of bug
- The staff try to solve the bug and send the solution to the admin or to the customer
- If the staff couldn't solve the bug he could assign it to another staff
- The staff also could check the bug case flow.

2.3.C: " Customer Use case"

Use Case For User:

- The system shall allow to the user to log in.
- Then, the user should choose one project from the projects in system.
- The user should track the project then catch the bug.
- Send the bug to the system so it can fix it.
- The user has the ability to see the bug case flow.
- the user receive the solution from system.

Detailed use case description:

Use Case Name	Log In
Include Use Case	Verify Password
Extend Use Case	Display Error Message
Description	This use case allows user to login to the site ,Then system will verify username and password (include use case) then, the user can view their functionality depending on his role.
Primary Actor	(Admin – User - Staff)
Secondary Actor	none
Preconditions	User must be registered to the system (User) His data has been entered to the system (Admin/Staff)
Postconditions	System will display the relevant home page
Main Flow	Log in. Insert username. Insert password. System validates if the user name and password are correct or not. System displays the relevant home page.
Alternative Flow	If Username or password aren't correct: System will display error message. User will enter the correct username and password. System displays the relevant home page.
Goal	User logs in to system successfully.

Use Case Name	Register
Include Use Case	Verify registration, Update user table.
Extend Use Case	Username exists
Description	Customers can make a registration to the web application to upload her/his software bug, he fills the form and click register then system verify registration and update user table.
Primary Actor	User
Secondary Actor	System
Preconditions	User is not logged in.
Postconditions	User can log into the system
Main Flow	1) Customer chooses from a login page that he doesn't have an account before and chooses to resist. 2) Customers fill the register form. 3) system approved the customer registration and update user table. 4) Customer Choose a username already in use. 5) system shows an error message. 6) Customer change username and make accepted registration.
Goal	Customers can create an account in e-bug tracking web application so can login to it and benefit from its features.

Use Case Name	Add / Edit / Delete
Description	This use case allows to (Add / Edit / Delete) a (Staff / projects) data to the system.
Primary Actor	Admin
Secondary Actor	none
Preconditions	- Staff must have been registered to the system and his data has been entered to the system. - Project data must be exist in the system.
Postconditions	Data are added or modified.
Main Flow	1) Log in 2) Verifying if having a role to add or delete or update 3) If having a role, then choose the field to modify 4) Editing or deleting or updating on the chosen field 5) Save the process 6) Logout
Goal	Data of (staff / projects) are added to the system successfully.

Use Case Name	Check bug
Include Use Case	none
Extend Use Case	none
Description	In this use case user track the project that he chooses to catch a bug and send it to the system.
Primary Actor	User
Secondary Actor	none
Preconditions	-User must be registered to the system (User) -he must choose project first.
Postconditions	-system must be opened.
Main Flow	1) User chooses the project from system. 2) User should track the project and choose the bug he catches.
Goal	User catches bugs and send it successfully.

Use Case Name	Send project.
Description	This use case allows to send the project that contain the bug we want be solved.
Primary Actor	User
Secondary Actor	Admin
Preconditions	User must login to the system. The message sent by the user must contain screen print of the bug.
Postconditions	- Admin view Bug and Assign it to a staff member.
Main Flow	User log in. Send bug with a screen print to it. Admin view bug message. Admin Assign the bug to a staff.
Goal	Bug is solved.

Use Case Name	Receive bug message.
Description	This use case is used to receive the bug message sent from user.
Primary Actor	Admin
Secondary Actor	Staff
Preconditions	The message sent by the user with a screen print of the bug.
Postconditions	- User see the state of his message. -Staff try to solve the bug.
Main Flow	Admin Receive bug message. Admin view bug message. Admin Assign the bug to a staff. Staff receives bug message to solve it
Goal	Bug received and assigned to the staff that will solve it

Use Case Name	Bug Statue
Include Use Case	none
Extend Use Case	none
Description	This use case make admin, staff and customer and staff could see the bug case flow and follow its fixing steps by the staff
Primary Actor	(Admin – User - Staff)
Secondary Actor	none
Preconditions	1) customer must be sent a bug 2) Admin must be pre- logged in
Postconditions	Admin can list the sent bug
Main Flow	1) admin, customer and staff can show the state of recent bug and follow it.
Goal	The system will be clearer to the user of system.

Use Case Name	Solve Bug
Description	This use case is used to solve the bug that the user sent.
Primary Actor	Staff
Secondary Actor	Another staff
Preconditions	Admin assign the staff that will solve the bug.
Postconditions	- bug solved and send the solution to the admin
Main Flow	Staff receives the assigned bug message. State the current statue of the bug (pending / solved / failed) If the staff failed to solve the bug, then it will be sent to another staff member to solve.
Alternative Flow	The bug message sent to another staff to
Goal	Bug is solved to send the solution to the Admin.

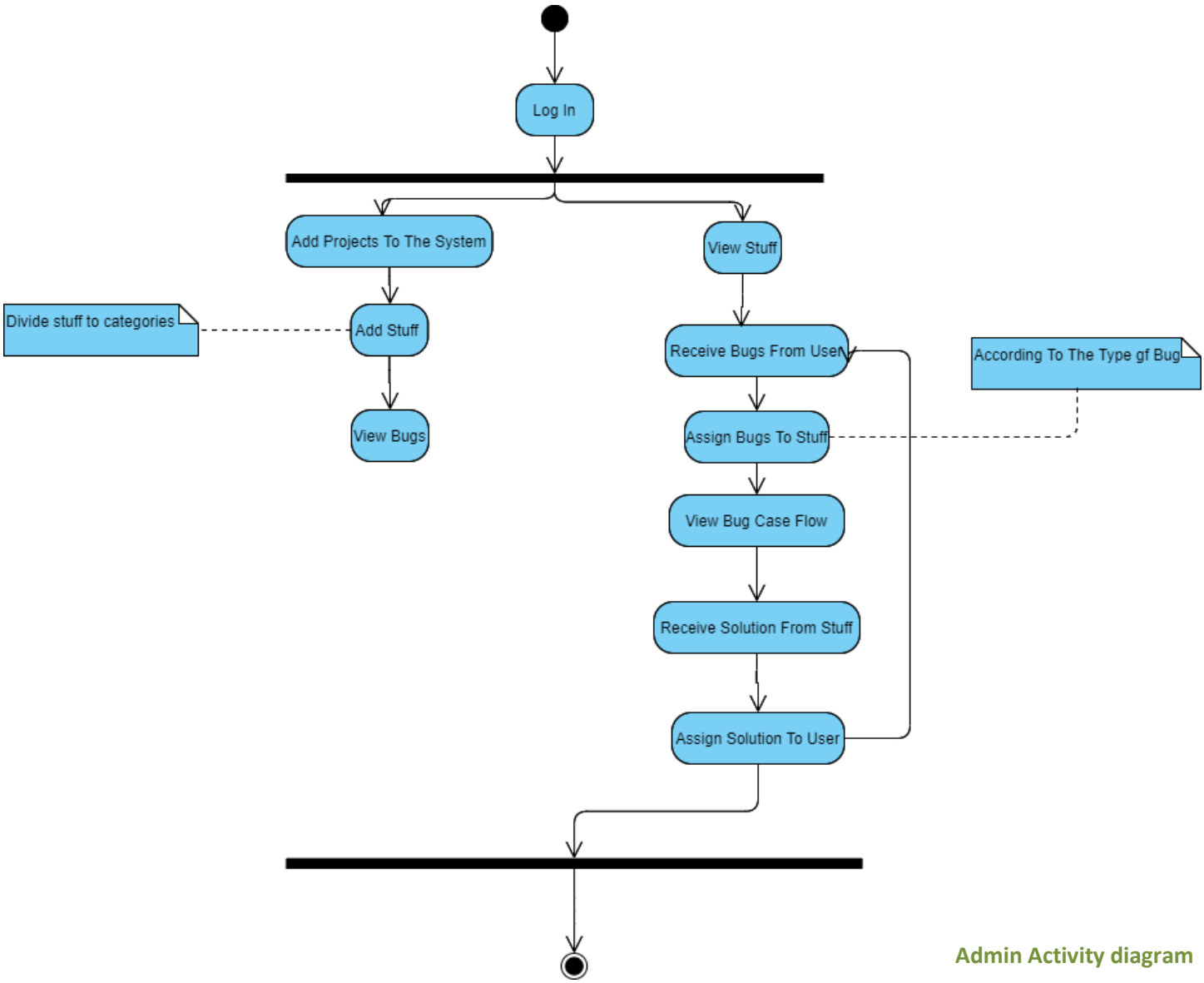
Use Case Name	Assign bug to staff.
Include Use Case	none
Extend Use Case	none
Description	This use case used by admin and staff, admin can use it to assign bugs to staff to work on solving sent bugs, staff can use it if he is involved by mistake to a bug which is not in his category or if it needs another category staff member
Primary Actor	Admin, Staff.
Secondary Actor	none
Preconditions	1)The Admin must have created the project. 2)The customer must be sent a bug. 3)The staff has spent more than specified time.
Postconditions	-the admin send bug to the staff depend on the staff category. -when the staff receive bug, he should solve it in specified time if he didn't, he could assign it to another staff
Main Flow	1)) Admin chooses to view sent bugs and assign each on depend on the category of bug and category of staff. 2)chosen staff try to solve the bug, if he couldn't and time is up, he assigns it to another staff.
Goal	The bug is solved as faster as possible.

Use Case Name	Send solution
Description	- This use case is used to send the solution to of the bug.
Primary Actor	Staff
Secondary Actor	Admin
Preconditions	- Bug is assigned to the staff. - Staff succeeded to solve the bug.
Postconditions	- solution is sent to the user
Main Flow	- staff solve the bug - Send the solution to the admin. - Admin send the solution to user.
Goal	Solution is sent to the user.

Use Case Name	Receive solution.
Description	This use case is used to receive the solution to of the bug.
Primary Actor	Admin
Secondary Actor	User
Preconditions	Bug is assigned to the staff. Staff succeeded to solve the bug. Bug solution is sent from the staff.
Postconditions	User receive the solution of his bug.
Main Flow	Admin receives solution from staff. Admin change the state of the message to solved. Admin send the solution to user. User receives the solution.
Goal	Bug is solved and received to the user successfully.

