



This Form is for Both the General & Medical Informatics Programmes

SE-I Course Project (COVER SHEET)

Discussions Scheduled for Week 13 (Thursday, May 9th, 2024).

- Print **1** copy of this cover sheet and attach it to a printed copy of the documentation (SRS, ... etc.). You must also 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 5 grades.

ProjectName: _____

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

	ID [Ordered by ID]	Full Name [In Arabic]	Attendance [Handwritten Signature]	Final Grade
1	20210028	احمد انور عبد العال		
2	20210096	احمد محمد احمد عبد الرؤف		
3	20210110	احمد محمد مرسي حسين		
4	20211119	بفلى عزيز حلمى فهمى		
5	20211130	محمد حسن علي عبدالله		
6	20220246	صفي الدين محمد عبد الفتاح		

Grading Criteria:

14 Items		Grade	Notes
1. Functional Requirements	1		
2. Non-Functional Requirements	1		
3. Use-Case Diagram(s) including general use-cases for the system and the detailed use-cases description	2.5		
4. System Architecture – including applied Architectural Pattern(s)	1		
5. Activity Diagrams	2		

Page 1 of 2



6. Object Diagrams (Including object diagrams that illustrate the preconditions and the post-conditions of selected functions)	2		
7. Package Diagram(s)	1		

8. Sequence Diagrams <i>including System Sequence Diagrams (SSDs)</i>	2.5		
9. Database Specification <i>(ERD, Tables)</i>	2		
10. Collaboration/Communication Diagrams	2		
11. Class Diagram <i>(3 versions)</i> 1. An initial version based on the requirements and Use-Case/Activity diagrams. 2. An intermediate version based on the interaction diagrams. 3. A final version after applying the design patterns and other modifications.	6		
12. Three Mandatory Design Patterns Applied <i>(Including a typed description)</i>	3		
13. Front End Design for all Functions <i>(HTML, Bootstrap)</i>	2		
14. Implementation <u>based on the submitted Requirements & Design.</u> Should include at least 4 of the following modules (in addition, of course, to modules specific to your projects): 4. User Role Management Module. 5. User manipulation Module <i>(Login, Add / Delete / Update / Search, List).</i> 6. Controlling Resources Module <i>(Rooms, Orders, Products, ... etc.).</i> 7. Reservation and Rescheduling Module. 8. Generating Reports Module <i>(PDFs, ... etc.).</i> 9. Sending Emails or Notifications Module.	7		

Teaching-Assistant's Signature: _____

Table of contents

Part 1: Overview & Software Requirements Specification

1- Introduction: -----	
1.1- Purpose: -----	
1.2- Scope: -----	
1.3- Glossary & Abbreviations: -----	
1.4- Stakeholders: -----	
1.5- References: -----	
2- Functional Requirements: -----	

- 2.1 User Requirements Specification: -----
- 2.2 System Requirements Specification: -----
 - 2.2.1 natural language: -----
 - 2.2.2 structured natural language: -----
 - 2.2.3 Design description languages: -----
 - 2.2.4 Graphical notations: -----
 - 2.2.5 Mathematical specifications: -----
- 2.3 Requirements' Priorities: -----
- 3- Non-Functional Requirements: -----
 - 3.1 Categories of Non-Functional Requirements: -----
 - 3.2 Non-Functional Requirements Specification: -----
 - 3.3 The fit criteria for every Non-Functional Requirement: -----
 - 3.4 How would the above-mentioned Non-Functional Requirements affect the System's overall Architecture: -----
- 4- Design & Implementation Constraints: -----
- 5- System Evolution: -----
 - 5.1 Anticipated changes: -----
 - 5.2 anticipated changes in the future: -----
- 8- requirements discovery approaches: -----
- 9- requirements validation techniques: -----

PART 2: System Design & Models

- 1- Functional Diagrams: -----
 - 1.1 Use-Case Diagram(s): -----
 - 1.2 Detailed Use-Cases Description: -----
 - 1.3 Package Diagram grouping relevant Use Cases into Packages: -----
- 2- Structural & Behavioral Diagrams: -----
 - 2.1 System Architecture: -----
 - 2.2 Activity Diagrams: -----
 - 2.3 List of User Interfaces required for the System and the corresponding users of each interface: -----
 - 2.4 Class Diagram V.1: -----
 - 2.5 Sequence Diagram(s): -----
 - 2.6 System Sequence Diagrams (SSDs): -----
 - 2.7 Collaboration/Communication Diagram(s): -----
 - 2.8 Use cases strategy: -----
 - 2.9 Class Diagram V.2: -----
 - 2.10 Design Patterns Applied: -----
 - 2.11 Class Diagram V.3: -----
 - 2.12 design smell: -----
 - 2.13 Class Structuring Criteria: -----
 - 2.14 Forks or Cascades: -----
 - 2.15 Package Diagram grouping relevant Classes into Packages: -----
 - 2.16 Object Diagrams: -----
 - 2.17 Database Specification: -----
 - 2.17.1 ERD: -----
 - 2.17.2 Tables: -----

PART 3: Development Phase (Implementation Details)

1- Front-End Design for all Functions (HTML, Bootstrap): -----

Part 1

Overview & Software Requirements Specification

Introduction

Purpose:

- The purpose of this system (defect tracking application) is to efficiently track software bugs observed by customers who use the system.

Scope:

- The scope of the bug tracking system is software that helps customers to have the solution for bugs of the system's created projects. Also, it will help developers (staff) to know bug details to solve it.

So, the system is a relationship between customers with their bugs and the staff who will solve those bugs. Under control the admin who manages which bugs are assigned to whom from the staff.

Glossary: [OBJ]

Term	definition
Software requirement	A capability or a condition that must be met to solve a certain problem or make specific achievement.
Functionality	The capabilities of a system as stated by its functional requirements.
Functional requirement	Something (Functionality) that the software does.
Non-functional requirement	Something (Functionality) that the software does without any additional technical effort.
Actor	1. Generally, in RE: A person, a system, or a technical device in the context of a system that interacts with the system. 2. Especially in goal oriented RE: a person, a system, or a technical device that may act and process information to achieve some goals.
Class	Represents a set of objects of the same kind by describing the structure of the objects, the ways they can be manipulated and how they behave
Class diagram	A diagrammatic representation of a class mode
Activity diagram	A diagram type in UML which models the flow of actions in a system or in a component including data flows and areas of responsibility where necessary
Entity-relationship diagram	A graphic representation of an entity-relationship model.
Scope (of a system)	The range of things that can be shaped and designed when developing a system.

Scenario	1. A description of a potential sequence of events that lead to a desired (or unwanted) result.
Sequence diagram	A diagram type in UML which models the interactions between a selected set of objects and/or actors in the sequential order that those interactions occur.
UML	Abbreviation for Unified Modeling Language, a standardized language for modeling problems or solutions.
Use case	A description of the interactions possible between actors and a system that, when executed, provide added value.
Use case diagram	A diagram type in UML that models the actors and the use cases of a system.
State machine	A model describing the behavior of a system or component by a finite set of states and state transitions. State transitions are triggered by events and can in turn trigger actions and new events.
Stakeholder	A person or organization that has a (direct or indirect) influence on a system's requirements.
Requirements template	A blueprint for the syntactic structure of individual requirements. A phrase template is a specific requirement template for requirements written in natural language.
System requirement	A requirement of a system or to a component of a system.
Validation (of requirements)	The process of checking whether documented requirements match the stakeholders' needs.
Priority (of a requirement)	Documents the importance of a requirement in comparison to other requirements according to given criteria.

Abbreviations:

ER	Entity-Relationship
ERD	Entity-Relationship Diagram
RE	Requirements Engineering

SRS	Software Requirements Specification
UML	Unified Modeling Language
WMC	Weighted Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Objects
RFC	Response for Class
LCOM	Lack of Cohesion of Methods
IEEE	Institute of Electrical and Electronics Engineers
HTML	Hypertext Markup Language
SRS	Software Requirements Specification

System Stakeholders:

Stakeholders	Category	DESCRIPTION
Admin	Internal	Someone who is responsible for adding staff to the site and assign bugs to those staff and manage the website.
Staff	Internal	Someone who can solve the bugs of the customers.
Customer	External	Someone Add bug details to the App.

References:

- 830-1984 - IEEE Guide for Software Requirements Specifications
- **(Book)Software requirement patterns: By Stephen Withall**
- **(Book, as quotes) The Timeless Way of Building (1977): By Christopher Alexander**

Functional Requirements:

➤ User Requirements Specification:

There are three actors in our system:(Admin – Staff – Customer).

- A user shall be able to:
 - Log in to the system.
 - Log out from the system.
 - Manage profile.

- An administrator shall be able to:
 - Register for the application.
 - Manage staff members.
 - Manage projects.
 - View bugs sent by the customers and their case-flow status & details.
 - Assign the work to (a) staff member(s).
 - Send a solution message to the customer.
 - Get a project report.

- Staff members shall be able to:
 - View assigned bugs.
 - Staff will be able to view bug flow.
 - Assign bugs to other staff members if the bug is related to them.
 - Send solution messages to the admin.
 - Change bug status.

- A customer shall be able to:
 - Register for the application.
 - Manage bug.
 - Send bug details with a print screen to the admin.
 - Monitor bug case-flow details, status, and solution details using the ticket number that is generated during bug creation.
 - Manage feedback.

➤ System Requirements Specification:

- Customer:

First customer register to our App (if he /she did not register before) by entering username and password if the username is already existing show error message if not show a registered successfully message and back to home page then login to our system then check the role and the open customer's pages. After the customer had logged to our website, the customer can add a new bug by selecting the project name which he/she found the bug then enter the bug details(error category – error details) and

upload a print screen of the bug generated from the project or the software .after the customer fill all details of the bug the system then presses the button (send) the system will generate a ticket number for the new bug and a view it in a table at this page. customer can use this ticket number for searching for the bug case flow details by put the bug's ticket number and press on the button (search), then the system will return all data about the bug then the customer can see that data (the status of bug – bug details).if the status of the bug (solved show solution message), press on the button (show the solution) and see the solution message if not he/she is waiting for the solution. The customer can give feedback after seeing the solution and the customer can log out by pressing on the button at the sidebar called (log out).

- Admin:

Admin login to the system by entering the username and the password. The system will check the role and verify the username and the password is correct or not and then the system will open the admin pages if not it will show an error message. Admins can create new projects by entering (description – name) which system the staff can work on it. The admin can add new categories of staff by entering the category name and press on the button (Add category) and he can delete and update the category and message will show that he is doing this operation is done successfully. The admin can add a new member to the Staff by entering staff details (username – password – E-mail ID – select category) then press on the button (Add) and message will show the staff is added successfully if the admin enter all the fields and he can also delete staff from the system. Admin can view all new bugs' details (project name- bug details – ticket number - print screen from the bug-error category) which sent from the customer then the admin selects the category of staff who can solve the bug then select one of them and put the due date for solving the bug. Admin can also view the customer's data. Admin can also view the staff solution then approved it then send a solution message to the customer. He can also send emails to the staff to communicate with them. He selects the staff name, writes a message, presses the button (send), and the system will show that the message is sent successfully. Then he/she logs out by pressing on a button at the sidebar called (log out).

- Staff:

Staff can login to the app using username and password and check for the role and verify if the username and the password are correct or not. If they correct the system will open the staff pages. If not, the error message will be shown. Staff can also view the bugs assigned to them. And show the details of the bug. and can also assign the bugs to other staff if the bug is related to them by selecting the category and select the member and press on the

button (assign). And can send the solution to the admin waiting if it approved or not if it is not, he will see the admin comment and resolve the bug again or assign it to another staff. Then he/she logs out by pressing on a button at the sidebar called (log out).

➤ **Requirements' Priorities:**

Requirement Name	Login & Register
Requirement ID	LR
Actor	Admin, Staff, Customer
Priority	Must
LR 1	The system must allow the admin to “login” with Username and password.
LR 2	The system must allow the staff to “log in” with Username and password.
LR 3	The system must allow the staff to “login” with Username and password.
LR4	The system must “verify” all user ‘s Username and password.

Requirement Name	Manage Staff
Requirement ID	MS
Actor	Admin
Priority	Should
MS 1	The system must allow the admin to “Add” Staff ‘s Category
MS 2	The system must allow the admin to “Enter” Staff ‘s (username – password – E-mail ID)
MS 3	The system must allow the admin to “select” staff’s category
MS 4	The system must allow the admin to “delete” staff
MS 5	The system must allow the admin to “update” staff

Requirement Name	Send Bugs
Requirement ID	SB
Actor	Customer
Priority	Should
SB 1	The system must allow the customer to “enter” bug details (selecting project name – error category – error details)
SB 2	The system must allow the customer to “upload” print screen of the bug
SB 3	The system must allow the customer to “have” a ticket number to flow the bug.

Requirement Name	Assign Staff
Requirement ID	AS
Actor	Admin, Staff
Priority	Must
AS 1	The system must allow the admin to “assign” the staff who will solve the bug by category
AS 2	The system must allow the staff to “assign” other staff the bug is related to them

Requirement Name	Solution Message
Requirement ID	SM
Actor	Admin, Staff, Customer
Priority	Must
SM 1	The system must allow the admin to “see” the solution messages which he sent before
SM 2	The system must allow the admin to “Approve” or “Refuse” the solution of the staff
SM 3	The system must allow the admin to “send” the solution of the customer’s bug to him
SM 4	The system must allow the customer to “show” the solution messages from staff or admin
SM 5	The system must allow the customer to “show” the solution messages from staff or admin

Requirement Name	Manage Project
Requirement ID	MP
Actor	Admin
Priority	Must
MP 1	The system must allow the admin to “Put” the project name
MP 2	The system must allow the admin to “put” the project description
MP 3	The system must allow the admin to “view” all projects is created
MP 4	The system must allow the admin to “delete” projects
MP 5	The system must allow the admin to “update” projects

Requirement Name	View Bugs
Requirement ID	VB
Actor	Admin, Staff
Priority	Must
VB 1	The system must allow the admin to “see” the new bugs which sent from the customer
VB 2	The system must allow the staff to “see” the new bugs which assigned to them

Requirement Name	Monitor Bug Case Flow
Requirement ID	MBCF

Actor	Customer
Priority	Should
MBCF 1	The system must allow the Customer to “ Search“with the ticket number to flow the bug
MBCF 2	The system must allow the Customer to “ View“the bug details and the status of the bug

Requirement Name	Give Feedback
Requirement ID	GF
Actor	Customer
Priority	Could
GF 1	The system must allow the customer to “Give” the feedback of the solution

Requirement Name	Get Report
Requirement ID	GR
Actor	Admin
Priority	Could
GR 1	The system must allow the admin to “get” report of all the bugs which that

Non-functional requirements

Categories:

- Product requirement
 - Requirements that specify that the delivered product must behave in a particular way like execution speed, reliability, etc.
- Organizational requirement
 - Requirements which are a consequence of organizational policies and procedures like process standards used, implementation requirements, etc.
- External requirement
 - Requirements that arise from factors that are external to the system and its development process like interoperability requirements, legislative requirements, etc.

Requirements Specification:

Requirement Name	Performance Requirements
Requirement ID	PR
Category	Product requirement
PR 1	The system must respond to the business operation in less than 5 seconds for the user
PR 2	The system should be compatible with all modern browsers
PR 3	The system should response the operation messages to the users within 2 seconds
PR 4	A website should be capable enough to handle 20 million users without affecting its performance

Requirement Name	Safety and Security Requirements
Requirement ID	SSR
Category	External requirement
SSR 1	The system cannot affect, harm damage to the user. it also cannot damage the user's computer while accessing the system over a network
SSR 2	The system must handle safe login and logout through the session
SSR 3	Hashing technology should be used to handle secure login for users. By using user role management
SSR 4	The database should be secured from my SQL injection to prevent leak or loss of information
SSR 5	The database should be secured from my SQL injection to prevent leak or loss of information
SSR 6	The system could use SSL (Secure socket layer) certificates to secure the data being transmitted

Requirement Name	Reliability
Requirement ID	R
Category	Product requirement
R 1	The system should be designed in a modular manner to ease in software maintenance .by designing modularly, we can reduce coupling allowing each module to perform a specific function
R 2	The program should be reliable and provide catching of exceptions so that unintended results do not occur such as system crashes, or data validation

Requirement Name	Maintainability
Requirement ID	M
Category	Product requirement
R 1	The product shall be able to be modified to cope with a new class of users
R 2	These are some metrics used to measure maintainability: <ul style="list-style-type: none"> • Technical debt. • Code smells. • Cyclomatic Complexity.

	<ul style="list-style-type: none"> Source Lines of Code (SLOC)
--	---

Requirement Name	Usability
Requirement ID	U
Category	Product requirement
U 1	The user interface shall be intuitive and accessible to all users.
U 2	The system should have a user - friendly interface to help the user to ease use the system

Non-Functional Requirements affect the overall Architecture of the System:

Of course they can change the whole software architecture, because they:

- Help in maintaining the website and find minor issues and fix them as soon as possible when the user reports an issue he faced and distributes his experience.
- ensure good user experience and ease of operating the software.
- help in formulating the security policy of the software system.

Design & Implementation Constraints

- The system is developed using (HTML5, CSS, JavaScript, bootstrap) as the tools for Frontend. Backend and data management is developed using pure PHP only, without any special, or additional framework to do the job.
- For databasing, and data management, we use SQL, why? Because data is a bit complicated to store as non-SQL services or JSON data type for storing. For instance, users have reviews, reviews have comments, or/and ratings. In some cases, we would like to search for a specific thing like comments review, or we can view a specific value to view, and of course, this is a bit complex data to store as something different than SQL
- Software Requirement:
 - Windows 7 or higher
 - Visual Studio
 - MySQL Server
 - Google Chrome Browser
- Hardware Requirement:
 - i3 Processor-Based Computer or higher
 - Memory: 1 GB
 - Hard Drive: 50 GB

- Monitor
- Internet Connection
- the software is that it will always be used on desktop computers & laptops so all Actors included in the software should be aware of computers & English language

System Evolution

Anticipated changes:

- Our software is very able to change easily, especially because we kept in mind the manageability and reliability as a non-functional requirement, so we can make good use of the feedback section for updating UI, or even the functionality. Changing one of the main functionalities is an anticipated change, for instance, the user might not like the way the feedback process is happening, or even the way of solving bugs or the solution is not effective, so we can develop our system and adding new modules to make the output of our system more effective
- Any software updates on time, and this project is no special case We can add new modules or actors like:
 1. Testers:
 - Who can test the solution of the bug and decide If this solution effective or not
 2. More Admins:
 - We can make for every category of staff one admin who can manage this category.
 3. Super Admin:
 - Who can manage the environment between the admins and staff
 4. Expert staff:
 - They can get effective solutions for any bug with a good quality and speed reply
 5. Communication module:
 - We can make the staff communicate with each other and their admins
 6. Projects module
 - We can have many projects to solve their bugs and we can add a payment for buying new projects.

anticipated changes in the future:

- Some changes in the future might not be handled with the current design. If we would like to make the commenting section on every solution and feedback section, we will have to add more architecture to observe and interfaces for any additional features or we might change the current architecture without adding a new one if we wanted to use SOCKETs for instance

requirements discovery approaches

We will use in our system two approaches of requirements discovery are.

1) Use cases approach

- A set of use cases should describe all possible interactions with the system.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system

2) Scenarios approach

Requirements Validation Techniques

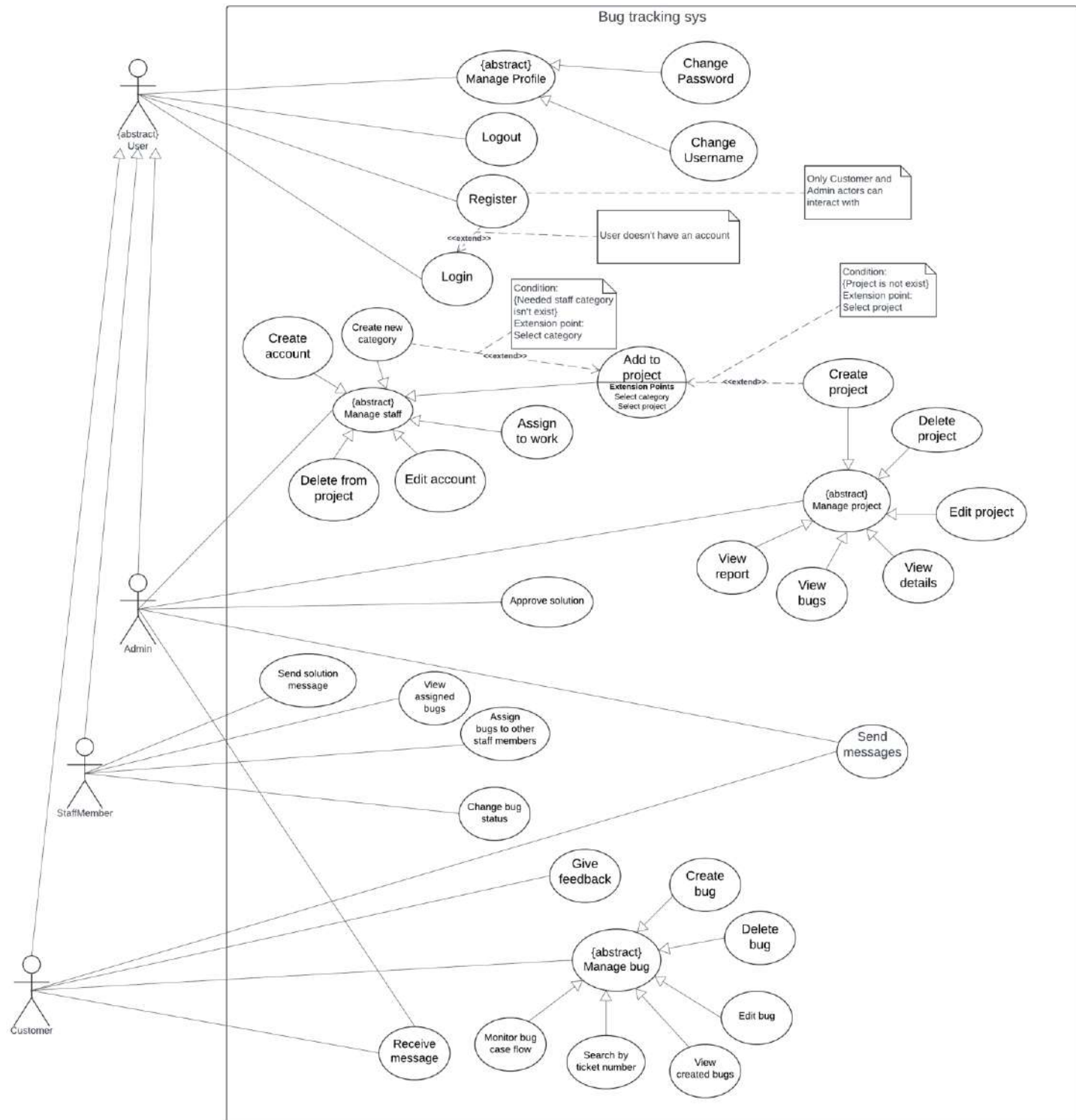
- Requirements validation
 - is the process that checks the requirements defined for development and defines the system that the customer wants.
- We perform requirements validation to check the issues related to the requirements.
- The requirements validation is used to check the error at the initial phase of the development as the error may increase when detected later in the development process.
- The requirements must be based on the available budget.
- In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software requirements specification (SRS)
- Several techniques used either individually or with other techniques to check all the system or part of it
 - 1) Test-case generation
 - The requirements mentioned in the (SRS) document should be testable.
Generally believed that if the test is impossible to design, this usually means that the requirements will be difficult to implement
 - 2) Requirements Reviews
 - The SRS is reviewed by a group of people; the reviewer will analyze the document to check the error and ambiguity
 - 3) Prototyping
 - In this validation technique, the prototype of the implemented system is presented before the client or end-user; they experiment with the prototype and check if it meets their need or not. The main goal of this type of model is to collect

Part 2

System Design & Models

Functional Diagrams

Use-Case Diagram:



Detailed Use-Cases Description:

Name:	Add to project.
Goal:	Adding staff member(s) to the project.
Preconditions:	Admin must be loggedin to the system.
Postconditions:	staff member(s) (was/ were) added to the project successfully.
Initiator(s):	The Admin.
Trigger:	The Admin requires adding staff member(s) to a project.
Standard Process:	(1) Admin selects project from list of projects and click on manage button. (2) Admin clicks on add staff button. (3) Admin types staff username(s). (4) Admin clicks on the add button. (5) The system checks username and finds that it exists on the system and is not added to the current project. (6) staff member(s) will be added to the project.
Alternative processes:	(5.a) username doesn't exist on the system or is already added to the project. (5.a.1) The system will display error message << "Username not found" >> or << "Username is already added to the project" >>. (5.a.2) Repeat from step 3.

Name:	Delete from project
Goal:	Deleting staff member from the project.
Preconditions:	Admin must be loggedin to the system.
Postconditions:	staff member was deleted from the project successfully.
Initiator(s):	The Admin.
Trigger:	The Admin requires deleting staff member from the project.
Standard Process:	(1) Admin selects project from list of projects and click on manage button. (2) Admin clicks on delete staff button. (3) Admin types staff username. (4) Admin clicks on the delete button. (5) The system checks username and finds that it was added to the current project. (6) staff member will be deleted from the project.
Alternative processes:	(5.a) username doesn't exist on the current project. (5.a.1) The system will display error message << "No staff member with that username" >> (5.a.2) Repeat at step 3.

Name:	Change bug status
Goal:	Staff members can change bug status from Open to in progress to indicate that they are working on fixing the bug or fixed to indicate that the bug is fixed and waits to be verified by the admin.
Preconditions:	Staff members must be logged in to the system.
Postconditions:	Bug's status was changed to in progress or fixed .
Initiator(s):	Staff member.
Trigger:	A staff member has initiated action to update the status of the bug.
Standard Process:	(1) A staff member selects a project from the projects list. (2) Staff member views assigned bugs. (3) A staff member changes the bug's status to open or fixed (4) The system shall send an email to inform the customer that the bug's status was changed and send an email to the admin only if the status of the bug is fixed to inform him that the bug waits to be verified .

Name:	Search by ticket number
Goal:	Finding the bug that match the ticket number.
Preconditions:	Customer must be loggedin to the system.
Postconditions:	None.
Initiator(s):	The customer.
Trigger:	The customer needs to track specific bug by its ticket number.
Standard Process:	(1) The customer opens the created bugs page. (2) The customer searches for the bug by its ticket number which is generated by the system when the bug was created. (3) The system finds the bug with that ticket number and shows to the customer the bug's details and its flow.
Alternative processes:	(3.a) The system doesn't find the bug. (3.a.1) The system will display error message << "No results" >> (3.a.2) Repeat from step 2.

Name:	Edit bug
Goal:	Editing bug's info.
Preconditions:	Customer must be loggedin to the system.
Postconditions:	bug's info has been edited successfully.
Initiator(s):	The customer.
Trigger:	The customer needs to edit bug info.
Standard Process:	(1) The customer opens the created bugs page. (2) The customer searches for the needed bug. (3) The system finds the bug and show it to the customer. (4) The customer clicks on the edit button. (5) The customer enters the new bug information. (6) The customer confirms the new info. (7) The system validates bug info and confirms it. (8) The system will send an email to the staff members to inform them about the new bug info.
Alternative processes:	(3.a) The system doesn't find the bug. (3.a.1) The system display error message " <<Not found>> ". (3.a.2) Repeat from step 2. (7.a) The system doesn't confirm the new bug info and display error message << "Plases enter valid information" >> (7.a.1) Repeat from step 5.

Name:	Send messages
Description :	the goal of sending messages is to convey information, ideas, thoughts, or emotions.
Primary actor	Admin & Customer
Precondition :	.1The user is authenticated and logged into the messaging application. .2The recipient's contact information (such as username or phone number) is valid and accessible within the application.
Postcondition :	.1The message is successfully delivered to the intended recipient. .2The message appears in the recipient's chat interface. .3The message is stored securely in the messaging platform's database for future reference.
Main flow:	.1The user navigates to the conversation window or selects the option to start a new conversation. .2The user types the message content into the message input field. .3The user initiates the sending action. .4The application validates the recipient's contact information and the message content and it is valid. .5 The message is sent successfully to the recipient.

Name:	Receive messages
Description :	the functionality of the system or application to receive incoming messages from other users or sources and deliver them to the intended recipients.
Primary actor:	Customer, Admin
Precondition :	1.The user is authenticated and logged into the messaging application or system. 2.The sender has successfully sent the message, and it has reached the messaging platform or system. 3.The recipient's messaging settings allow for receiving messages from the sender or the source of the message.
Postcondition :	1.The incoming message is successfully received and stored in the recipient's inbox or message queue. 2.The recipient can view, respond to, or interact with the received message as needed.
Main flow:	1) Recipient opens chat icon from the application. 2) Recipient choose which project he wants to recieve messages from. 3) System will show to him the messages.