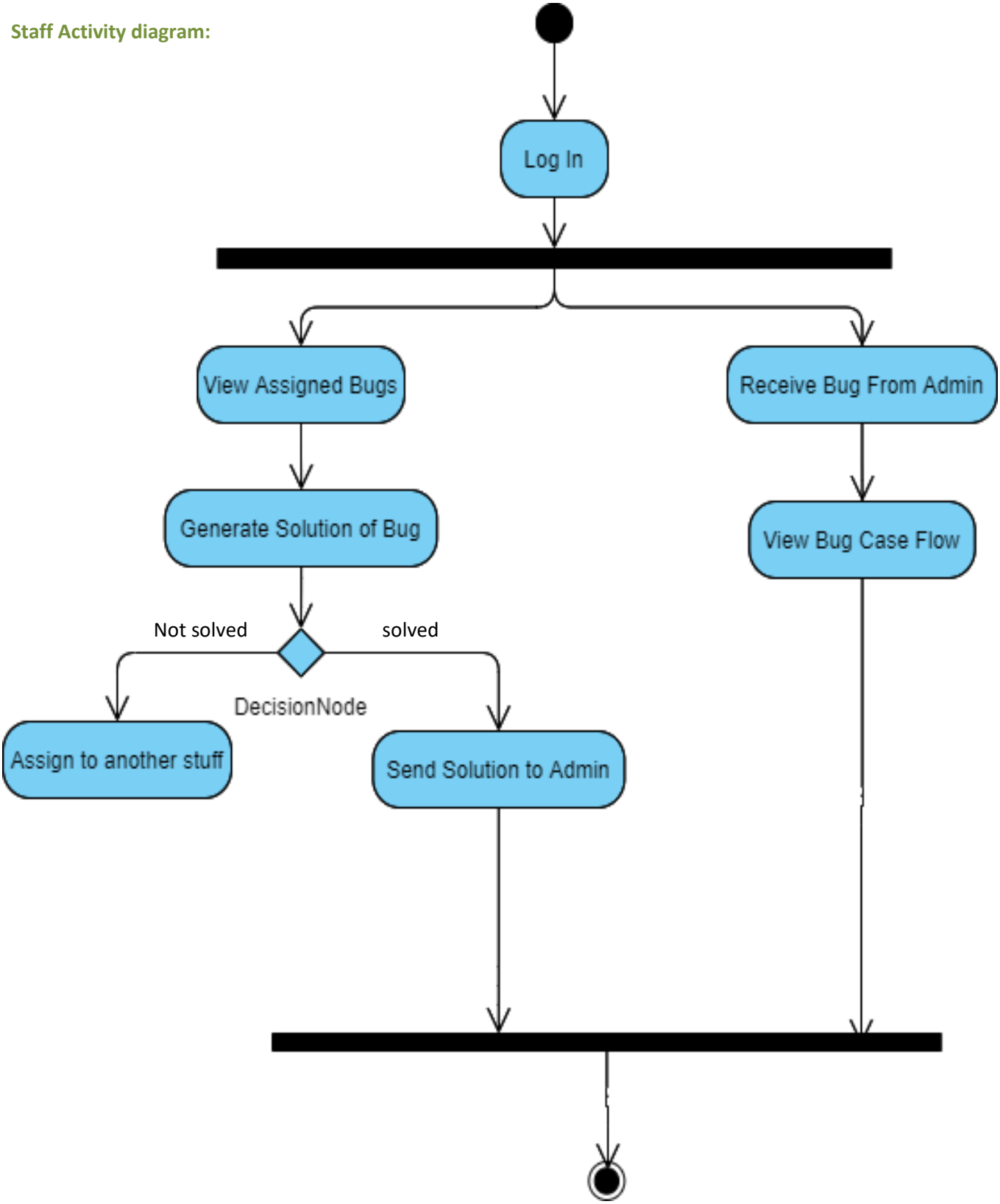
Use Case Name	Log out
Include Use Case	none
Extend Use Case	none
Description	This use case enables the customer, staff and admin to save their work and log out safely.
Primary Actor	Admin, Staff and customer.
Secondary Actor	none
Preconditions	1) The system is opened. 2) the user of system (customer/staff/admin) save their work before they log out.
Postconditions	-The system will be closed.
Main Flow	1) Admin, customer and staff log out from system.
Goal	The system is logged out safely with saving all data that has been changed.

2.4 Activity diagram(s):

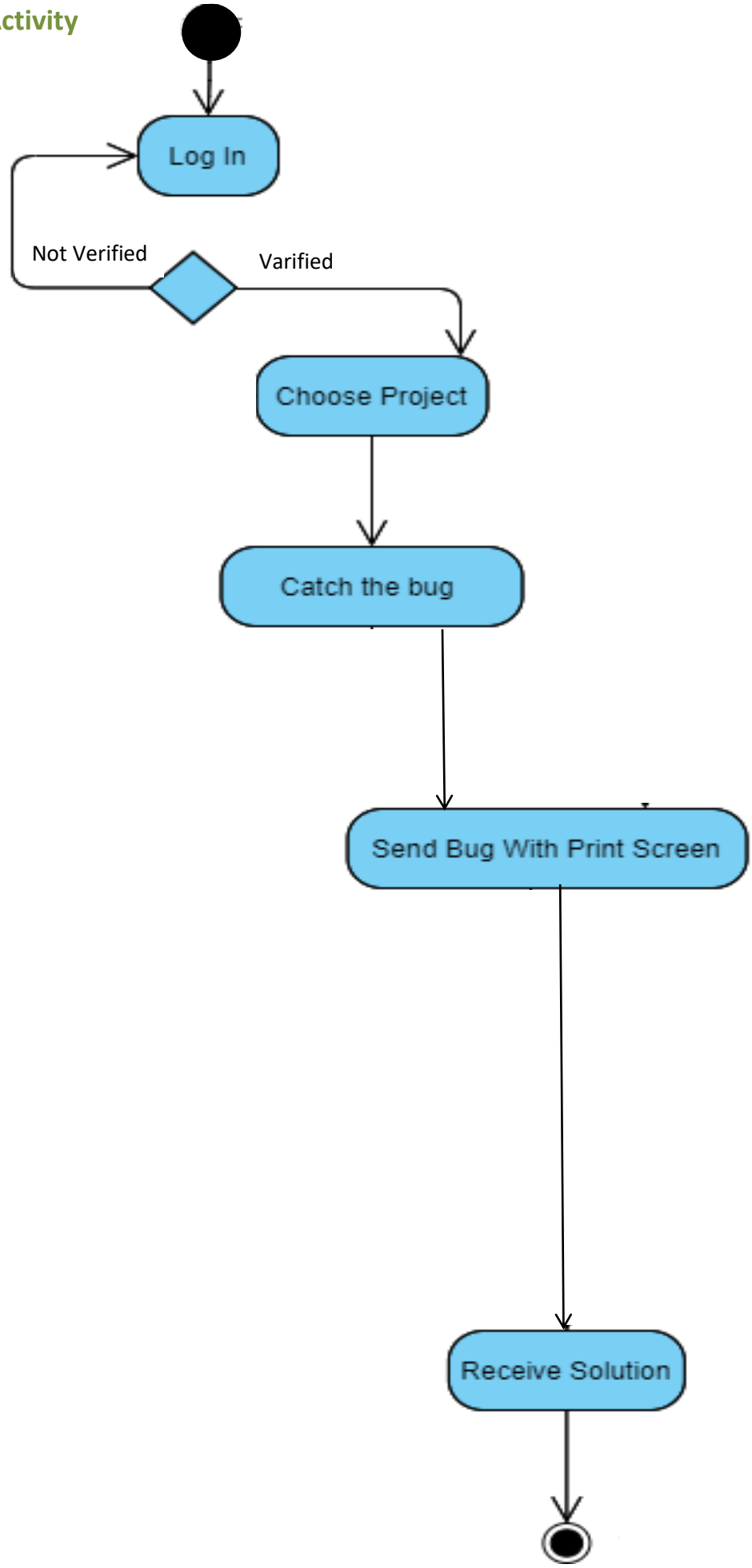


Admin Activity diagram

Staff Activity diagram:

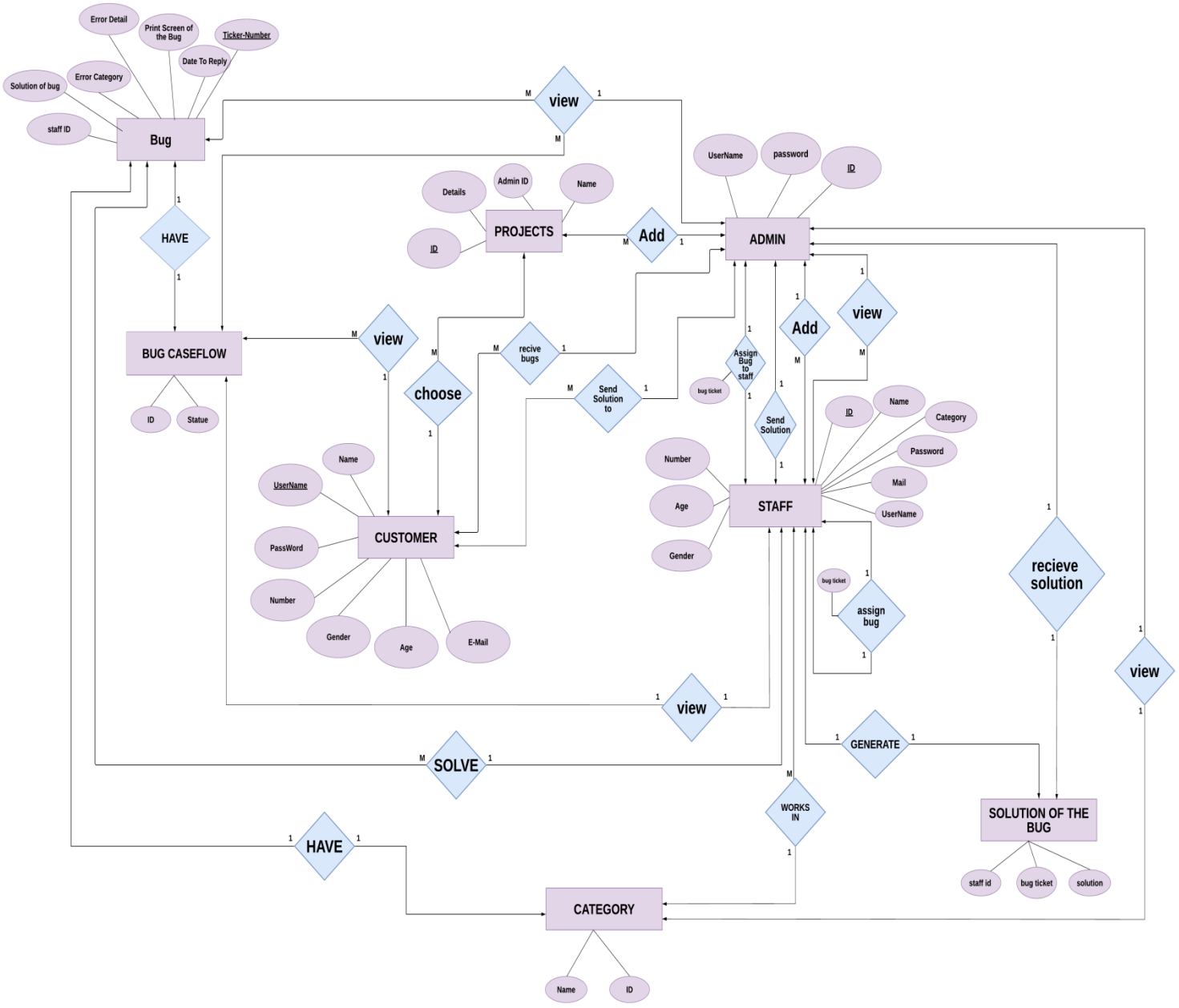


Customer Activity

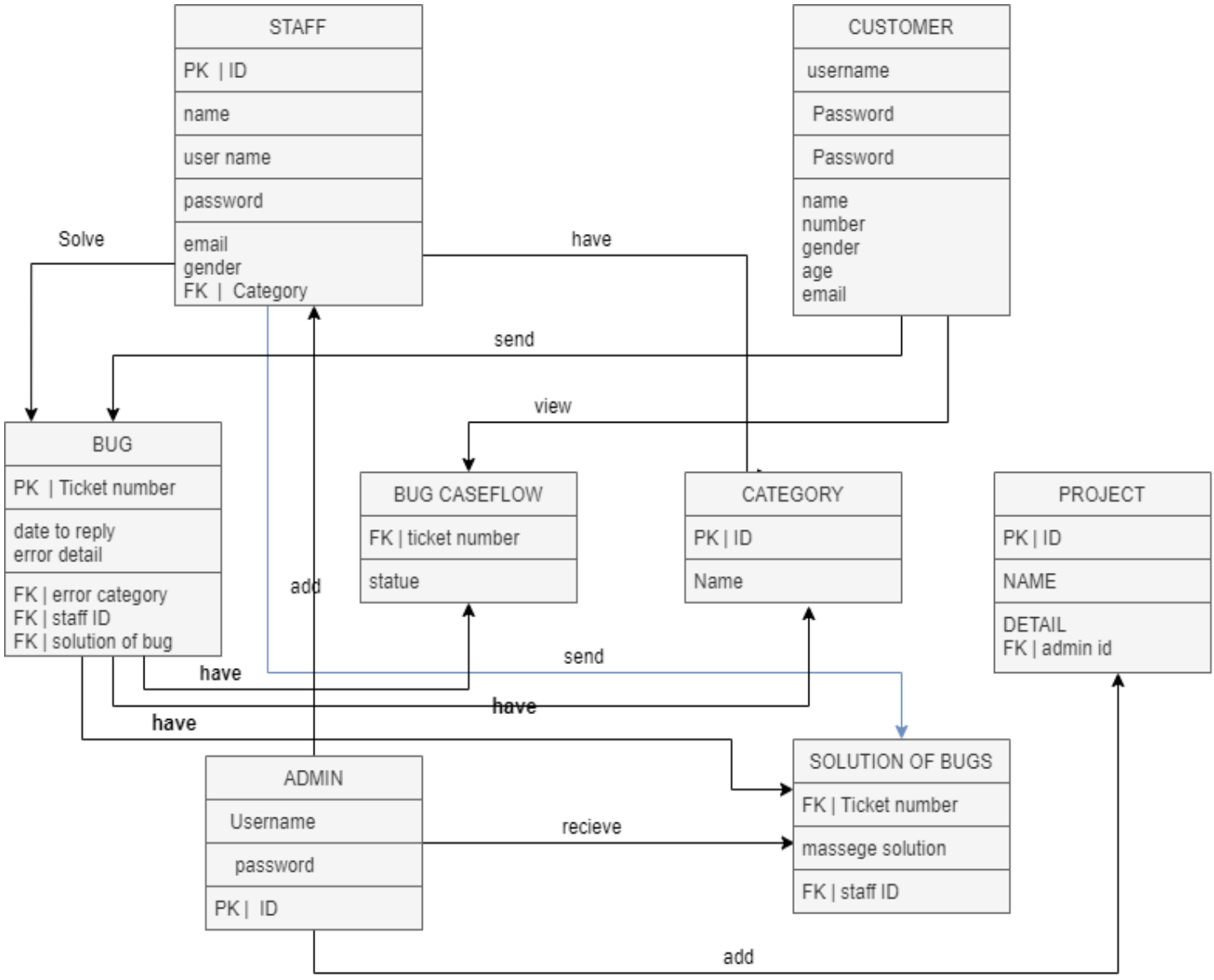


2.5 Database specification:

2.5.a “ERD”:



2.5.b “Table”:


STAFF
PK | ID
name
user name
password
email
gender
FK | Category
CUSTOMER
username
Password
Password
name
number
gender
age
email
Solve
have
send
view
BUG
PK | Ticket number
date to reply
error detail
FK | error category
FK | staff ID
FK | solution of bug
BUG CASEFLOW
FK | ticket number
statue
CATEGORY
PK | ID
Name
PROJECT
PK | ID
NAME
DETAIL
FK | admin id
add
send
have
have
have
ADMIN
Username
password
PK | ID
recieve
SOLUTION OF BUGS
FK | Ticket number
massege solution
FK | staff ID
add

Phase 2

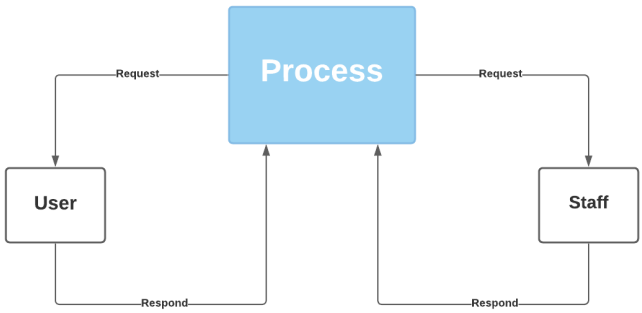
3.1. System Architecture

i. Definition:

• Systems Architecture is a general system to manage objects (existing or to be created), in a way that makes thinking about the structural properties of such objects easier.

• It shows the overall structure of the system, instead of showing individual components.

ii. Applied architectural pattern(s):

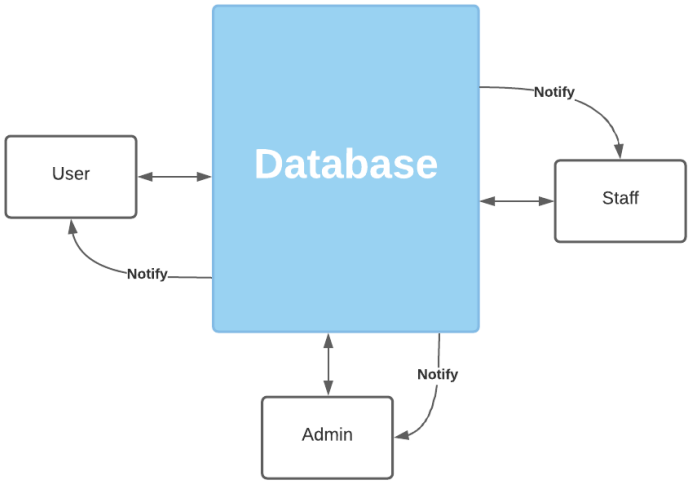


➢ Client/Server architecture:

• One of the best patterns to be used in web applications and pages.

• Our system is divided into a client-side (The web browser & site and the user or the customer.) and a server-side (The admin and his access to database and PHP server.).

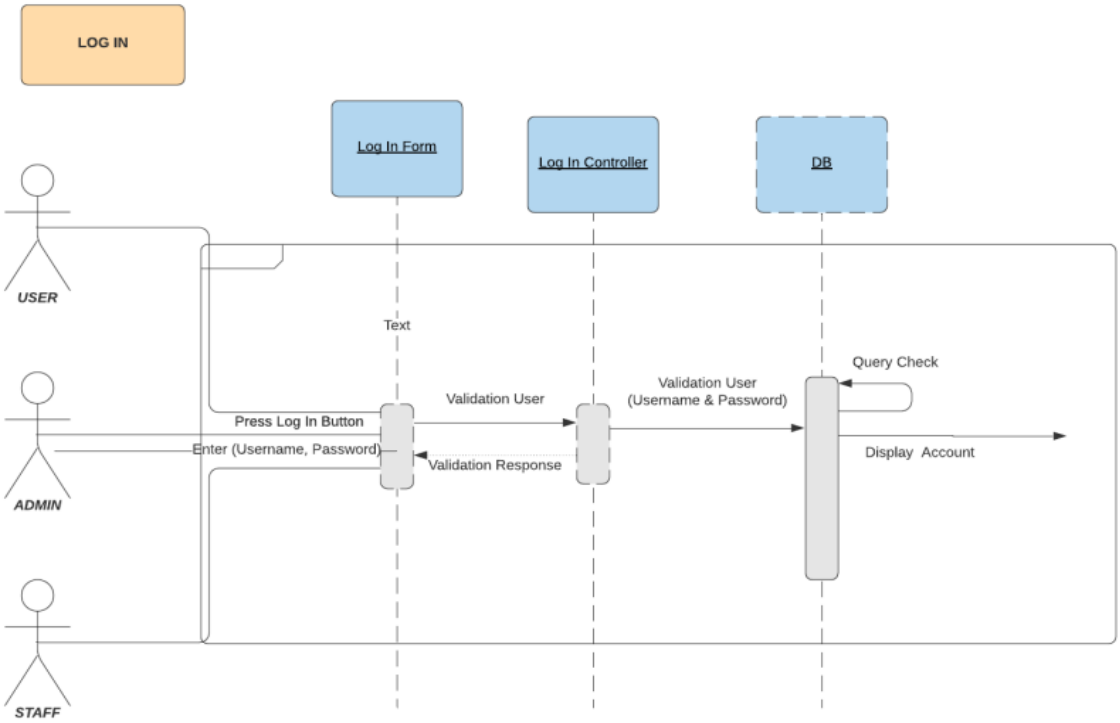
-Client/server pattern-

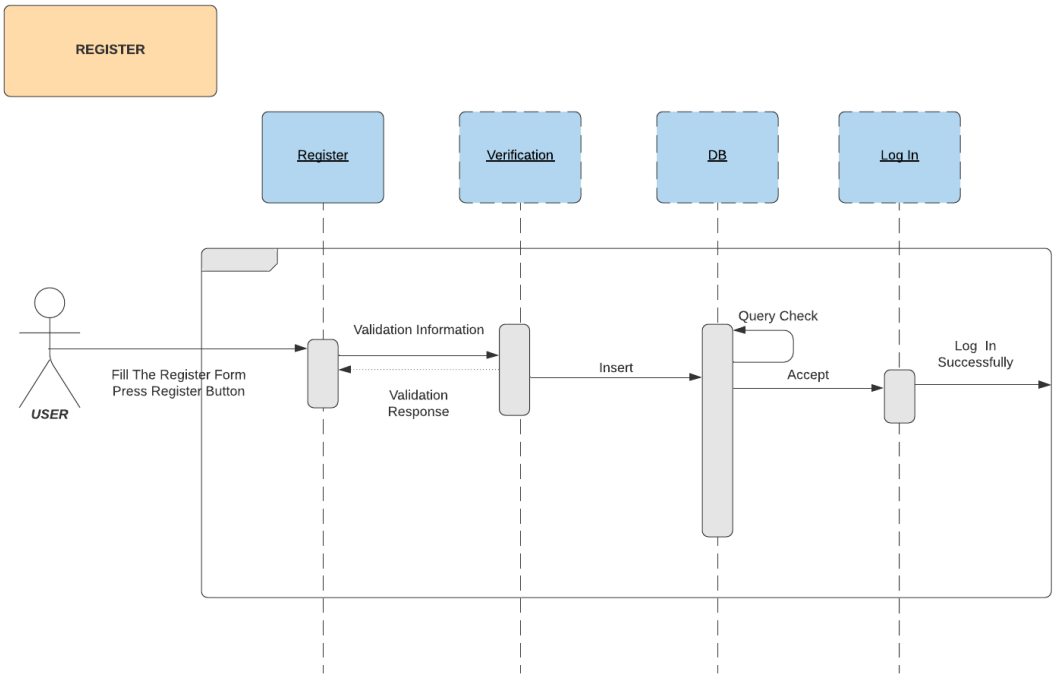


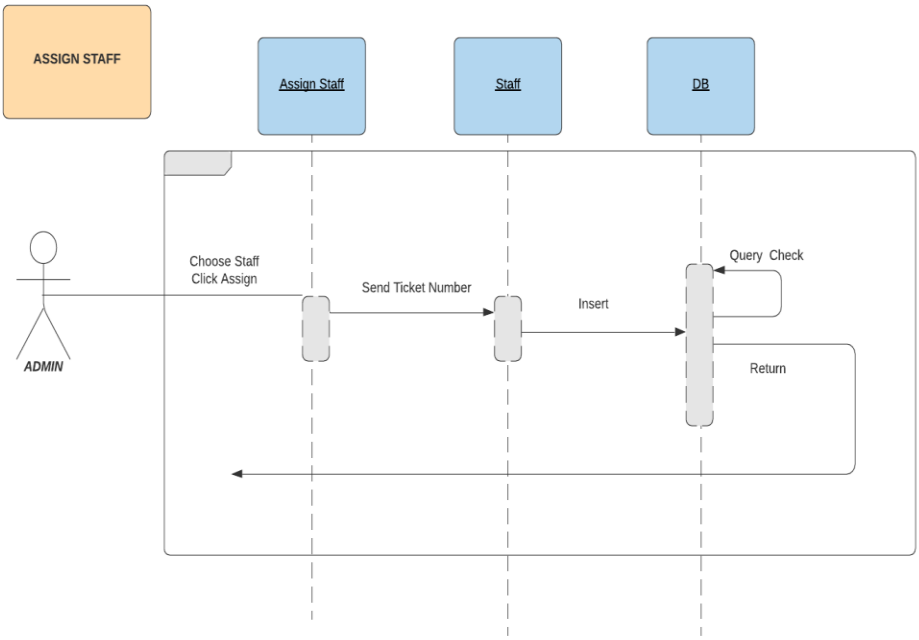
➢ Data-Centered architecture (Black-Board):