Name:	Delete bug
Description :	the functionality for authorized users to remove or delete bug reports from the bug tracking system. This use case involves identifying and selecting specific bug reports for deletion, confirming the deletion action, and updating the bug tracking database accordingly
Primary actor	customer.
Precondition :	<ol style="list-style-type: none"> 1.The user is authenticated and logged into the bug tracking system or software application. 2.At least one bug report exists within the bug tracking system that the user has permission to delete.
Postcondition :	<ol style="list-style-type: none"> 1.The selected bug report(s) are successfully deleted from the bug tracking system. 2.Any references to the deleted bug report(s) are removed from the bug tracking database and associated records. 3.The bug tracking system updates its internal state to reflect the deletion of the bug report(s).
Main flow:	<ol style="list-style-type: none"> 1.The user navigates to the list of bug reports within the bug tracking system. 2.The user selects one or more bug reports that they want to delete. 3.The user initiates the deletion action by selecting the "Delete" or similar option from the user interface. 4.The bug tracking system prompts the user to confirm the deletion action to prevent accidental deletions. 5.The user confirms the deletion action. 6.The bug tracking system removes the selected bug report(s) from the bug database. 7.Any references or links to the deleted bug report(s) are updated or removed from associated records or reports. 8.The bug tracking system updates its internal state to reflect the deletion of the bug report(s).

Name:	Monitor bug case flow
Description :	functionality for users, to track the progress and status of bug reports within the bug tracking system. This use case involves monitoring the lifecycle of bug reports, including their creation, assignment, resolution, and closure, to ensure timely and effective bug management.
Primary actor	customer.
Precondition :	<ol style="list-style-type: none"> 1.The user is authenticated and logged into the bug tracking system or software application. 2.There exist active bug reports within the bug tracking system that are relevant to the user's role or responsibilities.
Postcondition :	<ol style="list-style-type: none"> 1.The user has successfully monitored the progress and status of bug reports within the bug tracking system. 2.The user is aware of the current status of each bug report, including whether it is open, assigned, in progress, resolved, or closed.
Main flow:	<ol style="list-style-type: none"> 1.The user navigates to the bug tracking dashboard or relevant section within the bug tracking system. 2.The user views a list of active bug reports sorted by status, priority, or other relevant criteria. 3.The user selects a specific bug report to monitor its progress and status. 4.The user reviews the details of the selected bug report, including its title, description, severity, assignee, and current status.

Name:	Create bug
Description :	the functionality for users to report issues, glitches, or defects encountered within a software application or system. This use case involves capturing relevant information about the bug, such as its description, severity, steps to reproduce, and any accompanying files or screenshots, to facilitate the debugging and resolution process.
Primary actor	Customer
Precondition :	<ol style="list-style-type: none"> 1.The user is authenticated and logged into the bug tracking system or software application. 2.The user has identified a legitimate issue or bug within the software application or system.
Postcondition :	<ol style="list-style-type: none"> 1.The bug report is successfully created and stored within the bug tracking system. 2.The relevant stakeholders, such as developers or project managers, are notified of the newly created bug report. 3.The bug report includes all necessary information to accurately describe the issue, including its title, description, severity level, steps to reproduce, and any attached files or screenshots. 4.The bug report is assigned to the appropriate individual or team responsible for investigating and resolving the reported issue. 5.The status of the bug report is updated to reflect its current state in the bug tracking workflow (e.g., open, assigned, fixed, verified).
Main flow:	<ol style="list-style-type: none"> 1.The customer identifies a potential bug or issue within the software application or system. 2.The customer navigates to the bug tracking system or application. 3.The customer selects the option to create a new bug report. 4.The customer fills out the bug report form, providing details such as the bug title, description, steps to reproduce, expected behavior, actual behavior, and severity level. 5.The user attaches screenshot the bug. 6.The user submits the bug report to the bug tracking system. 7.The bug tracking system validates the submitted bug report and creates a new entry in the bug database. 8.The bug tracking system assigns a unique identifier or bug ID to the newly created bug report. 9. the bug tracking system sends notifications to relevant stakeholders, such as developers or project managers, to inform them of the newly created bug report.

Name:	Delete Project.
Goal:	involves the removal of an existing project from the bug tracking system. It allows administrators to permanently delete a project and its associated data.
Preconditinos:	Admin must be loggedin to the system.
Postconditions:	The project and its associated data are deleted form the system.
Initiator(s):	Admin.
Trigger:	Admin requires deleting a project from the system.
Standard Process:	1) Administrator navigates to the "Projects section" section. 2) System presents a list of existing projects. 3) Administrator selects the project to be deleted. 4) The system checks all the bugs in the project and they are all resolved. 5) System prompts for confirmation. 6) Administrator confirms deletion. 7) System permanently removes the project and all associated data.
Alternative Process:	4.a) If the selected project has associated bugs, the system may prompt the administrator to reassign or resolve them before deletion. 4.a.1) the project won't be deleted and the system won't continue in the process.

Name:	Create Project.
Goal:	enables Admins to initiate a new project within the bug tracking system.
Preconditinos:	Admin must be loggedin to the system.
Postconditions:	Project is created successfully in the system.
Initiator(s):	Admin.
Trigger:	Admin requires creating a project to enable them to manage project bugs.
Standard Process:	1) Admin opens projects list. 2) Admin clicks add btn. 3) Admin fills the form and click create button. 4) The system validate the input and it's valid. 6) The system creates a project successfully.
Alternative Process:	4.a) The system finds that the input is not valid or used by another project. 4.a.1) display error message. 4.a.2) repeat from step 3.

Name:	Create Account.
Goal:	enables administrators to create new staff accounts in the bug tracking system.
Preconditinos:	Admin must be loggedin to the system.
Postconditions:	Account is created sucessfully and staff member can use it.
Initiator(s):	Admin.
Trigger:	Admin requires creating staff member accounts to enable them to use the system.
Standard Process:	1) Admin opens create staff form. 2) Admin fills the form and click create button. 3) The system validate the input and it's valid. 4) The system creates an staff member account successfully.
Alternative Process:	3.a) The system finds that the input is not valid or used by another account. 3.a.1) display error message. 3.a.2) repeat from step 2.

Name:	Give Feedback.
Goal:	enables a customer to provide feedback for a bug solution.
Preconditinos:	The customer must be loggedin to the system.
Postconditions:	System created a feedback to a bug solution.
Initiator(s):	Customer.
Trigger:	A customer requires to rate the solution provided by staff members.
Standard Process:	1) Customer chooses project. 2) System shows created bugs. 3) User chooses bug. 4) if the bug has a solution the customer can fill a from and click add btn. 5) System create a feedback and send success message.

Name:	Create category.
Goal:	enables admin to create staff members categories which will help in handling bugs in the project and each category will have its own jobs.
Preconditinos:	Admin must be loggedin to the system.
Postconditions:	nwe category is created successfully in the system.
Initiator(s):	Admin.
Trigger:	Admin requires creating a new category.
Standard Process:	(1) Admin Chooses Add Staff. (2) Admin Chooses Add category. (3) Admin fills the form and clicks add btn. (4) The system checks if the category exists and it isn 't exist. (5) Category will be added. (6) Category table will be updated.
Alternative Process:	4.a) The system finds that the category already exist. 4.a.1) display error message. 4.a.2) repeat from step 3.

Alternative flows	Change username
Description :	the process by which a user can update or modify their username within a system or application.
Primary actor	User.
Precondition :	the user must be logged in to the system.
Postcondition :	the user's username is successfully updated, and they can continue using the system with their updated username.
Main flow :	<ol style="list-style-type: none"> 1.User navigates to the profile or account settings section of the system or application. 2.User selects the option to change their username. 3.User provides the new desired username 4.The system validates the new username to ensure it meets requirements. 5.If the new username is valid, the system updates the user's profile with the new username. 6. The system displays success message.
Alternative flows	<p>(5.a) the username is not valid.</p> <p>(5.a.1) the system shall display error message.</p> <p>(5.a.2) repeat from step 3</p>

Name:	Register.
Description :	the interaction between users and the system or application during the account creation process, ensuring a smooth and secure registration experience while enabling users to access the system's functionalities.
Initiators	User (admin, customer) .
Precondition :	the user not already having an existing account.
Postcondition :	the user's account is successfully created and user can use the system with the account.
Main flow :	<p>(1) User provides necessary information to create an account, such as username, email address, password.</p> <p>(2) The system validates the provided information to ensure it meets requirements.</p> <p>(3). f the information is valid, the system creates a new user account and stores the provided data.</p> <p>(4) the system shall display a success message and navigate the user to his home page.</p>
Alternative flows	<p>(3.a) the data is not valid.</p> <p>(3.a.1) the system shall display error message.</p> <p>(3.a.2) repeat from step 1</p>

Name:	Logout .
Description :	Using this use case you can get out from your page and return to login page by clicking on Logout button.
Initiator:	User.
Precondition :	The user must be pre-logged in .
Postcondition :	User logged out and returned to login page .
Main flow :	1- Click the logout button in the system interface 2- The system returns the user to the login page .

Name:	Login
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 .
Initiators:	User.
Precondition :	User must register to site (customer, admin) or her data has been entered in system (staff).
Postcondition :	System will display the relevant home page .
Main flow :	1- User will click on login . 2- User will enter/insert username . 3- User will enter/insert password . 4- System validates username and password and they are correct, 5- System display the relevant home page .
Alternative flows	(4.a) username or password are not correct . (4.a.1) System will display error message <<"username or password not correct ">> .
Goal :	User login to site successfully .

Name:	Change password
Description:	the process by which a user can update or modify their password within a system or application. It details the steps involved in changing the password, including input requirements, validation procedures, and updating user credentials.
Primary actor:	User.
Precondition :	Specifies any conditions that must be met before the password change process can begin, such as the user being logged in to their account and having appropriate permissions to modify their password.
Postcondition :	Specifies the state of the system and the user after the password change process is completed, such as the user's credentials being updated with the new password.
Main flow :	<ol style="list-style-type: none"> 1.User navigates to the profile or account settings section of the system or application. 2.User selects the option to change their password. 3.User provides their current password and the new desired password. 4. The system validates the provided current password to ensure it matches the user's existing credentials and it matches. 5. The system displays success message.
Alternative flows	<p>(4.a) the username is not valid.</p> <p>(4.a.1) the system shall display error message.</p> <p>(4.a.2) repeat from step 3</p>

Use case ID :	
Use case name :	Edit Project
Include use case :	none
Description :	Editing a project involves making changes to project details, configurations, or settings to ensure alignment with project objectives
Primary actor :	Admin
Secondary actor :	None
Precondition :	1-The project management system is accessible and operational. 2-The Admin has appropriate permissions to edit projects
Postcondition :	The project is updated with the changes made by the Admin
Main flow :	Access Project Initiate Edit Modify Project Details Adjust Project Settings Update Timeline and Milestones
Goal :	The goal of this use case is to enable the project administrator to efficiently make necessary adjustments to the project details, configurations, and settings to ensure the project's continued alignment with organizational objectives and successful execution

Use case ID :	
Use case name :	View Created Bugs
Include use case :	None
Description :	This use case outlines the process by which a customer views bugs they have reported within a software application or system. Customers may need to monitor the status and resolution progress of reported bugs to ensure their concerns are addressed effectively.
Primary actor :	Customer
Secondary actor :	None
Precondition :	1-The software application or system has a bug reporting feature enabled. 2-The customer has previously reported one or more bugs.
Postcondition :	The customer has successfully viewed the list of bugs they have reported along with relevant details.
Main flow :	Access Bug Reporting Interface: View Reported Bugs: Review Bug List: Access Bug Details: Monitor Bug Status:
Goal :	to enable customers to easily access and review a list of bugs they have reported within the software application or system, allowing them to monitor the status and resolution progress of reported issues effectively.

Use case ID :	
Use case name :	Send solution message .
Description :	This use case allows staff to send solution messages to admins to be approved and sent to customer.
Include use case :	Sate state.
Primary actor :	Staff .
Secondary actor :.	admin
Precondition :	Admin and staff must be pre-logged in . Staff assigned to solve the bug .
Postcondition :	The message sent to customer then customer can view sent message .
Main flow :	1- staff choice sends a message . 2- Admin receives the solution message and approves it . 3- System sent the message to the customer
Goal :	Solution messages sent to customer and he can see it.

Use case ID :	
Use case name :	View bug .
Description :	By staff using this use case he can view all assigned bugs that are assigned by Admin and he can see all bug details including the ticket number generated by the system to solve it .
Primary actor :	Staff.
Secondary actor :	None .
Precondition :	1-customer must be sent a bug. 2-The Admin must have created the project and assigned bug and this staff to it. 3-staff must be pre-logged in .
Postcondition :	Staff viewed the bug and worked to solve it and send the solution message to the customer .
Main flow :	1-Staff chooses to list the assigned bug. 2-Staff worked to solve the bug. 3- Bug was solved. 4-Customer received the solution message from admin or staff .
Alternative flows :	1-Bug category not like that belongs to staff that bug assigned to 2-the staff assign bug to other staffs if the bug is related to them. 3-the new staff involved to solve this bug can solve this or can assign it to other staffs again if it is related to them .
Goal :	Staff can list all bugs that are assigned by the admin to solve it .

Use case ID :	
Use case name :	View report.
Include use case :	None.
Description :	System generate reports to be viewed by Admin to track all things happen the application during the month
Primary actor :	Admin.
Secondary actor :	System.
Precondition :	1- projects created. 2- Sent bugs . 3- Bugs solved. 4- Customer give feedback. 5- Admin logged in application.
Postcondition :	Admin can view this reports to truck application's activity
Main flow :	1- Admin login to application. 2- Admin click on view reports
Goal :	Admin can view this reports to truck application's activity

Name:	Assign work to other staff members.
Initiator(s):	Staff member.
Precondition:	Staff member loggedin to the system.
Postcondition:	The work has been assigned to more staff members.
Main flow:	<ol style="list-style-type: none"> 1) StaffMember opens bugs page. 2) System displays assigned bugs. 3) StaffMember chooses a bug. 4) StaffMember clicks on add staff button. 5) user fills the form with staff username. 6) System checks the username and it's not already exist in the project. 7) System displays sucess message and staffMember assigned to the work.
Goal:	Allow staffMembers to add other staff members to work that will be related to them.

Use Case ID:

Use Case Name: Edit Account

Include Use Case: None

Primary Actor: User (Project Manager, Team Member, or any user with an account)

Secondary Actor: N/A

Precondition:

1. User is logged into the system.

Postcondition: The user's account information is updated with the new details.

Main Flow:

1. User selects the option to edit their account information (e.g., "Settings" or "Account Profile").
2. The system displays a form containing the user's current account details.
3. User edits the information they want to change, such as name, email address, password, or other relevant profile details.
4. User submits the changes.
5. The system validates the information and prompts the user to confirm changes (especially for sensitive information like password).
6. Upon confirmation, the system updates the user's account information in the database.

Goal: Allow users to modify their account details to keep information accurate and manage their preferences.

Use Case ID

Use Case Name: View Project Details

Include Use Case: None

Primary Actor: Project Manager (or Team Member with appropriate permissions)

Secondary Actor: N/A

Precondition:

1. User (Project Manager or authorized Team Member) is logged into the system.
2. The project to be viewed exists in the system.

Postcondition: User can see detailed information about the selected project.

Main Flow:

1. User selects the option to view projects.
2. (Optional) The system displays a list of available projects. User chooses the specific project to be viewed.
3. The system displays detailed information about the selected project, including its name, description, tasks, team members, deadlines, files, and other relevant data.

Goal: Gain a comprehensive understanding of a specific project's status and data

Use Case ID:

Use Case Name: View Bugs

Include Use Case: Manage Project (if bug reporting is a project management feature)

Primary Actor: Project Manager (or Team Member with appropriate permissions)

Secondary Actor: N/A

Precondition:

1. User (Project Manager or authorized Team Member) is logged into the system.
2. The project with reported bugs exists in the system (and bug reporting is enabled for the project, if applicable).

Postcondition: User can see a list of reported bugs for the selected project.

Main Flow:

1. User selects the option to view bugs (or navigate to the project management section and select "View Bugs" for a specific project).
2. User chooses the project for which they want to view reported bugs (if viewing bugs within project management).
3. The system displays a list of reported bugs for the selected project, including their status, severity, assigned fixes, and any relevant comments (if applicable).

Goal: Identify and track reported issues within a project.

Notes:

- View Bugs includes (Manage Project) because bug reporting might be a feature within project management.
- You can adjust the access permissions (Project Manager vs. Team Member) based on your specific system.

Use Case ID:

Use Case Name: Assign to work

Include Use Case: - Manage Project (assigning tasks is a subfunction of managing a project)

Primary Actor: Project Manager

Secondary Actor: N/A

Precondition:

1. Project Manager is logged into the system.
2. The project to which the task will be assigned exists in the system.

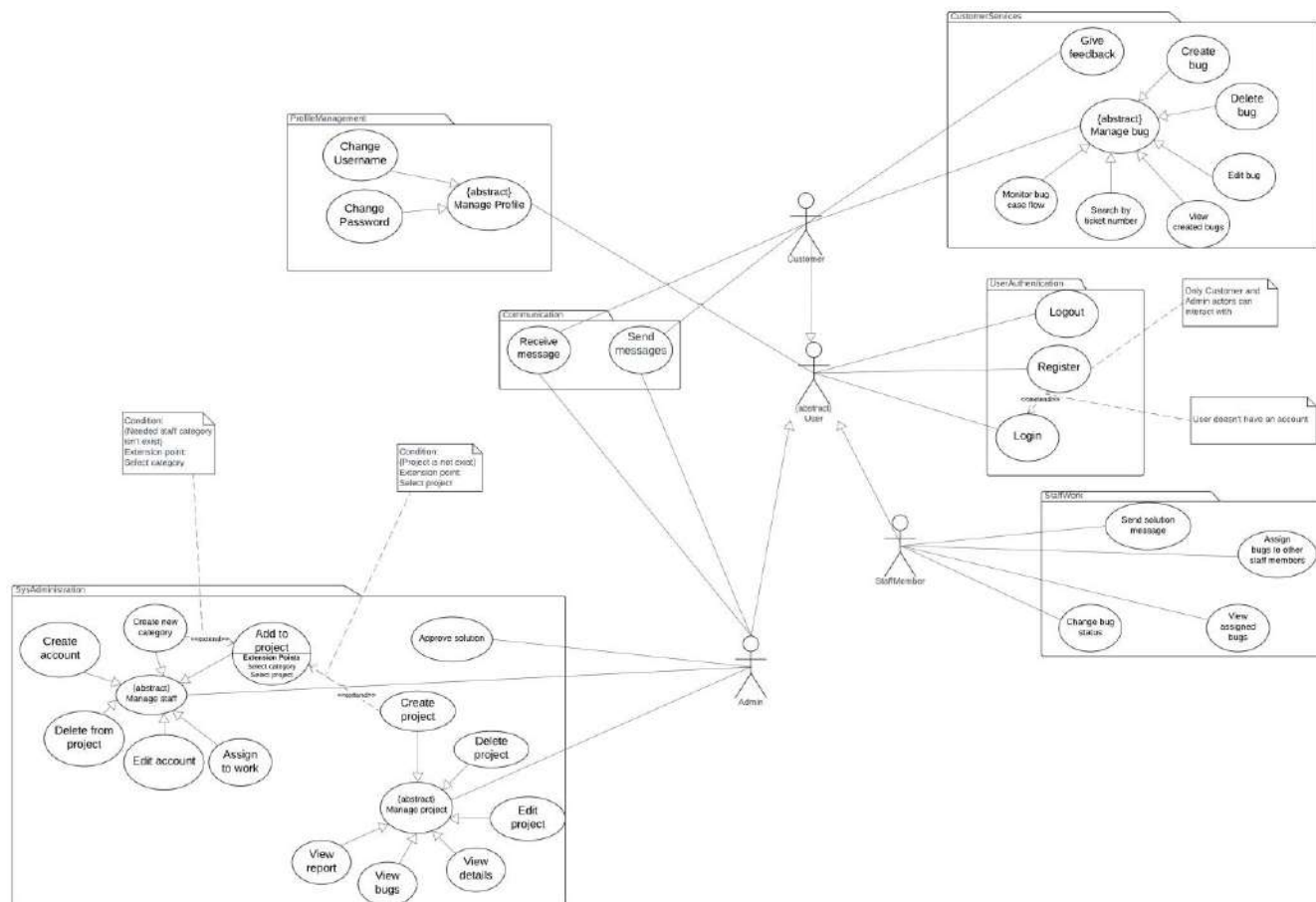
Postcondition: A new task is assigned to a specific team member within the chosen project.

Main Flow:

1. Project Manager selects the option to manage a project (or goes to the specific project).
2. Project Manager chooses the project where they want to assign a task.
3. Project Manager selects the option to "Assign Task" or similar functionality.
4. The system displays a form to enter task details.
5. Project Manager fills out the form with information such as task name, description, due date, priority level, and any attachments.
6. Project Manager selects a team member from a list or enters their name to assign the task.
7. (Optional) Project Manager can set notifications for the task.
8. The system validates the information and confirms the task assignment.
9. The system creates a new task within the project and assigns it to the chosen team member.

Goal: Efficiently delegate tasks to specific team members within a project.

Package Diagram grouping relevant Use Cases into Packages



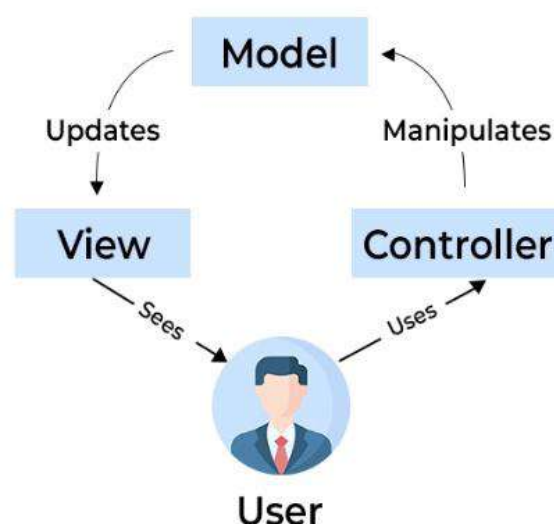
Structural & Behavioral Diagrams

System Architecture

- A system architecture is the conceptual model which defines a system's structure, actions and more views. A description of an architecture is a systematic description and representation of a system, structured in a manner that facilitates thinking about system mechanisms and behaviors.

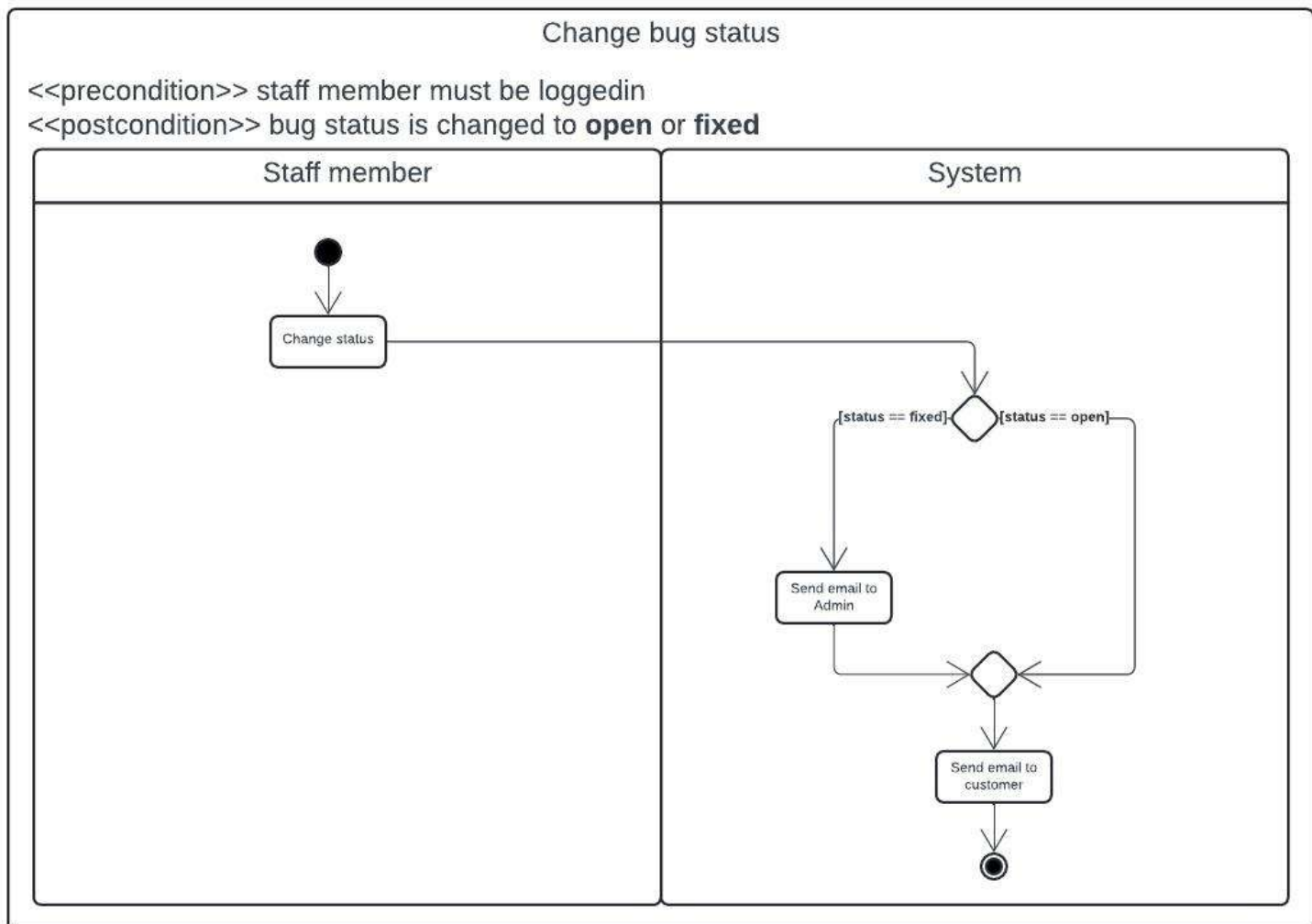
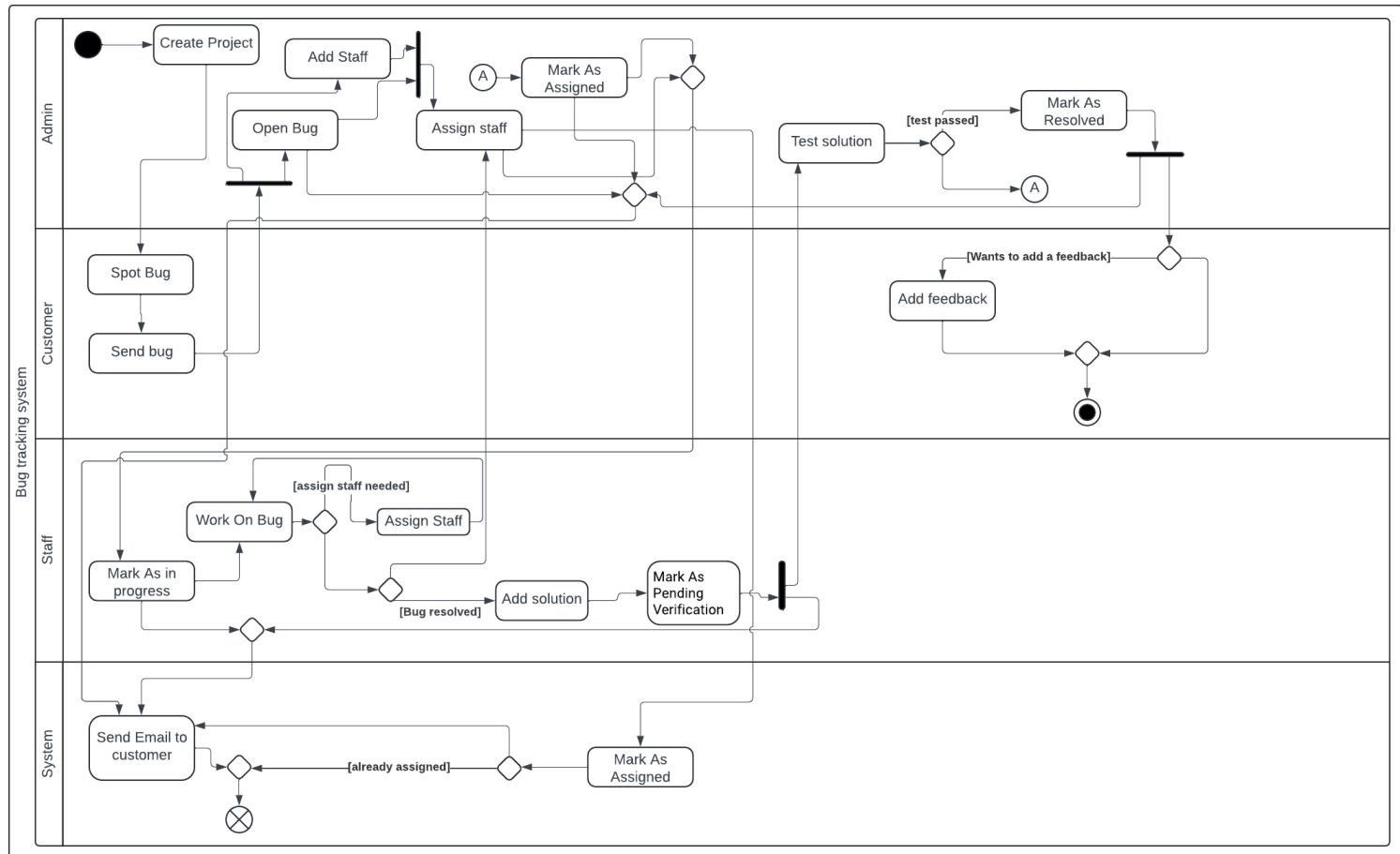
In our project we've chosen MVC architecture pattern because of:

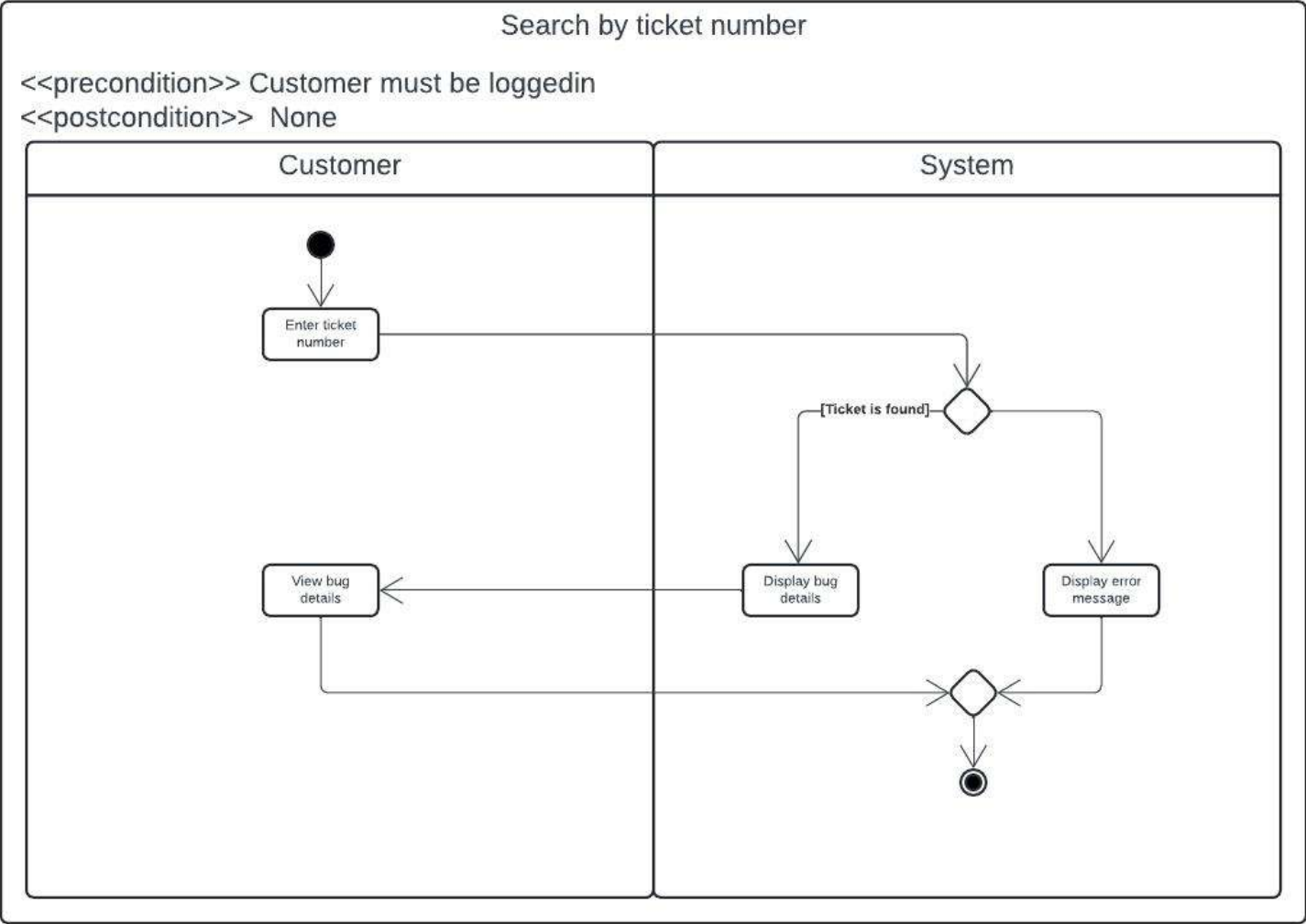
- Ease of modification: the separation of responsibilities, future development or modification is easier
- Multiple views for a model: Models can have multiple views
- Testability: with the clearer separation of concerns, each part can be better tested independently



- **Model:** Holds all the data, state and application logic. Oblivious to the View and Controller. Provides API to retrieve state and send notifications of state changes to “observer”.
- **View:** Gives user a presentation of the Model. Gets data directly from the Model.
- **Controller:** Takes user input and figures out what it means to the Model.