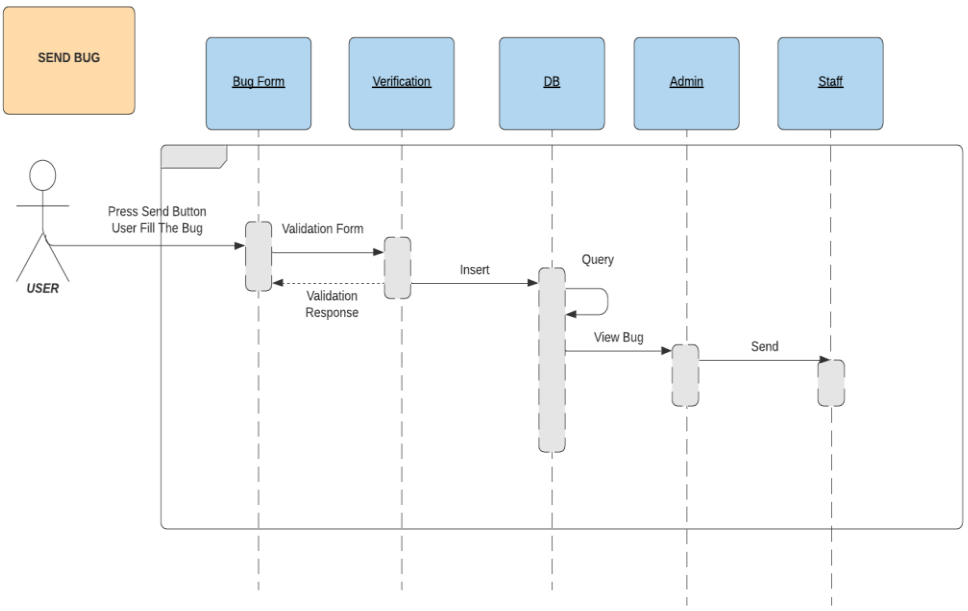
• Our system has a database holding all personal data of all Customers, Staffs and the admin.

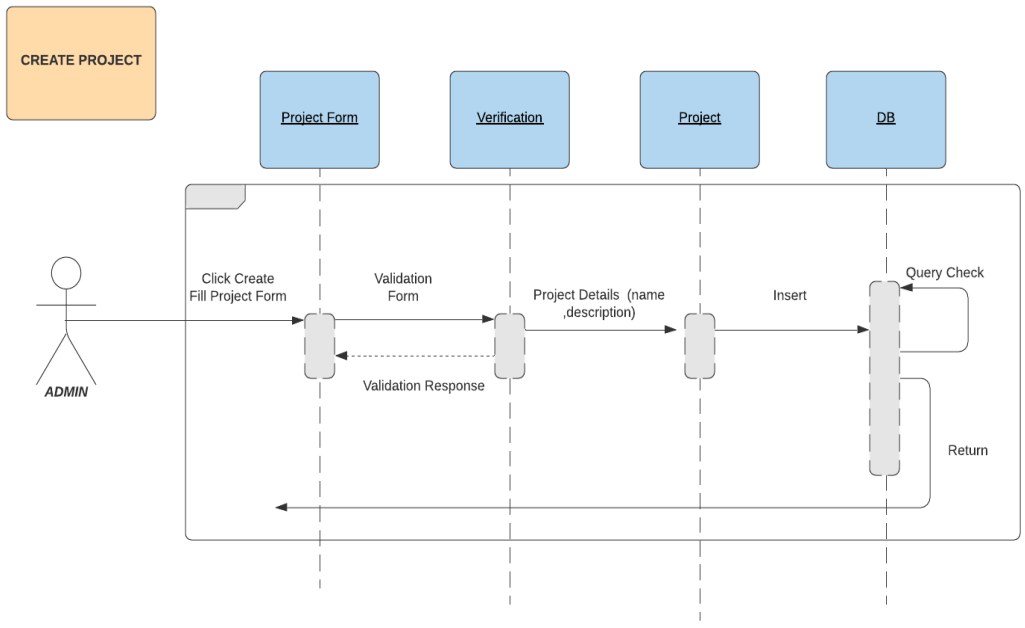
3.2 Sequence Diagram

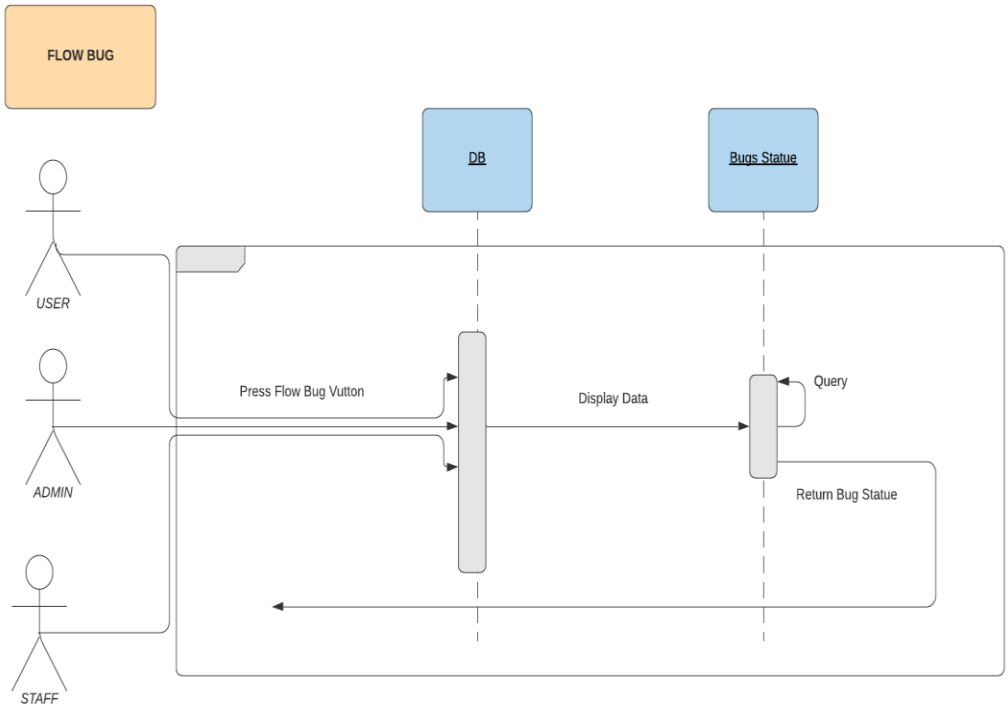


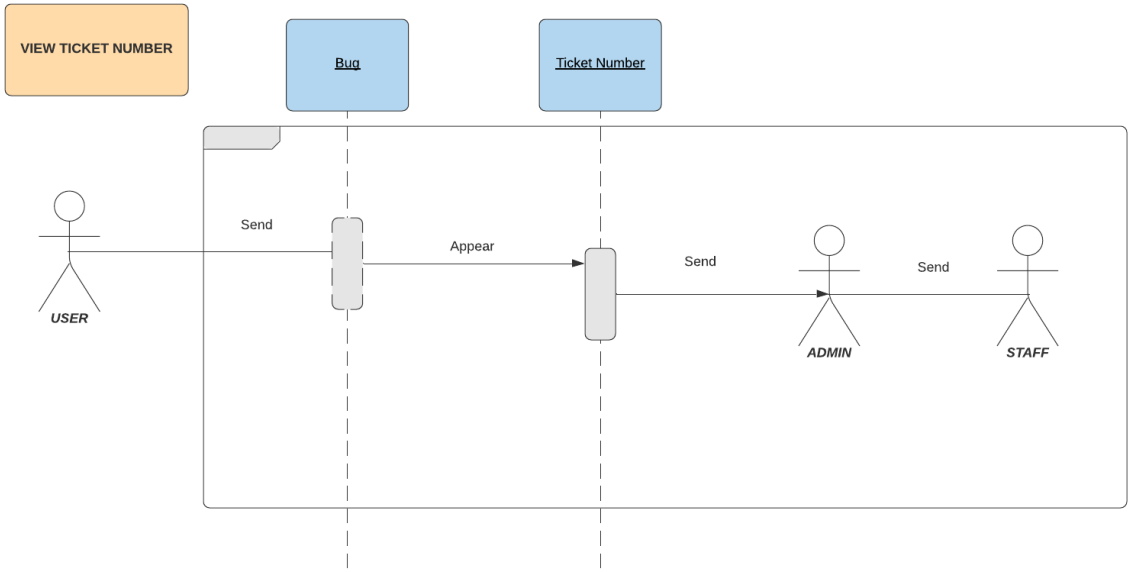


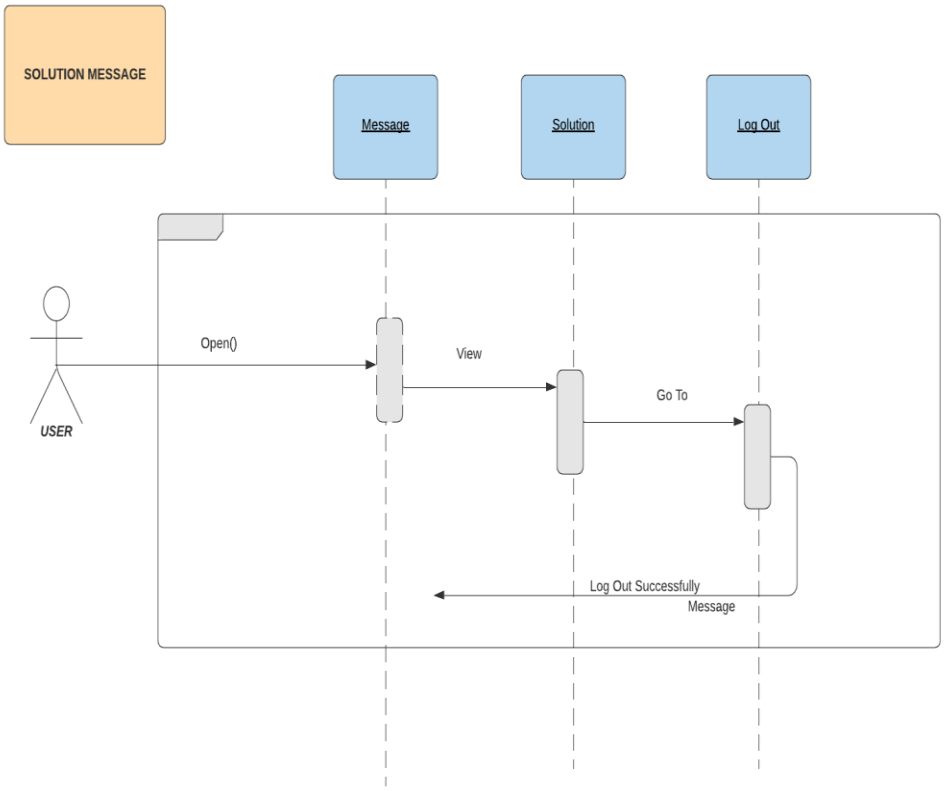

ASSIGN STAFF
Assign Staff
Staff
DB
ADMIN
Choose Staff
Click Assign
Send Ticket Number
Insert
Query Check
Return