Activity Diagrams

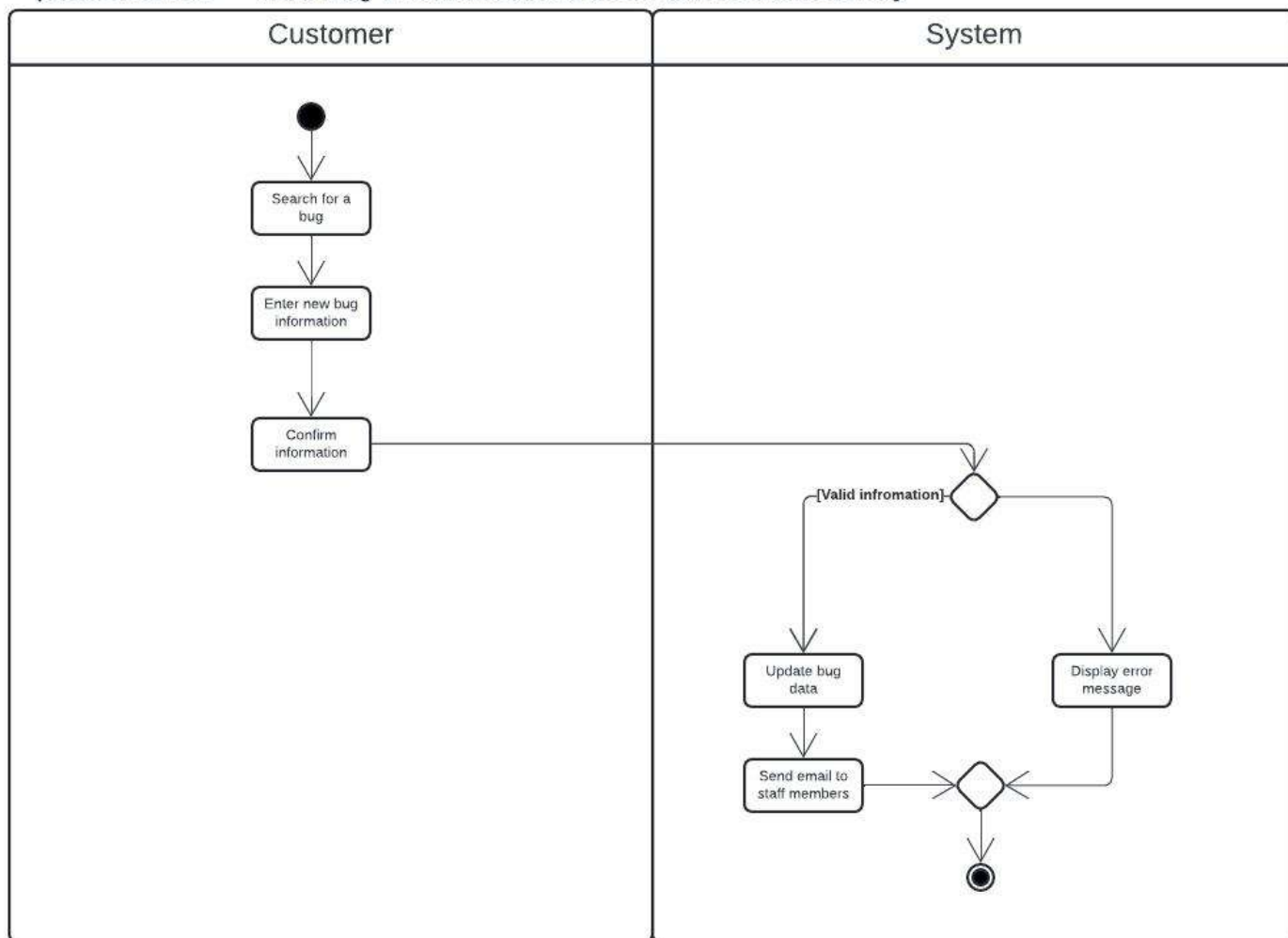


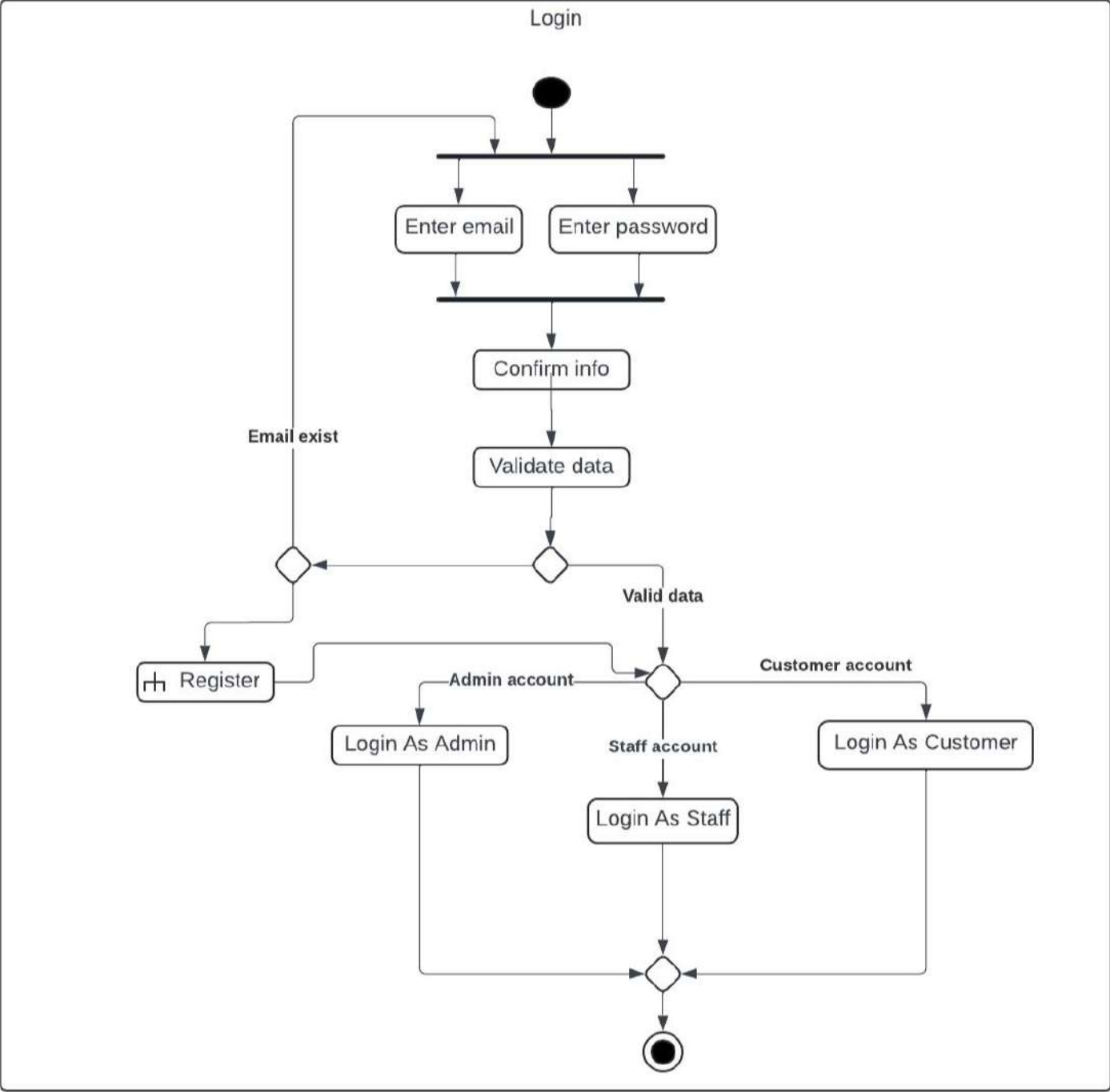


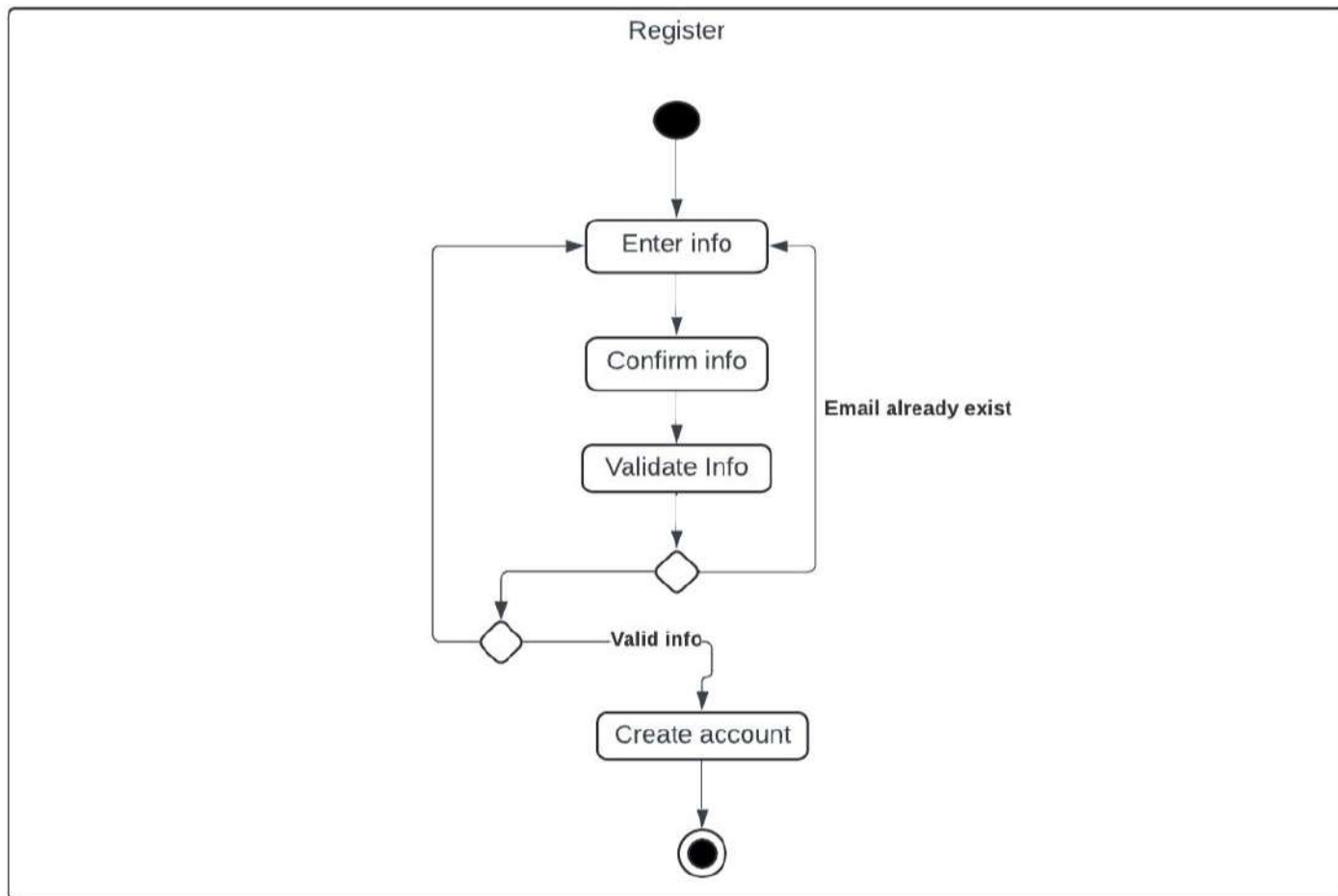
Edit bug

<<precondition>> Customer must be logged in

<<postcondition>> New bug information has been added successfully



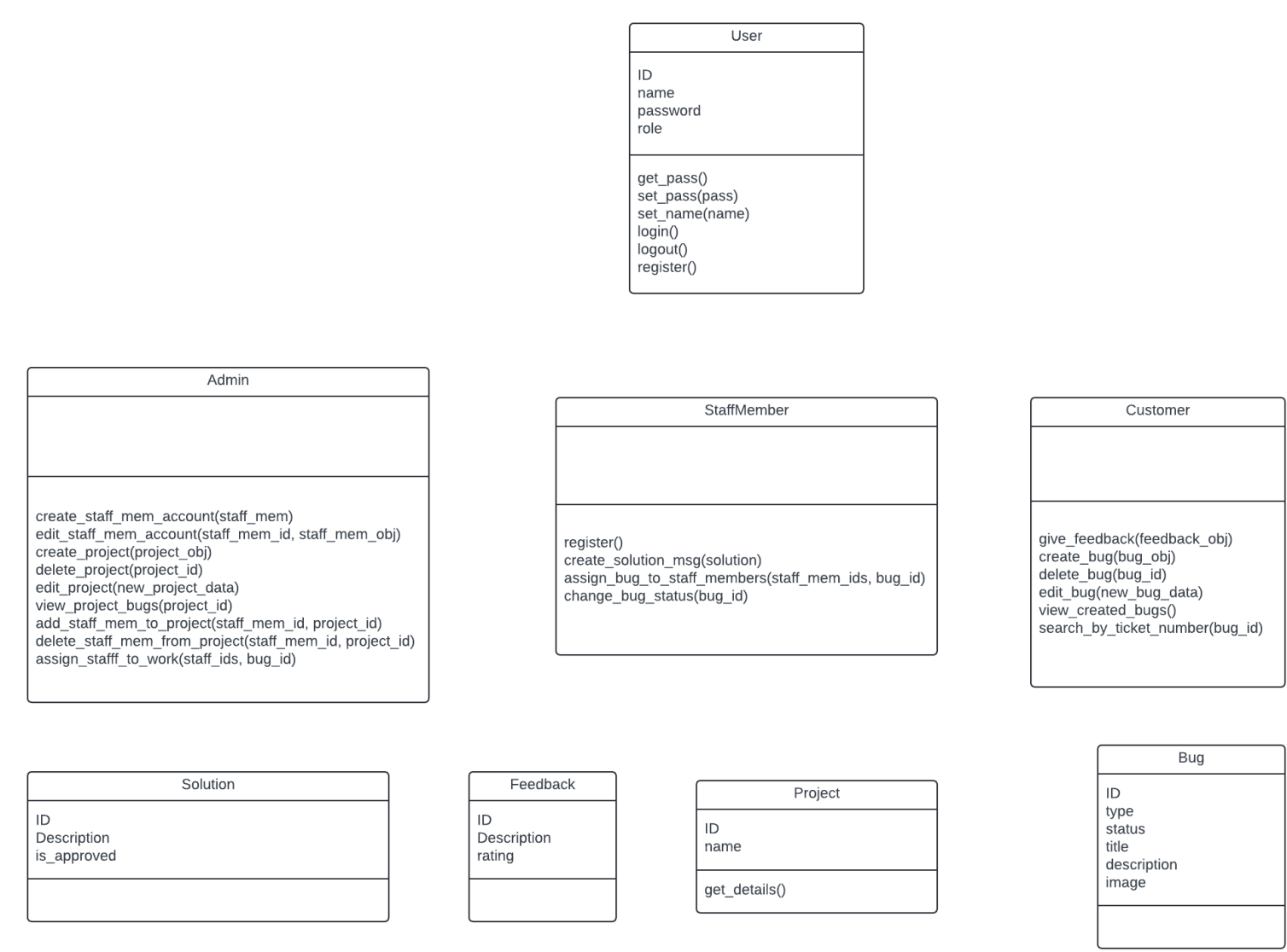




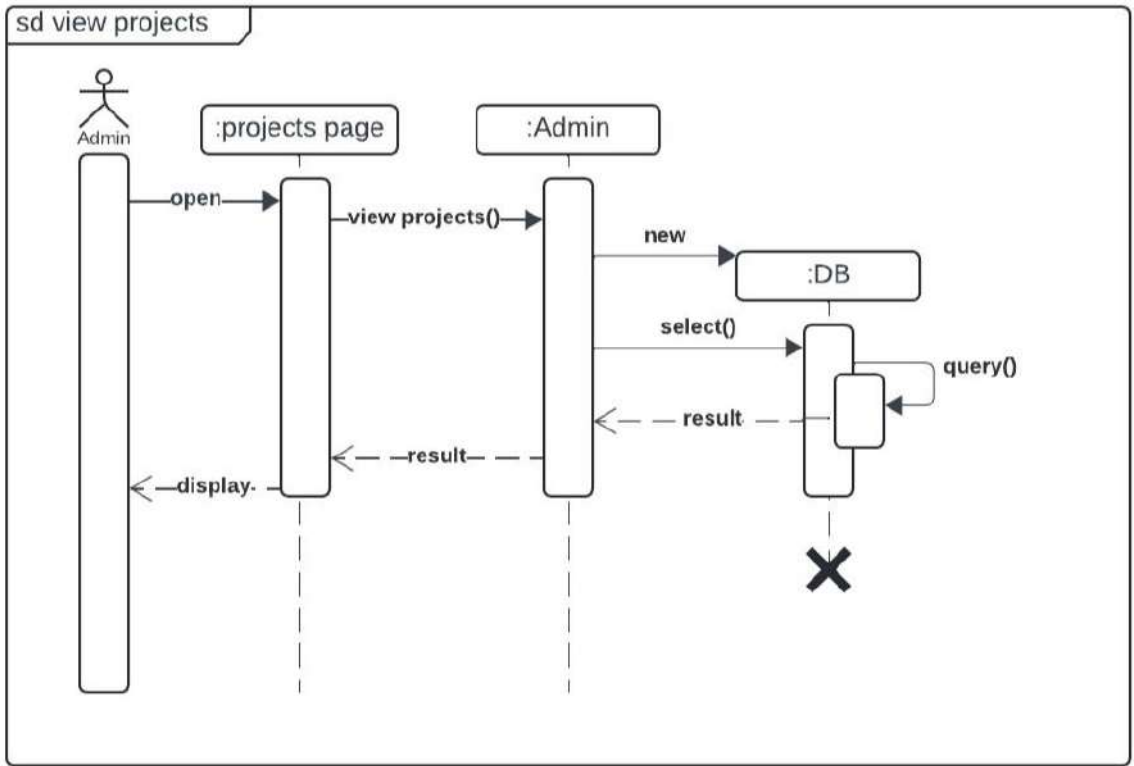
List of User Interfaces required for the System and the corresponding users of each interface

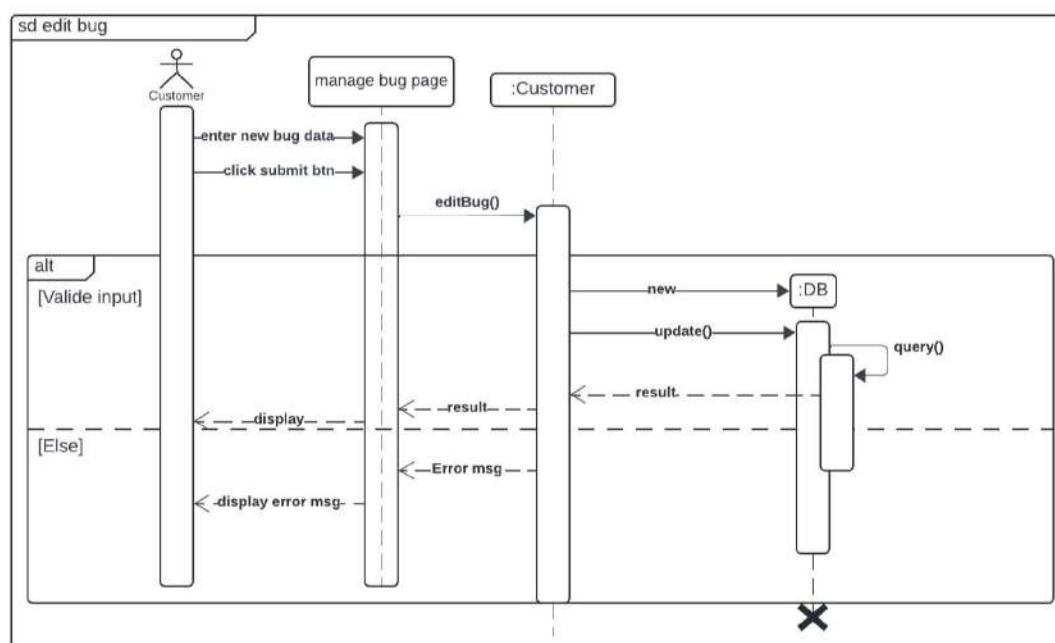
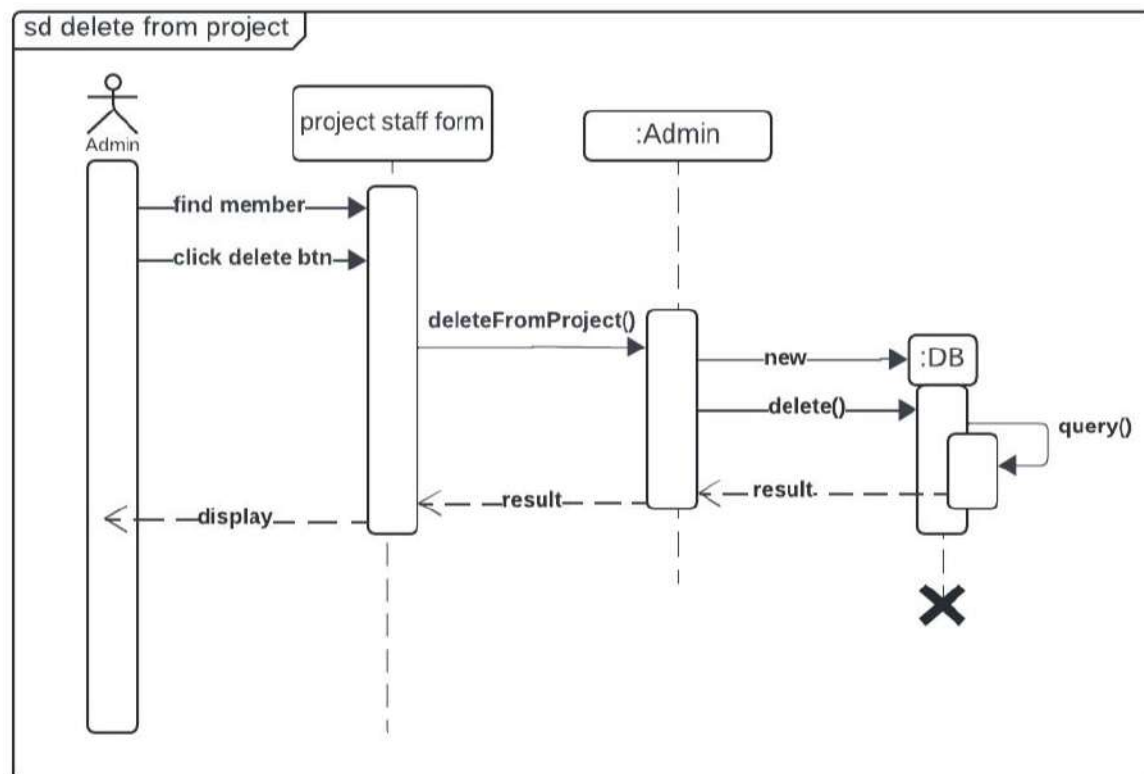
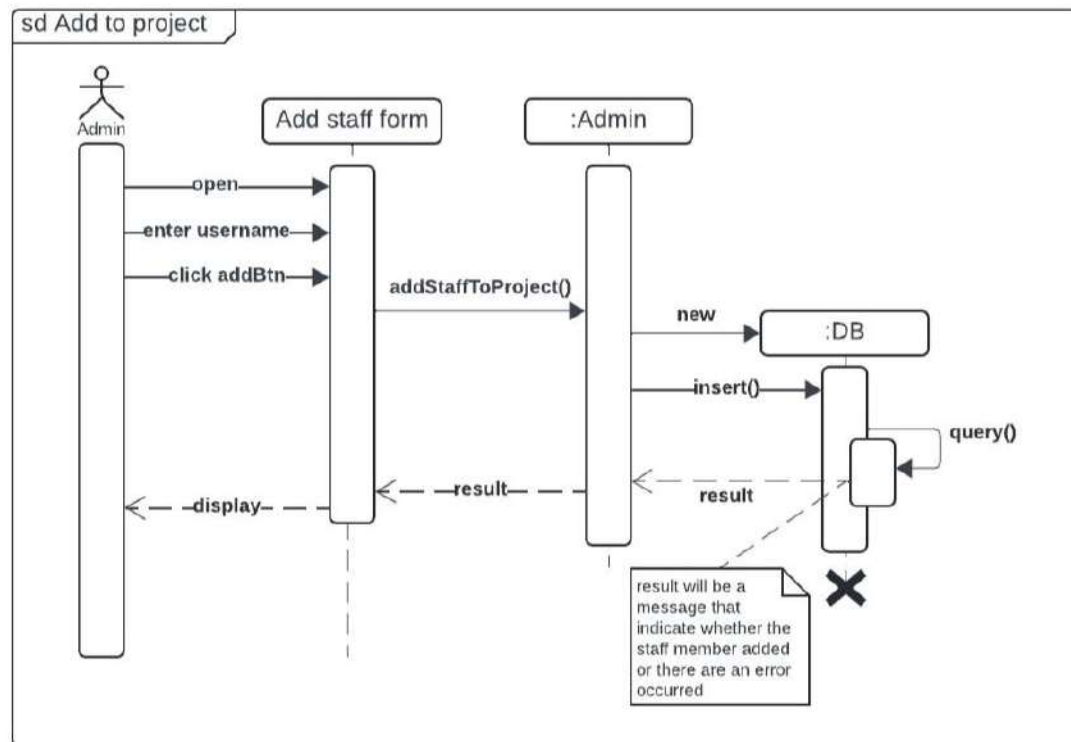
- User
 - Login form
 - Register form
 - Manage profile form
- Admin
 - Manage project interface
 - Mange staff interface
 - Created project list and its spotted bugs interfaces
 - Admin customer messages form.
- Staff
 - The list of projects that staff added to it and assigned bugs to each project interfaces.
 - Assign other staff members to a project form
 - Add solution to a bug form
- Customer
 - The list of projects that customer has spotted bugs in it and its spotted bugs interfaces.
 - Search by ticket number form.
 - Customer Admin messages form.
 - Manage bug form.