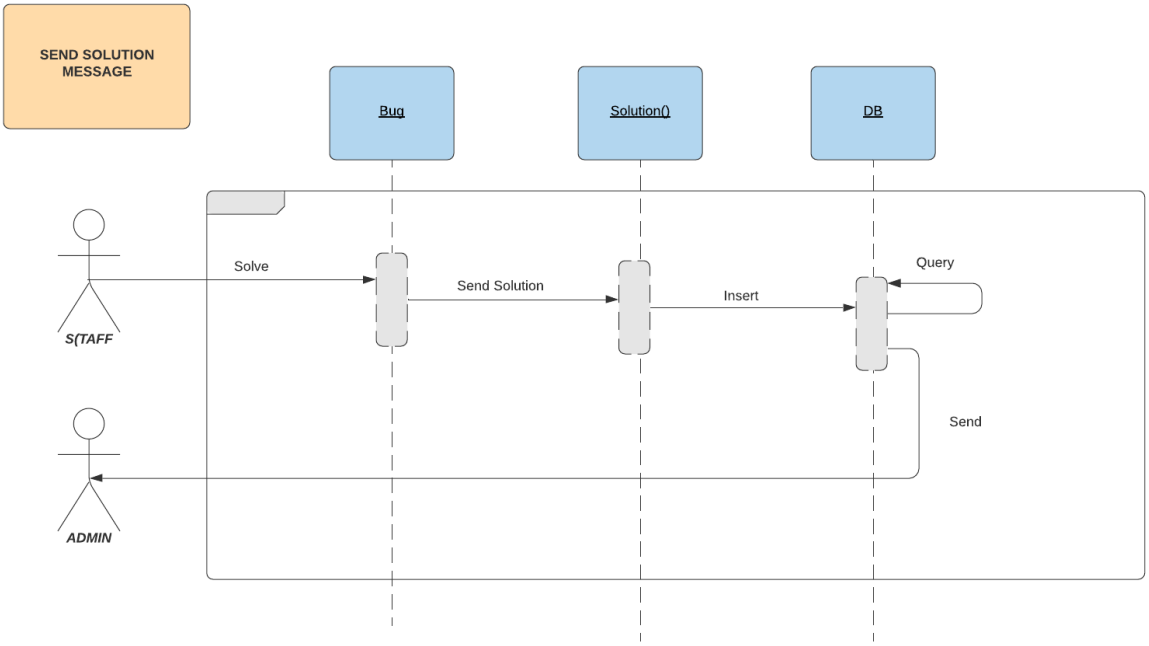

SEND BUG
Bug Form
Verification
DB
Admin
Staff
USER
Press Send Button
User Fill The Bug
Validation Form
Validation
Response
Insert
Query
View Bug
Send


CREATE PROJECT
Project Form
Verification
Project
DB
ADMIN
Click Create
Fill Project Form
Validation
Form
Validation Response
Project Details (name
,description)
Insert
Query Check
Return

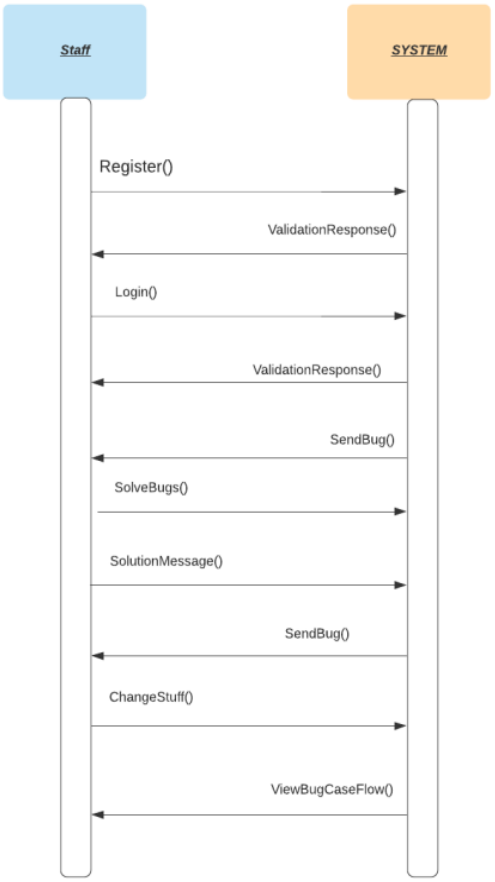

FLOW BUG
DB
Bugs Statue
USER
ADMIN
STAFF
Press Flow Bug Vutton
Display Data
Query
Return Bug Statue

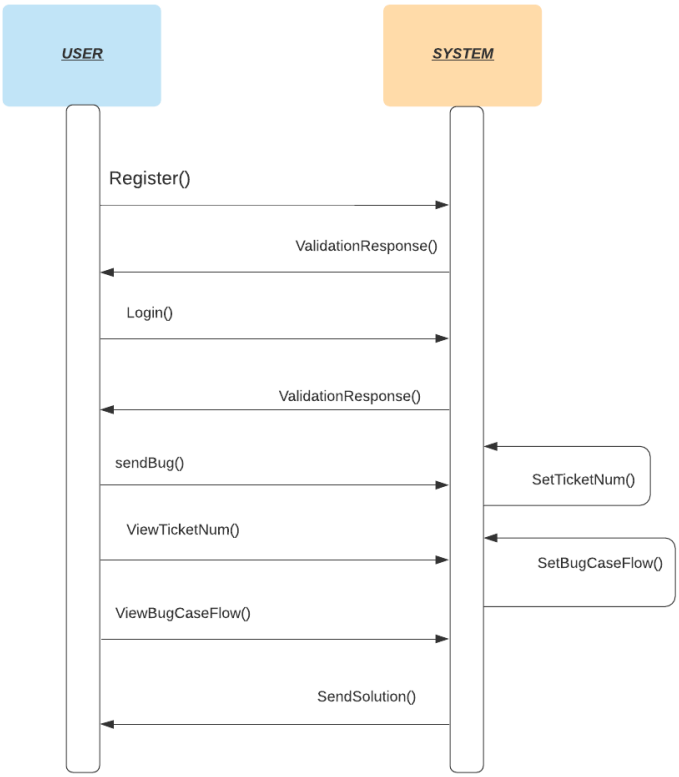

VIEW TICKET NUMBER
Bug
Ticket Number
USER
Send
Appear
Send
ADMIN
Send
STAFF

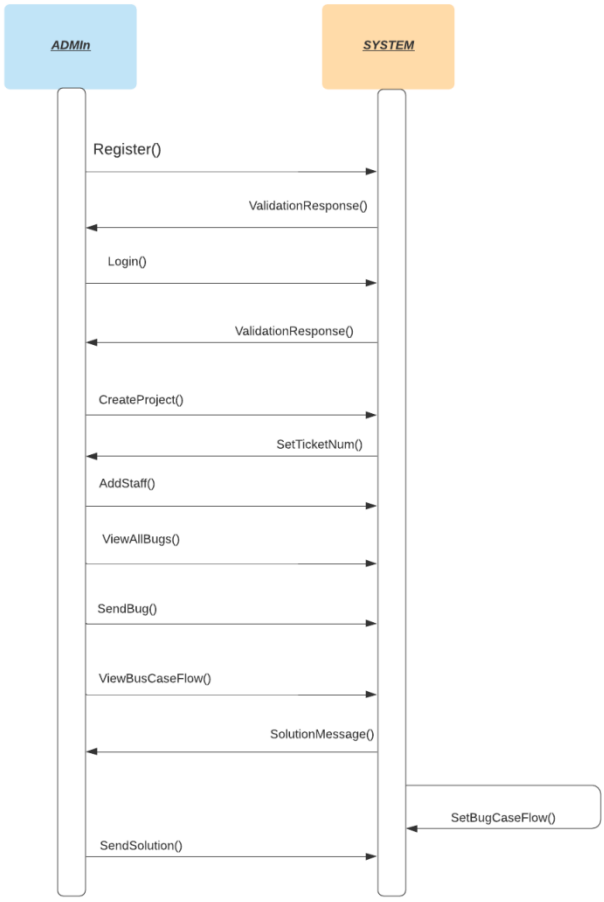

SOLUTION MESSAGE
Message
Solution
Log Out
USER
Open()
View
Go To
Log Out Successfully
Message


SEND SOLUTION MESSAGE
Bug
Solution()
DB
S(TAFF
Solve
Send Solution
Insert
Query
Send
ADMIN

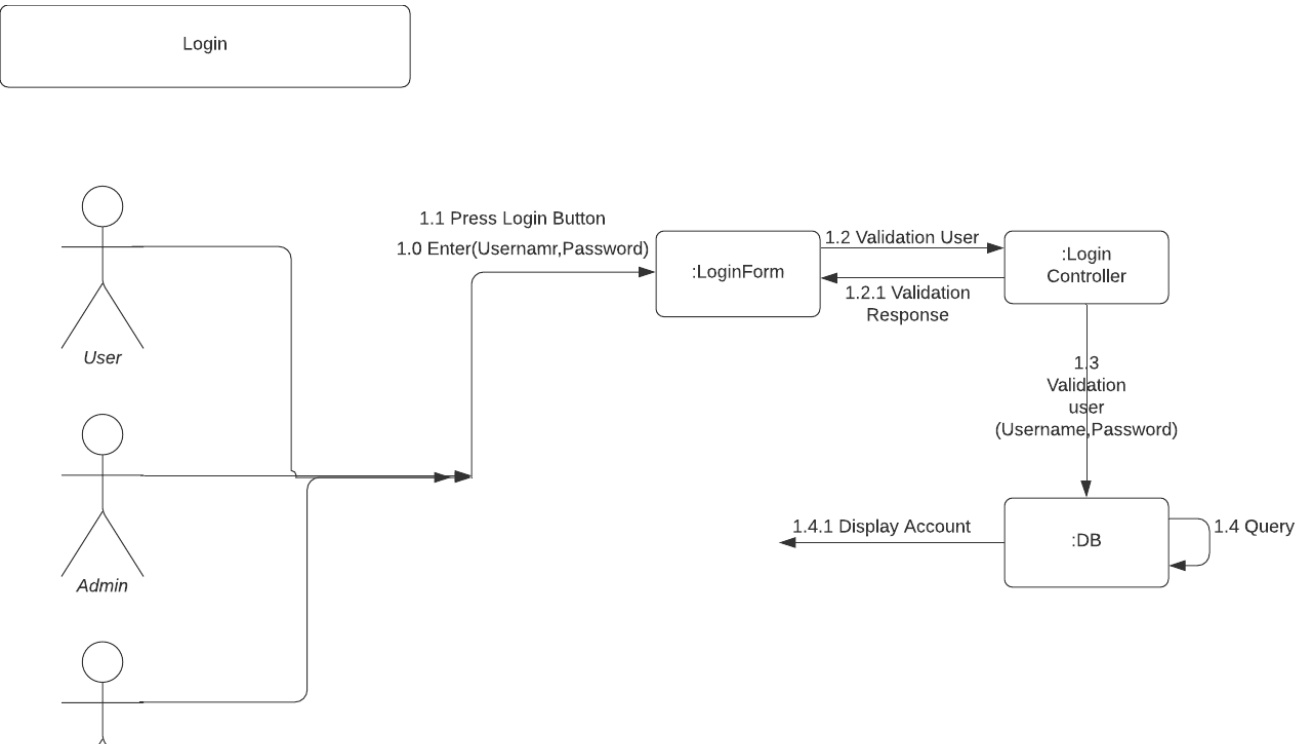
3.2.1 System Sequence Diagram

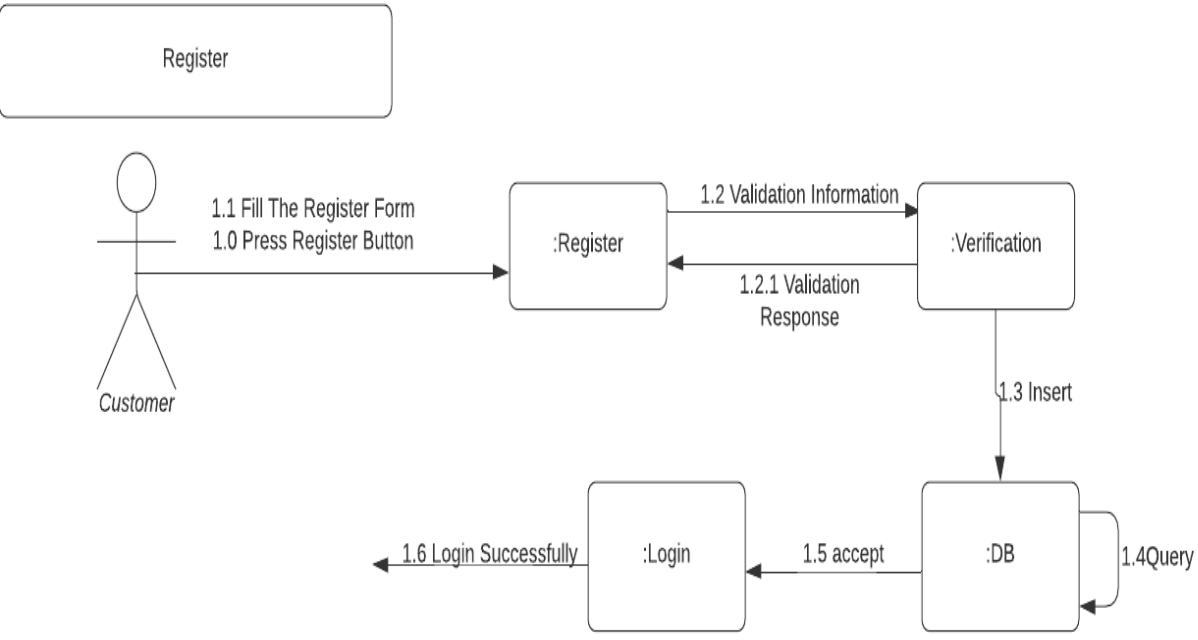



USER
SYSTEM
Register()
ValidationResponse()
Login()
ValidationResponse()
sendBug()
SetTicketNum()
ViewTicketNum()
SetBugCaseFlow()
ViewBugCaseFlow()
SendSolution()

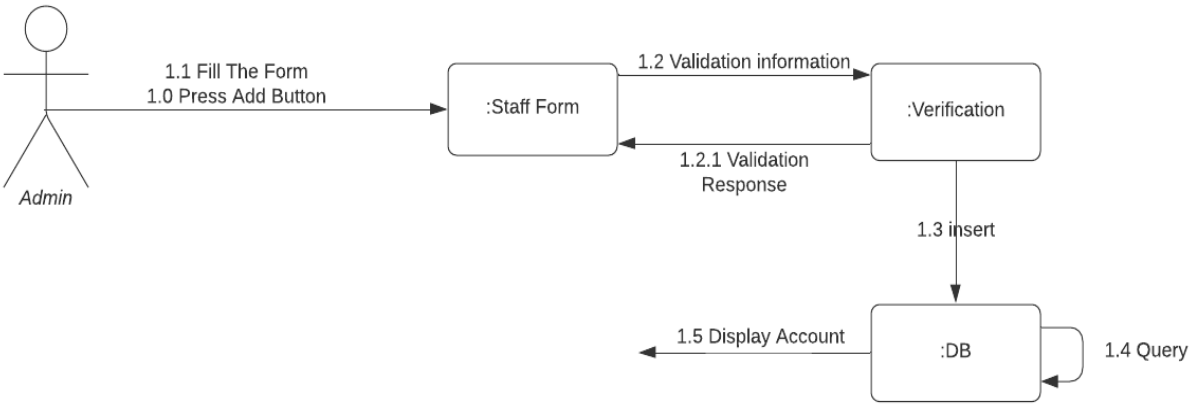

ADMIn
SYSTEM
Register()
ValidationResponse()
Login()
ValidationResponse()
CreateProject()
SetTicketNum()
AddStaff()
ViewAllBugs()
SendBug()
ViewBusCaseFlow()
SolutionMessage()
SetBugCaseFlow()
SendSolution()