Class Diagram V.1

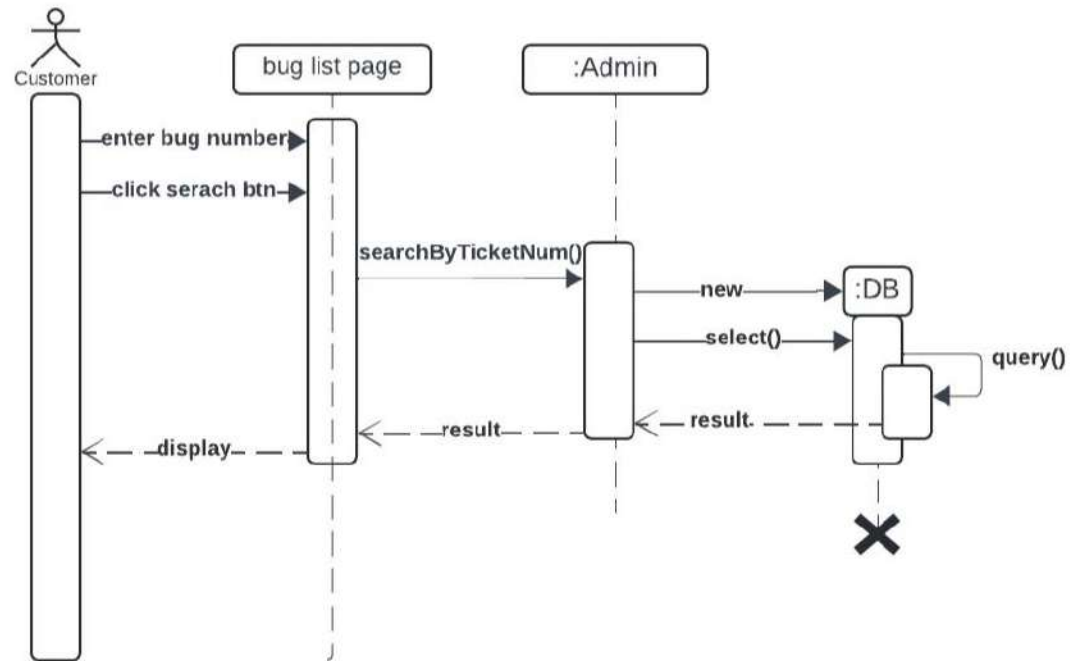


Sequence Diagram

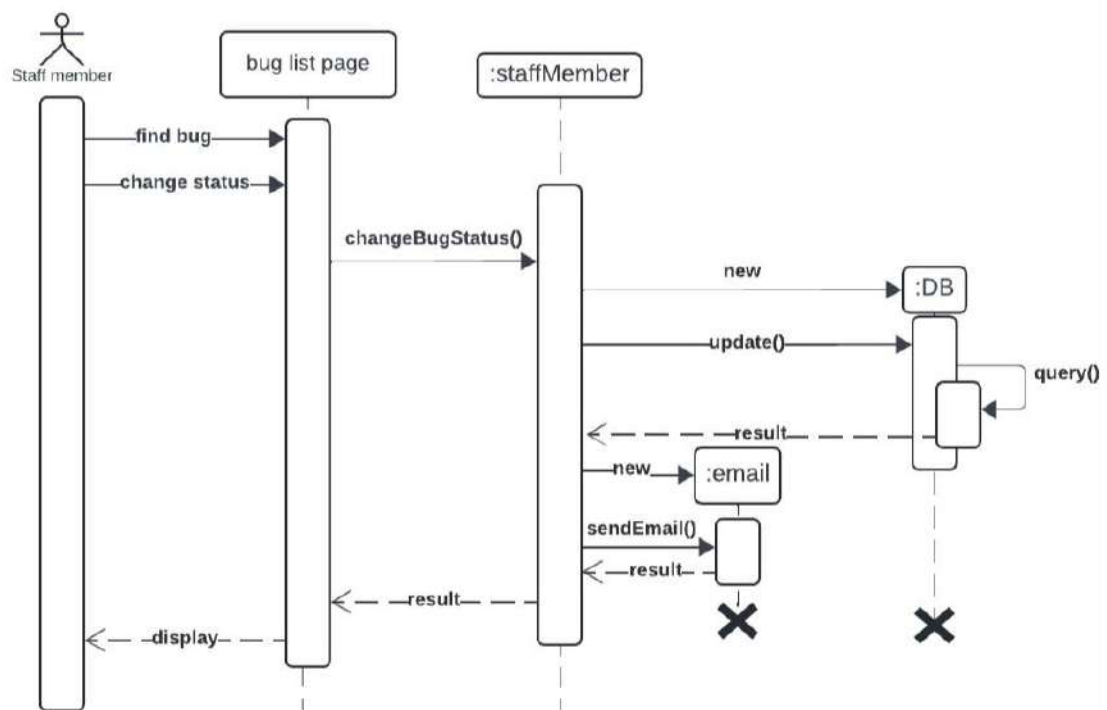


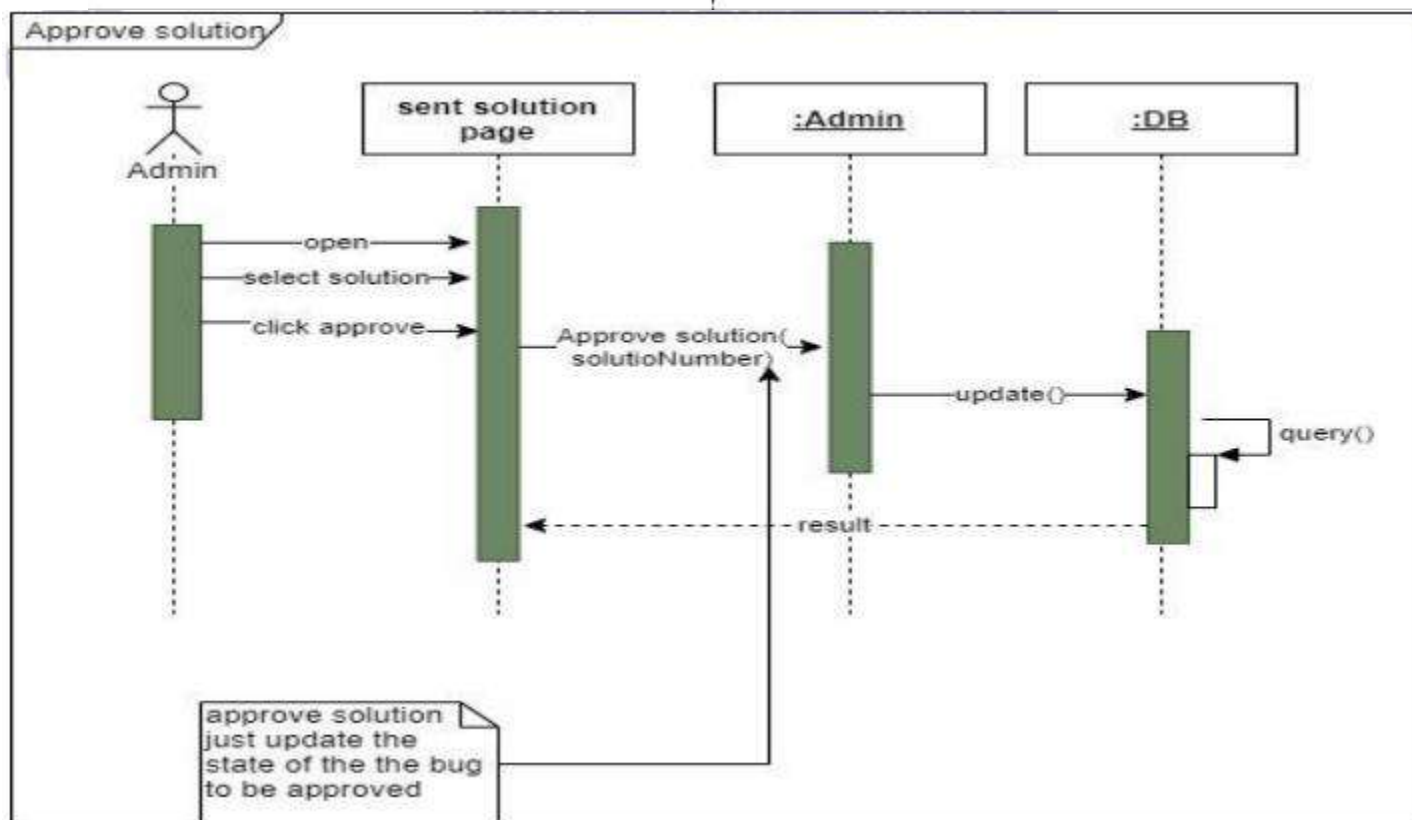
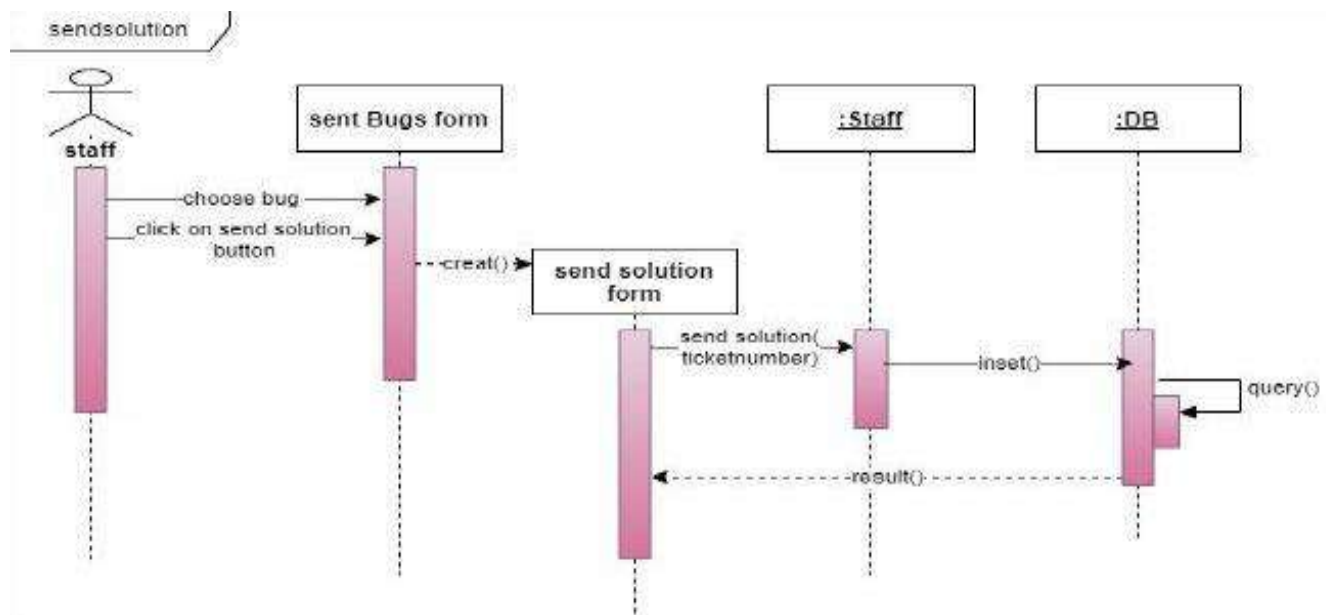


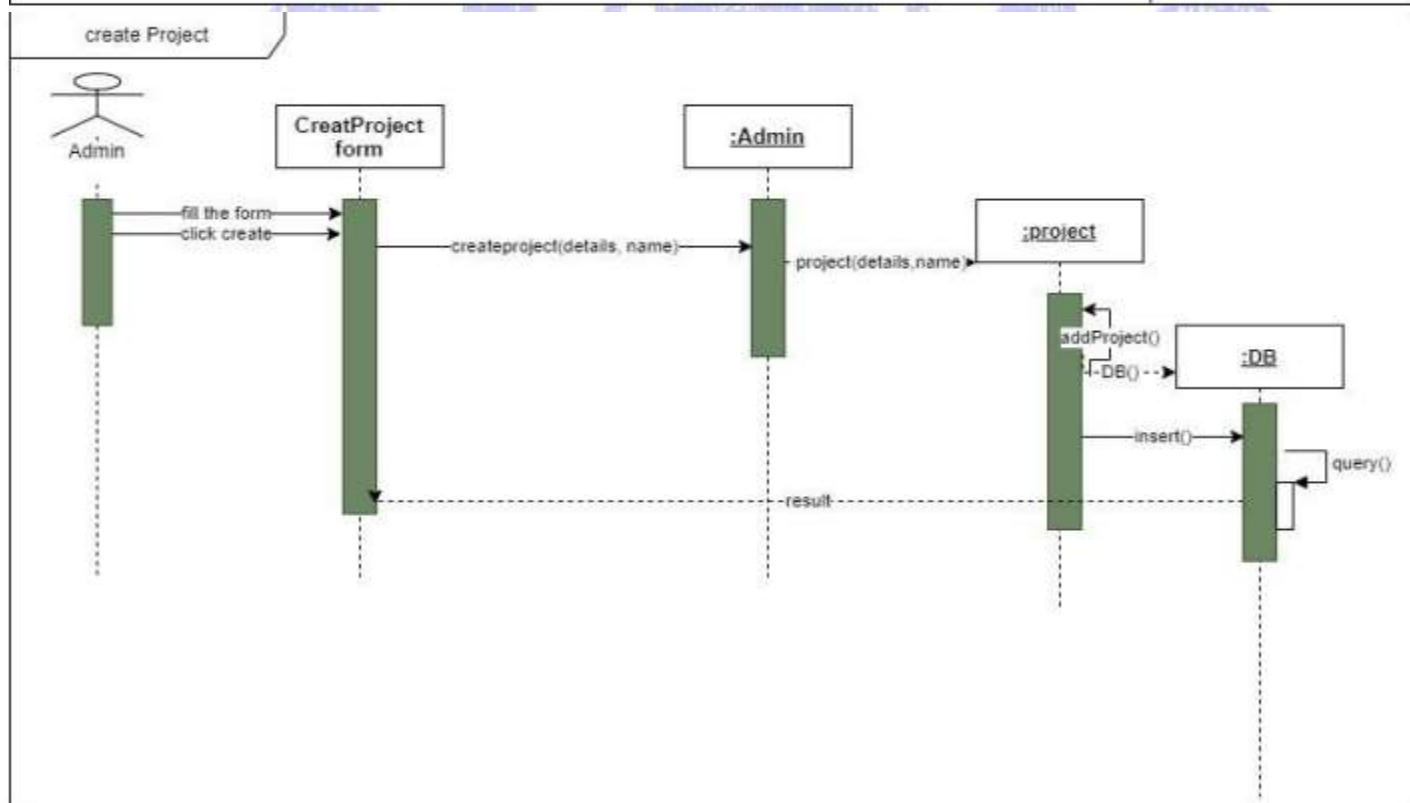
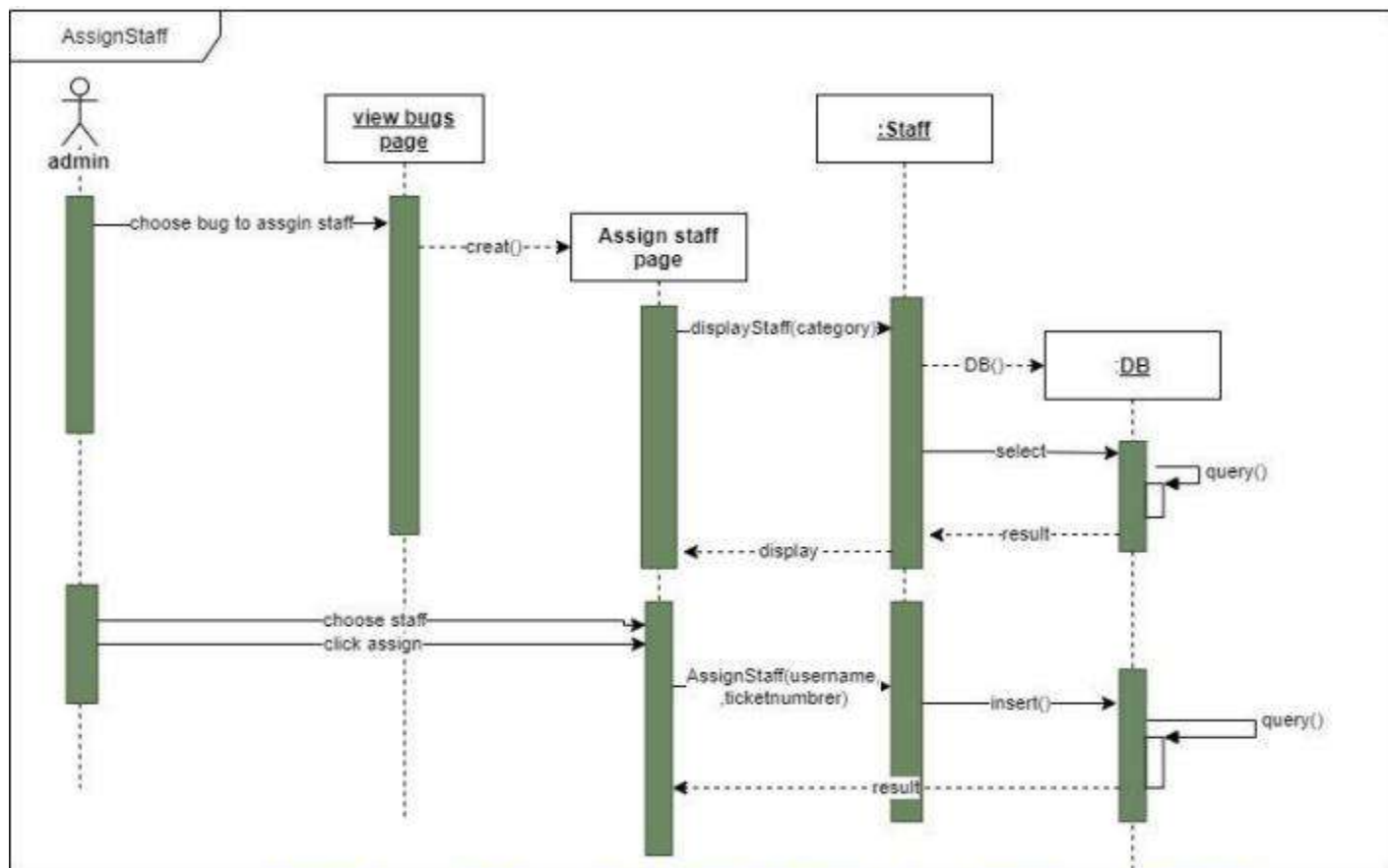
sd search by ticket number

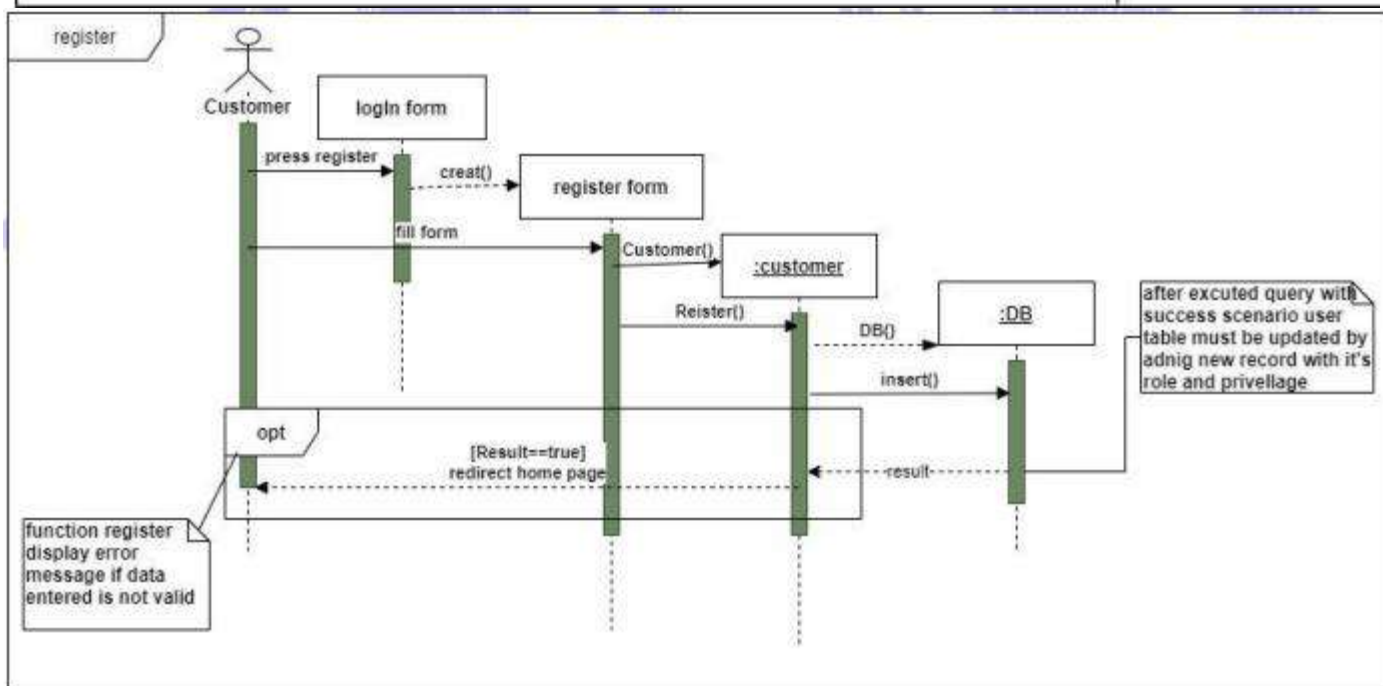
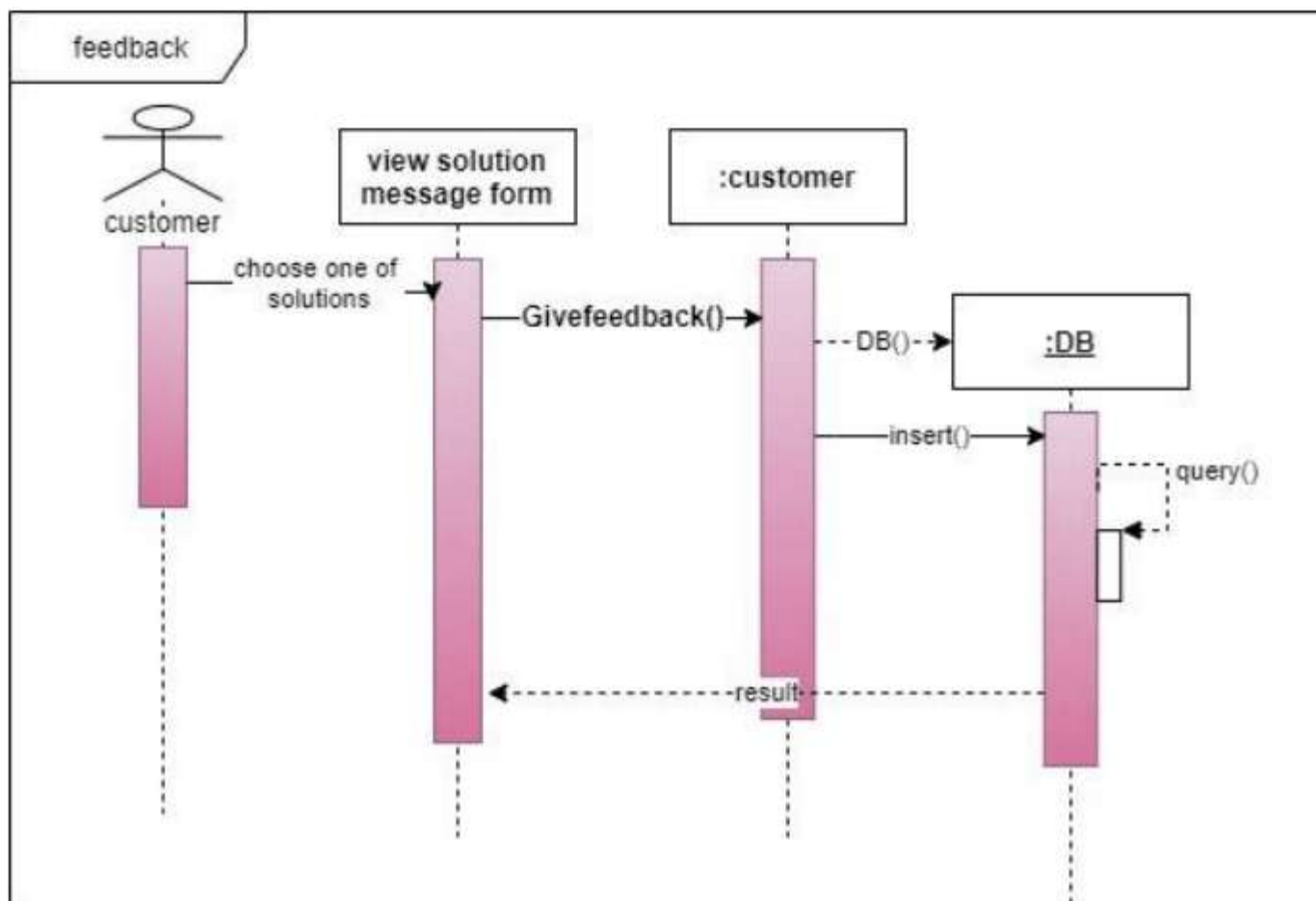


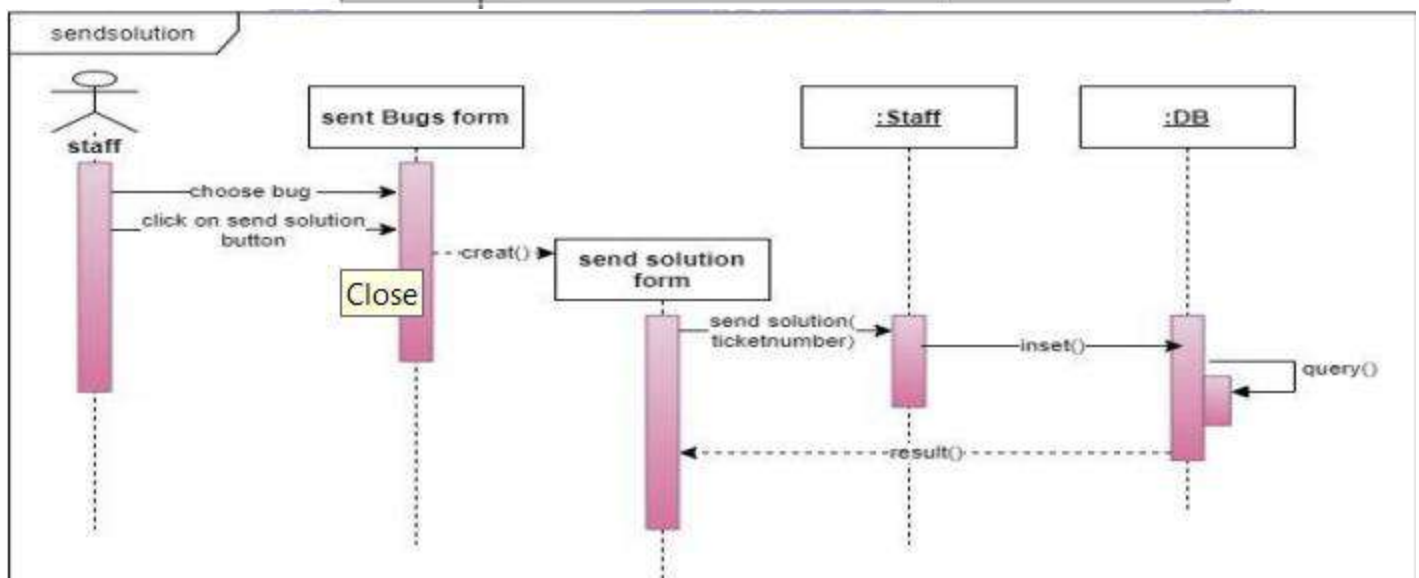
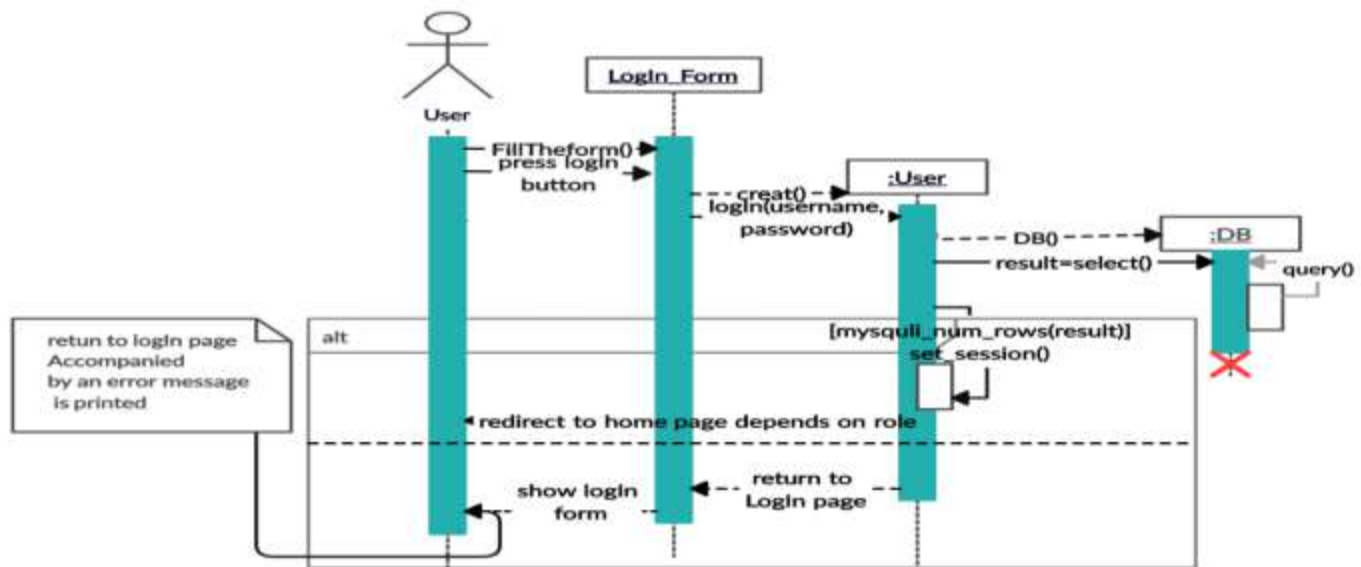
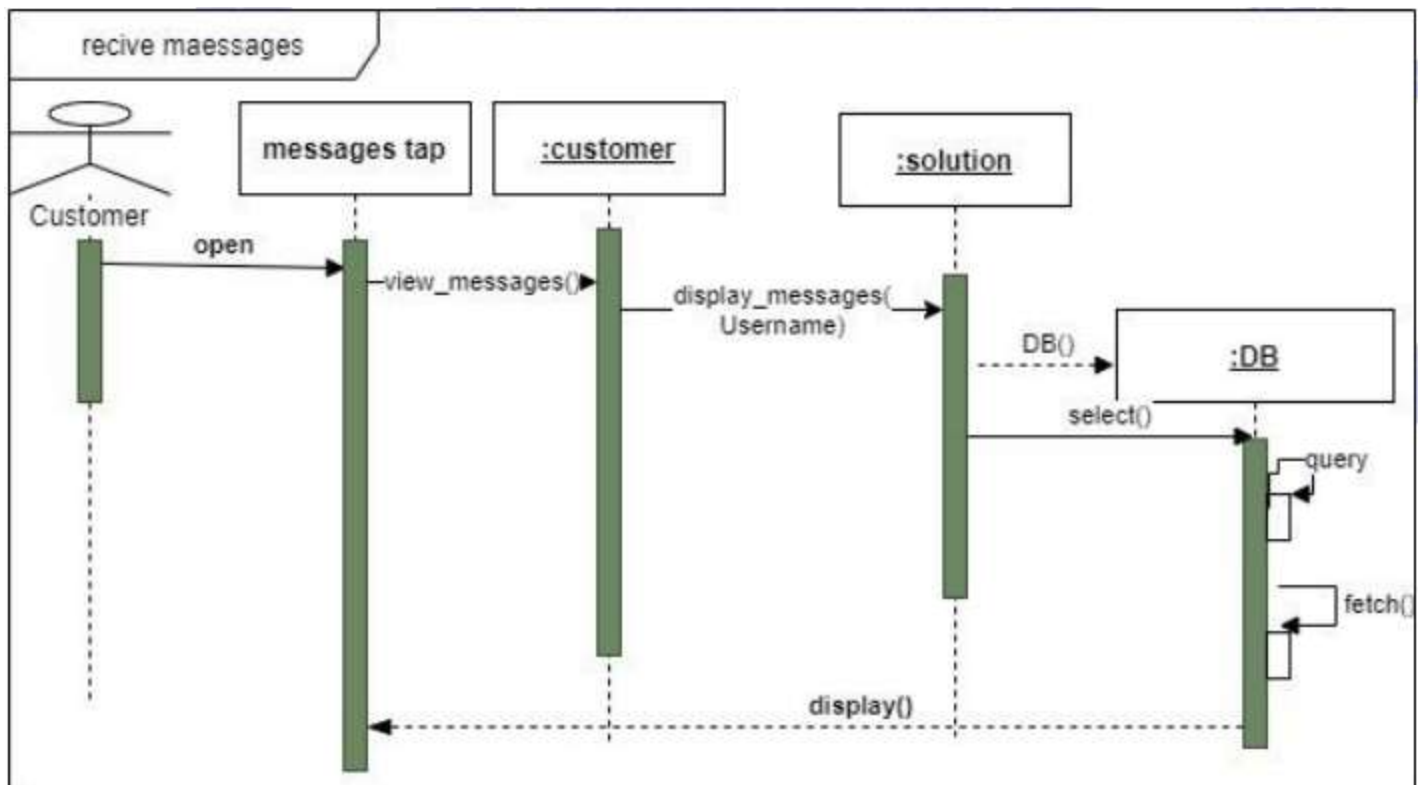
sd change bug status

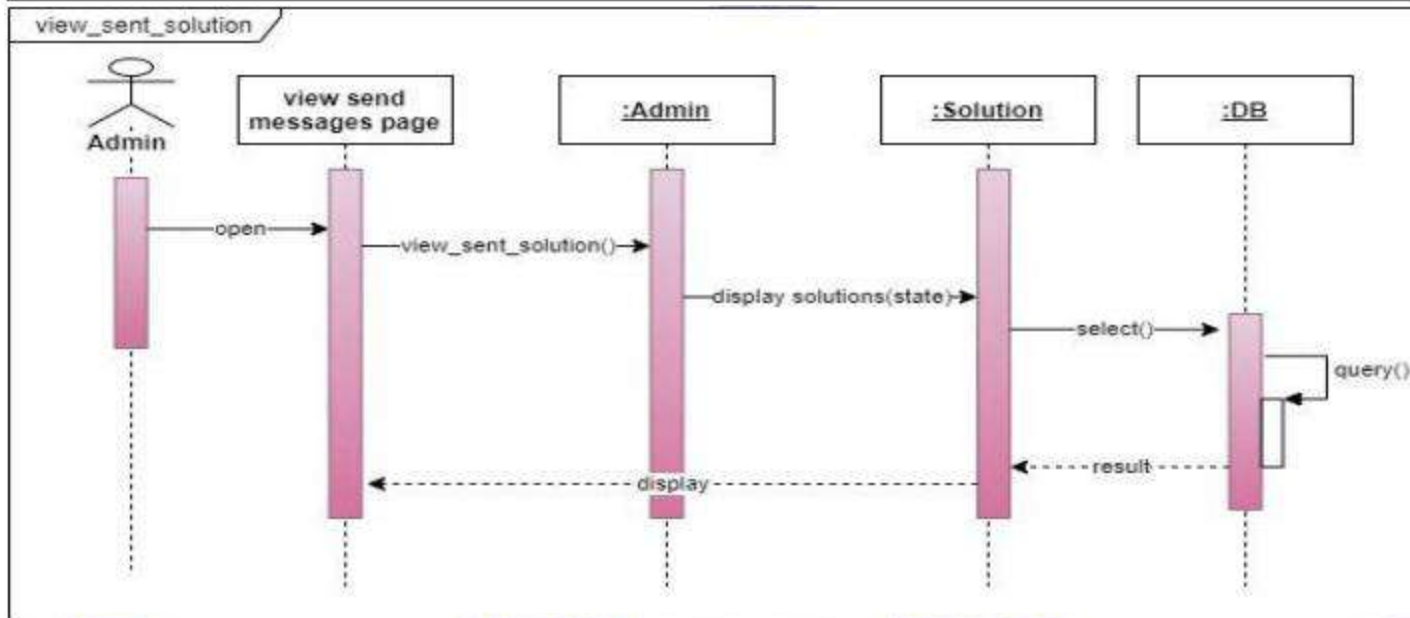
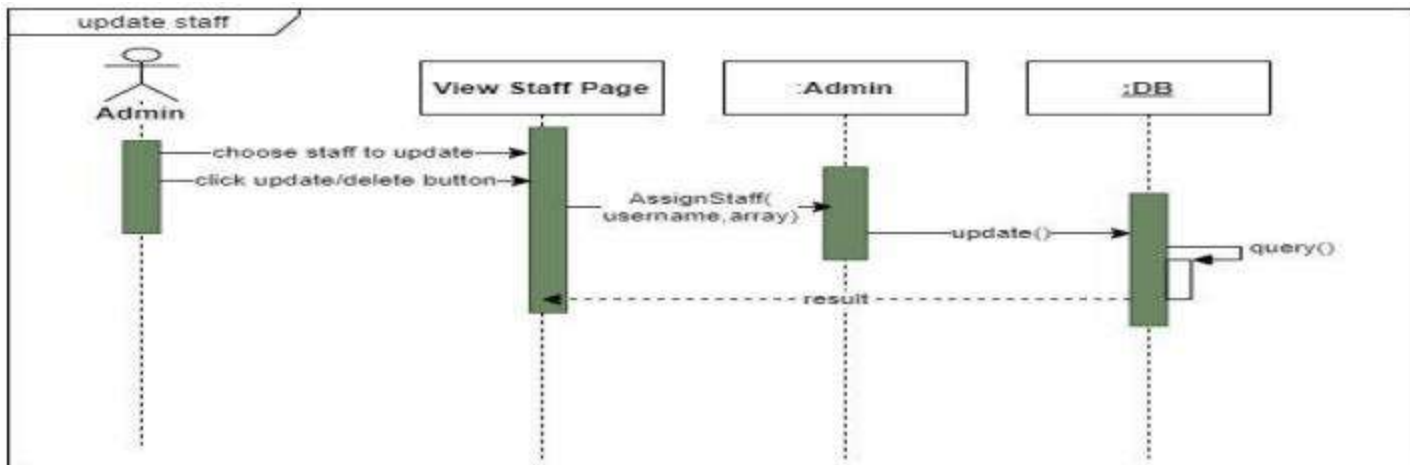




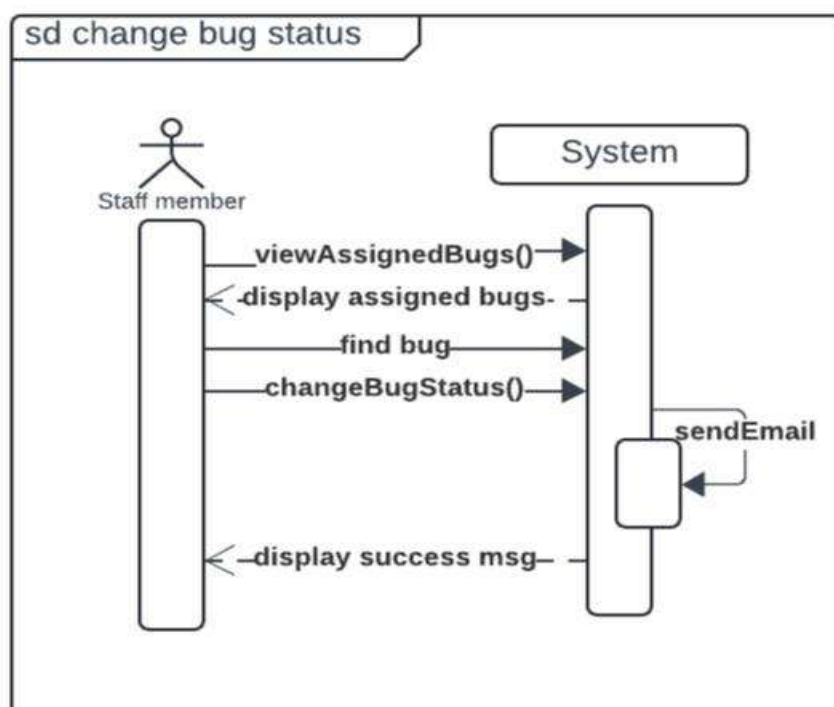
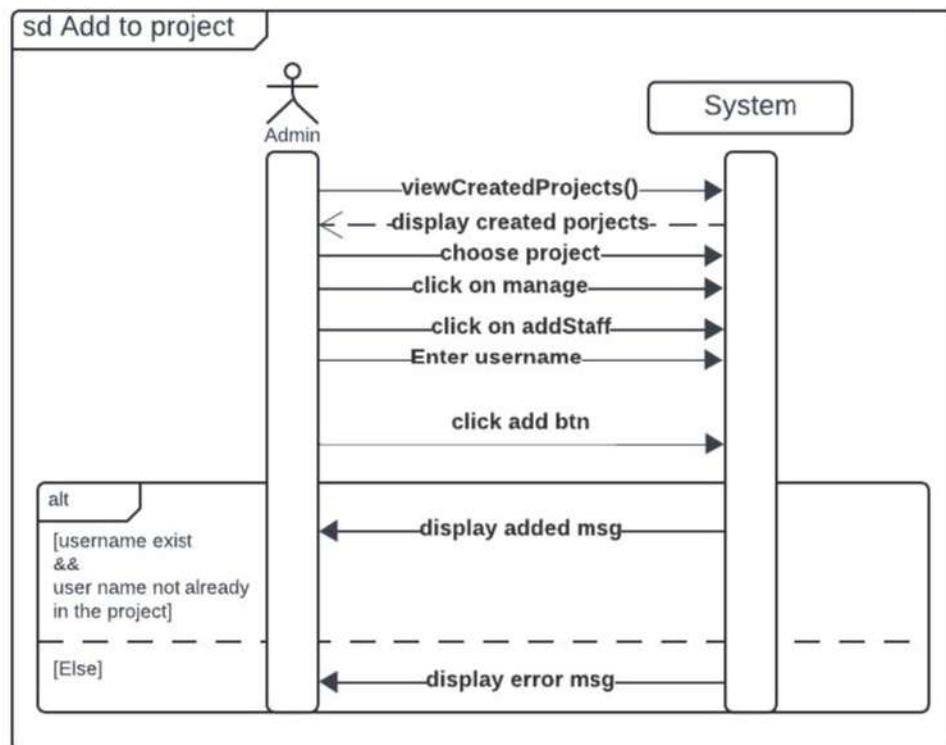


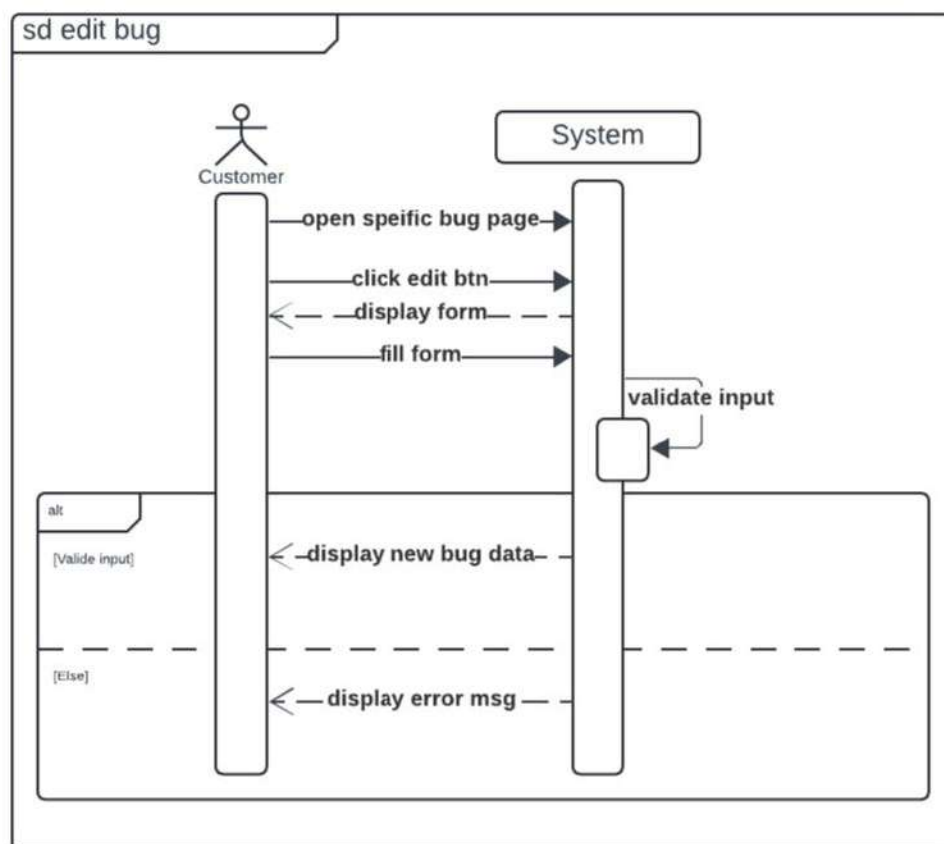
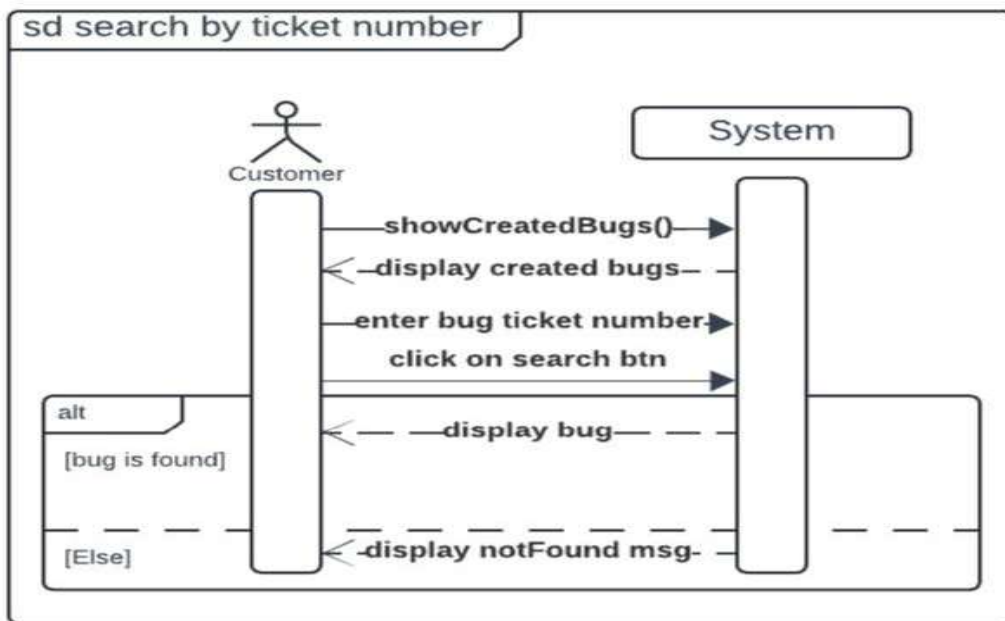


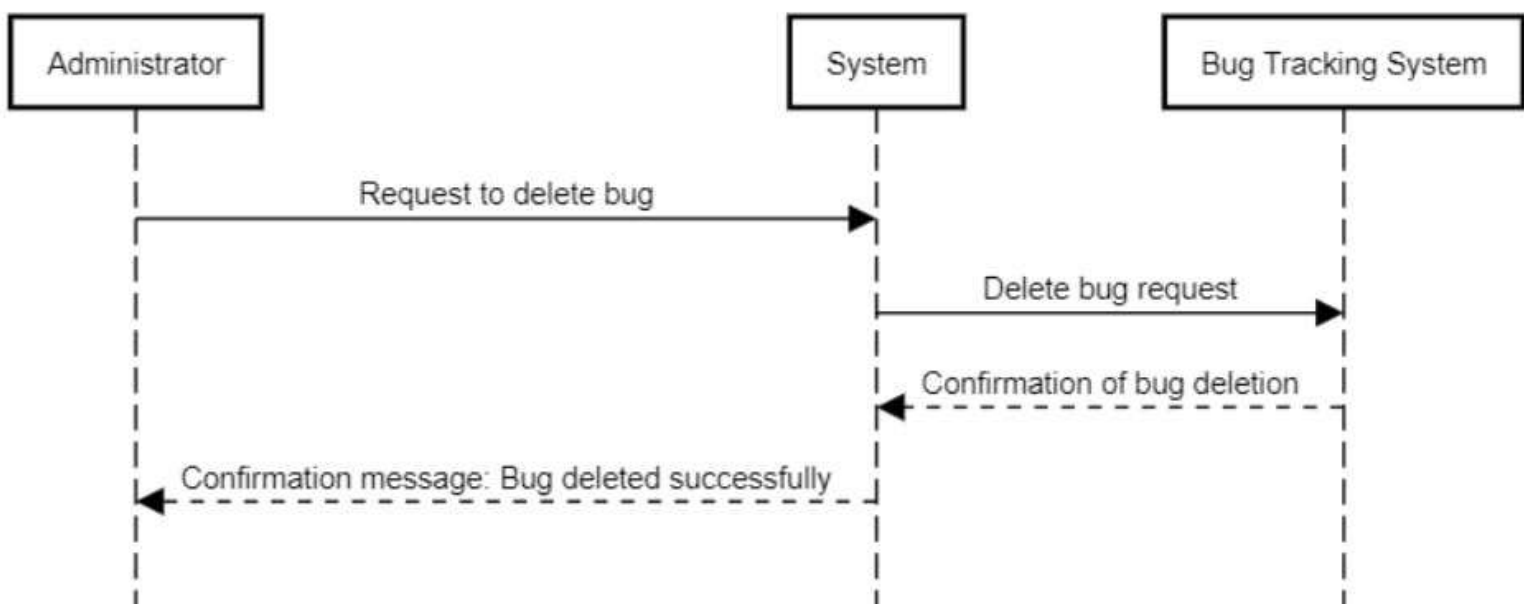
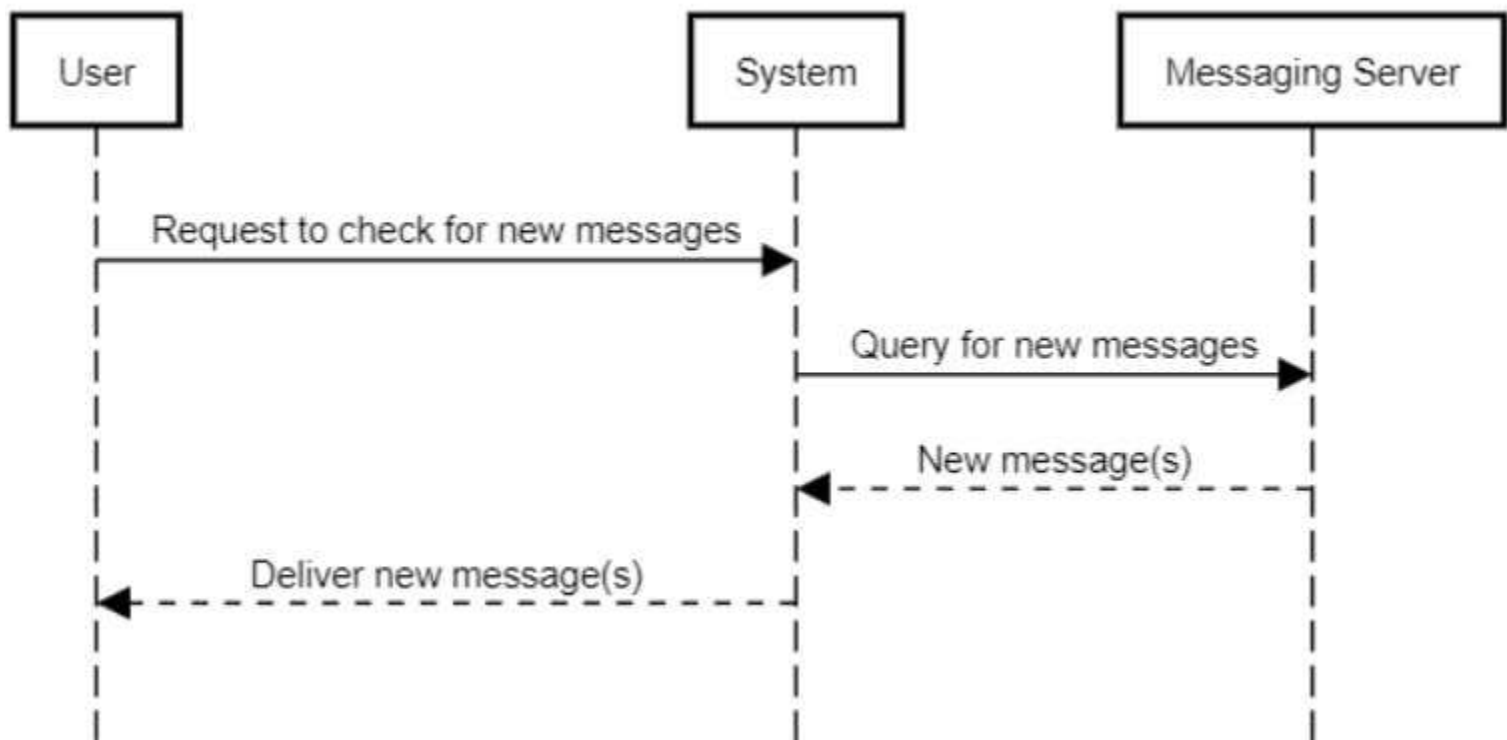
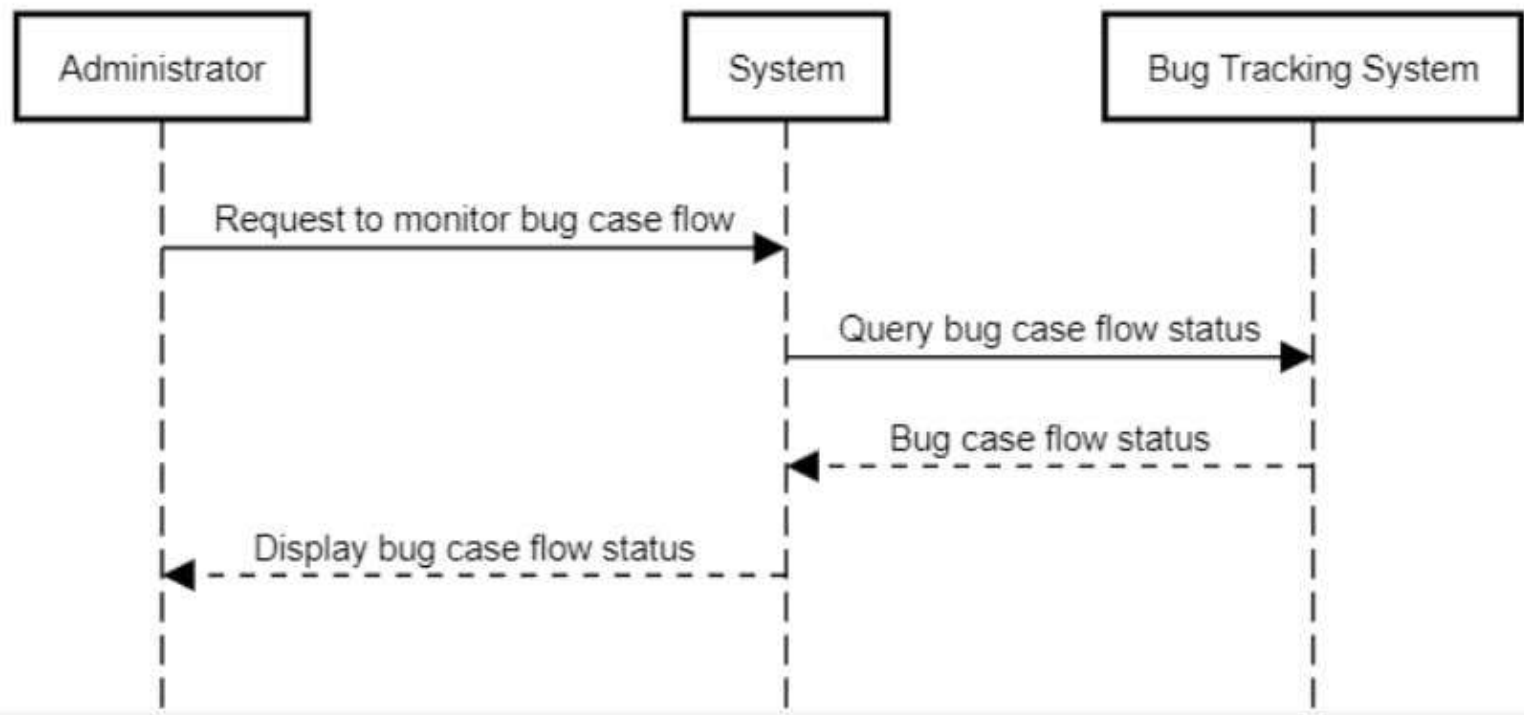


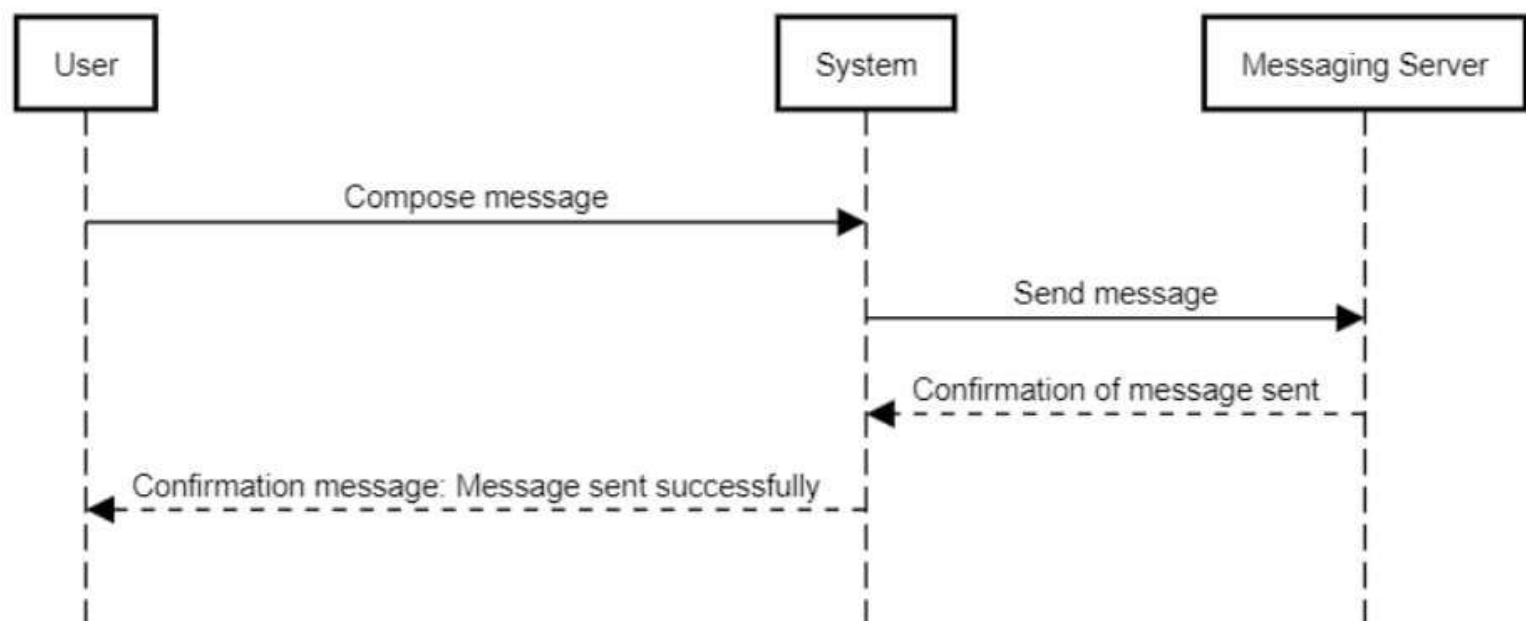
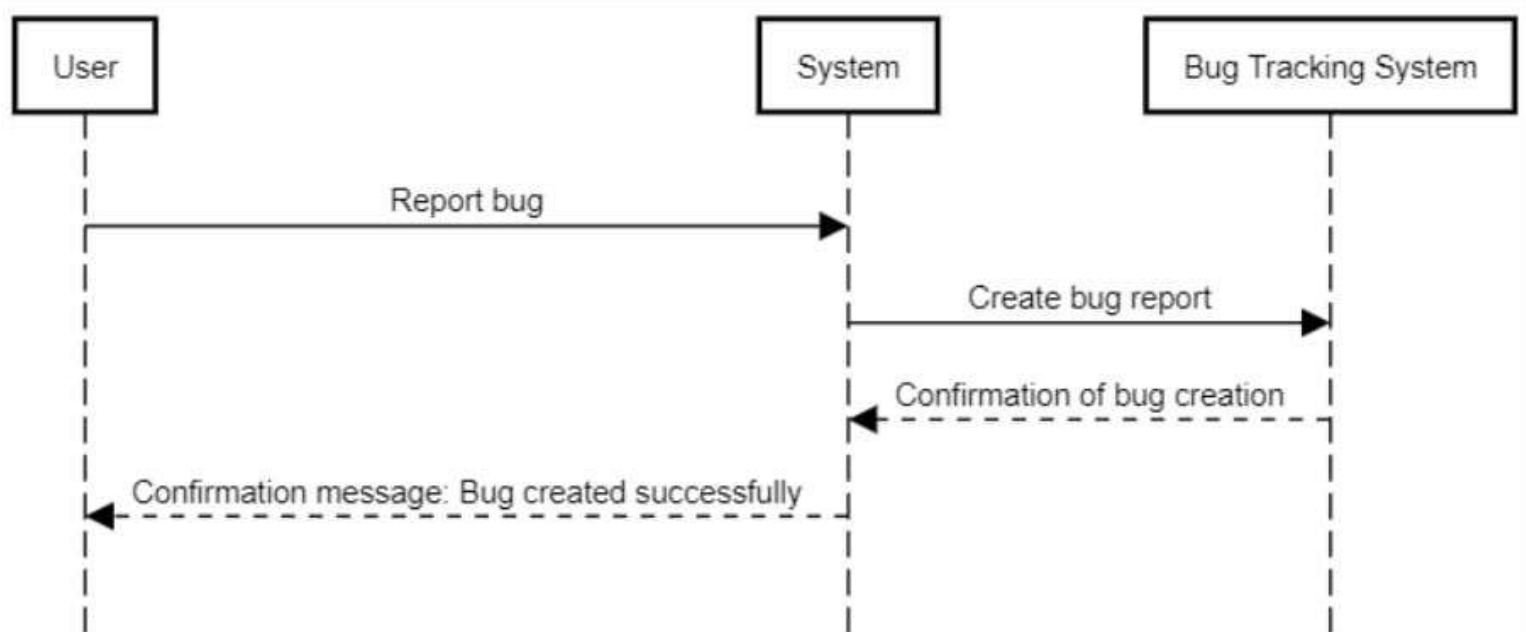


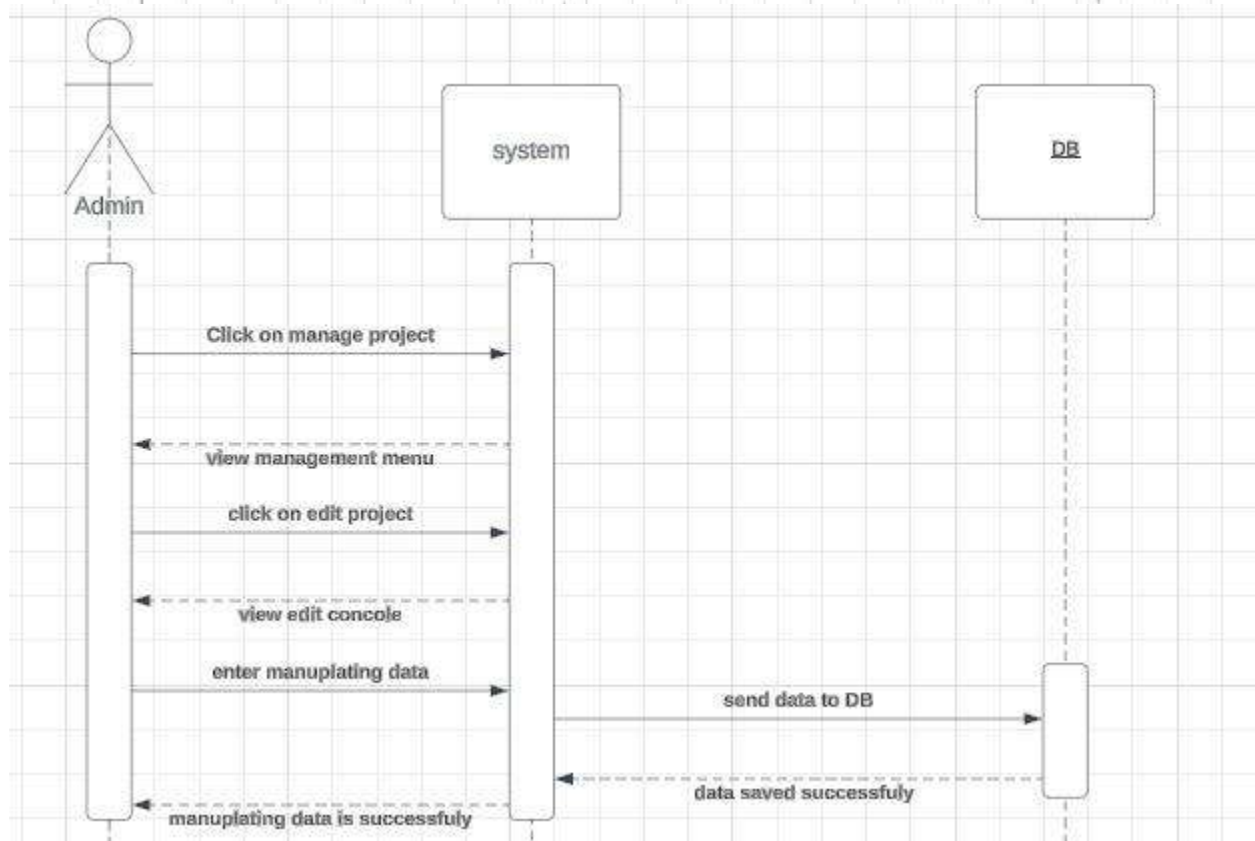
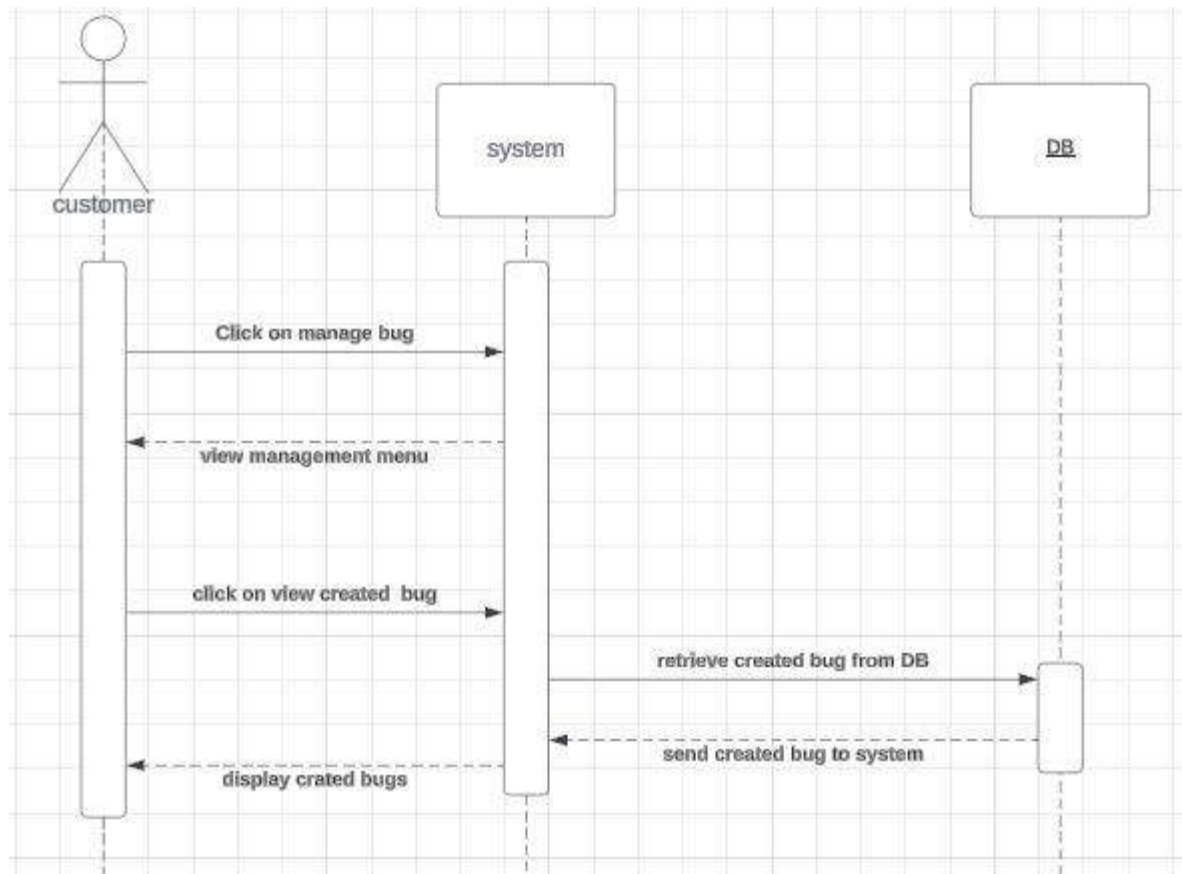
System Sequence Diagrams

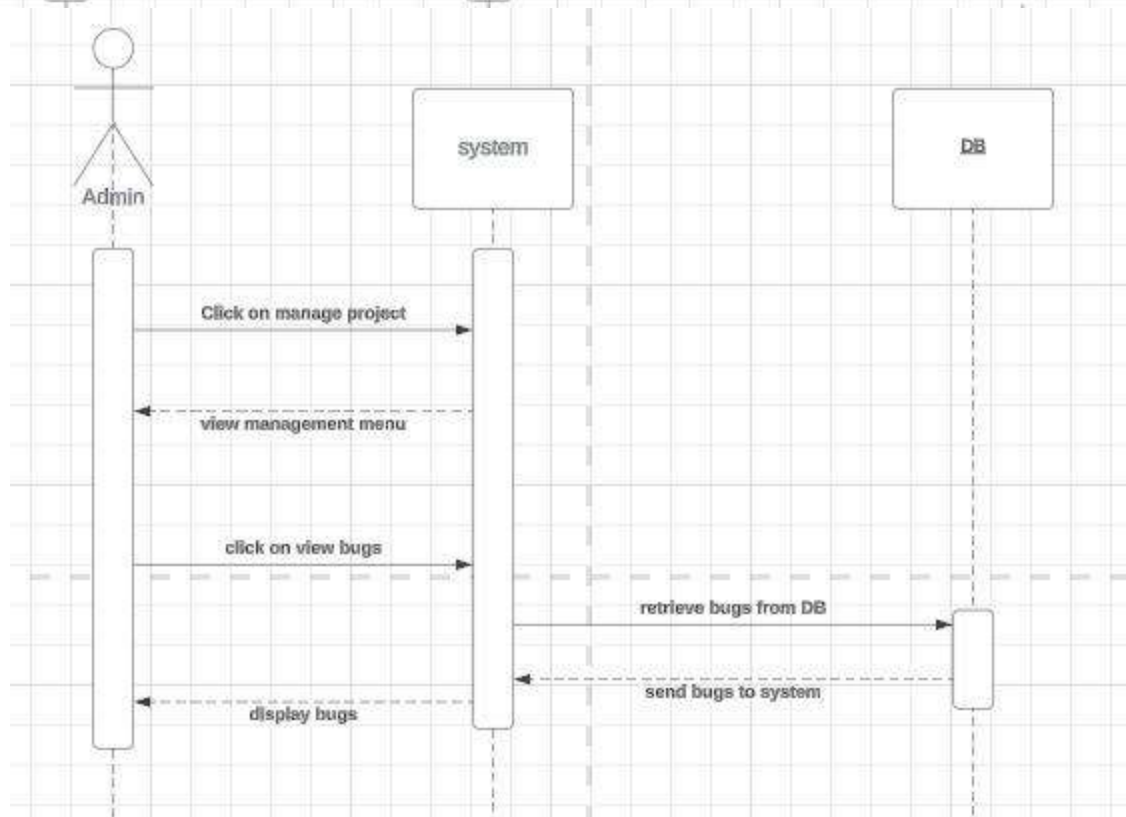
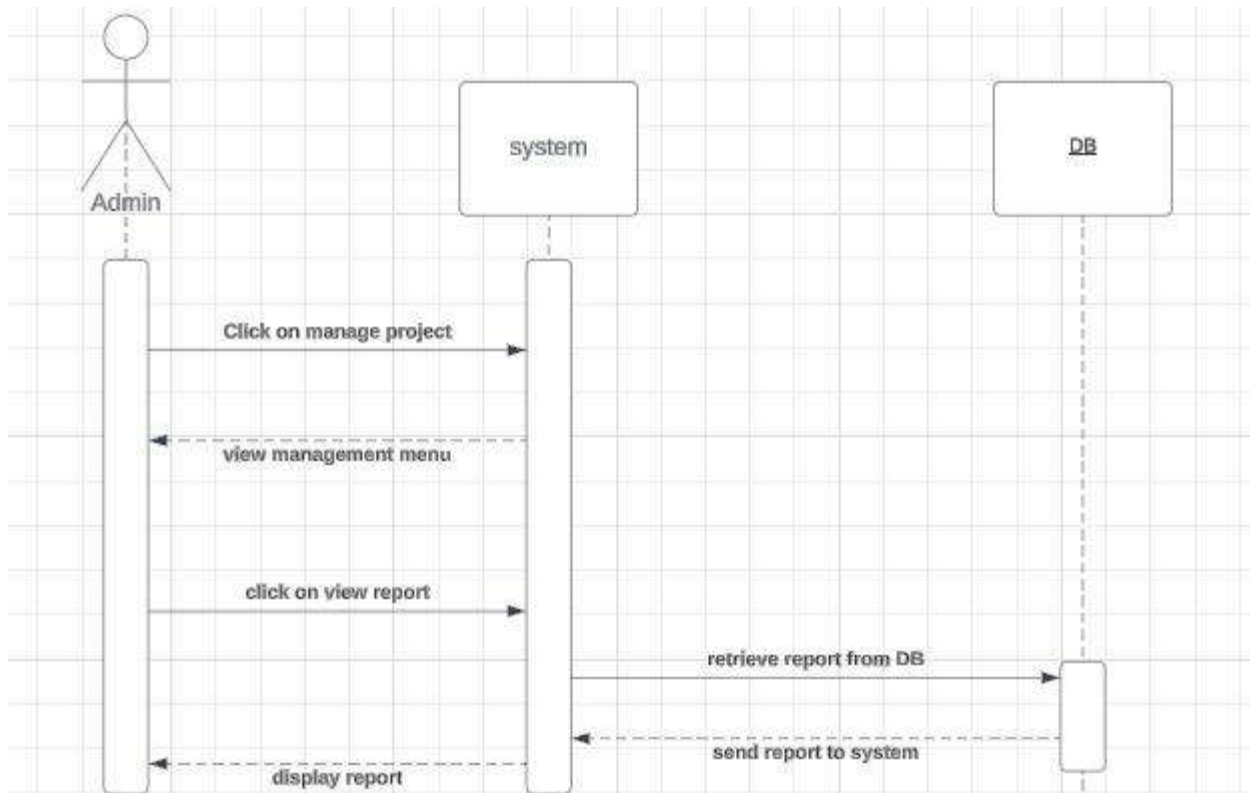


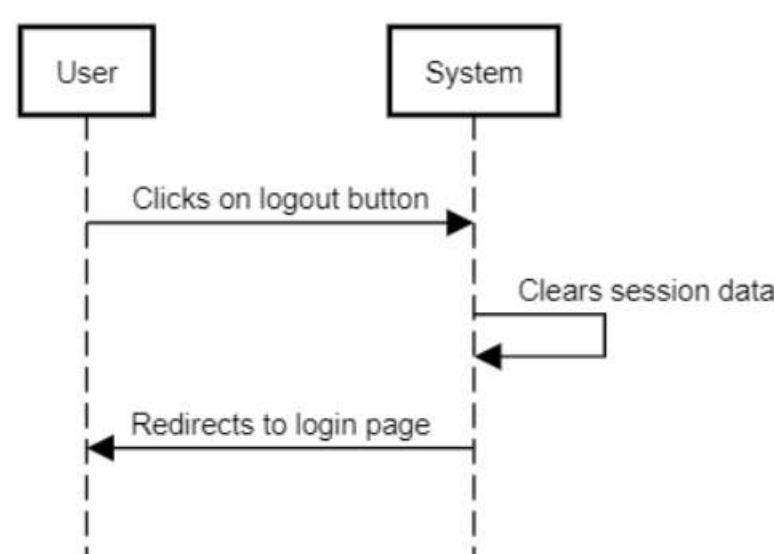
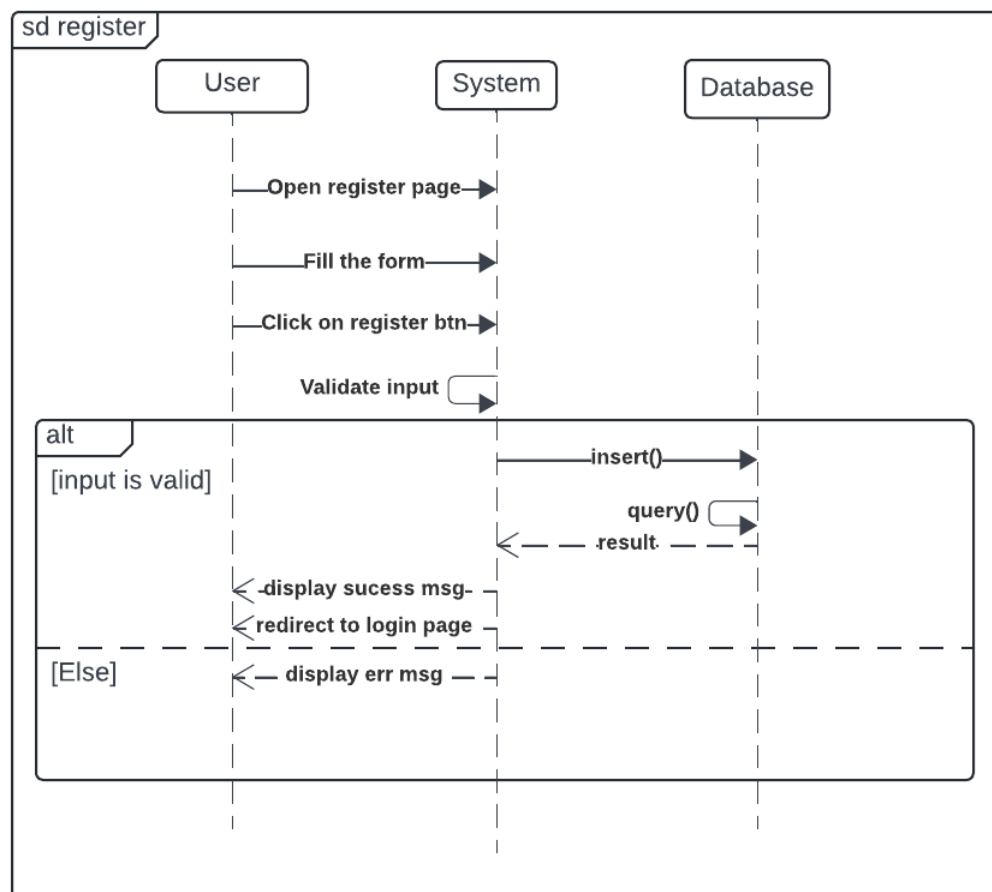


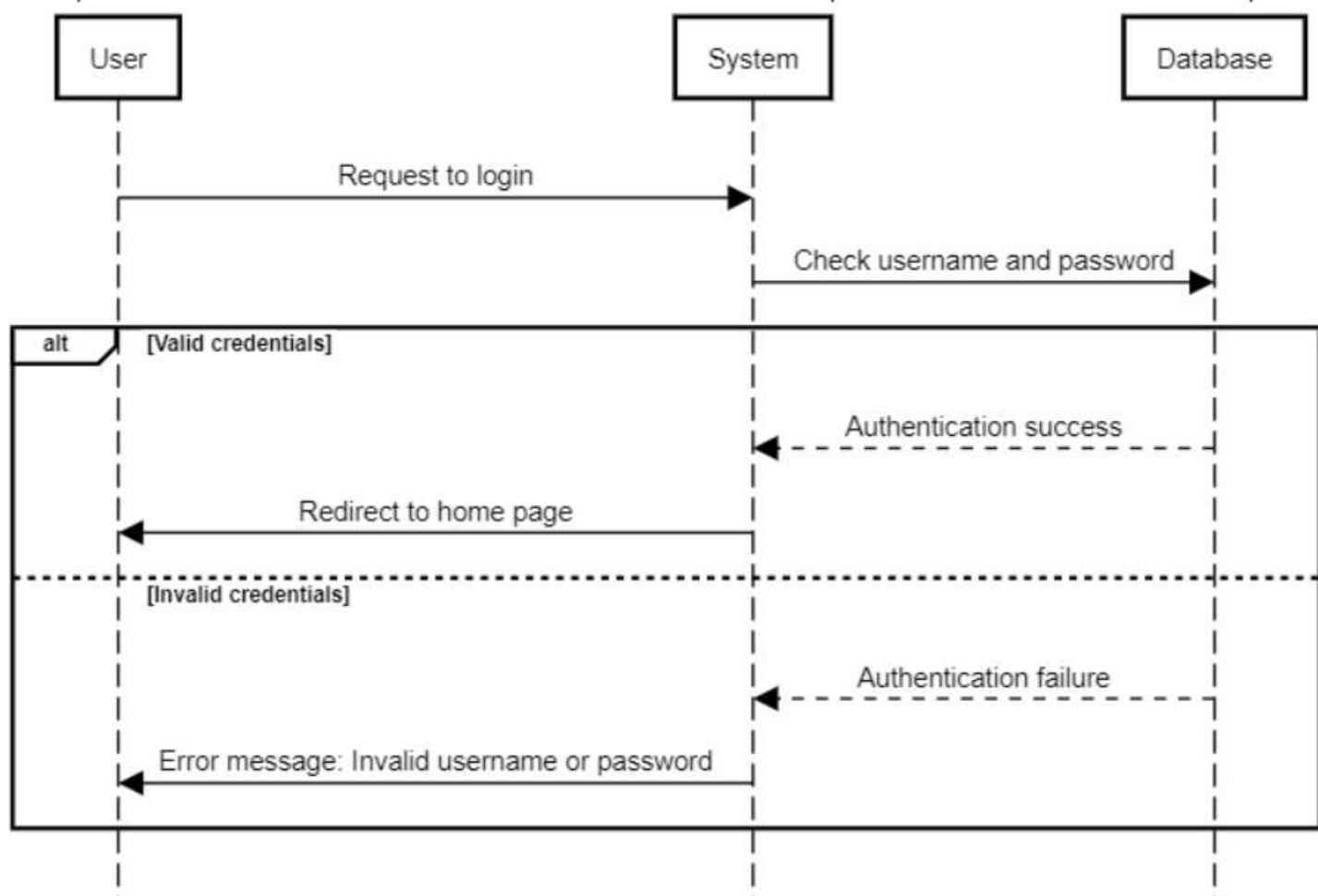
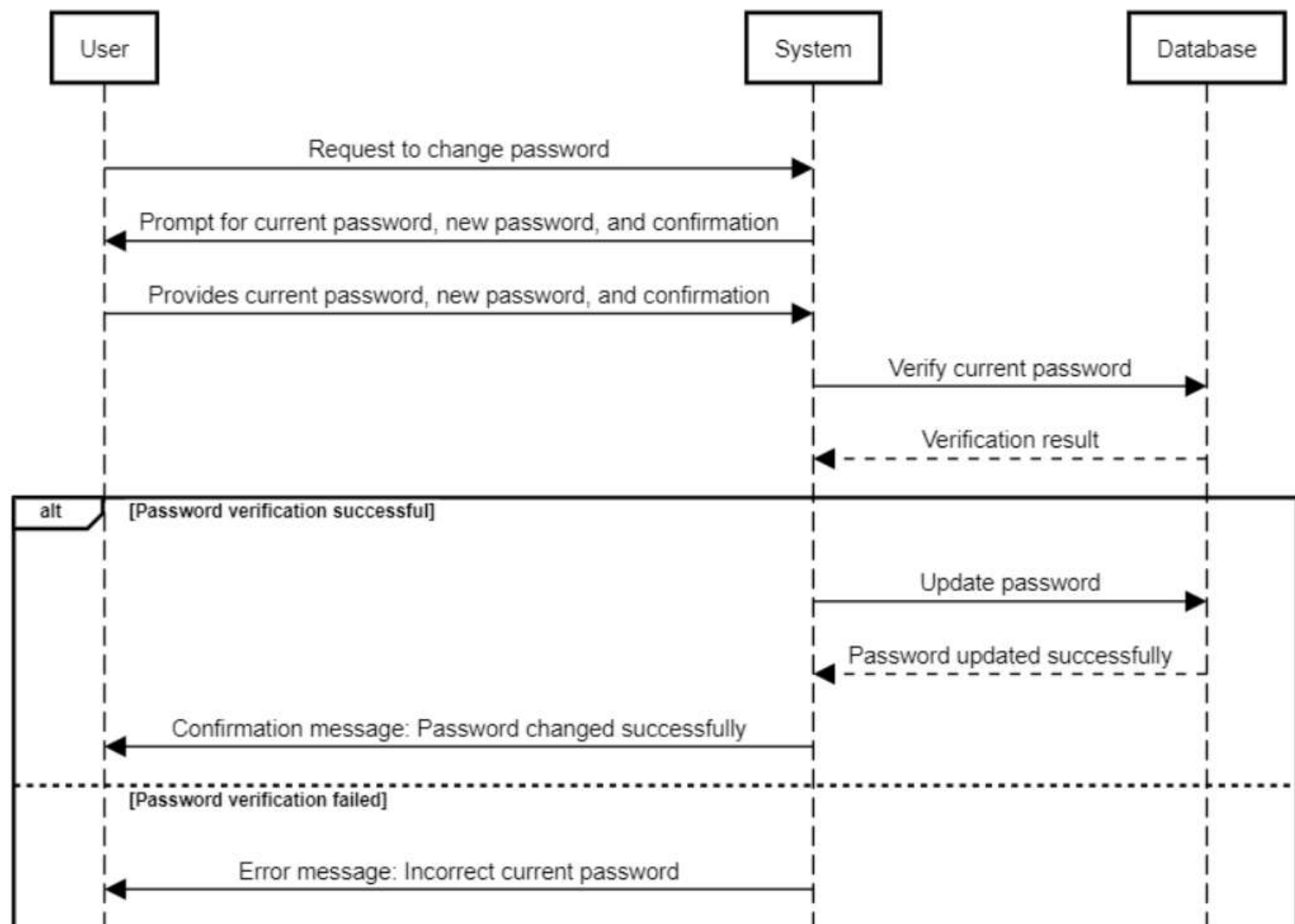


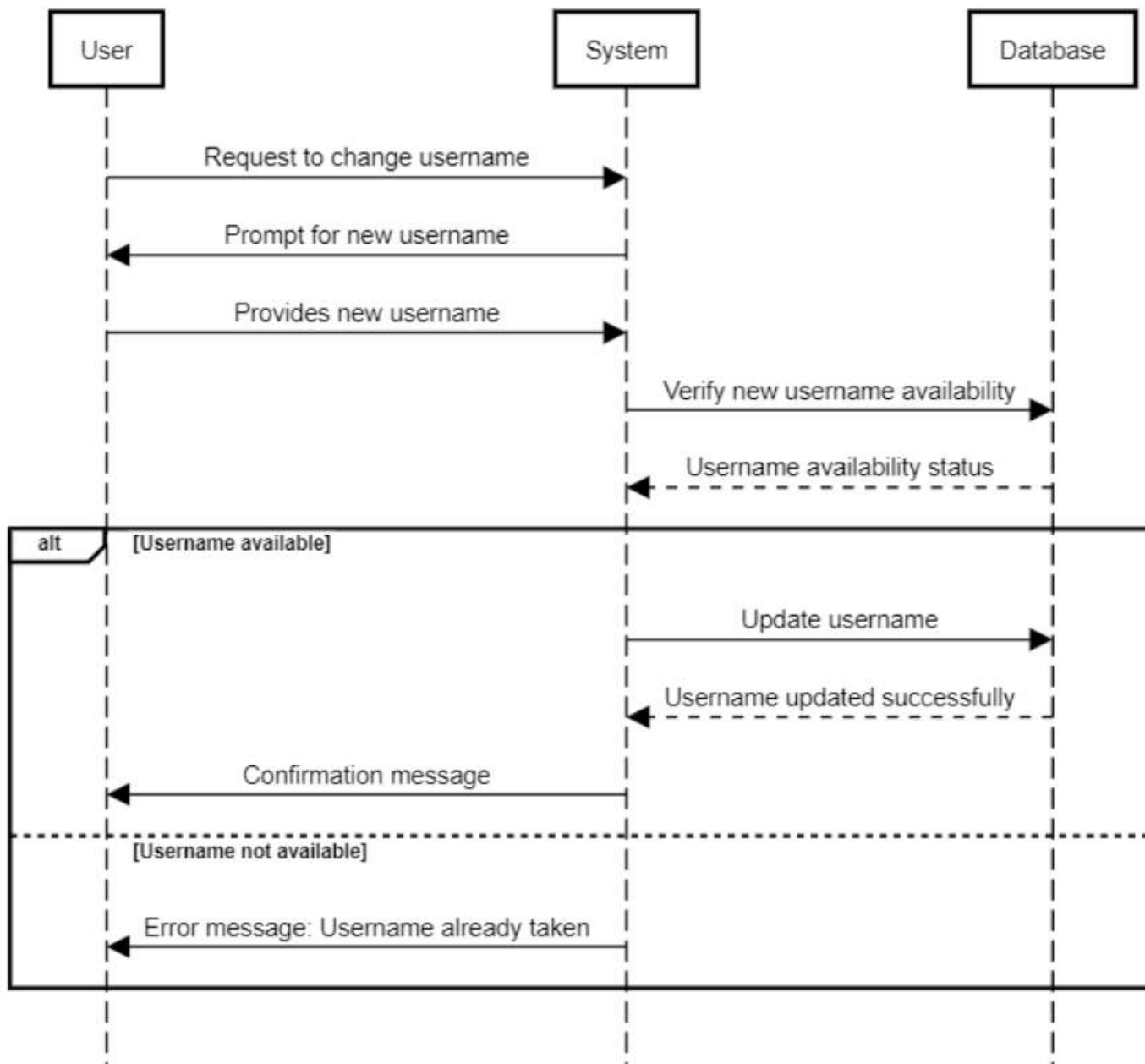






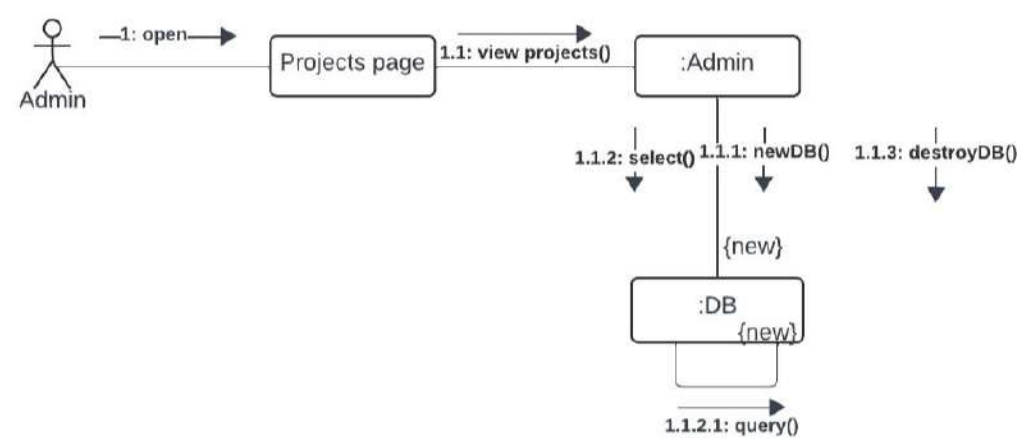




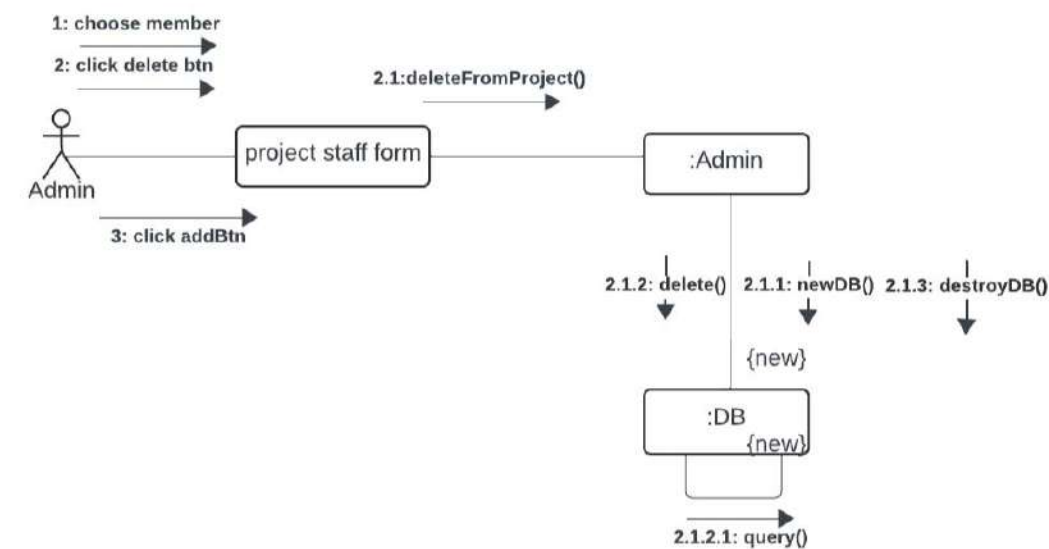


Collaboration/Communication Diagram

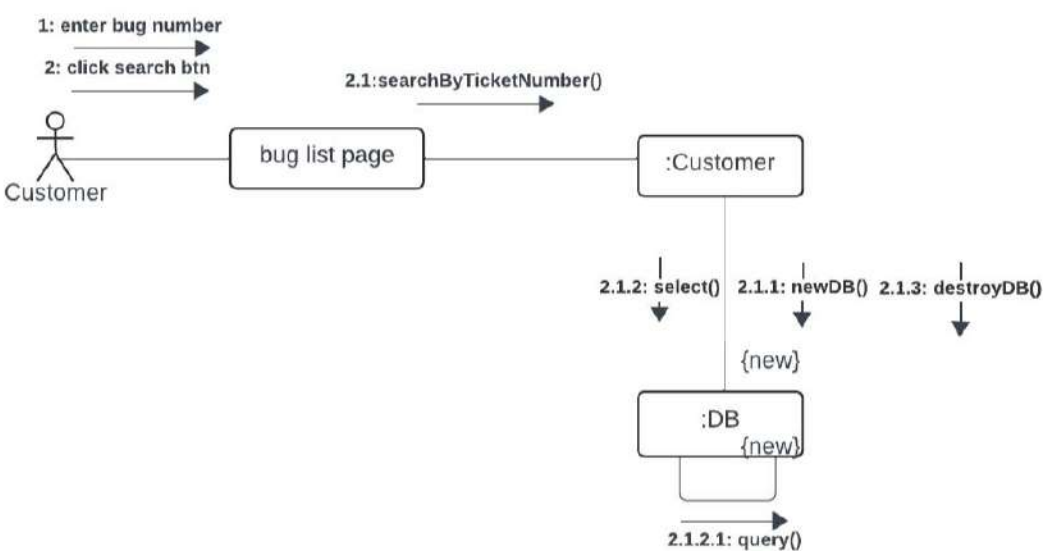
View projects



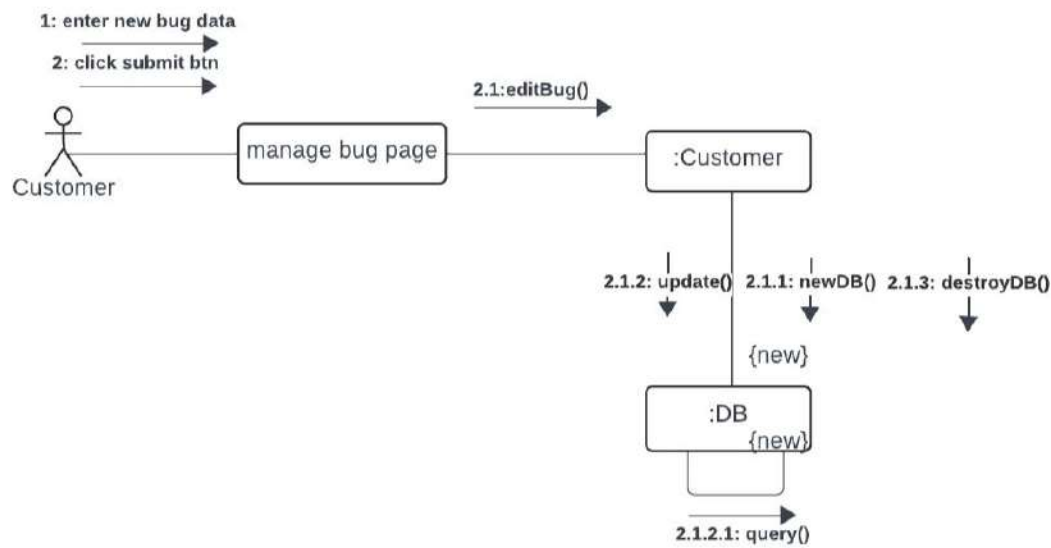
delete from project



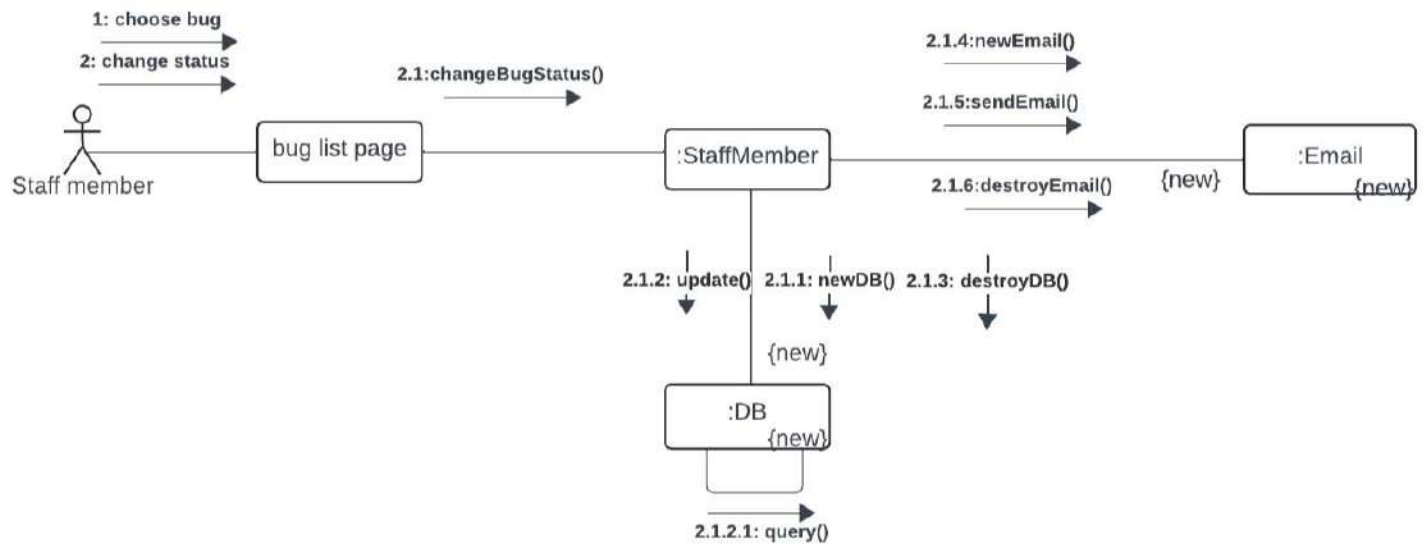
search by thicket number



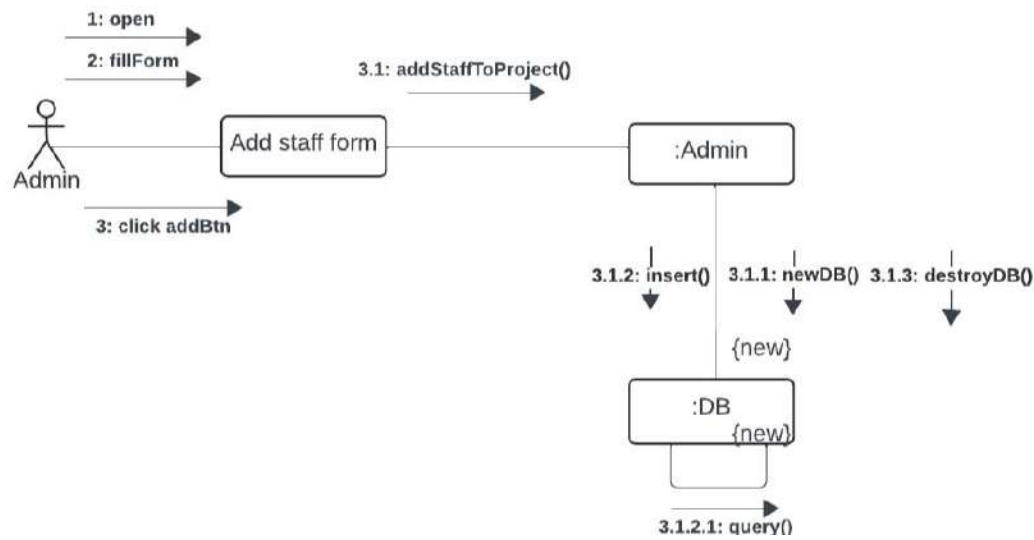
edit bug



change bug status



Add to project

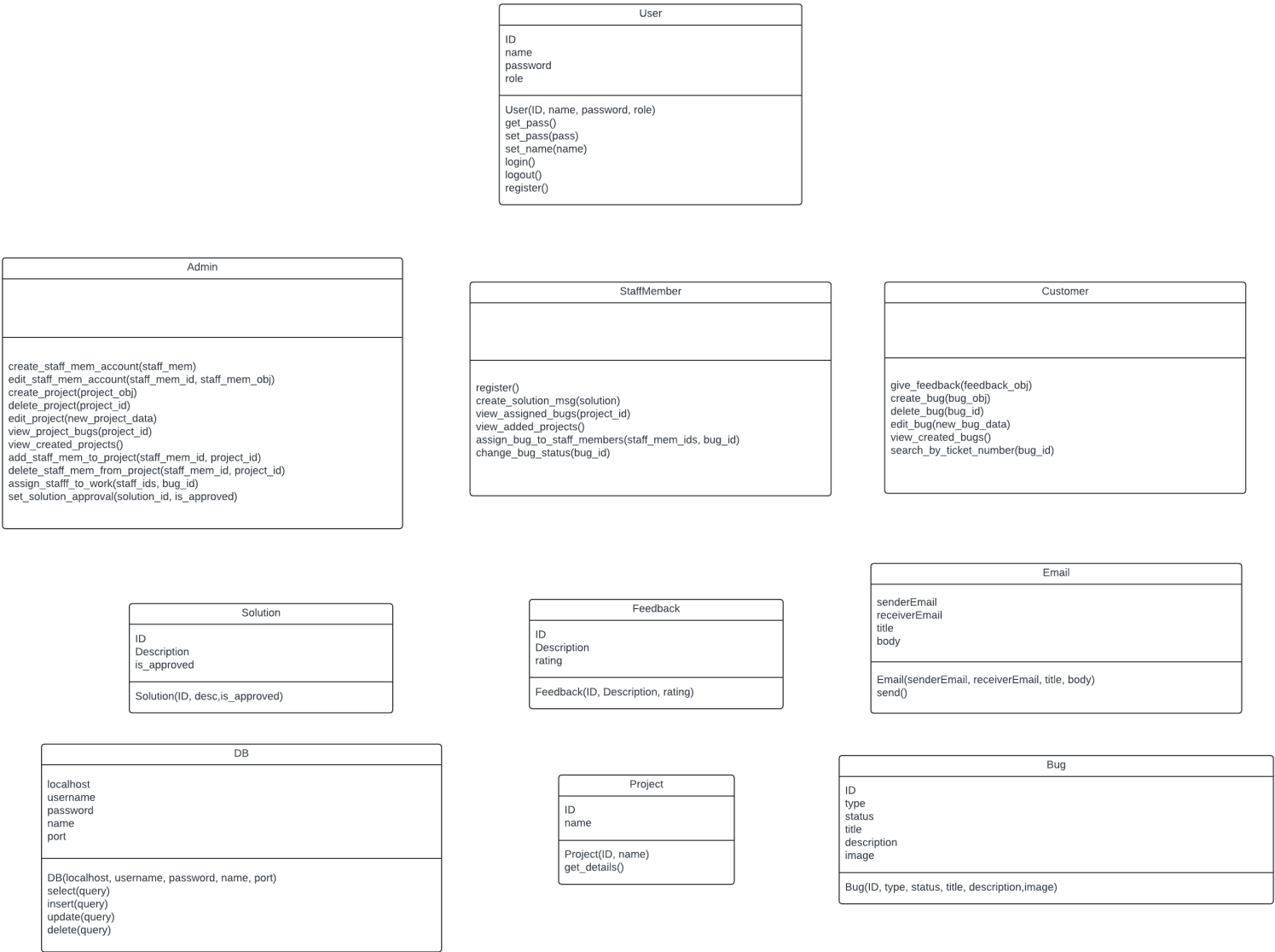


Use cases strategy

We implemented the use-cases using actor class strategy, because its easier to implement as code and its clearer than one central class and Use-Case class.

- Advantages
 - From some designers' point of view this approach has a clearer structure than the previous one.
- Disadvantages
 - Traceability is more difficult than one central class
 - Limitation of reusability; For example: an Actor-Class as Guest includes methods that are related to the role of the guest in this application, thus reducing its usefulness in another application
 - There is a further complication when there are two actors that can initiate an interaction

Class Diagram V.2



Design Patterns Applied

We've applied 3 Design patterns:

- CREATIONAL
 - SINGLETON PATTERN
 - IMMUTABLE PATTERN
- STRUCTURAL
 - ABSTRACTION-OCCURRENCE PATTERN

➤ **SINGLETON PATTERN**

Context:

- It is very common to find classes for which only one instance should exist (singleton).
- Examples: Company or university class, Main Window class in GUI

Problem:

- How do you ensure that it is never possible to create more than one instance of a singleton class. And provide a global point of access to it.

Forces:

- The use of a public constructor cannot guarantee that no more than one instance will be created.
- The singleton instance must also be accessible to all classes that require it; therefore, it must often be public.

Solution:

- 1) Have the constructor private to ensure that no other class can create an instance of the class singleton.
- 2) Define a public static method, The first time this method is called, it creates the single instance of the class "singleton" and stores a reference to that object in a static private variable.

➤ **IMMUTABLE PATTERN**

Context:

- An immutable object is an object that has a state that never changes after creation

Problem:

- How do you create a class whose instances are immutable?

Forces:

- There must be no loopholes that would allow 'illegal' modification of an immutable object

Solution:

- 1) Ensure that the constructor of the immutable class is the only place where the values of instance variables are set or modified.

- 2) Instance methods which access properties must not change instance variables.

ABSTRACTION-OCCURRENCE PATTERN

Context:

- In a domain model you find a set of related objects “occurrences”; the members of such a set share common information but also differ from each other in important ways.
- Example: All copies of the same book in a library, Episodes of a television series have different story lines, Flights that leave at the same time every day for the same destination.

Problem:

- What is the best way to represent such sets of occurrences?

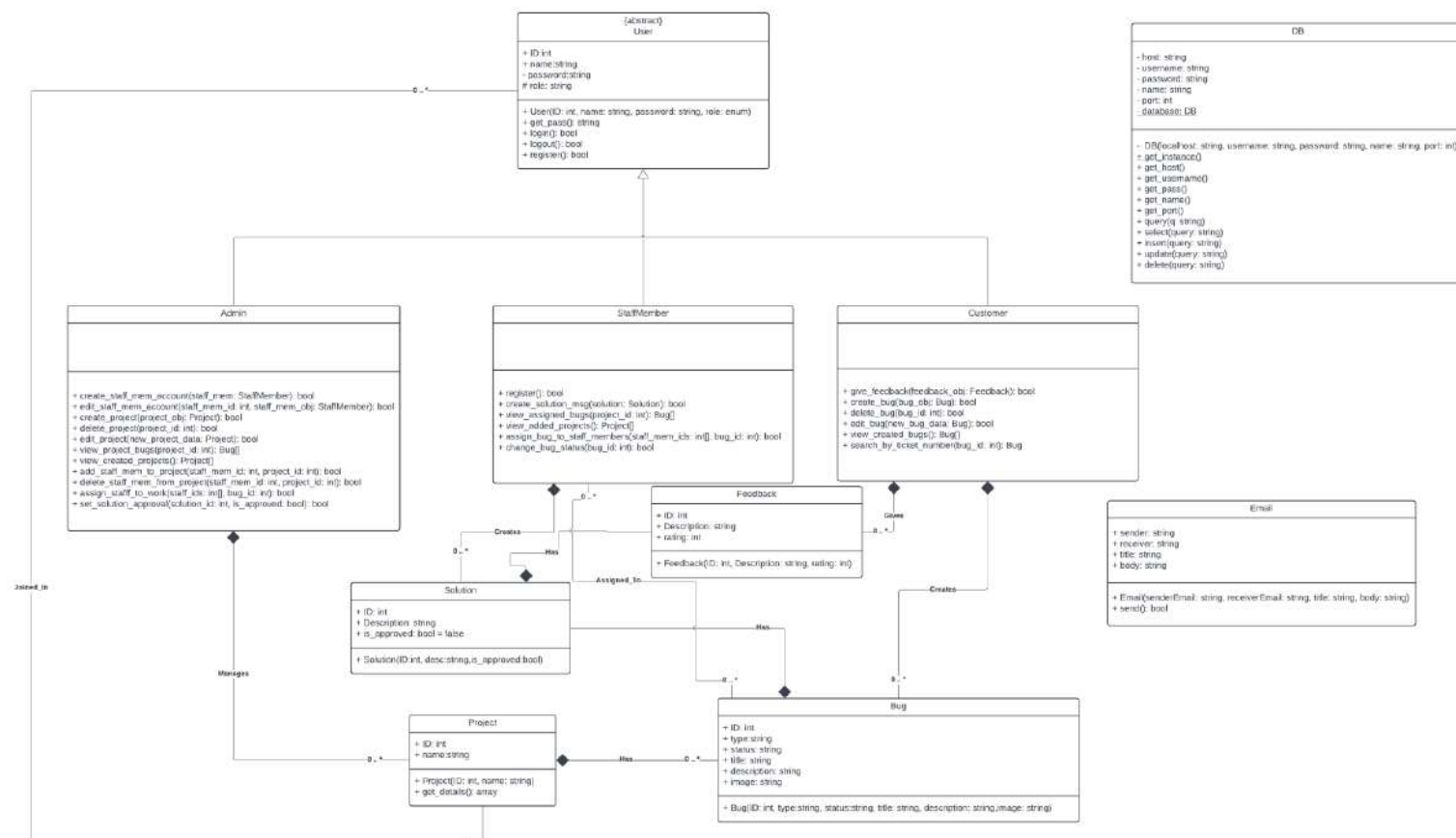
Forces:

- You want to represent the members of each set of occurrences without duplicating the common information.

Solution:

- 1) Create an “abstraction” class that contains the common data.
- 2) Then create an “occurrence” class representing the occurrences of this abstraction.
- 3) Connect these classes with a one-to-many association.

Class Diagram V.3



design smell

- design smell is a structure that affects the design and quality design negatively .
- Some of the smell design examples is
 - Large class
 - Long parameters
 - Long method
 - Duplicate code

Large class: when class has an inconsequential large number of attributes and methods.

Long parameters: when any function has a large number of parameters which affects the readability, complexity, Testing, and maintainability of the system.

Long method: when a method has a lot of lines of codes which makes it hard to analysis the code and determine the task of this function.

Duplicate code: when you write the same code more than once (copy and past) it affects the cost, maintainability, and lines of code of the system.

Class Structuring Criteria

There are three class structuring criteria:

(1) Boundary class

- the user interacts only with the boundary classes.

(2) Controller class

- These classes typically don't contain any business functionality. However, their main task is to transfer control to the appropriate business logic class, depending on a few inputs received from the boundary classes.

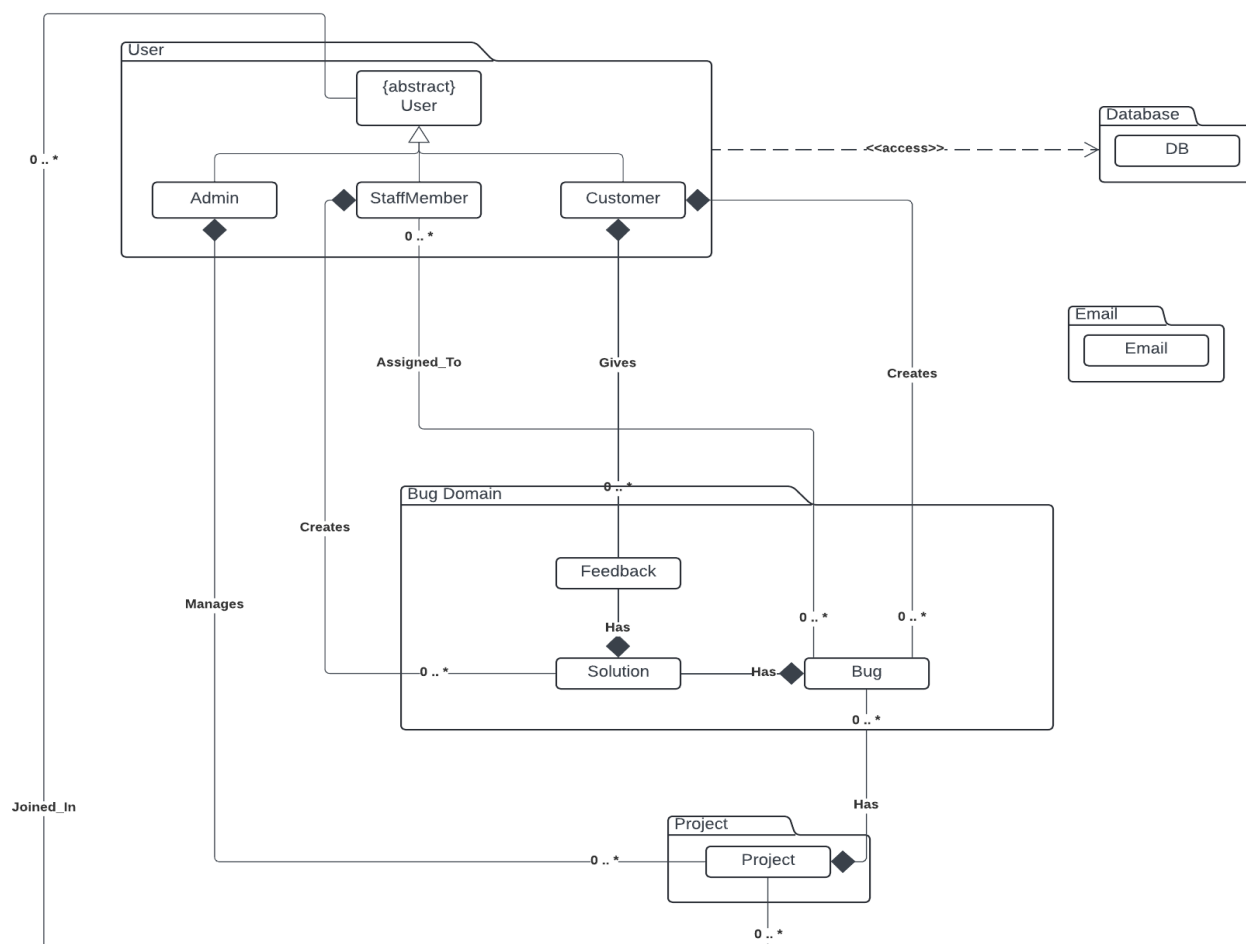
(3) Entity class

- These classes are those that contain the business functionality. Any interactions with back-end systems are generally done through these classes

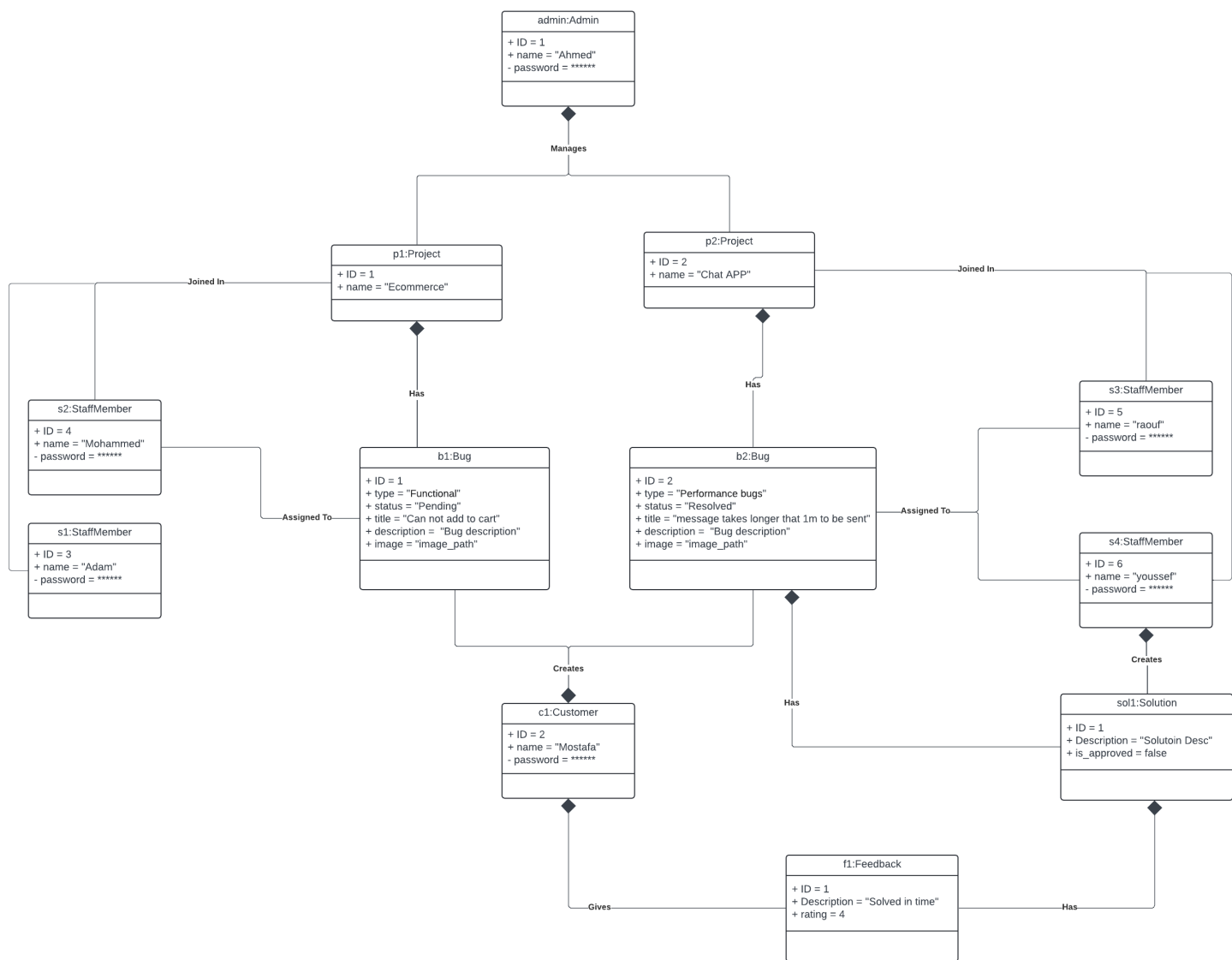
Our class diagram has:

- **Controller class:** User, Admin, Customer, StaffMember, Project, Bug, Solution, Feedback
- **Entity class:** DB

Package Diagram grouping relevant Classes into Packages

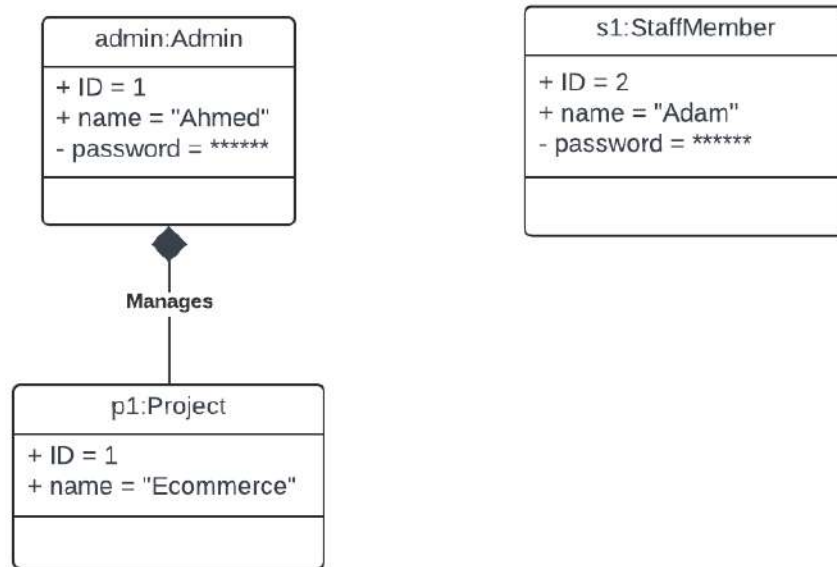


Object Diagrams

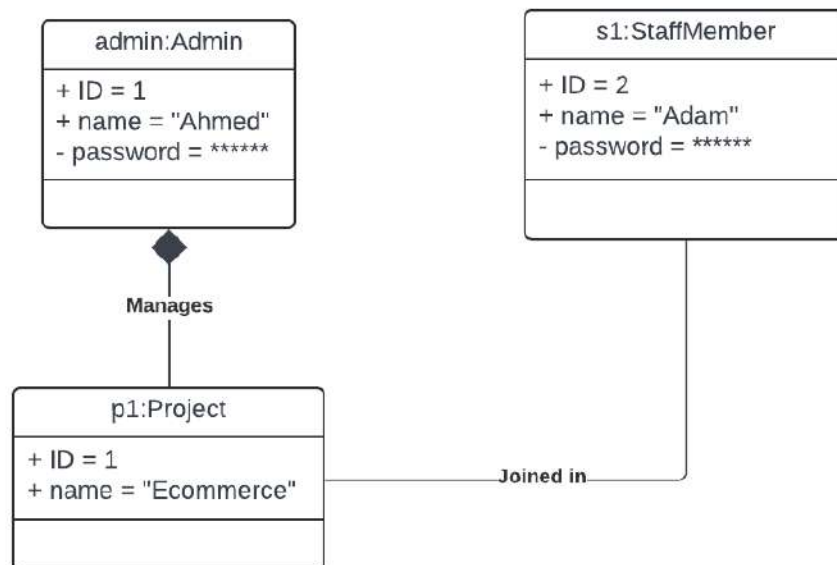


Add to project

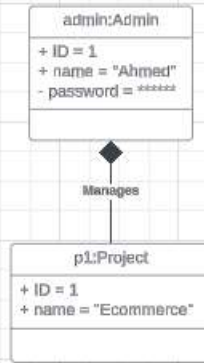
Precondition



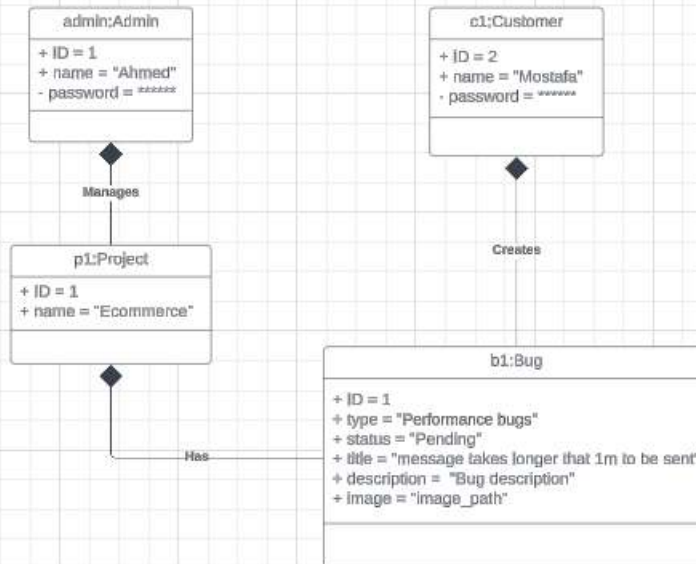
Postcondition



Precondition

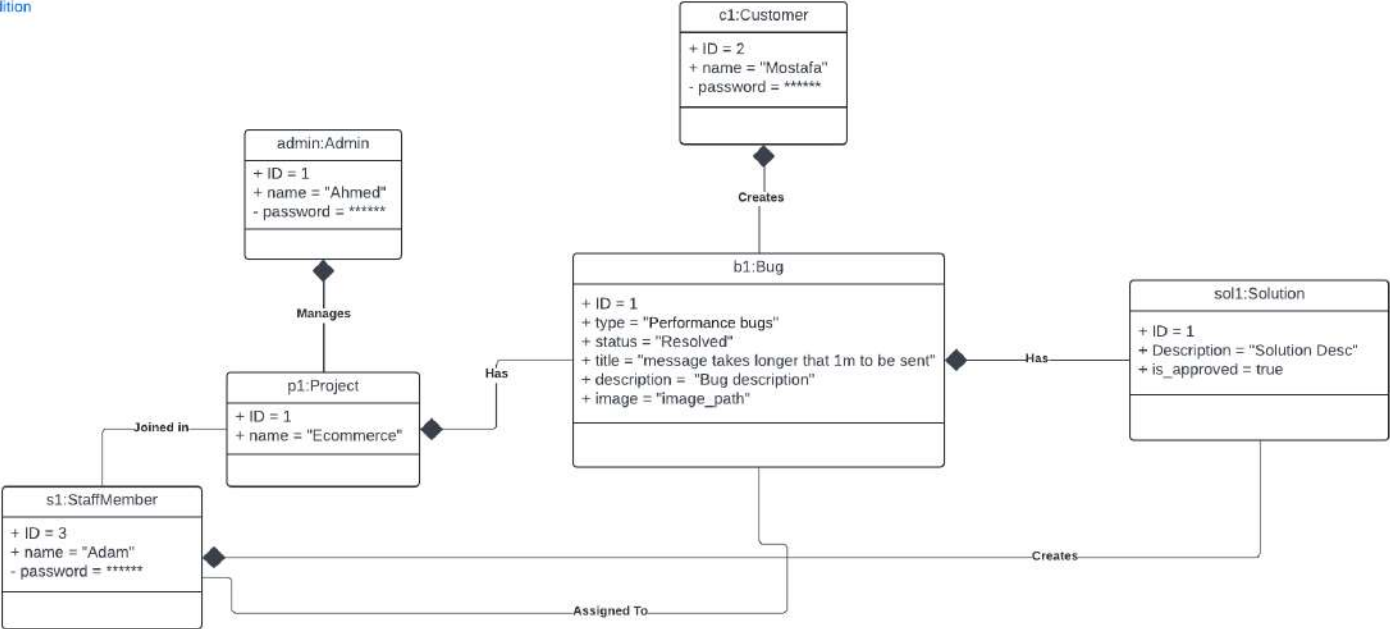


Postcondition

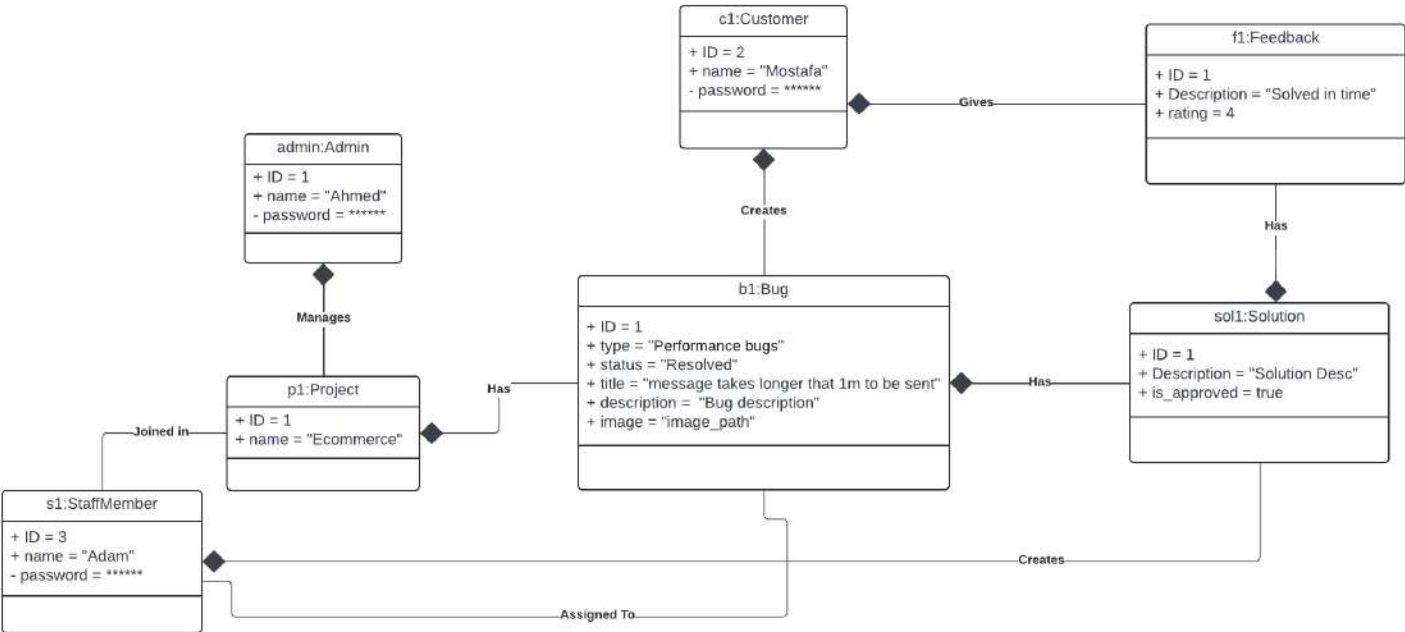


Give feedback

Precondition

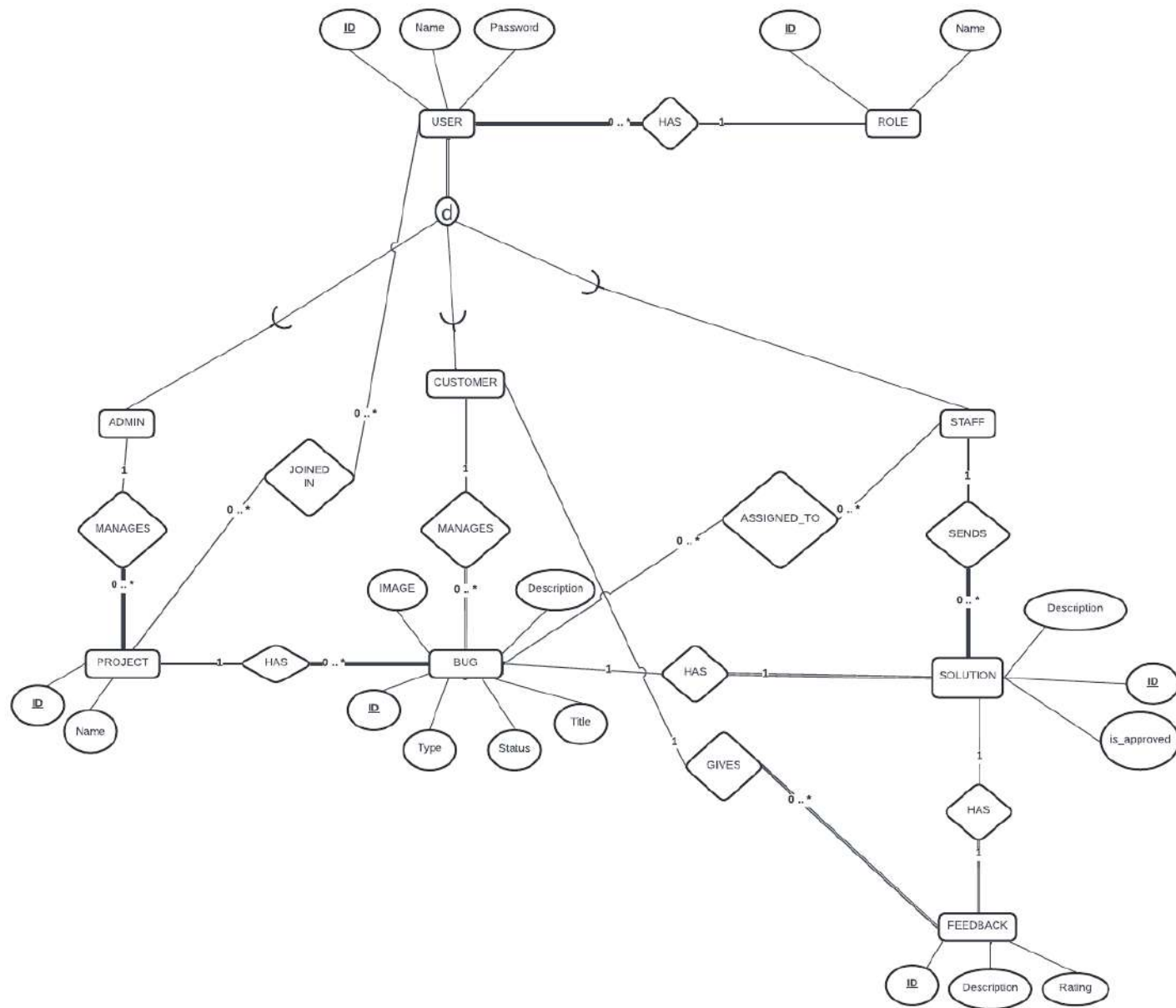


Postcondition

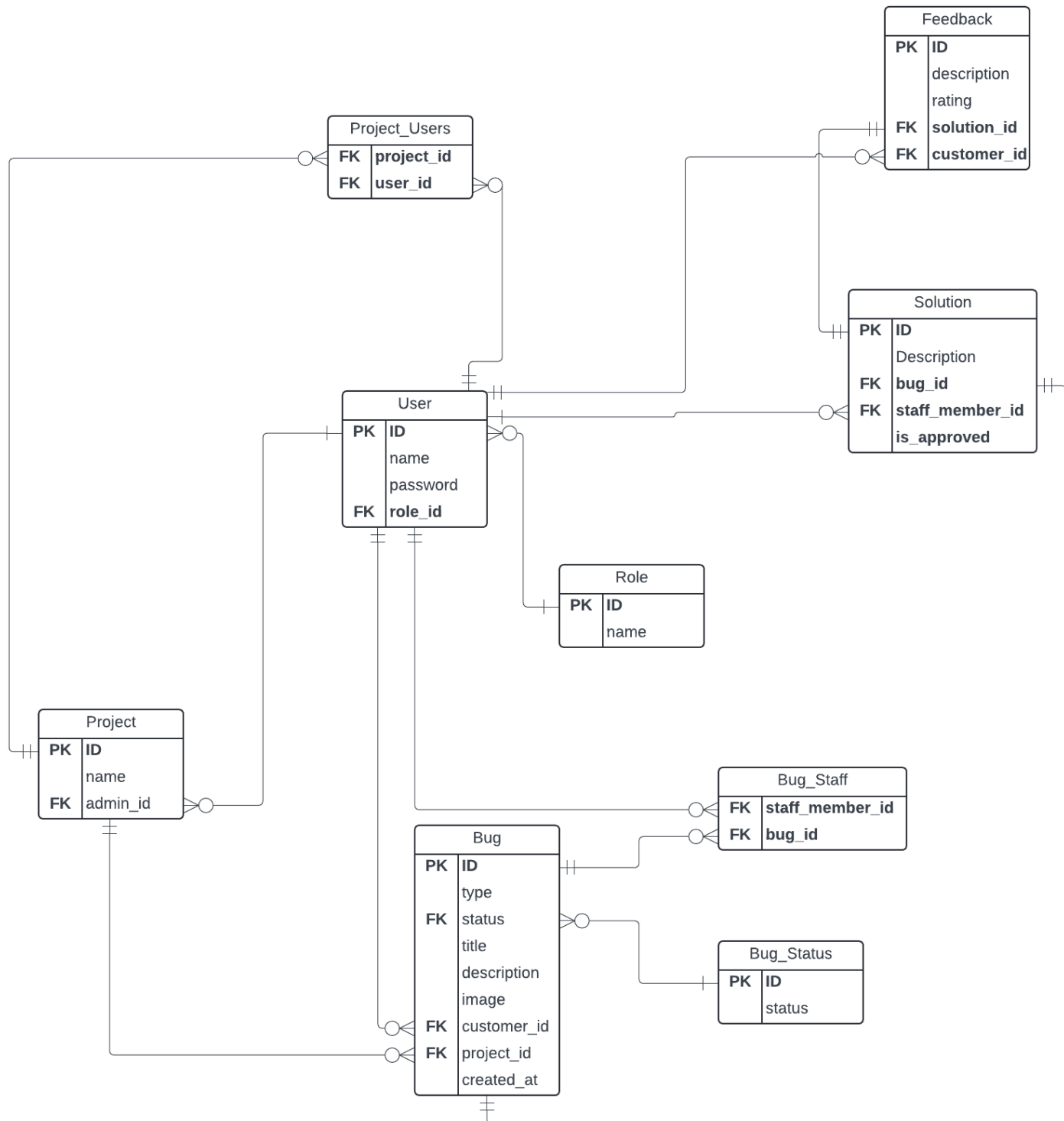


Database Specification

➤ ERD



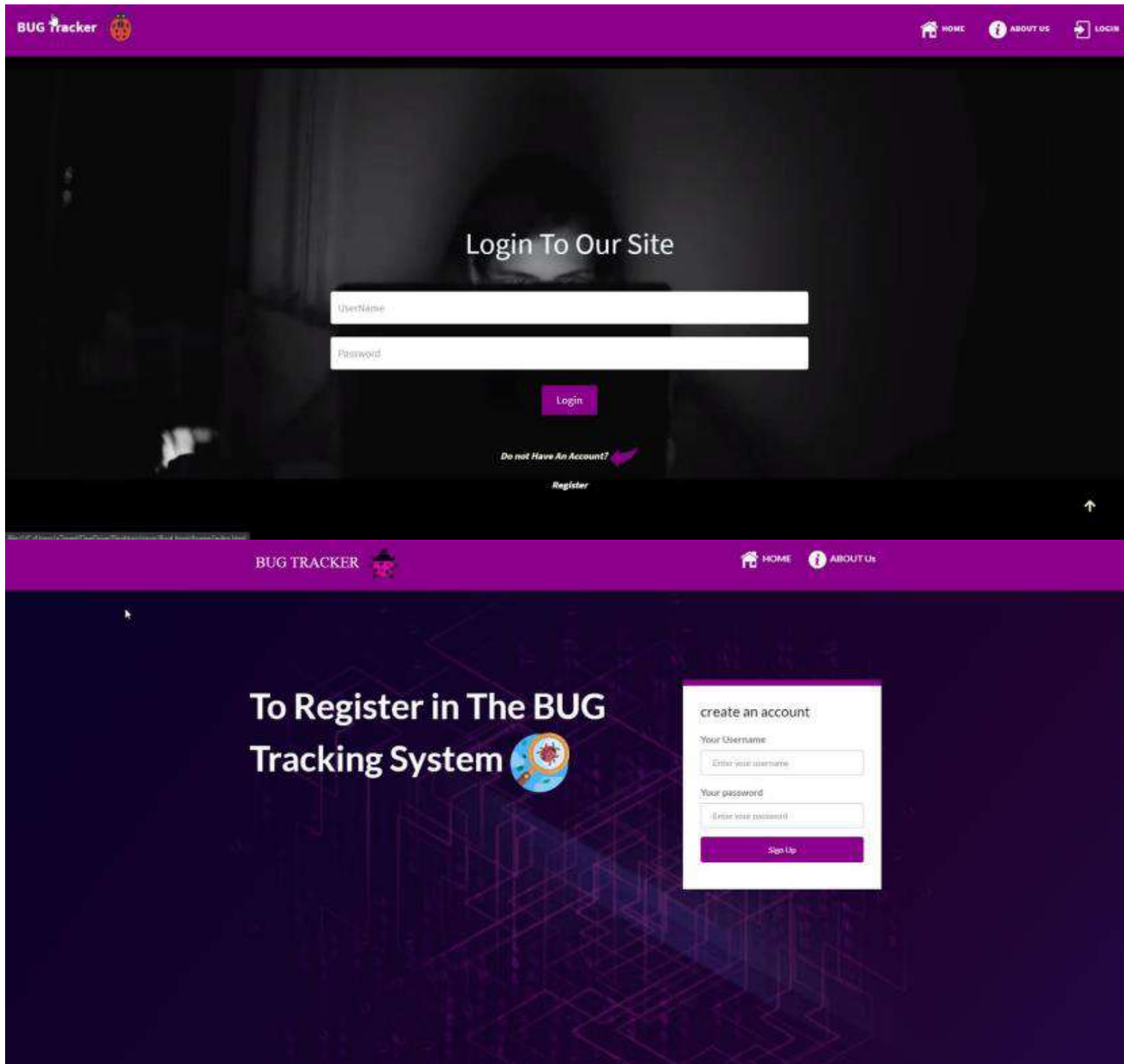
➤ Tables



Part 3

Development Phase (Implementation Details)

Front-End Design for all Functions (HTML, Bootstrap):



BUG TRACKER



Add BUGs



Flow Bugs



Messages



Log Out



Customers Page

Messages

Ticket Number	Solution
1	Show
2	Show
3	Show
4	Show



BUG TRACKER



New BUGs



Projects



Staff



Messages




Log Out




Admin Page

New Bugs


	Project Name	Customer username	Bug State	Error Category	
1					Create Project
2					Create Project
3					Create Project




BUG TRACKER




Add BUGs



Flow Bugs



Messages




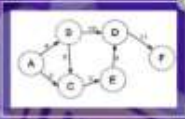
Log Out

Error Category

Error Details

Print Screen From Your Bug







Submit

Show Ticket Numbers

#	Project Name	Ticket Number
1		
2		
3		

Search



BUG Details


Project Name

Error Category

Error Details

Date

Print Screen



Status

Assign Staff

Project Name

Ticket number

Description

Error details

Error category

Choose Staff Category

Select Staff


	Employee 	Category 	Select Staff 
1			
2			
3			

Date of the bug

Due Date For Reply


Screen Photo 

Submit



BUG TRACKER

- New BUGs
- Projects
- Staff
- Messages
- Log Out



Admin Page

Staff


Category :

	User Name	E-mail ID	Category	
1				
2				
3				
4				

Add Category


Categories

1	database		
---	----------	--	--



BUG TRACKER

- New BUGs
- Projects
- Staff
- Messages
- Log Out



Admin Page

Staff Solution Messages

Ticket number	Project Name	Customer Name	Solution	Approve
1			<input type="button" value="Show"/>	<input type="button" value="Approved"/>
2			<input type="button" value="Show"/>	<input type="button" value="Approved"/>
3			<input type="button" value="Show"/>	<input type="button" value="Approved"/>
4			<input type="button" value="Show"/>	<input type="button" value="Approved"/>

Add BUGs

Flow Bugs

Messages

Log Out

Customers Page

Fill this form

Project Name

Enter Your Project Name

Error Category

Enter Your Error Category

Error Details

Enter Your Error Details

Print Screen From Your Bug

Submit

Show Ticket Numbers

#	Project Name	Ticket Number
---	--------------	---------------

New BUGs

Projects

Staff

Messages

Log Out

Customers Page

Solution Message

Close

Messages

	Owner Name	Solution	Approve
1		Show	Approved
2		Show	Approved
3		Show	Approved
4		Show	Approved