3.3 Collabration Diagram

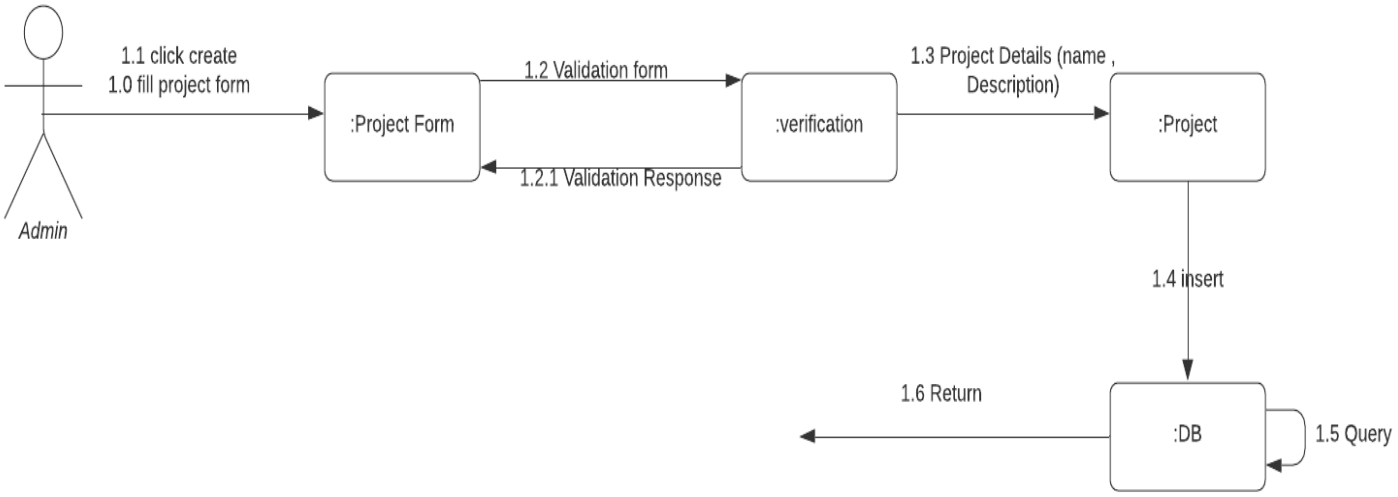




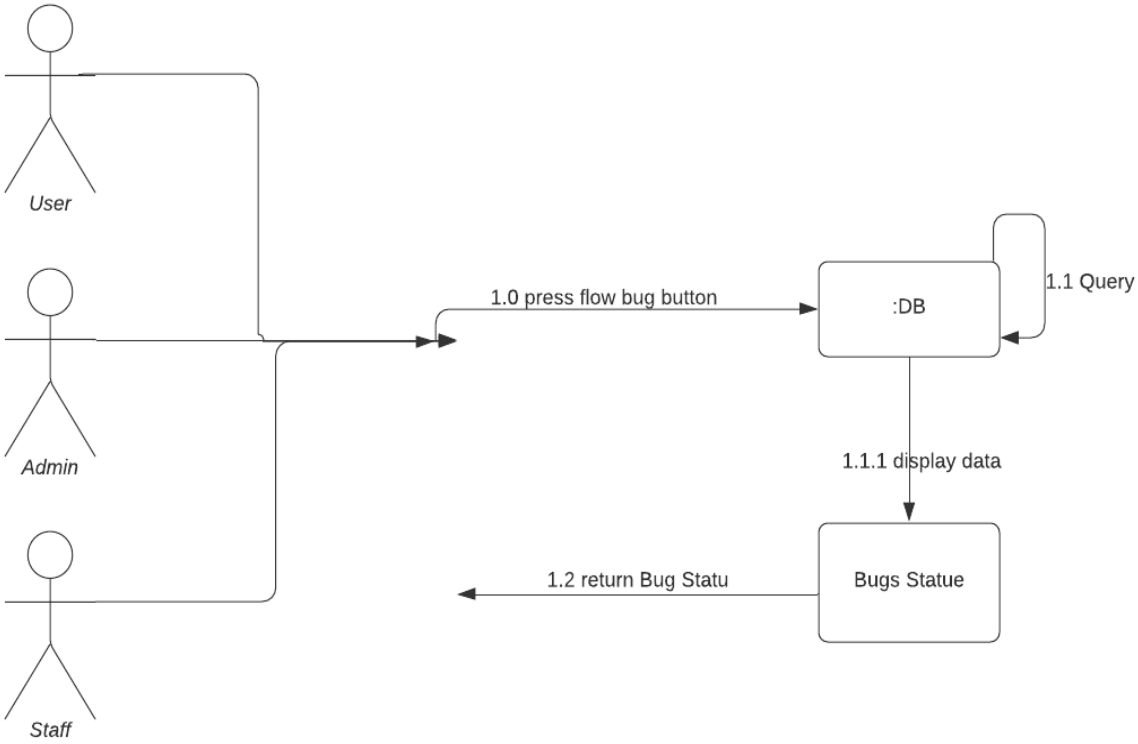
Add Staff



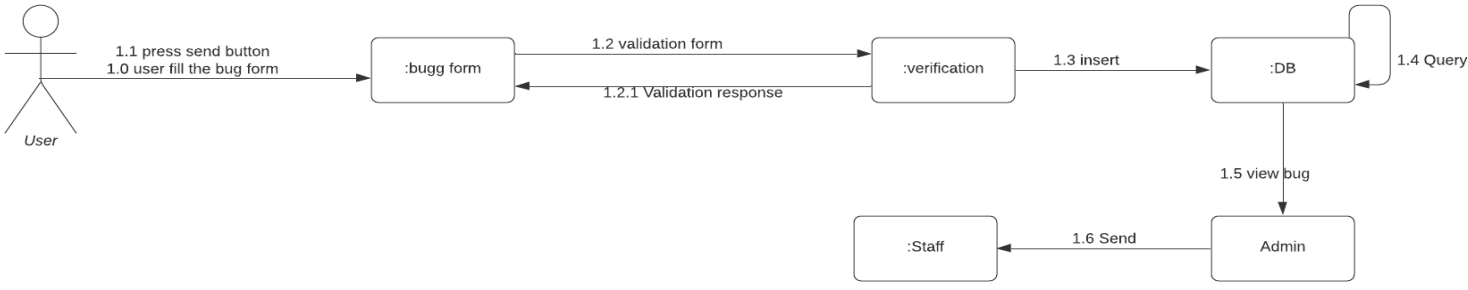
Create Project



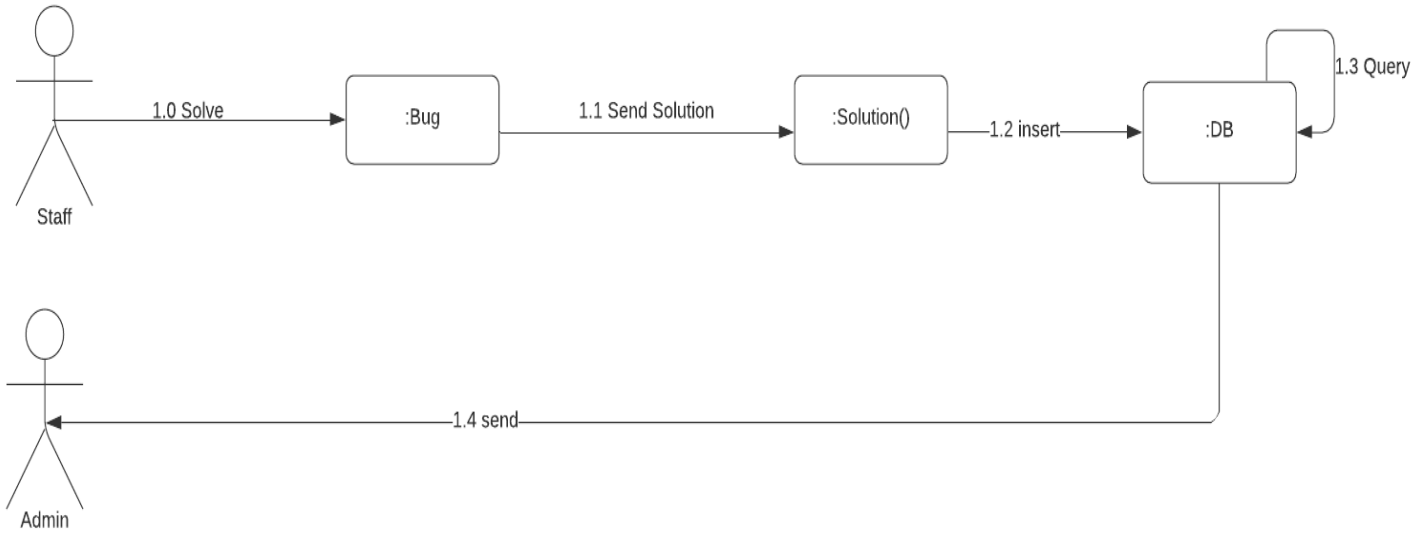
Flow Bug



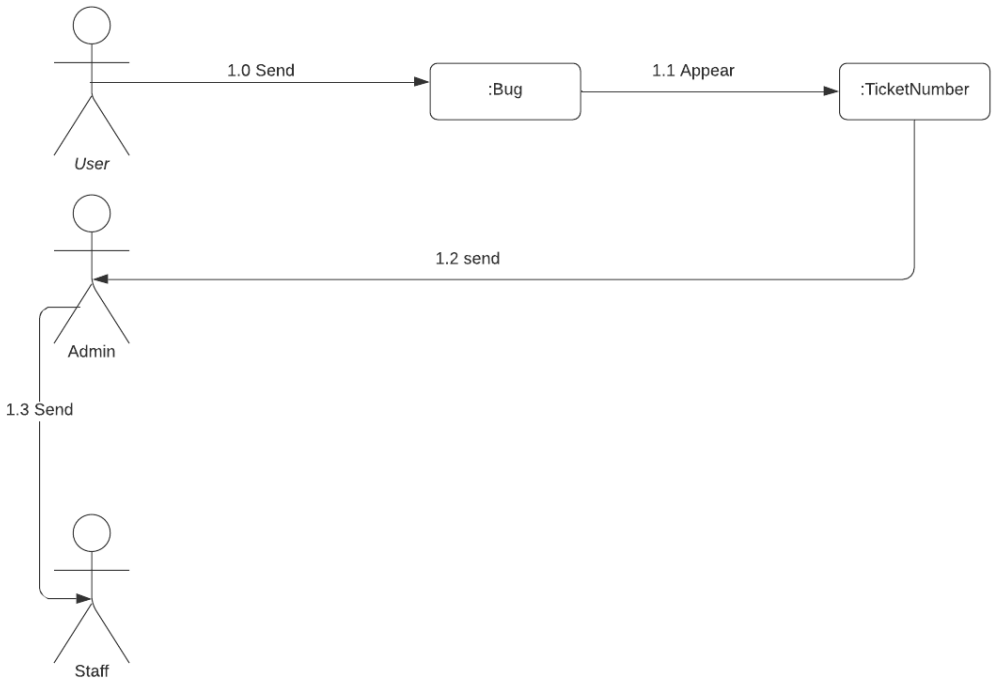
Send Bug



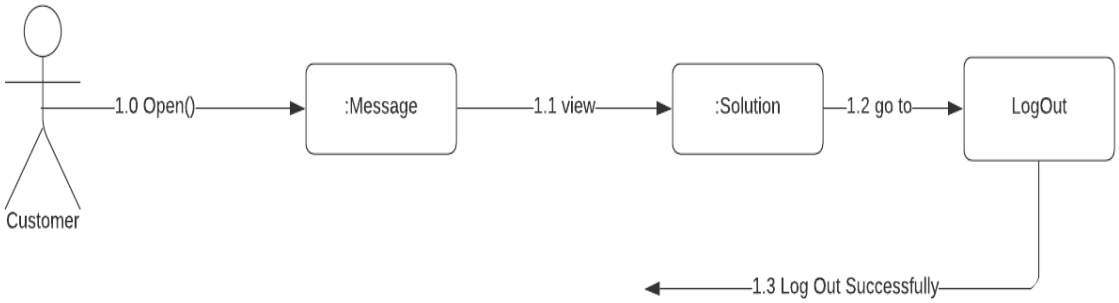
Send Solution message



View Ticket number

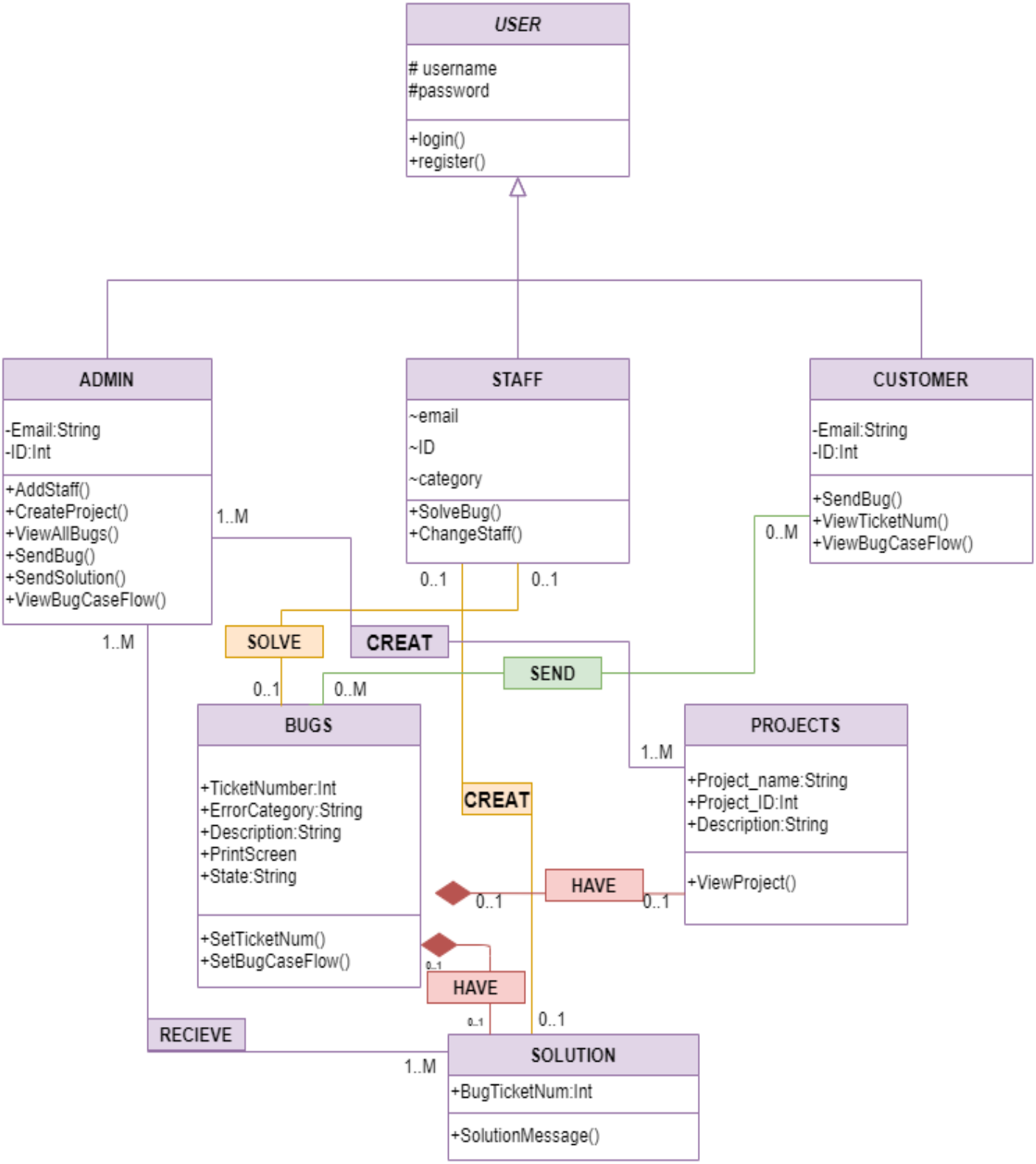


Solution Message

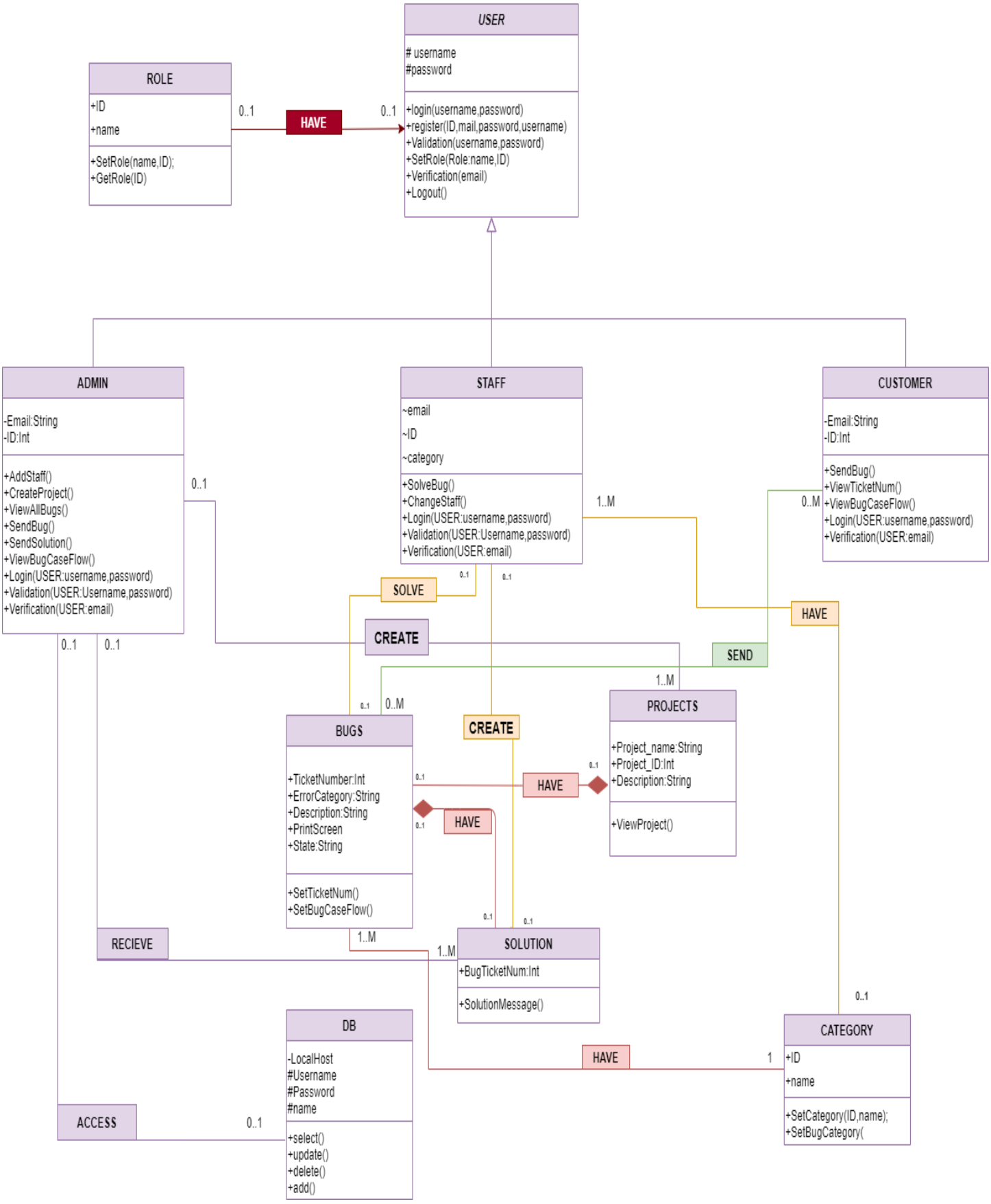


3.4 Class Diagram

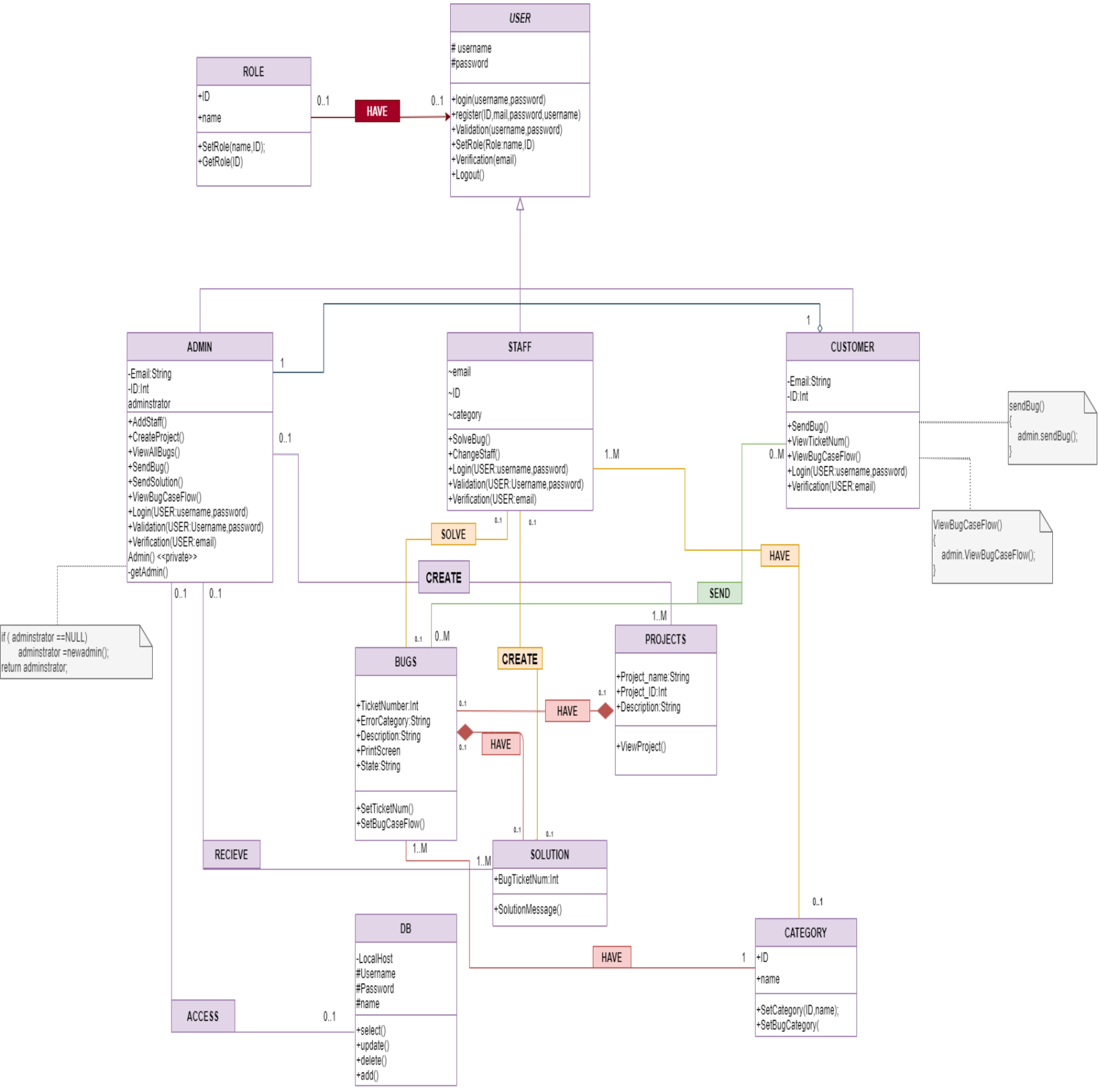
3.4.1 An Initial Version



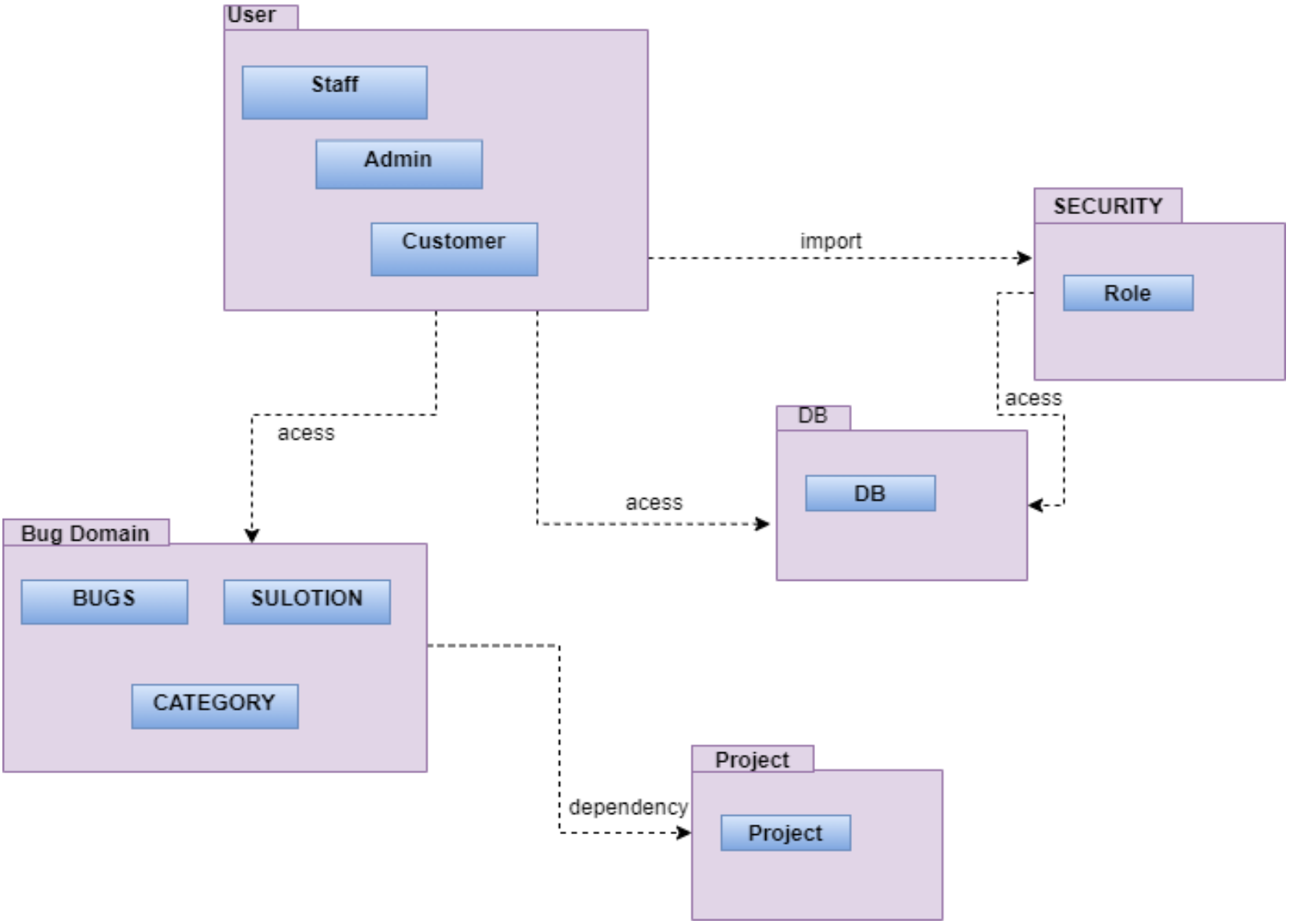
3.4.2 Intermediate Version



3.4.3 Final Version



3.5 Packet Diagram



3.6 Mandatory Design Patter Applied

1) Singleton pattern (creational):

Intent:

➢ Using when we need to ensure that only one object of a particular class need to be created.

Problem:

➢ how to make class have one object and never possible to make more?

Solution:

➢ 1- Make the constructor private.

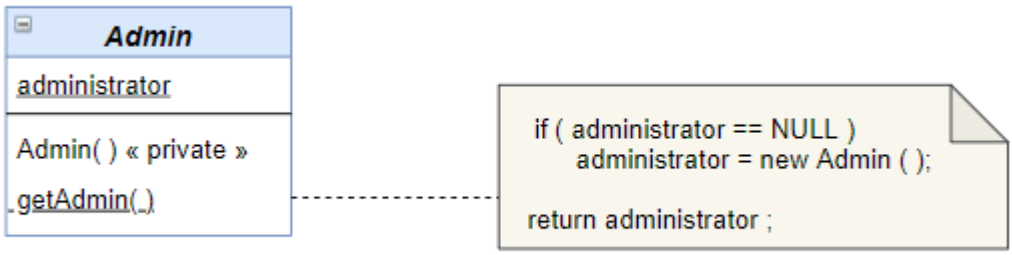
2- Make Method Public Static.

3- Make Variables Static Private.

Participants:

➢ Admin Class

Structure:



2) Delegation Pattern(Structural):

Intent:

➢ It is designing a method in the class, and then realizing that another class has a method that provides the required service.

Problem:

➢ How can you most effectively make use of a method that already exists in the other class?

Solution:

➢ the delegating method in <<customer>> class calls a method in <<Admin>> class to perform the required task. An association must exist between the <<Customer >> and <<Admin>> classes.

Participants:

➢ Customer and Admin Classes.

Consequences:

➢ Avoiding duplication of chunks of code.

➢ Minimize development cost by reusing methods.

Structure: