BYTE PRO

An online version control, file management and collaborative tasks management system

BYTE PRO (Version Control System)

Software Requirement Specification

Course: SE 505 Software Project Lab 2

Submitted to

Manager SPL 2

Institute of Information Technology

University of Dhaka

Supervised By

Dr. B M Mainul Hossain

Associate Professor

Institute of Information Technology

University of Dhaka

Submitted by

Chinmoy Acharjee (BSSE 0819)

Sabik Abtahee (BSSE 0829)

Institute of Information Technology

University of Dhaka

20-03-2018

LETTER OF TRANSMITTAL

19th March, 2018

Dr. B M Mainul Hossain

Associate Professor

Institute of Information Technology,

University of Dhaka

Subject: Submission of Software Requirement Specification report of SPL -2

Sir,

With due respect, we are submitting our software requirement specification report of our project 'BYTE PRO' – a version control, project management and collaborative system. We have given our best effort to fulfill the requirements of our stakeholders and prepare a detailed documentation.

Therefore, we hope that you will be kind enough to excuse our errors and oblige thereby.

Yours sincerely,

Chinmoy Acharjee (BSSE 0819)

Sabik Abtahee (BSSE 0829)

BSSE08 batch

Institute of Information Technology,

University of Dhaka

_____

Signature of Supervisor

ACKNOWLEDGEMENT

We are highly indebted for getting such a tremendous opportunity to prepare the report of 'BYTE PRO'. We would like to extend our gratitude to all of the individuals who supported us. We are highly indebted to our supervisor Dr. B M Mainul Hossain, Associate professor, Institute of Information Technology. We would like to thank our SPL-2 coordinators Mohammed Shafiul Alam Khan (Assistant Professor), Amit Seal Ami (Lecturer) and Md. Saeed Siddik (Lecturer) for their guidance and encouragement. Finally, we would like to appreciate and thank our stakeholders who helped us gather our requirements of this project.

ABSTRACT

This document contains all the requirements and specifications of our project 'BYTE PRO' an online version control, project management, collaborative system. The document incorporates inception, elicitation, scenario-based model, data-based model, class-based model and behavioral model. We defined our requirements based on our stakeholders' point of view.

BYTE PRO is an online version control system. It will let users work collaboratively on projects and manage versions of their project files.

This document will be helpful to guide us during implementation.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

This chapter is intended to specify the purpose if this document and the intended audiences of it.

1.1 Purpose

This document briefly describes the Software Requirement Analysis of 'Byte Pro' an online project management, version control and collaborative tasks managements system. It contains functional, non-functional and supporting requirements and establishes a requirements baseline of the development of the system. The requirements contained in the SRS are independent, uniquely numbered and organized by topic. The SRS serves as an official means of communicating user requirements to the developer and provides a common reference point for both the developer team and the stakeholder community.

1.2 Intended Audience

This SRS is intended for several audiences including the project managers, designers, developers, users.

- The user will use this SRS to verify that the developer team has created a product that is acceptable to the user.
- The project managers of the developer team will use this SRS to plan milestones and a delivery date and ensure that the developing team is on track during development of the system.
- The designers will use this SRS as a basis for creating the system's design. The designers will continually refer back to this SRS to ensure that the system they are designing will fulfil the users' need.
- The developers will use this SRS as a basis for developing the system's functionality. The developers will link the requirements defined in this

SRS to the software they create to ensure that they have created software that will fulfill all of the users' documented requirements.

1.3 Conclusion

This analysis of the audience helped us to centralize our attention in people who will be using our analysis. This document will help each and every one who is related to the project to have a better view about the project.

CHAPTER 2: INCEPTION

2.1 Introduction

Inception is the beginning phase of requirements engineering. It defines how a software project gets started and what the scope and nature of the problem is to be solved. The goal of the inception phase is to identify concurrent needs and conflicting requirements among the stakeholders of a software project. At project inception, we establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired and the effectiveness of preliminary communication and collaborations between other stakeholders and the software team. The purpose of the document is to represent a short description of 'BYTE PRO' – a version control system and project management. To establish the groundwork, we have worked with the following factors related to the inception phases:

- List of stakeholders
- Recognizing multiple viewpoints
- Working towards collaboration
- Requirements questionnaire

2.1.1 List of stakeholders

A stakeholder is any group or individual who can affect or get affected by the system. Stakeholders are end users who will interact with the software or anyone who will have business value of the software. To identify the stakeholders, we consulted with some project managers and software developers and asked them the following questions:

- Who will use the BYTE PRO software's outcome?
- Who can get benefited by using the software?
- Who has the resources and ideas to make the software better?
- What type of problems do they face during version control or project management?

We identified some of the following stakeholders through these questions:

1. **Project manager**: A project manager is a professional in the field of project management. Project managers have the responsibility of planning, procurement and execution of a project.
2. **Software developer**: Developers are the main stakeholder. Because they have the skills and resources. They work on big projects of millions of lines of code. In the industry they work with many other developers, project manager and tester. So, they need to keep updated with other developers and their activities.
3. **Tester:** Tester are the employer those who find bugs and issues within a project before it gets deployed to clients or customers.
4. **Others:** In this category all the users who will use the system as online backup storage of code and personal projects.

2.2 Recognizing Multiple Viewpoints:

Different stakeholders achieve different benefits from the system. Consequently, each of them has a different view of the system. So we have to recognize the requirements from multiple points of view, as well as multiple views of requirements. Assumptions are given below:

- Project manager:
    - Fast and efficient system
    - Good communication
    - Project analytics
- Software Developer:
    - Keep track of versions of file
    - Error free system
    - Collaborative working environment
    - Upload project or file
    - Download project or file
    - Organized file management system
    - Code referencing during issue creation
    - View other projects and progress
    - Strong authentication
    - User friendly
- Tester:
    - Can create issues facility
    - Can post bugs and developers can see those bugs
    - Strong authentication
- Others
    - Online storage facility
    - See other projects and files
    - Can ask question
    - User friendly
    - Good graphical representation
    - Can seek help
    - Upload or download files

2.3 Working Towards Collaborations:

Every stakeholder has their own thoughts which result into these requirements. I needed to merge all of the requirements together.

- By identifying the common and conflicting requirements
- By categorizing requirements
- By prioritizing points for each requirement from stakeholders
- By making final decision about the requirements

CHAPTER 3: ELICITATION

After discussing on the inception phase, we need to focus on Elicitation phase. So, this chapter specifies the Elicitation phase.

3.1 Introduction:

Requirements Elicitation is a part of requirements engineering that is the practice of gathering requirements from the users, customers and other stakeholders. We have faced many difficulties, like understanding the problems, making questions for the stakeholders, problems of scope and volatility. Though it is not easy to gather requirements within a very short time, we have surpassed these problems in an organized and systematic manner.

3.2 Elicitation Requirements:

We have seen Question and Answer (Q&A) approach in the previous chapter, where the inception phase of requirement engineering has been described. Requirements Elicitation (also called requirements gathering) combines problem solving, elaboration, negotiation and specification. The collaborative working approach of the stakeholders is required to elicit the requirements. We have finished the following tasks for eliciting requirements:

- Collaborative Requirements Gathering
- Quality Function Deployment
- Usage scenarios
- Elicitation work products

Approaches:

1. Visualization. Using tools that promote better understanding of the desired end-product such as visualization and simulation.

2. Consistent language. Using simple, consistent definitions for requirements described in natural language and use the business terminology that is prevalent in the enterprise.

3. Guidelines. Following organizational guidelines that describe the collection techniques and the types of requirements to be collected. These guidelines are then used consistently across projects.

4. Consistent use of templates. Producing a consistent set of models and templates to document the requirements.

5. Documenting dependencies. Documenting dependencies and interrelationships among requirements.

6. Analysis of changes. Performing root cause analysis of changes to requirements and making corrective actions.

3.3 Quality Function Development:

Quality Function Deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. It concentrates on maximizing customer satisfaction from the software engineering process. So we have followed this methodology to identify the requirements for the project. The requirements, which are given below, are identified successfully by the QFD.

3.3.1 Normal Requirements:

Normal are generally the objectives and goals that are stated for a product or system during meetings with the stakeholders. The presence of these requirements fulfils stakeholders' satisfaction. The normal requirements of our project:

1. Upload files or projects
2. Manage versions
3. Issue report
4. Web based system
5. Version release
6. Create repository
7. Code referencing
8. Manage project planning
9. Collaborative works

3.3.2 Expected Requirements:

1. Strong authentication
2. Account recovery
3. Download repository or file
4. Search project
5. View project
6. Profile modification
7. Create public or private repository
8. User friendly interface
9. Fast performance

3.3.3 Exciting Requirements:

1. Q/A option where anyone can ask question or seek help
2. Project analytics

3.4 Usage Scenario

Byte pro is an online based version control system. It is divided into five sub systems.

3.4.1 Authentication

Users need to be authenticated before they access into the system.

Sign Up

User has to register into the system by providing some credentials. She/he has to provide her name, username, email and password. The system will check if the given credentials are valid. Furthermore, it will verify if the username and email are unique. If not, the user has to provide another username or email. If every credential is validated and verified then her account will be registered in the system. Her information will be inserted in the database. A confirmation email will be sent to her.

Sign In

Any registered user can sign-in into the system by providing her username or email and her password. The system will validate and verify the given information. If the information is valid and verified, the sign-in phase will be completed. Otherwise it will not let her access the system and ask her to try again.

Account Recovery

If any user forgets her username or password, she can recover her account by providing her email. The email has to be the same email as she used during signing up. The system will send an email containing the user's username and password. User can use that username and password to sign in into the system.

Sign Out

During signing out the system will save all her unsaved file and sub systems. Then the system will let the user to sign out from the system.

3.4.2 Project Management

Everything related to projects or files will be handled in project management.

Project Planning

User can create project and she will have a repository of that project. To create a project, she needs to provide project name, project description, project tags and other information. She can make the project private or public. She can assign other users as developers to her project. The assigned developers will get notified.

Any user can create an issue description about any public project. If the issue is created from project progress, the corresponding codes will have a reference of that issue.

File Management

In a project a user can upload files or folders. To upload file or folder she needs to provide a description (version description) of that upload. She can mark the progress of the uploaded file or folder. The system will store all the files or project information (file-name, file-size, file-type, modification-time, file-uploader, project-access-type, project-tags). The system will control the versions of that file or folder.

User can download file or folder from a particular project.

User can release a certain project to a particular version. That release version will be stored in the database. If the released project is public, any user can access and download that particular release. Users who have the same skill tag of the project will get notified about the project release.

User can rebase a particular file to the current repository. Rebase means she can replace a particular file version to any of its other version.

User can delete any project or file or folder. All information connected with that file or folder or project will be deleted.

Manage Version

The system will manage all versions of a particular file. If any existing file is reuploaded, the system will check for difference of that file and the new file. It will store both files but keep different versions. If any existing file is uploaded by different users then the system will detect any confliction between the files. The recent user can choose which file to upload between the two. She can also manually update the file and upload it as she wishes. In the case where a user uploads a folder which exists in the project already then, all the files in that folder will be checked individually for version control.

3.4.3 Profile Management

User can edit her profile information. She can change her profile picture, name, skill tags, biography. If she wants to change her username, the system will validate and verify the new username. After username verification the system will store the new username. If she wants to change her password, she has to provide her old password. After password verification the system will prompt for new password. The system will update her information into the database.

3.4.4 Search

User can search files or folders that is in her own project repositories. If the file or folder exists, it will show all the information related to it. Any user can search other users by providing their name. She can also search all public projects by providing project's name.

3.4.5 Communication

A user who gets assigned by another user to her project will get notification about that project. The notification will have content (project description) and a link to access the project. The assigned user will be the developer of the project and the user who assigned her to the project will be the master user. If any project is released, all users having some of the skill tags of the project will be notified about the project. Any user can comment her thoughts on any public project or file. The owner or assigned developers can see who commented and the comment description.

CHAPTER 4: SCENARIO BASED MODEL

This chapter briefly describes the scenario-based modeling of BYTE PRO.

4.1 Introduction

To characterize requirements and build meaningful analysis and design models, we need to understand how user ends interact with the system. Hence, requirements modeling begins with the creation of scenarios in the form of Use Cases, activity diagrams and swim lane diagrams.

4.2 Definition of use case

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

Primary Actor:

The primary actor of a use case is the stakeholder that calls on the system to deliver one of its services. It has a goal with respect to the system – one that can be satisfied by its operation. The primary actor is often, but not always, the actor who triggers the use case**.**

Secondary Actor:

Actors that the system needs assistance from to achieve the primary actor's goal. Secondary actors may or may not have goals that they expect to be satisfied by the use case, the primary actor always has a goal, and the use case exists to satisfy the primary actor.

4.3 Use case diagrams

4.3.1 Level 0 Use case diagram

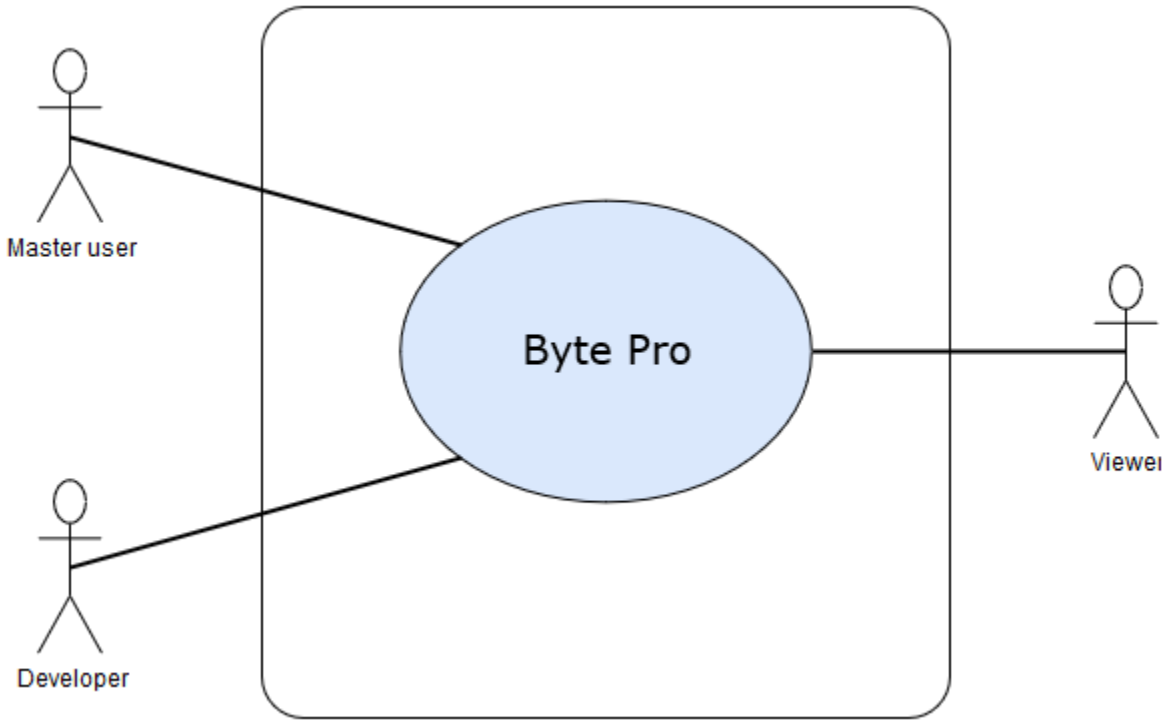


Figure 1: Level 0 Use case diagram of BYTE PRO

4.3.2 Level 1 Use Case diagram

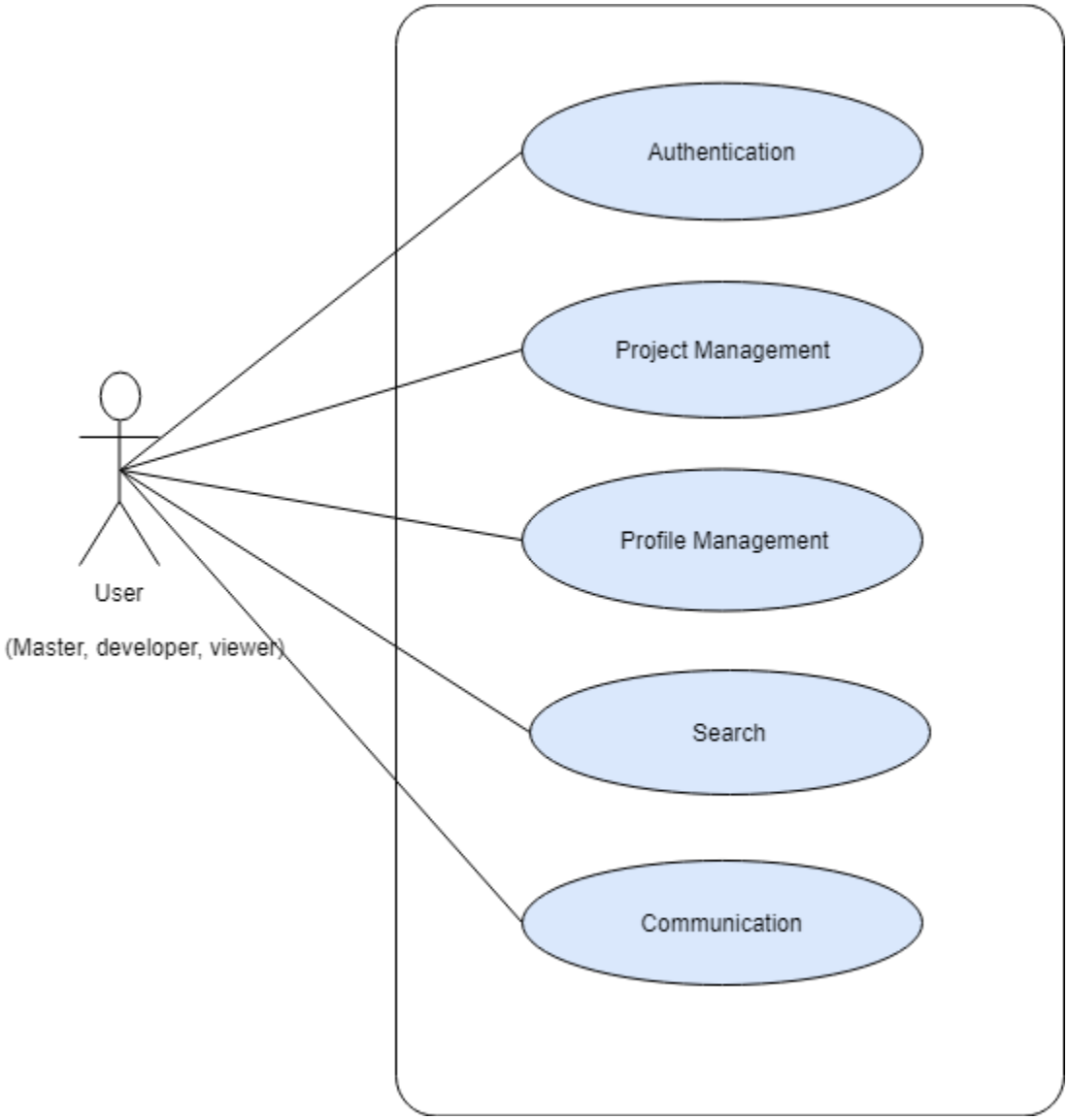


Figure 2: Level 1 Use case diagram

4.3.3 Level 1.1 use case diagram

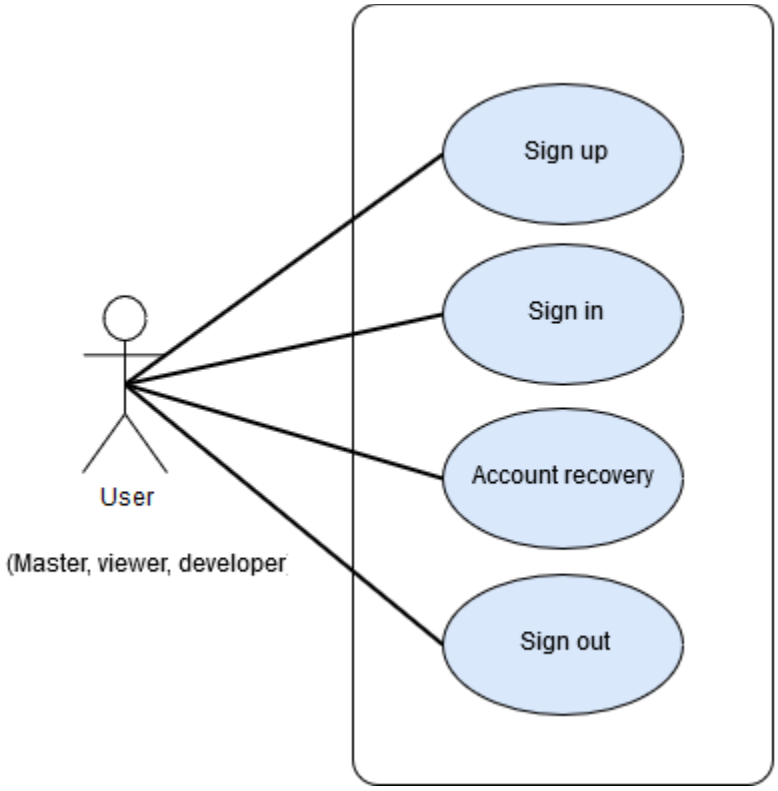


Figure 3: Level 1.1 use case diagram

Action: User sign up with user information.

Reply: System registers an account of that user in the system.

Action: User provides email and password to login

Reply: System verifies and signs in the user

Action: User provides email to recover account

Reply: System sends an email containing username and password

Action: User signs out the system

Reply: System saves unsaved files and logs out the user

4.3.4 Level 1.2 use case diagram

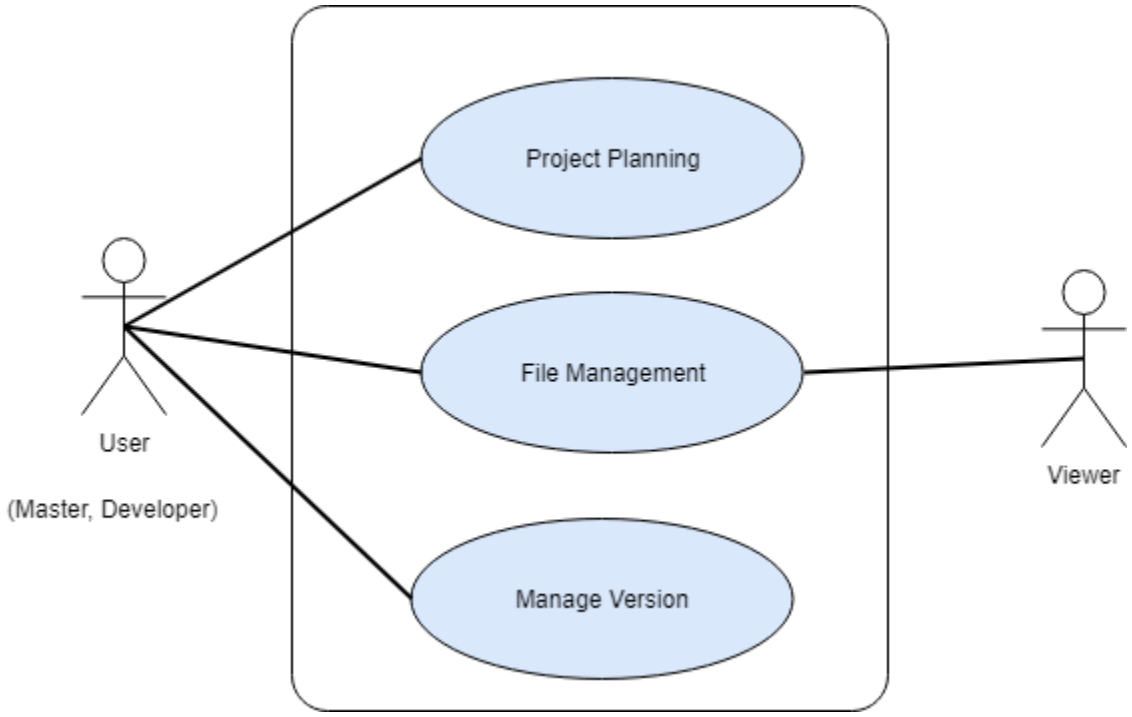


Figure 4: Level 1.2 use case diagram

Action: Master user plans project details

Reply: System stores all project plans

Action: Users interacts, upload, download, rebase files

Reply: System stores and manages the file

Action: User uploads same file

Reply: System manages versions of that file

4.3.5 Level 1.4 use case diagram

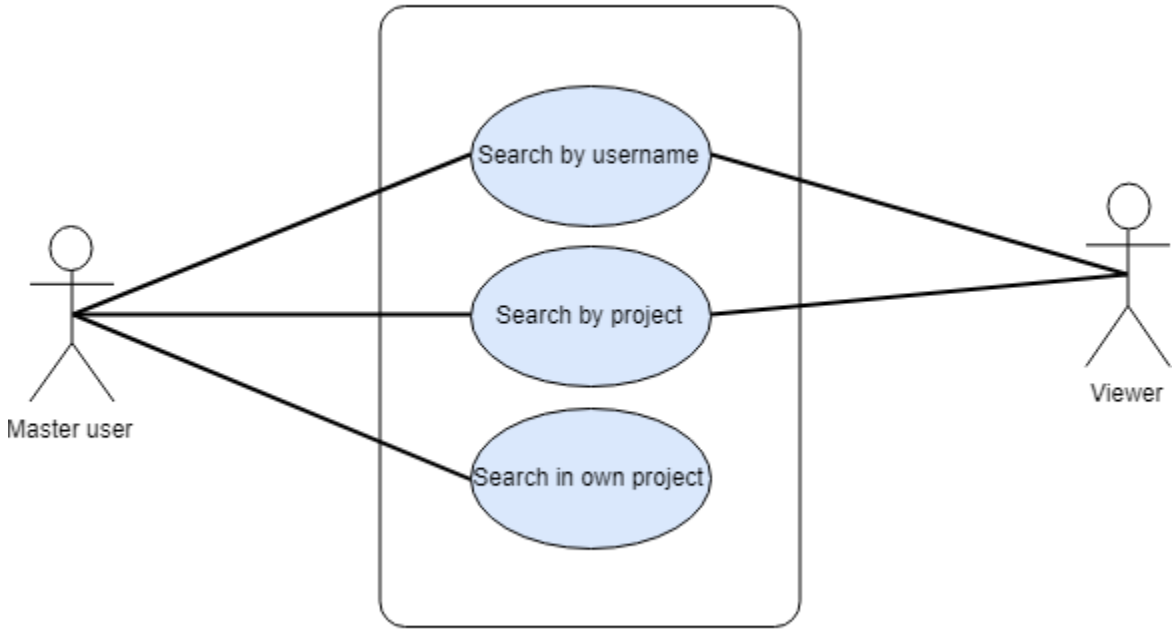


Figure 5: Level 1.4 use case diagram

Action: User searches by providing username

Reply: System shows information if user exists

Action: User searches with a project name

Reply: System shows related projects if exists

Action: User searches file in his repository

Reply: System shows files if exists

4.3.6 Level 1.5 use case diagram

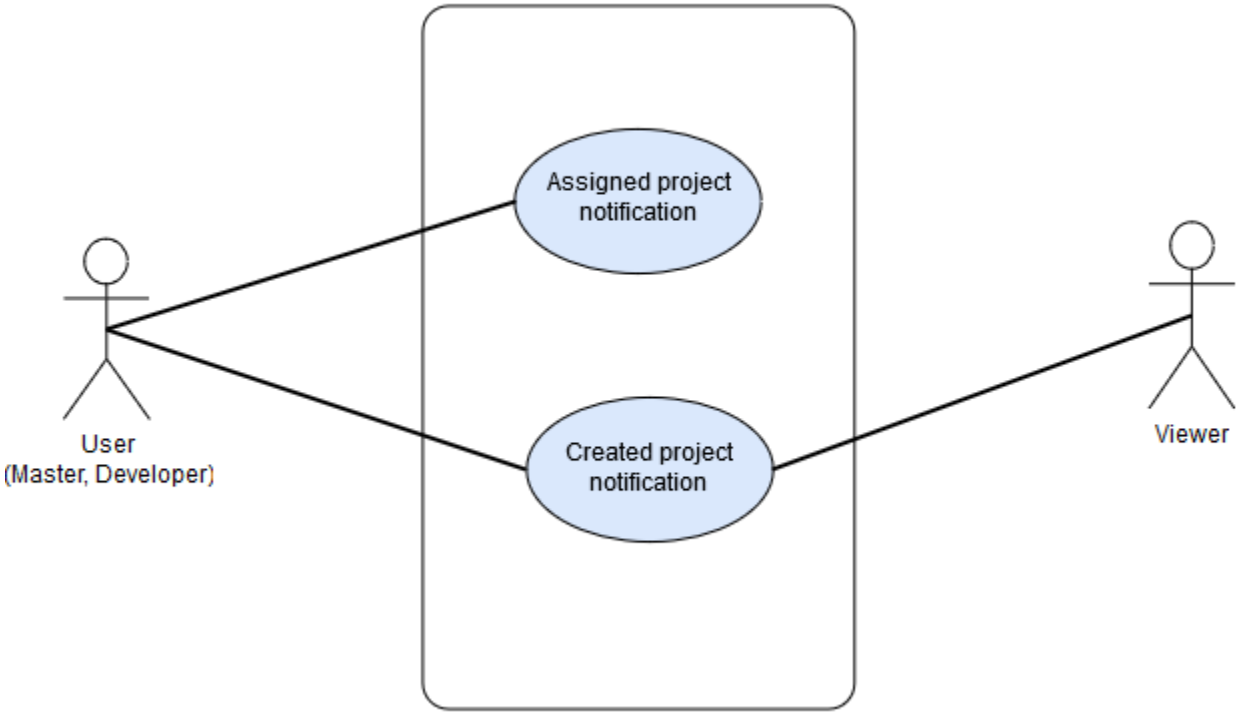


Figure 6: Level 1.5 use case diagram

Action: Master assigns a developer to his project

Reply: Assigned developer receives notification

Action: User releases a project

Reply: Users having some of the skill tags of that project gets notified

4.4 Activity Diagrams

Level 1.1.1 Authentication (sign up)

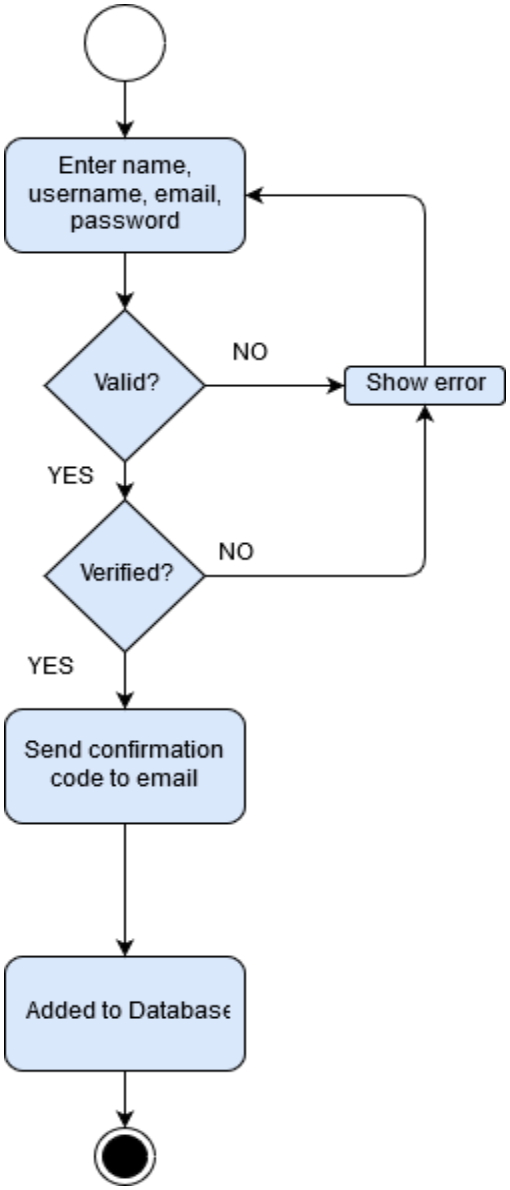


Figure 7: Level 1.1.1 activity diagram authentication (sign up)

Level 1.1.2 Authentication (Sign in)

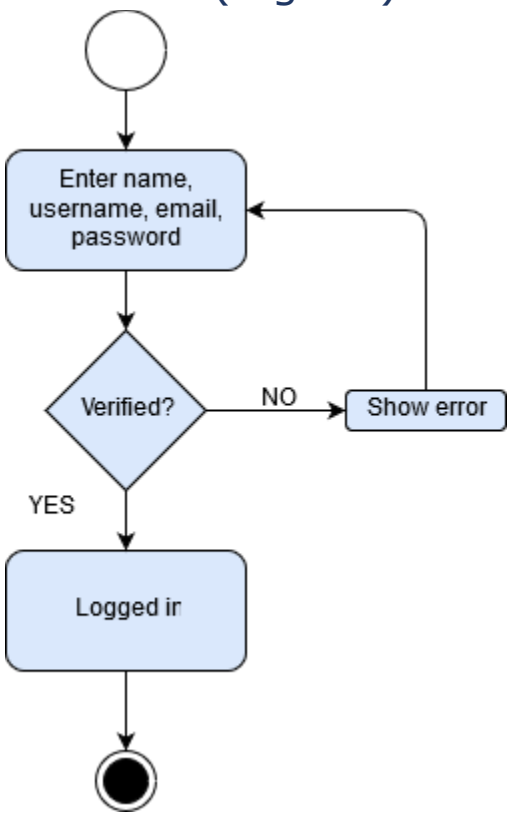


Figure 8: Level 1.1.2 activity diagram authentication (sign in)

Level 1.1.3 Authentication (Recovery)

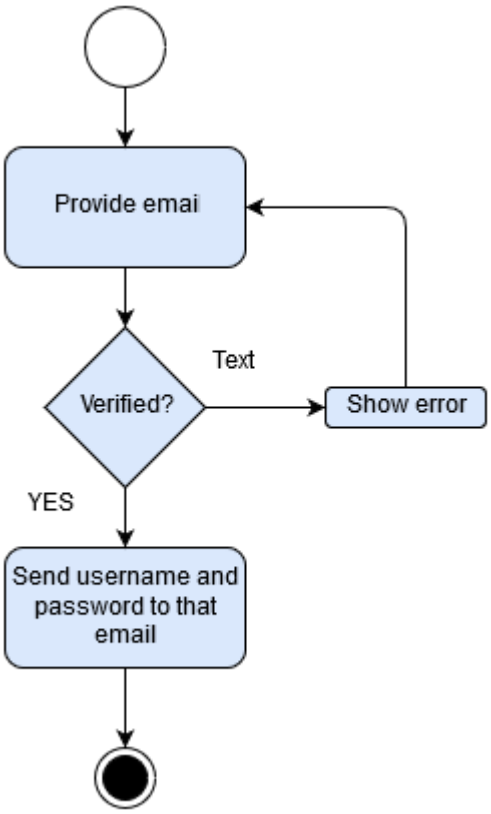


Figure 9: Level 1.1.3 activity diagram authentication (recovery)

Level 1.1.4 Authentication (sign out)

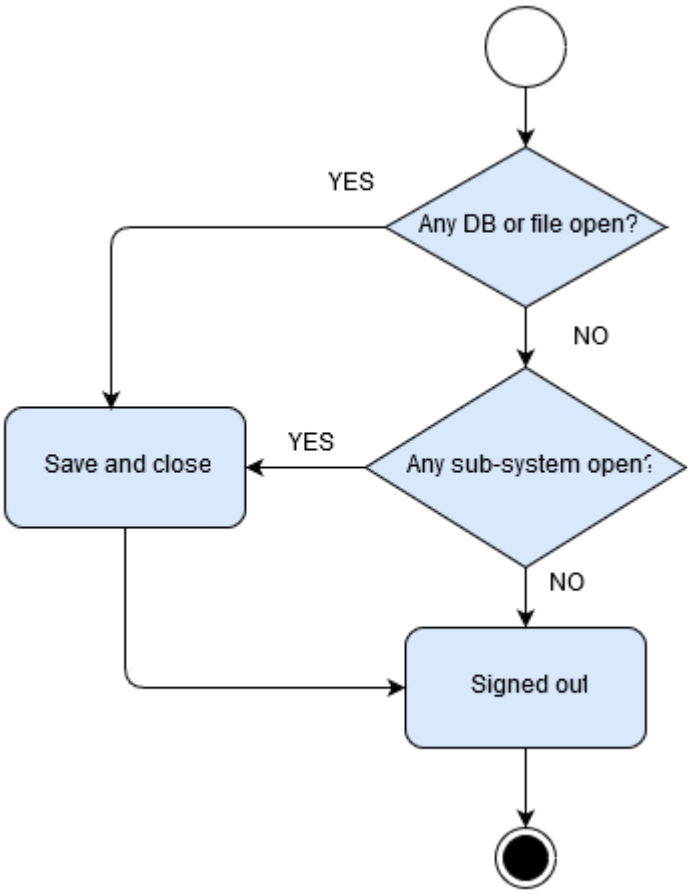


Figure 10: Level 1.1.4 activity diagram authentication (sign out)

Level 1.2.1 Project Management (project planning)

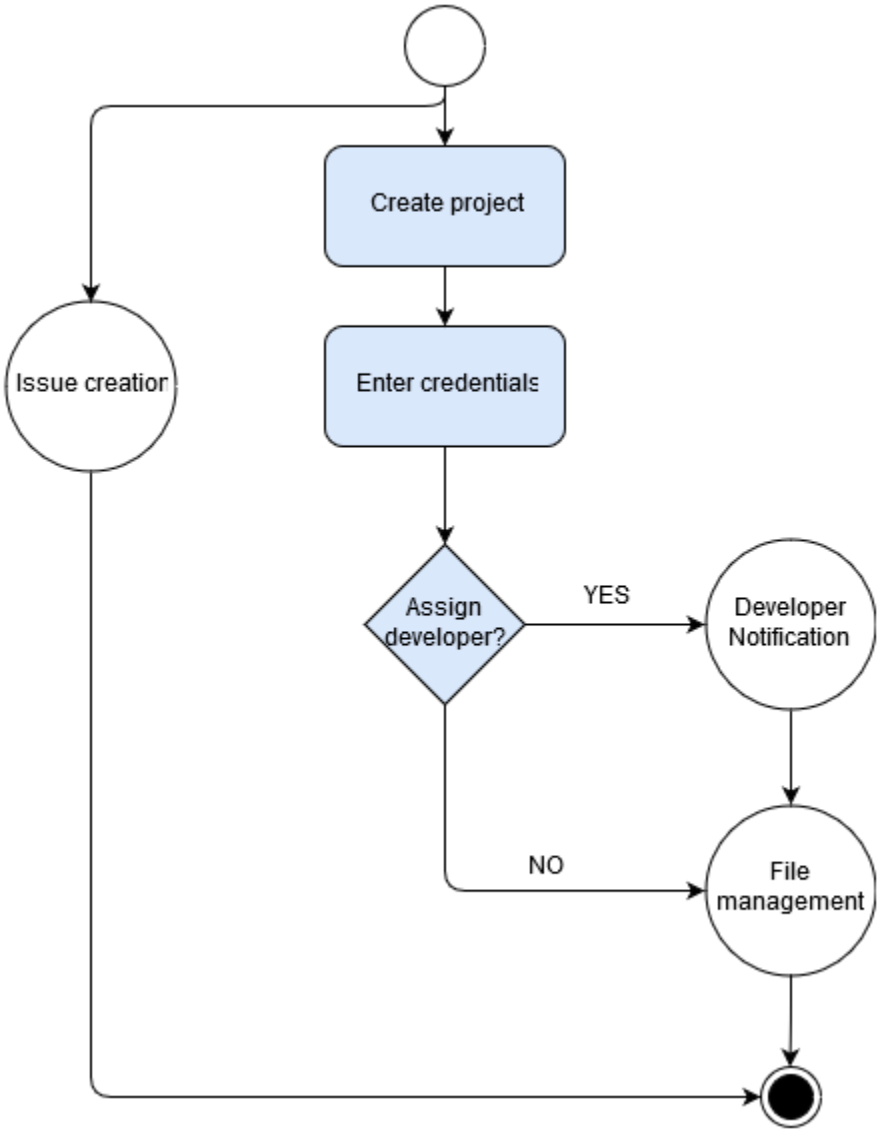


Figure 11: Level 1.2.1 Project Management (project planning)

*Issue creation – Level 1.2.1.1

*Developer Notification- Level 1.5.1

*File management – Level 1.2.2

Level 1.2.1.1 Project Management (Issue Creation)

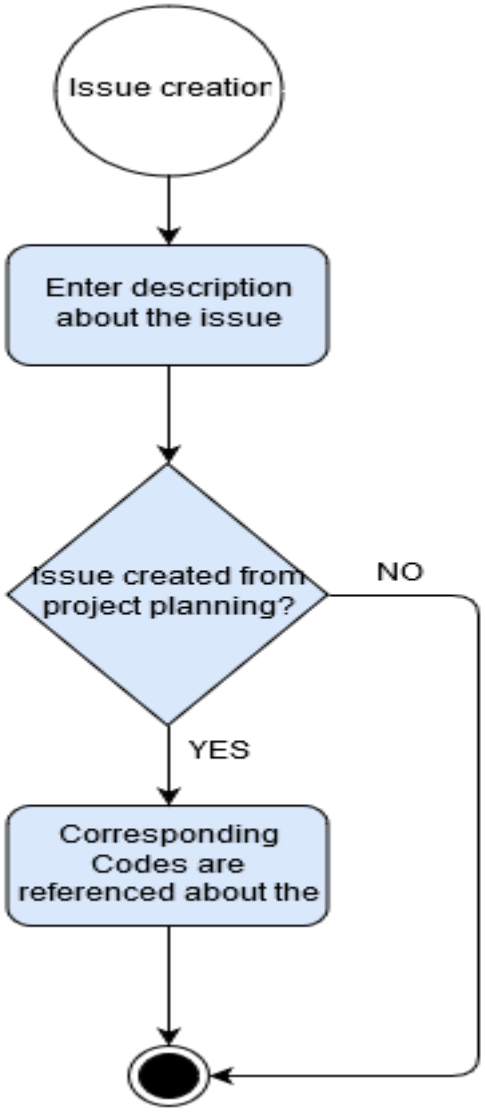


Figure 12: Level 1.2.1.1 Project Management (Issue creation)

Level 1.2.2 Project Management (file management)

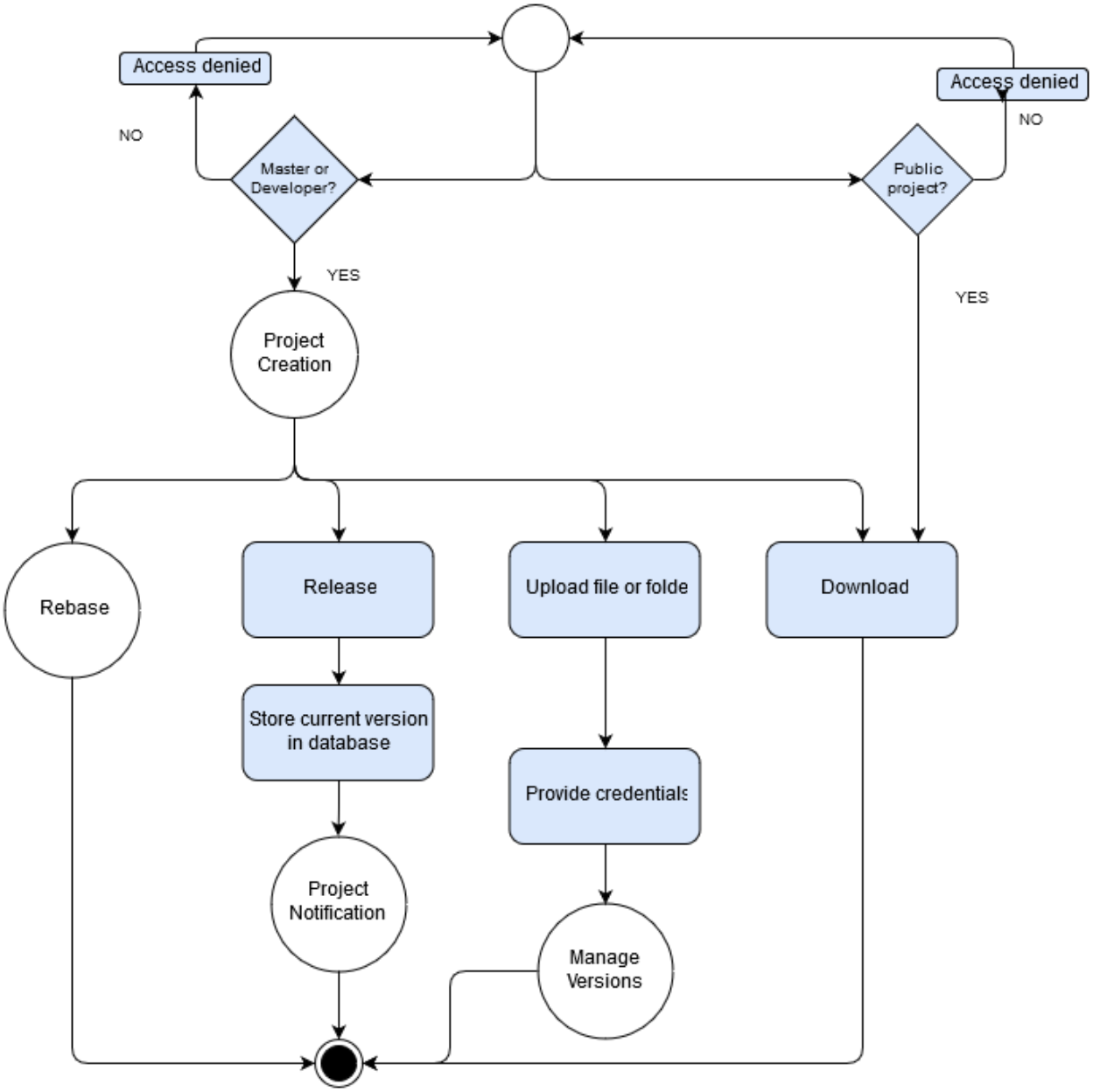


Figure 13: Level 1.2.2 Project Management (file management)

* Project creation – Level 1.2.1

* Rebase – Level 1.2.2.1

* Project Notification - Level 1.5.2

* Manage versions – Level 1.2.3

Level 1.2.2.1 Project Management (rebase)

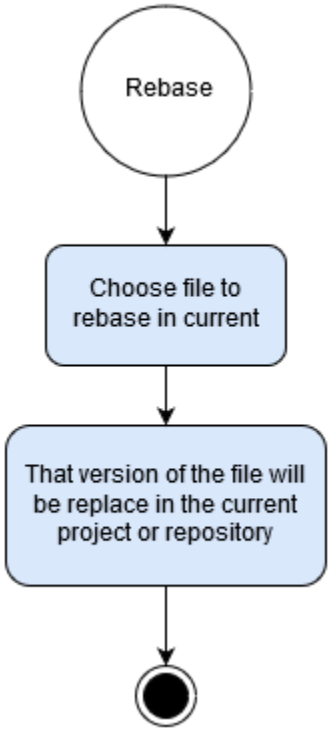


Figure 14: Level 1.2.2.1 Project Management (rebase)

Level 1.2.3 Project Management (manage version)

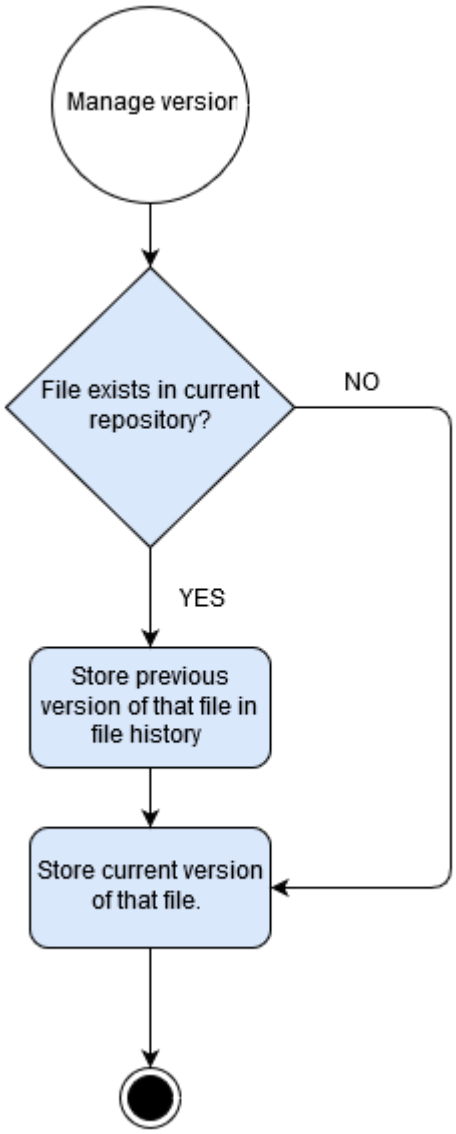


Figure 15: Level 1.2.3 Project Management (manage version)

Level 1.3 Profile Management

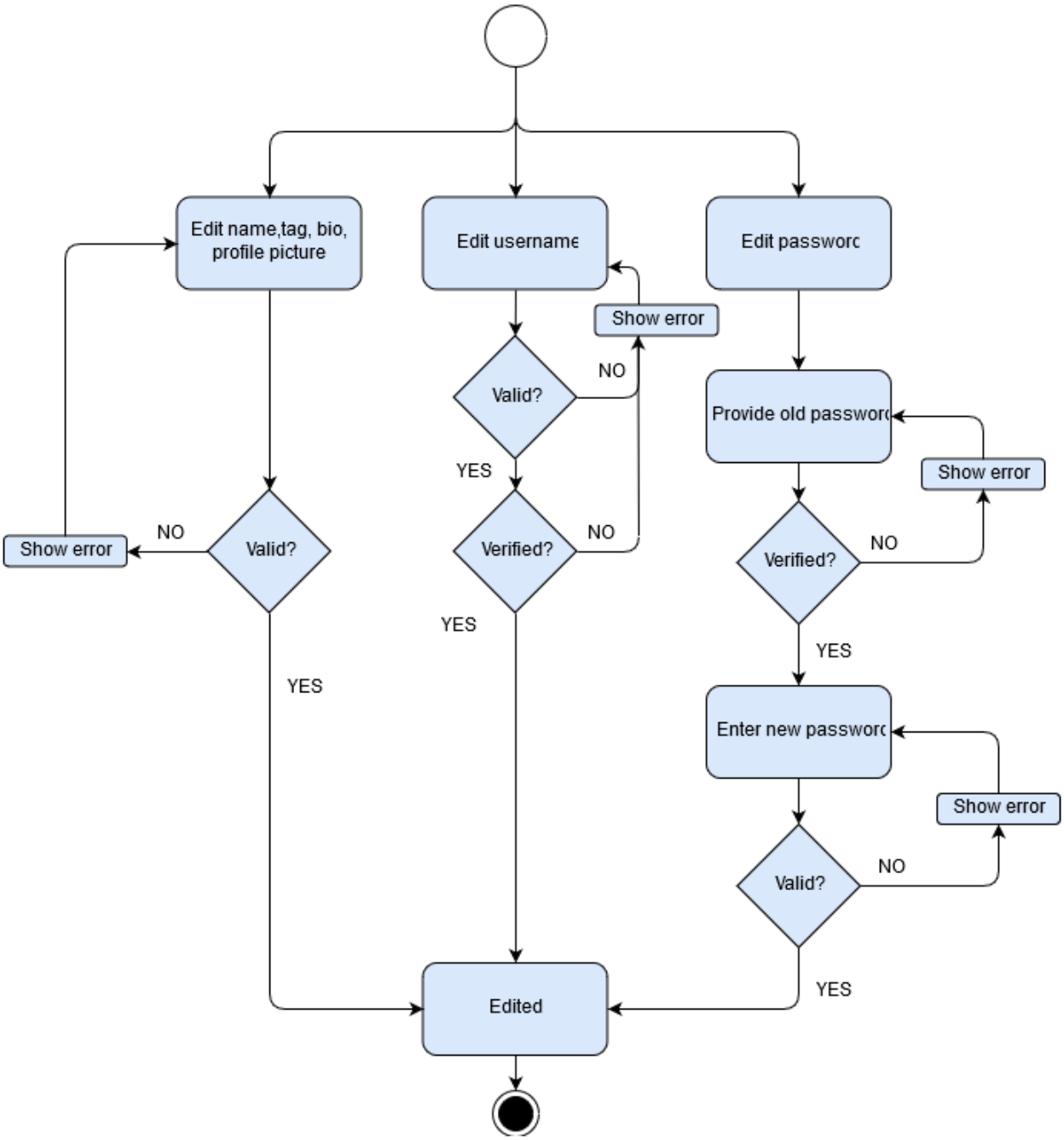


Figure 16: Level 1.3 Profile Management

Level 1.4 Search

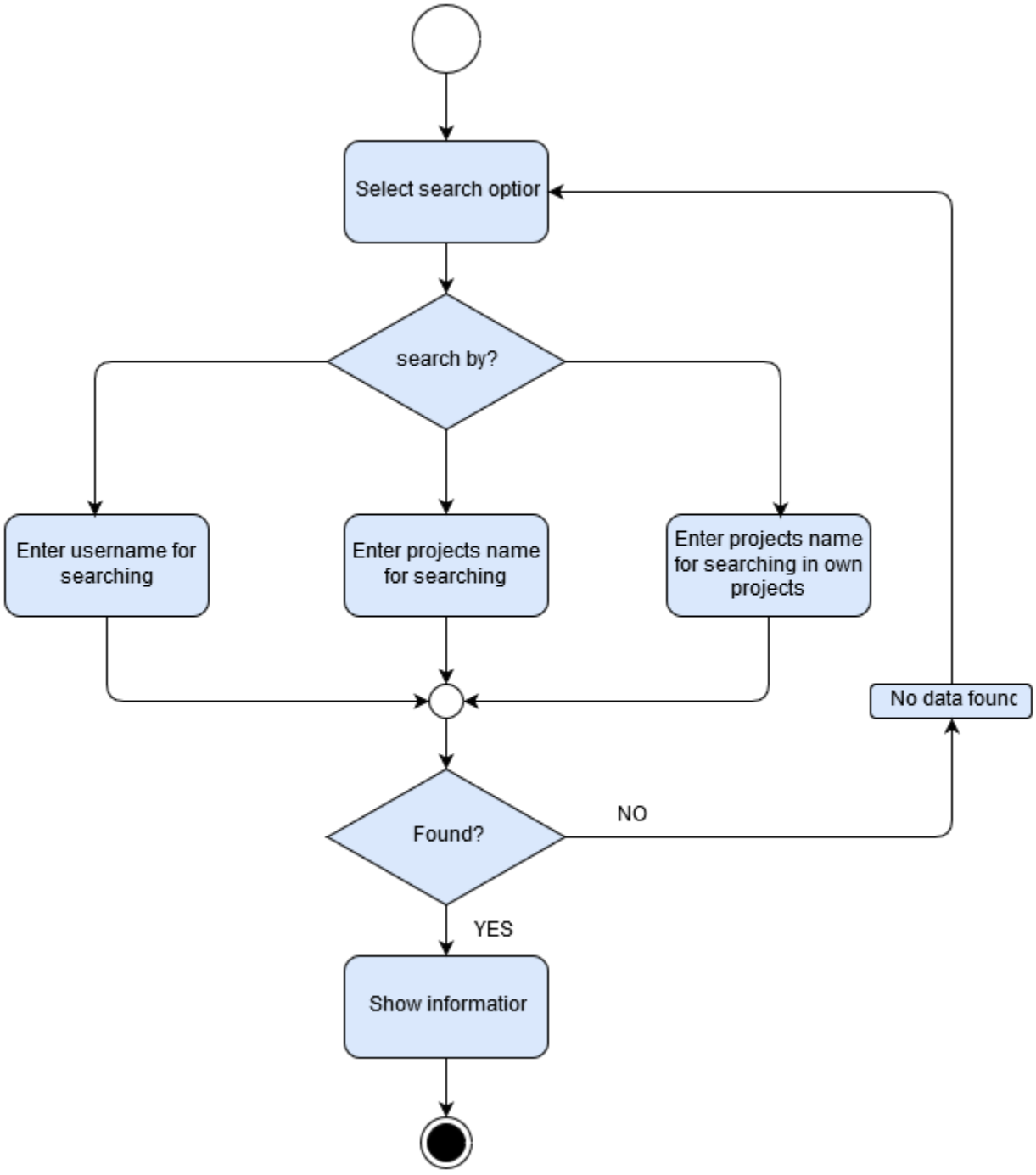


Figure 17: Level 1.4 Search

Level 1.5.1 Communication (Developer notification)

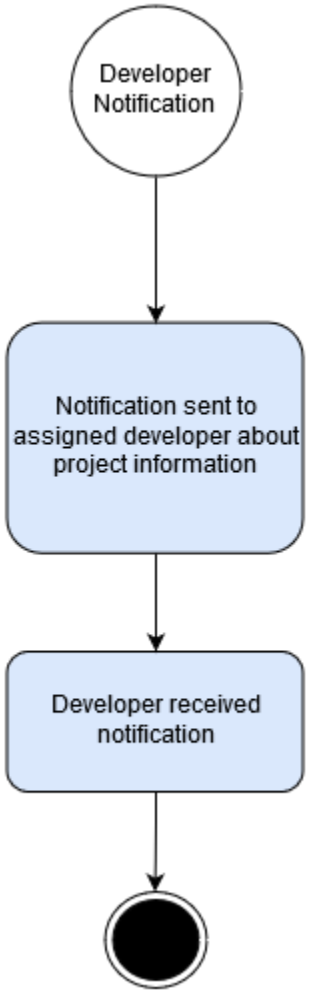


Figure 18: Level 1.5.1 Communication (Developer notification)

Level 1.5.2 Communication (Project notification)

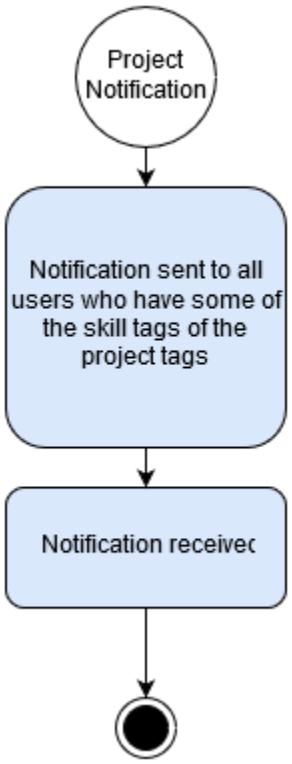


Figure 19: Level 1.5.2 Communication (Project notification)

4.5 Swimlane Diagrams

Level 1.1.1 Authentication (sign up)

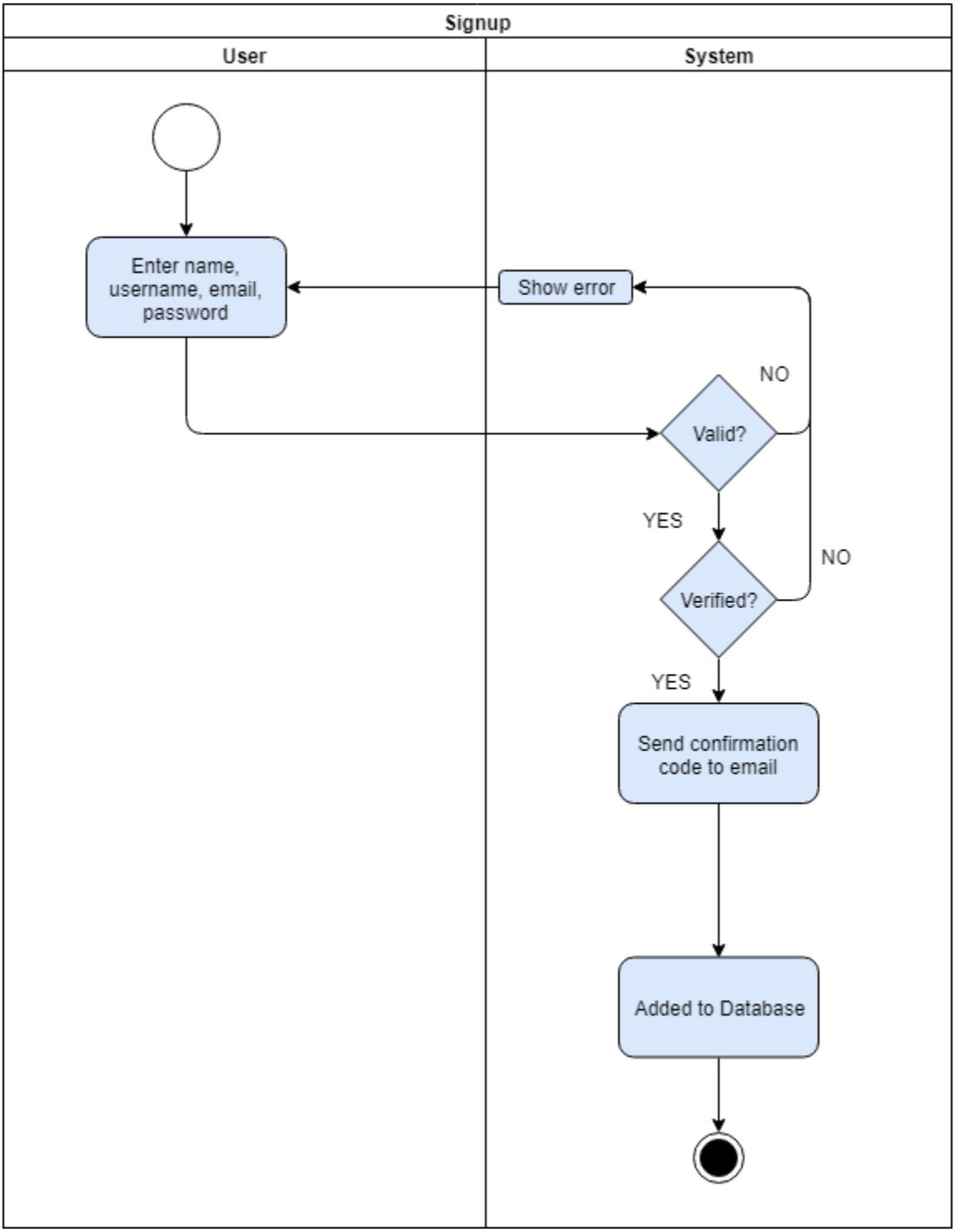


Figure 20: Level 1.1.1 swimlane diagram authentication (sign up)

Level 1.1.2 Authentication (Sign in)

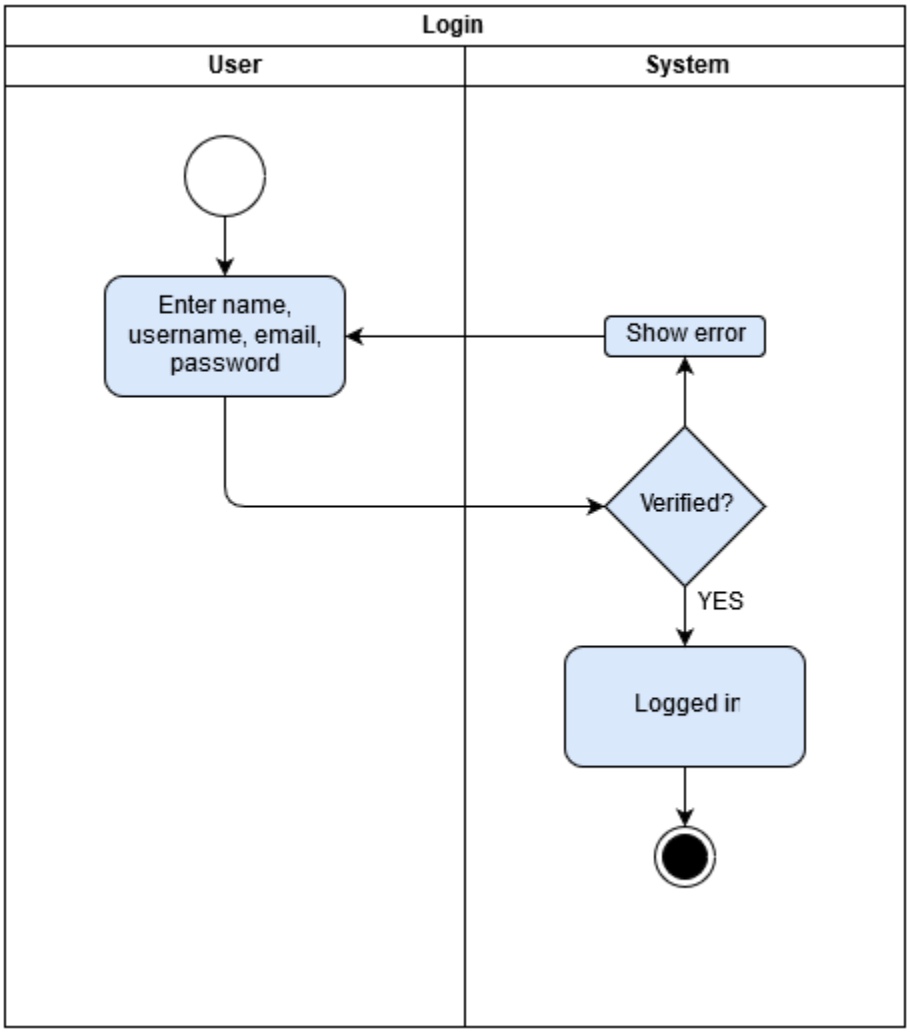


Figure 21: Level 1.1.2 swimlane diagram authentication (sign in)

Level 1.1.3 Authentication (Recovery)

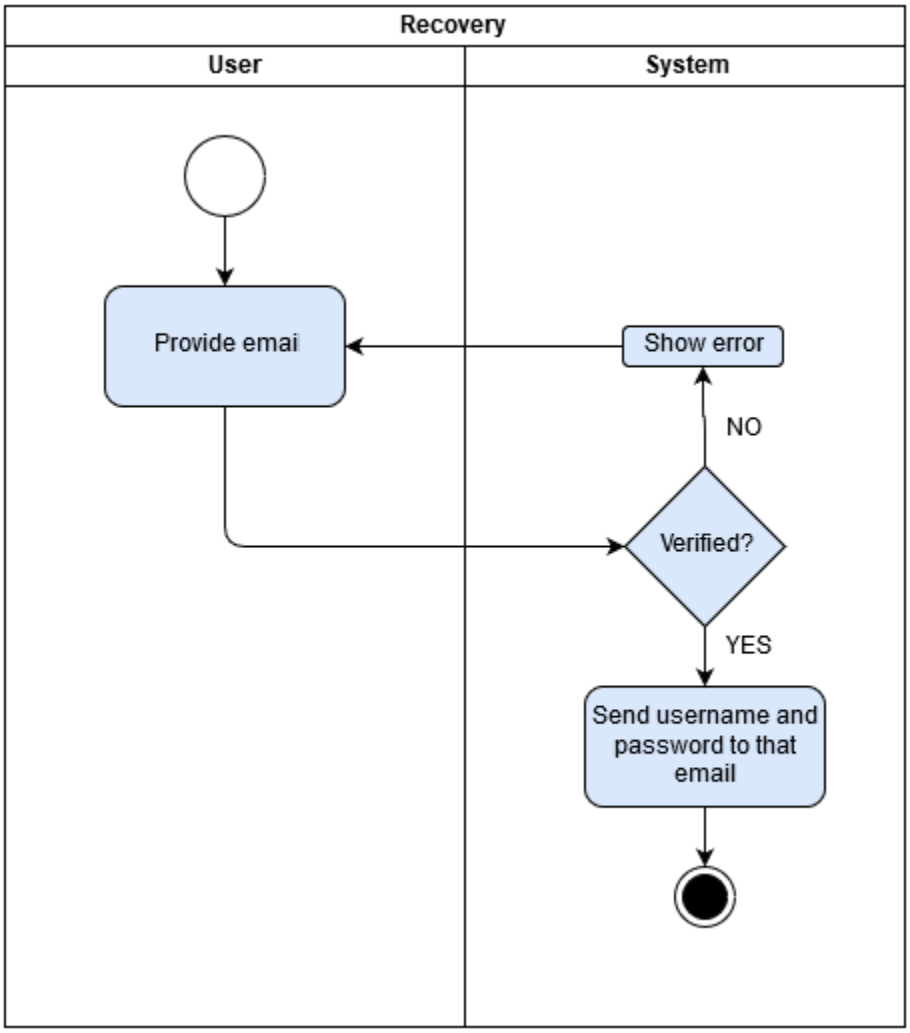


Figure 22: Level 1.1.3 swimlane diagram authentication (recovery)

Level 1.1.4 Authentication (sign out)

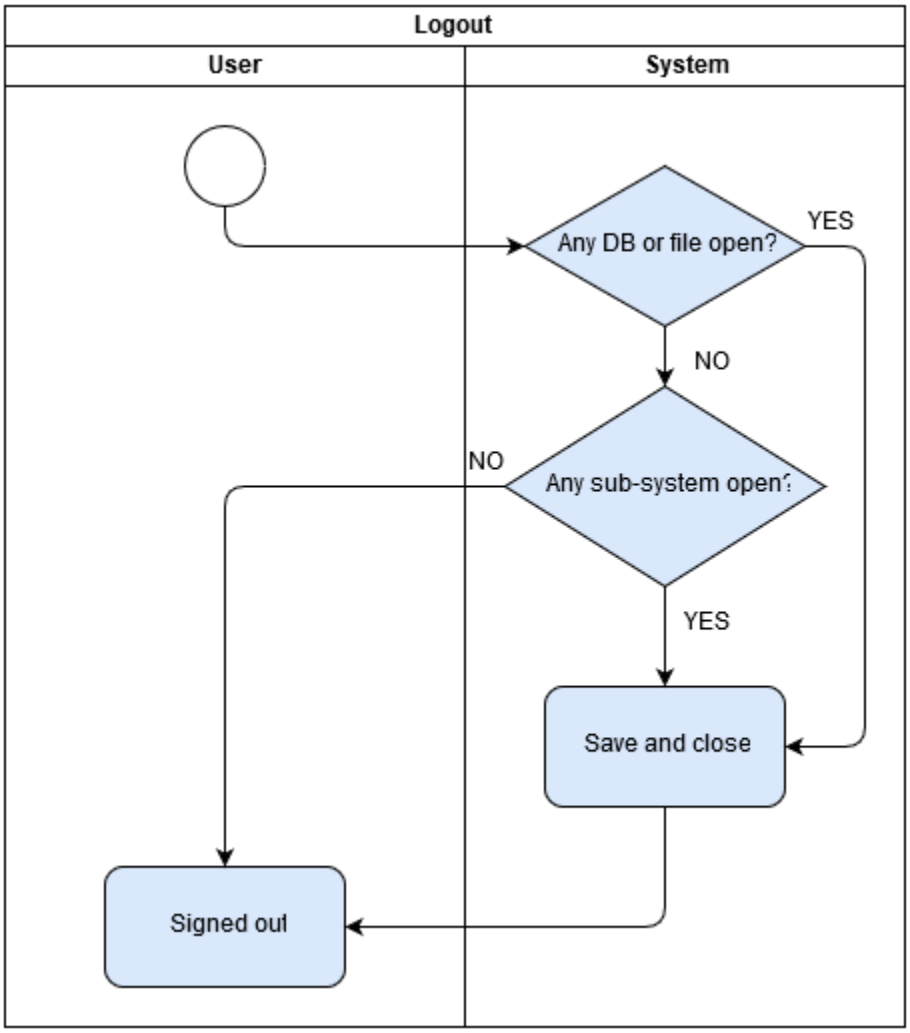


Figure 23: Level 1.1.4 swimlane diagram authentication (sign out)

Level 1.2.1 Project Management (project planning)

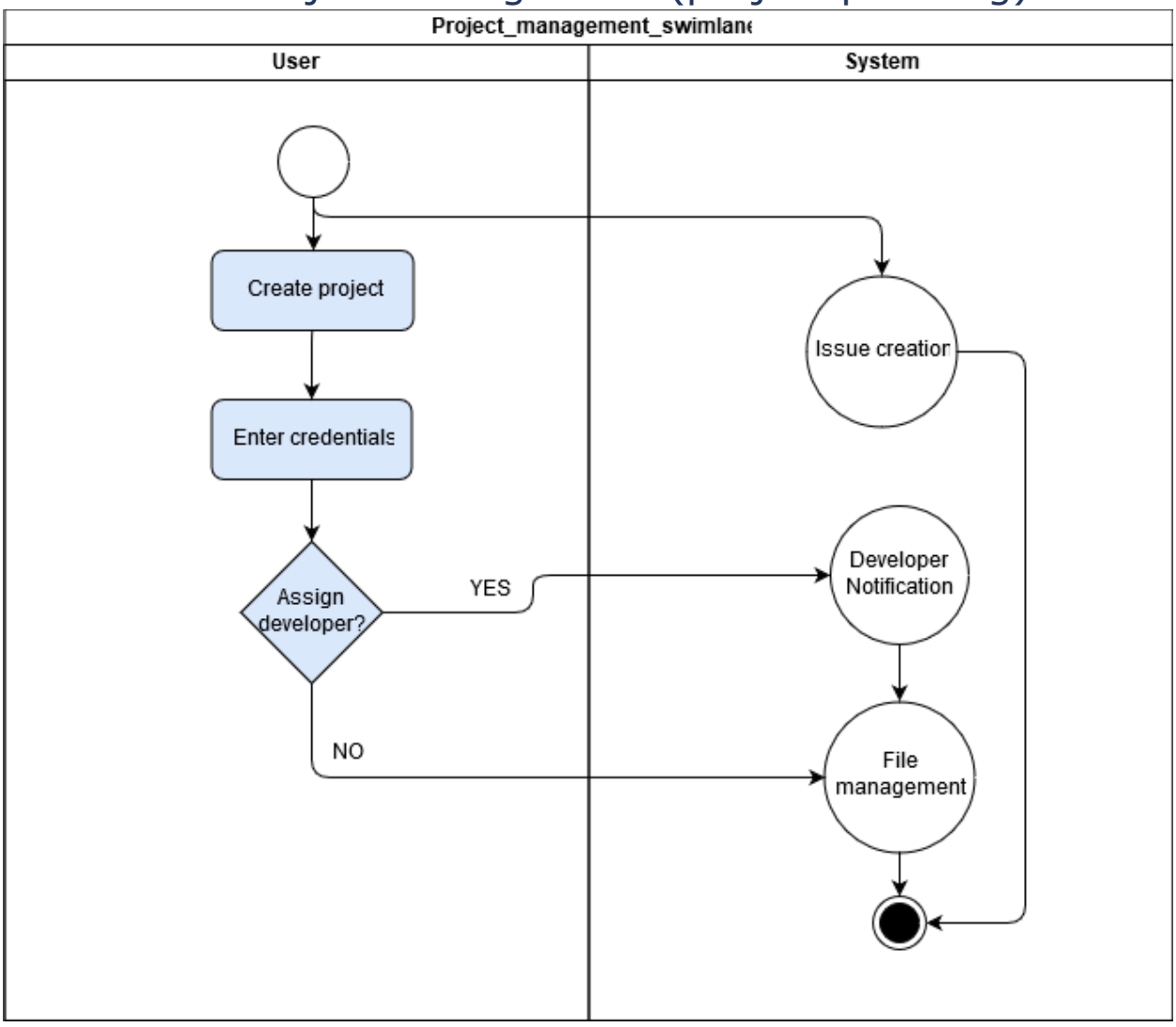


Figure 24: Level 1.2.1 swimlane diagram Project Management (project planning)

*Issue creation – Level 1.2.1.1

*Developer Notification- Level 1.5.1

*File management – Level 1.2.2

Level 1.2.1.1 Project Management (Issue Creation)

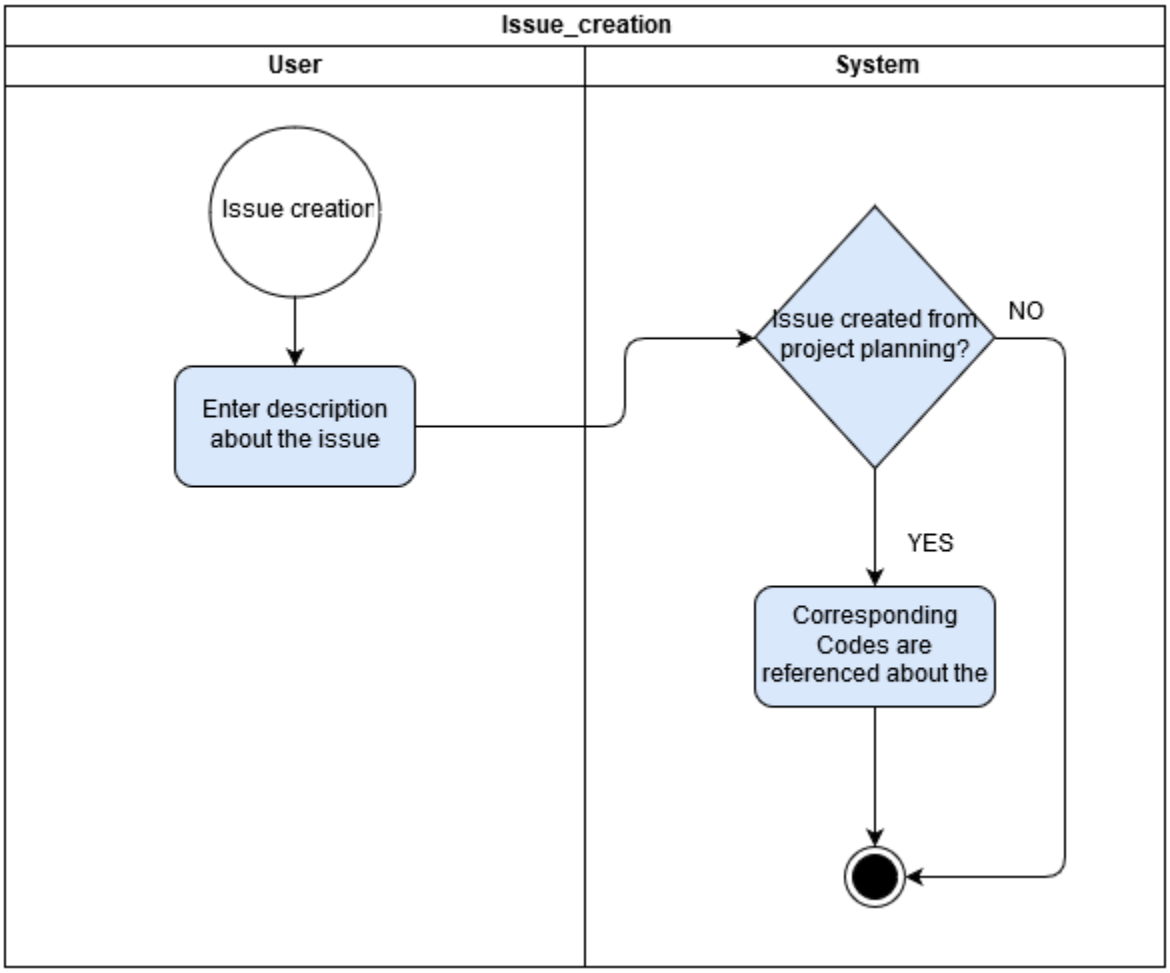


Figure 25: Level 1.2.1.1 swimlane diagram Project Management (Issue creation)

Level 1.2.2 Project Management (file management)

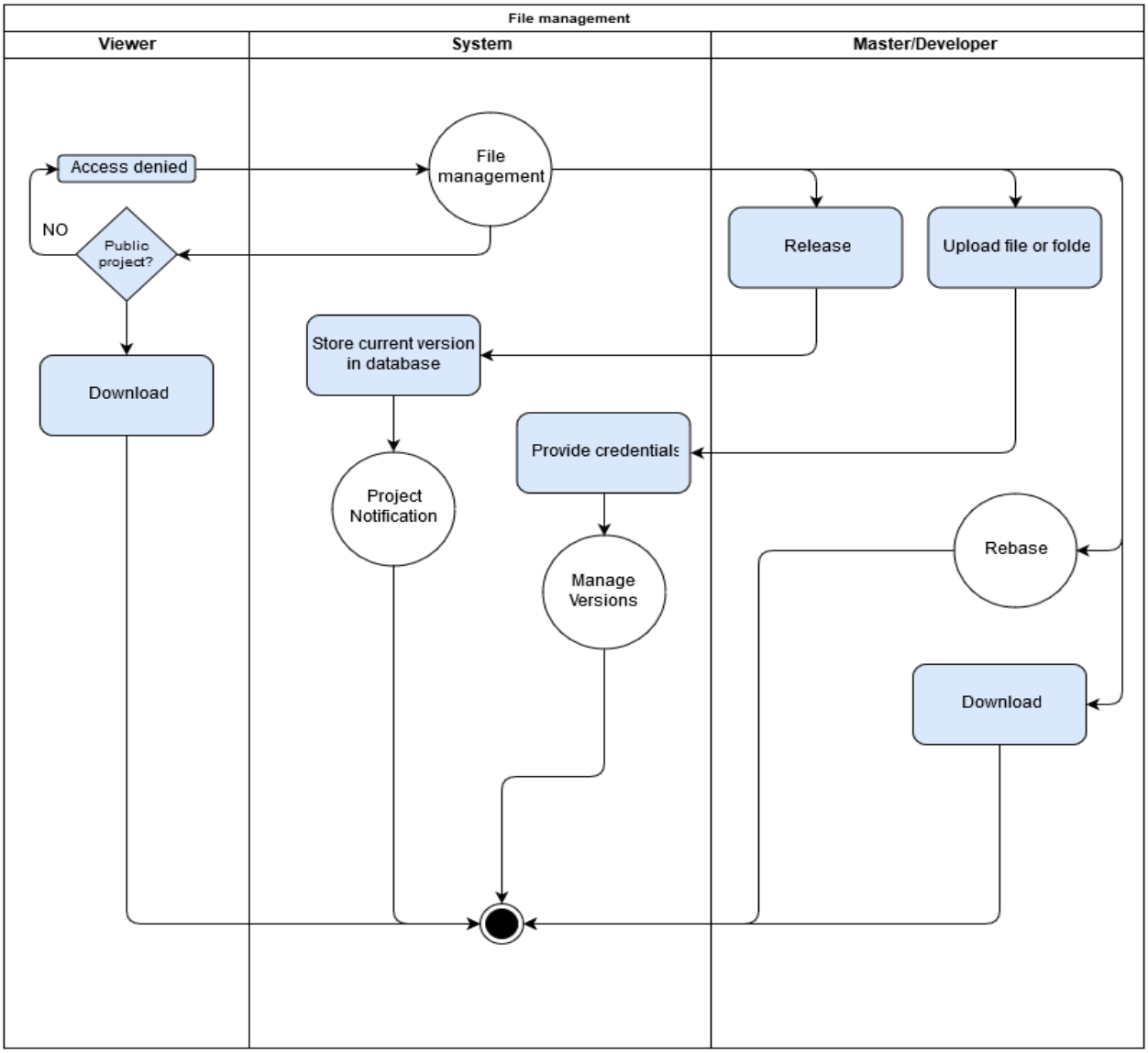


Figure 26: Level 1.2.2 swimlane diagram Project Management (file management)

* Project creation – Level 1.2.1
* Rebase – Level 1.2.2.1
* Project Notification - Level 1.5.2
* Manage versions – Level 1.2.3

Level 1.2.2.1 Project Management (rebase)

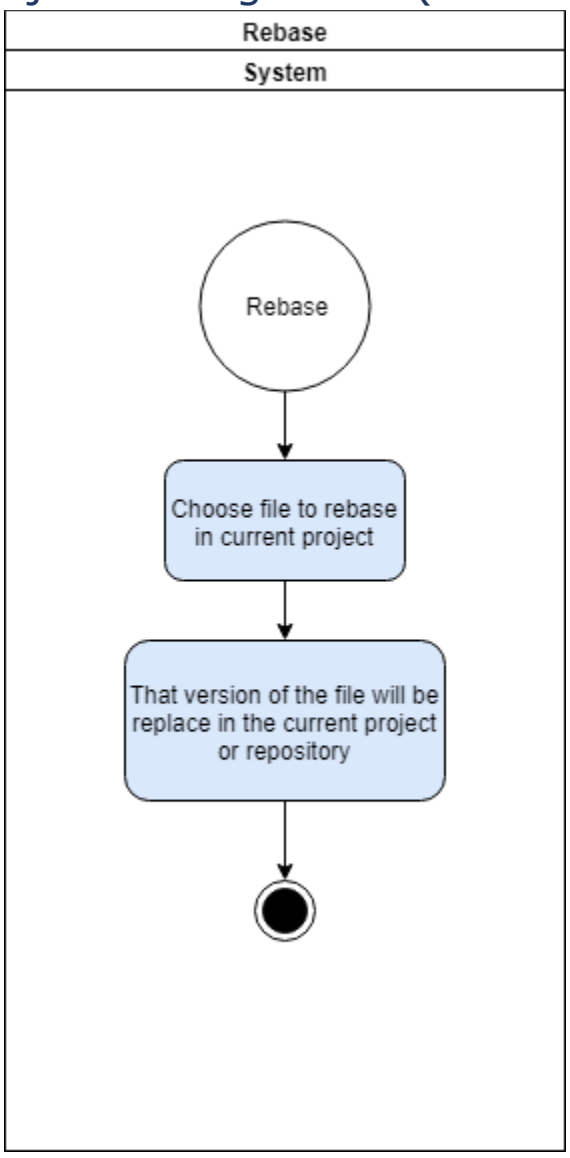


Figure 27: Level 1.2.2.1 swimlane diagram Project Management (rebase)

Level 1.2.3 Project Management (manage version)

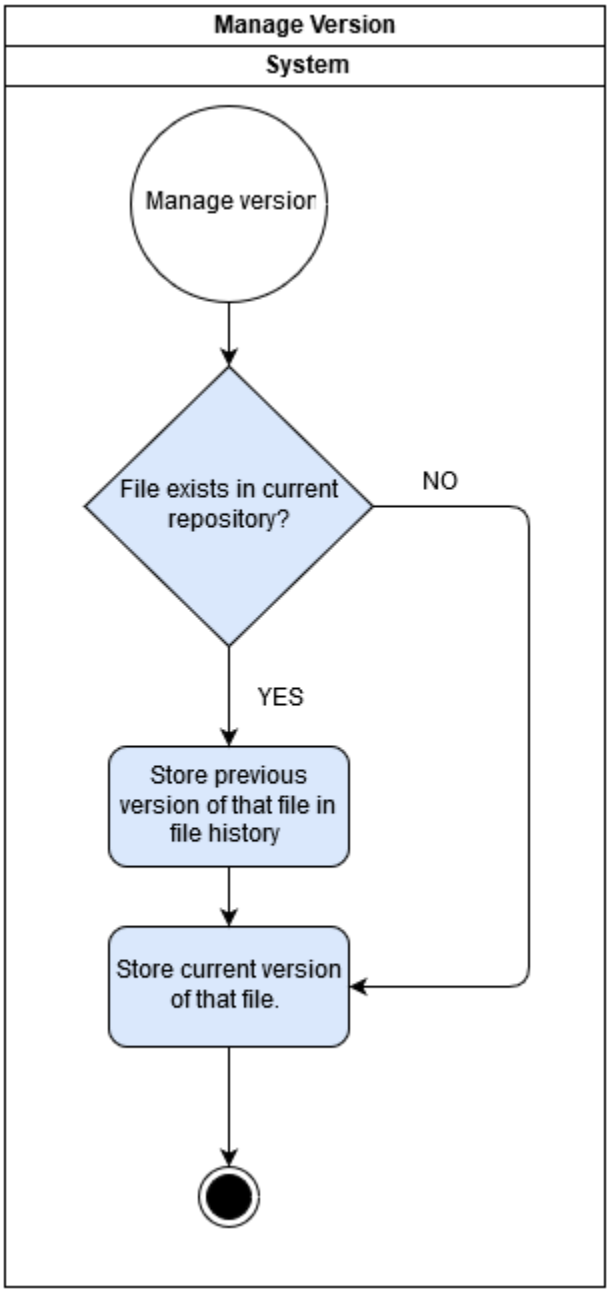


Figure 28: Level 1.2.3 swimlane diagram Project Management (manage version)

Level 1.3 Profile Management

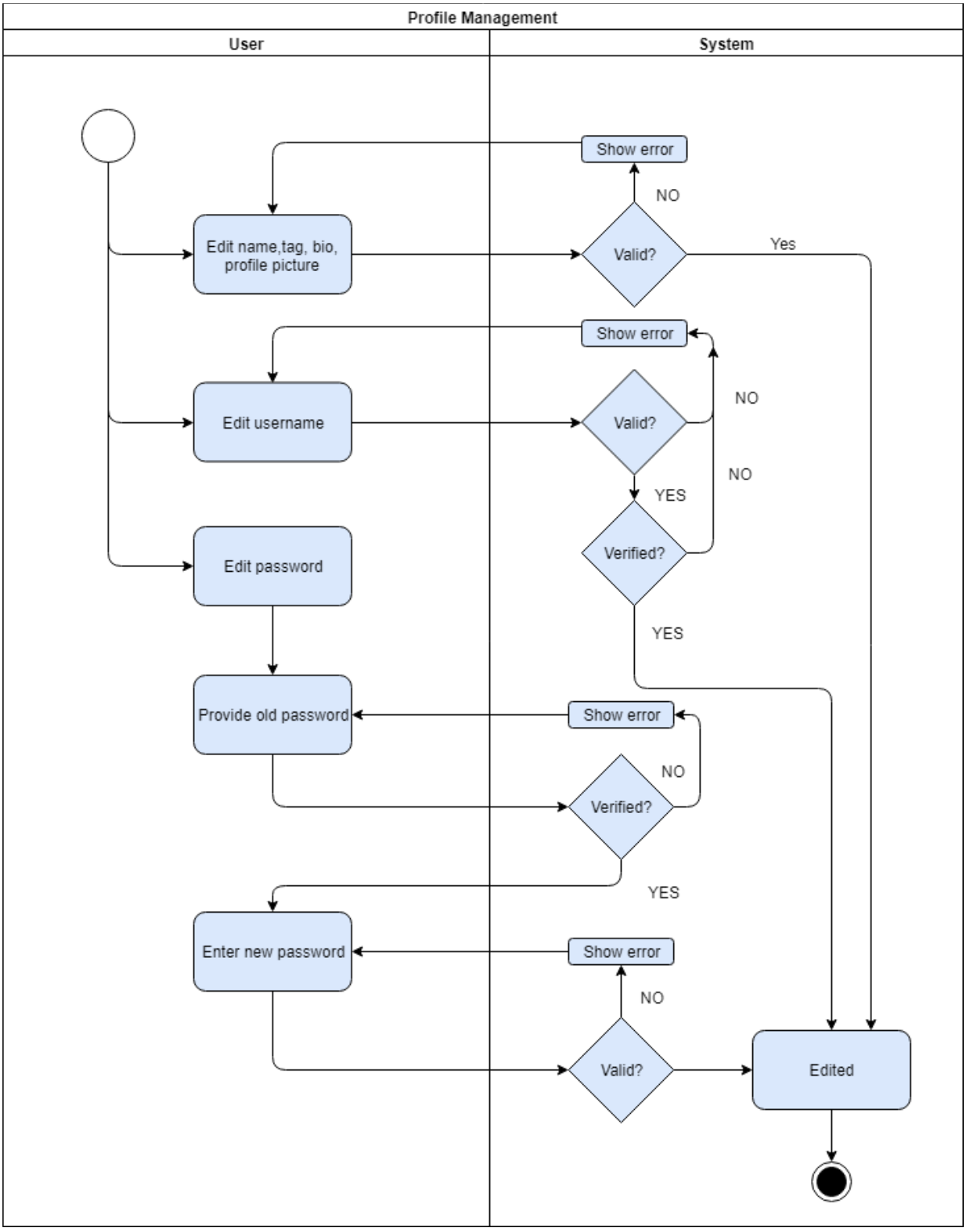


Figure 29: Level 1.3 swimlane diagram Profile Management

Level 1.4 Search

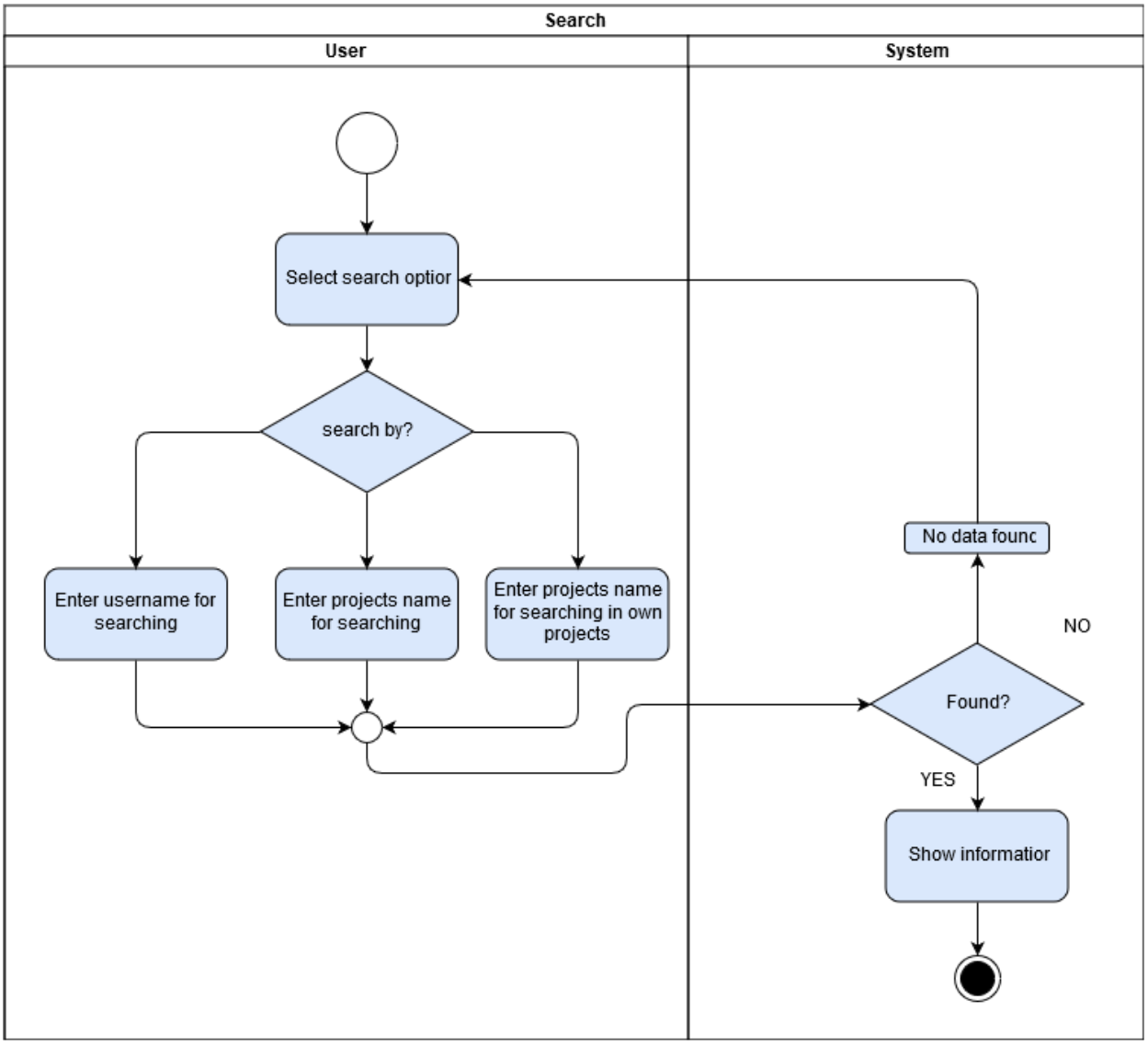


Figure 30: Level 1.4 swimlane diagram Search

Level 1.5.1 Communication (Developer notification)

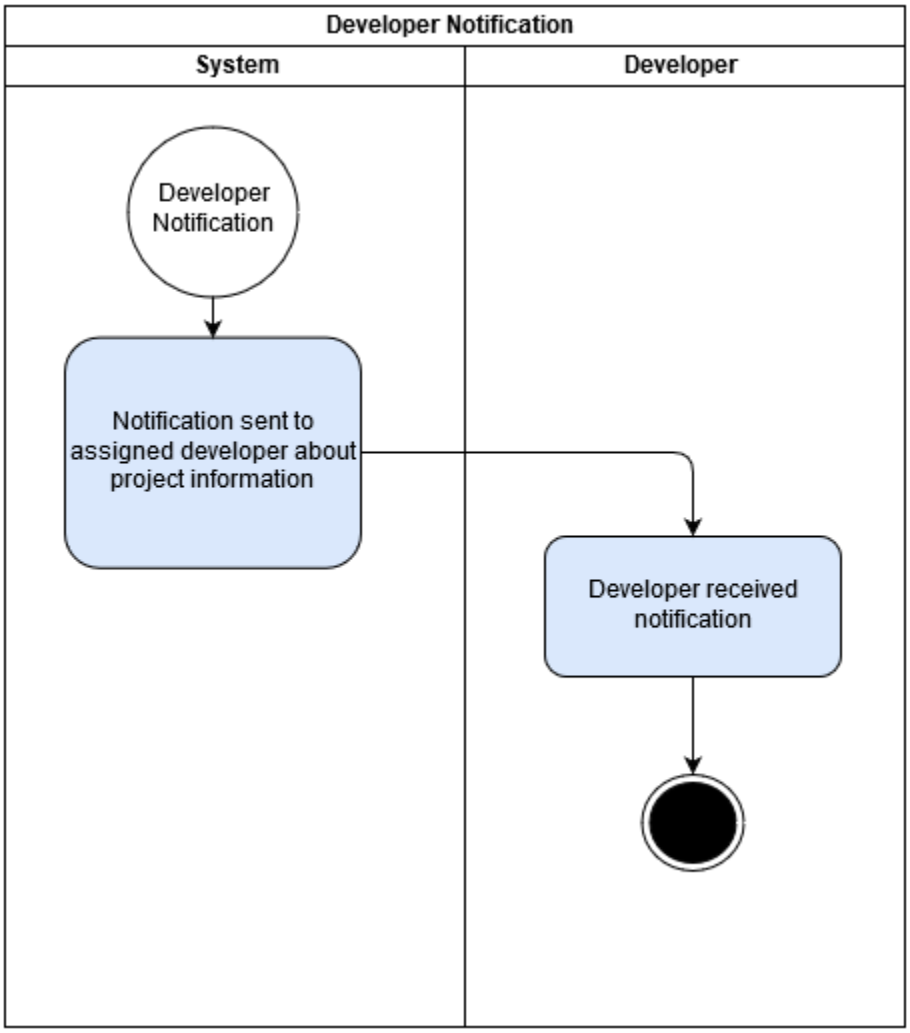


Figure 31: Level 1.5.1 swimlane diagram Communication (Developer notification)

Level 1.5.2 Communication (Project notification)

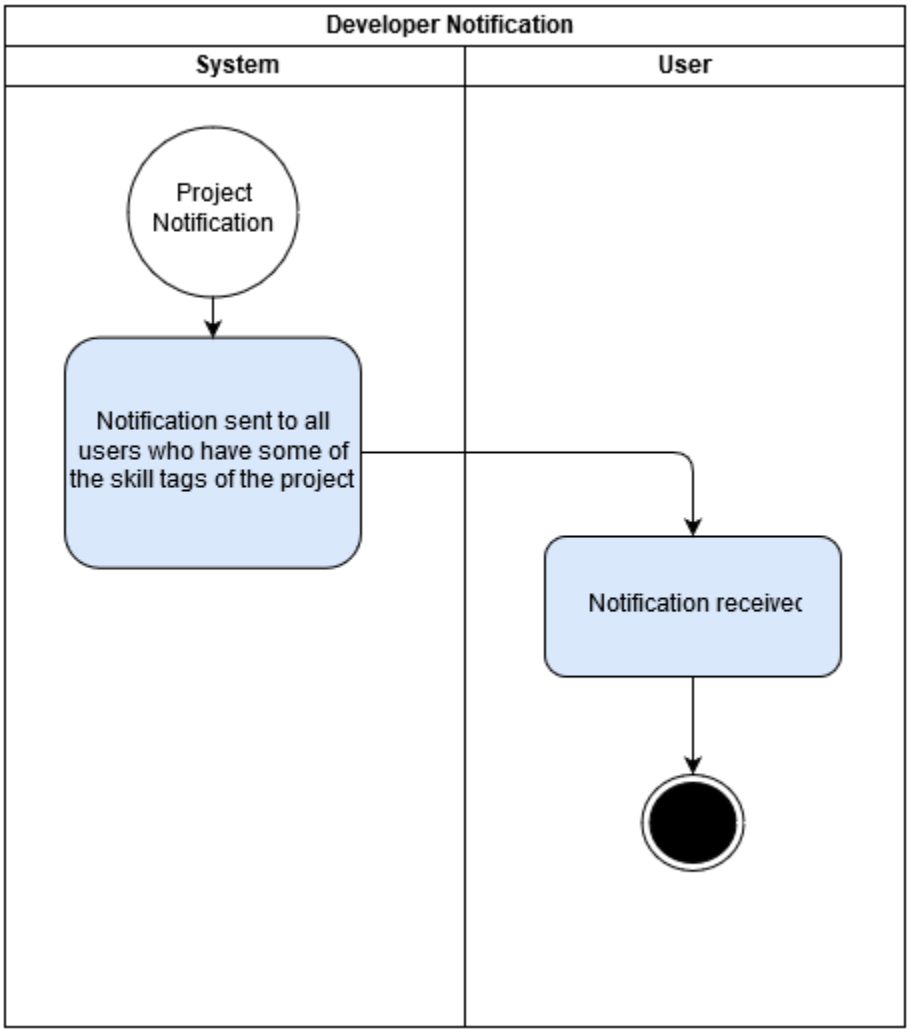


Figure 32: Level 1.5.2 swimlane diagram Communication (Project notification)

CHAPTER 5: DATA MODELING

5.1 Data Modeling Concepts

The process of creating a data model by applying formal data model descriptions using data modeling techniques. If software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated, then the software team chooses to create data models as part of overall requirements modeling. The Entity-Relation diagram is a data modeling technique used in software engineering to produce a conceptual data model of an information system. Entity Relationship Diagrams illustrate the logical structure of databases.

5.2 Data objects

A data object is a representation of composite information that must be understood by the software. Here, composite information means an information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role, an organizational unit, a place or a structure.

5.2.1 Noun Identification

We identified all the nouns whether they are in problem space or in solution space from our usage scenario.

Table 1: Noun identification for data modeling

NO.	Noun	P/S	Attributes
1	Users	S	5, 6, 7, 8, 42, 43, 44
2	System	P	
3	Credentials	P	
4	name	S	
5	username	S	
6	email	S	

7	password	S	
8	account	P	
9	registration	P	
10	Information	P	
11	Recovery	S	
12	File	S	14, 15, 16, 17, 18, 19
13	file_name	S	
14	file_type	S	
15	file_size	S	
16	file_description	S	
17	modification_time	S	
18	file_uploader	S	
19	Sub-System	S	
20	Folder	S	24, 25
21	Project	S	23,27,28,29,30,31
22	Project_created_at	S	
23	Folder_name	S	
24	Folder_created_at	S	
25	Repository	S	
26	Project_name	S	
27	Project_description	S	
28	Project_tags	S	
29	Private	S	
30	Public	S	
31	Developer	S	5, 6, 7, 8, 42, 43, 44
32	Project_progress	S	60, 61, 62
33	Code	P	
34	Reference	S	

35	Version	S	68, 69, 70
36	Download	S	
37	Release	S	
38	Rebase	S	
39	Current	P	
40	ReUpload	P	
41	Profile_picture	S	
42	Skill_tag	S	
43	Biography	S	
44	Verification	S	
45	Master	S	5, 6, 7, 8, 42, 43, 44
46	Comment	S	64, 67
47	Project_management	S	
48	Authentication	P	
49	Signup	P	
50	Signin	P	
51	Singout	P	
52	Project_planning	P	
53	File_management	P	
54	Version_management	P	
55	Profile_management	P	
56	Search	S	
57	Communication	P	
58	Notification	S	1, 73, 74
59	To-Do	S	
60	In-Progress	S	
61	completed	S	
62	Issues	S	65, 66

63	Comment_description	S	
64	Issues_description	S	
65	Issue_time	S	
66	Comment_time	S	
67	Version_number	S	
68	Version_file_name	S	
69	Version_description	S	
70	Database	S	
71	Validation	S	
72	content	S	
73	link	S	

5.2.2 Potential Data Objects

- User: 5-8, 42-44
- File: 14-19
- Folder: 24, 25
- Project: 23, 27-31
- Developer: 5-8, 42-44
- Master: 5-8, 42-44
- Version: 68-70
- Comment: 64, 67
- Issues: 65, 66
- Project-progress: 60-62
- Notification: 1, 73, 74

5.2.3 Final Data Objects

Table 2: Final Data Objects

No	Data objects	Attributes
1	User	name, username, password, email, profile_picture, skill_tag, biography
2	Project	project_name, project_description, access_type, project_tag, project_created_at
3	Folder	folder_name, folder_created_at
4	File	file_name, file_description,modification_time, file_type
5	Comment	comment_description, comment_time
6	Issues	Issue_description, issue_time
7	Version	version_number, version_file_name, version_description
8	Project-progress	to_do, in_progress, completed
9	Notification	content, status, link, sender, receiver

5.3 Data Object Relations

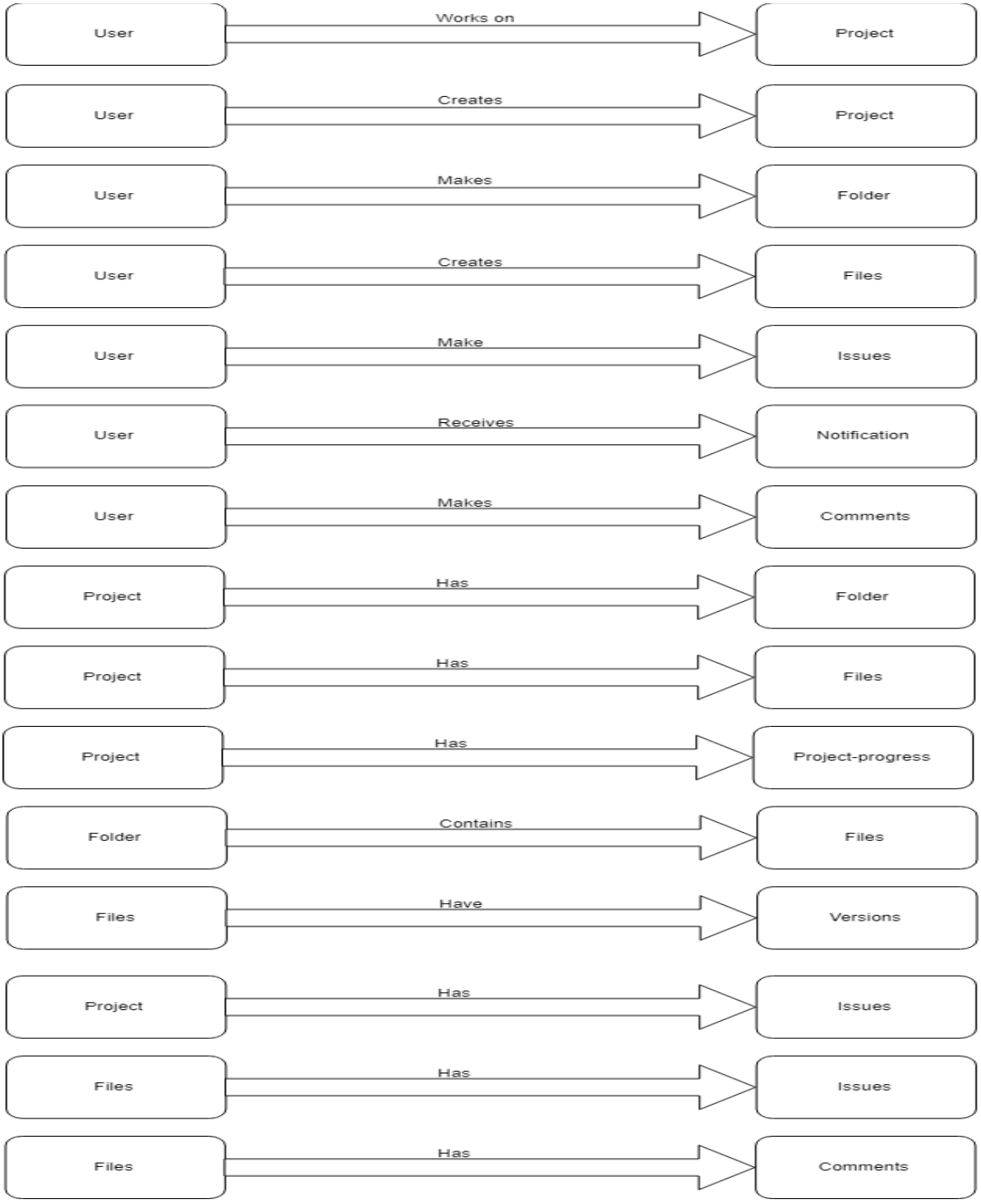


Figure 33: Data object relations

5.4 Entity-Relationship Diagram

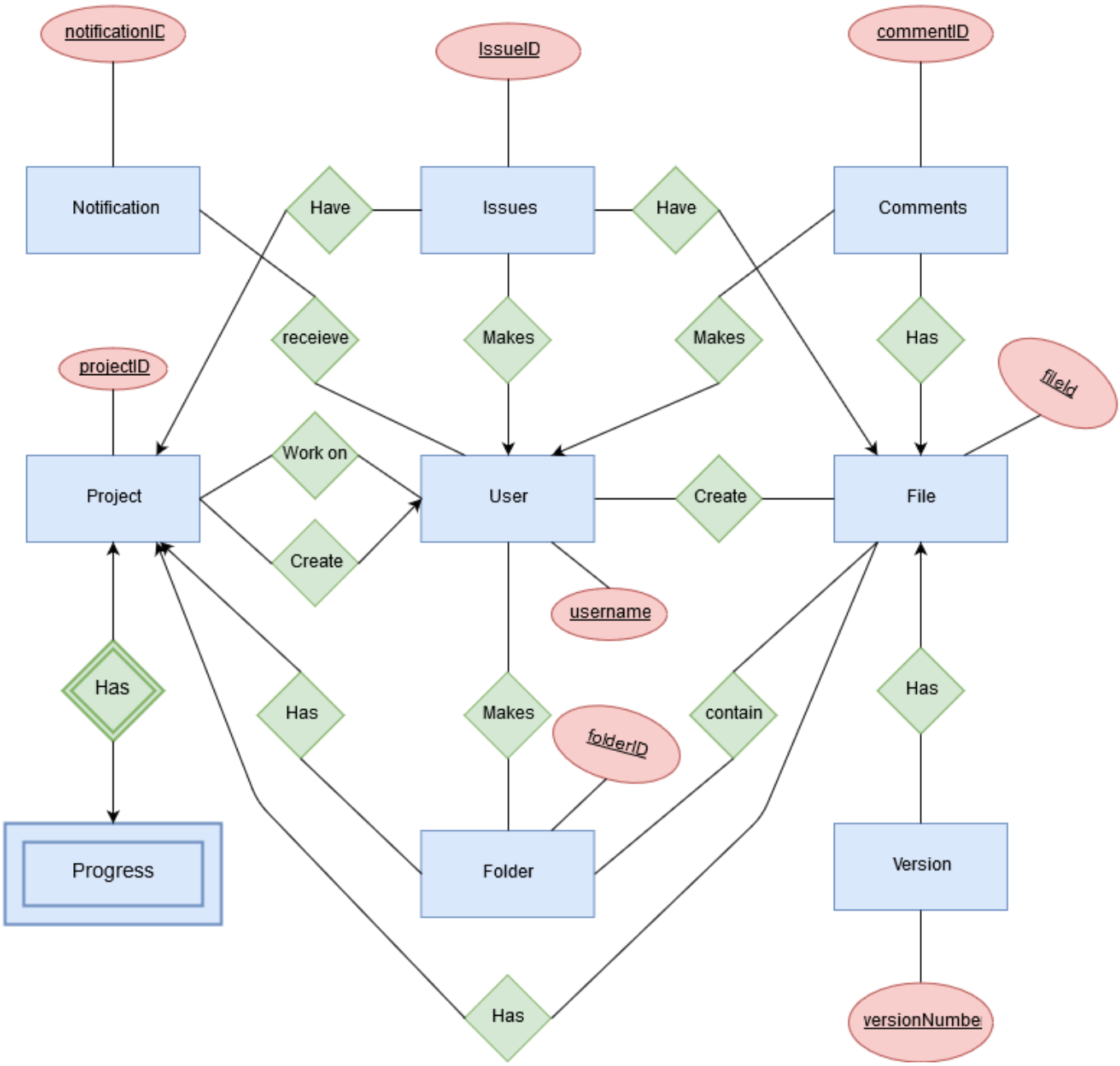


Figure 34: Entity-Relationship Diagram

Table 3: Schema for User

User		
Attributes	**Type**	**Size**
Username	VARCHAR2	100
name	VARCHAR2	50
password	VARCHAR2	50
profile_picture	VARCHAR2	100
skill_tag	VARCHAR2	100
biography	VARCHAR2	200
email	VARCHAR2	100

Table 4: Schema for Project

Project		
Attributes	**Type**	**Size**
projectID	NUMBER	20
projectName	VARCHAR2	50
projectDescription	VARCHAR2	200
accessType	VARCHAR2	20
projectTag	VARCHAR2	100
projectCreatedAt	DATE	15

Table 5: Schema for Folder

Folder		
Attributes	**Type**	**Size**
folderID	NUMBER	20
projectID	NUMBER	20
folderName	VARCHAR2	50
folderCreatedAt	DATE	15

Table 6: Schema for File

FILE		
Attributes	**Type**	**Size**
fileID	NUMBER	20
filename	VARCHAR2	50
fileDescription	VARCHAR2	200
file_size	NUMBER	100
file_uploader	VARCHAR2	20
modification_time	DATE	15
file_type	VARCHAR2	20

Table 7: Schema for Comment

Comment		
Attributes	**Type**	**Size**
commentID	NUMBER	20
username	VARCHAR2	100
fileID	NUMBER	20
commentDescription	VARCHAR2	200
commentTime	DATE	15

Table 8: Schema for issue

Issue		
Attributes	**Type**	**Size**
issueID	NUMBER	20
username	VARCHAR2	100
fileID	NUMBER	20
issueDescription	VARCHAR2	200
issueTime	DATE	15

Table 9: Schema for Version

Version		
Attributes	**Type**	**Size**
versionNumber	NUMBER	20
fileID	NUMBER	20
versionFileName	VARCHAR2	50
verisonDescription	VARCHAR2	200

Table 10: Schema for Project-Progress

Project-Progress		
Attributes	**Type**	**Size**
projectID	NUMBER	20
Todo	VARCHAR2	100
Inprogress	VARCHAR2	100
completed	VARCHAR2	100

Table 11: Schema for Notification

Notification		
Attributes	**Type**	**Size**
notificationID	NUMBER	20
content	VARCHAR2	100
link	VARCHAR2	100
sender	NUMBER	20
receiver	NUMBER	20

Table 12: Schema for UserNotification

UserNotification		
Attributes	**Type**	**Size**
notificationID	NUMBER	20
username	VARCHAR2	100

Table 13: Schema for UserFile

UserFile		
Attributes	**Type**	**Size**
fileID	NUMBER	20
username	VARCHAR2	100

Table 14: Schema for UserFolder

UserFolder		
Attributes	**Type**	**Size**
folderID	NUMBER	20
username	VARCHAR2	100

Table 15: Schema for UserNotification

UserNotification		
Attributes	**Type**	**Size**
notificationID	NUMBER	20
username	VARCHAR2	100

Table 16: Schema for UserProject

UserProject		
Attributes	**Type**	**Size**
projectID	NUMBER	20
username	VARCHAR2	100

Table 17: Schema for FileFolder

FileFolder		
Attributes	**Type**	**Size**
fileID	NUMBER	20
folderID	NUMBER	20

CHAPTER 6: CLASS BASED MODELING

6.1 Class based modeling concepts

A class-based model is the most fundamental model for a system to be done. This is because it describes what is changing within a system. Through class-based modeling we can identify the analysis classes.

6.2 General classifications

To identify the potential classes, we have first selected the nouns from the solution space of the story. These were then characterized in seven general classifications. The seven general characteristics are as follows:

1. External entities

2. Things

3. Events

4. Roles

5. Organizational units

6. Places

7. Structures

Following are the specifications of the nouns according to the general classifications:

Table 18: General Classification of Noun

Noun	P/S	General Classification
Users	S	4,5,7
System	P	
Credentials	P	
name	S	

username	S	
email	S	
password	S	
account	P	
registration	P	
Information	P	
Recovery	S	1,3
File	S	1,2
file_name	S	
file_type	S	
file_size	S	
modification_time	S	
file_uploader	S	4
file_description	S	
Sub-System	P	
Folder	S	2,6
Project	S	2, 6
Project_created_at	S	
Folder_name	S	
Folder_created_at	S	
Repository	S	5,6
Project_name	S	
Project_description	S	
Project_tags	S	
Private	S	
Public	S	
Developer	S	4,7
Project_progress	S	

Code	P	
Reference	S	
Version	S	5
Download	S	3
Release	S	3
Rebase	S	3
Current	P	
ReUpload	P	
Profile_picture	S	
Skill_tag	S	
Biography	S	
Verification	S	
Master	S	4,5
Comment	S	
Project_management	S	3,5
Authentication	S	3,5
Signup	S	
Signin	S	
Singout	S	
Project_planning	P	
File_management	S	3,6
Version_management	S	3,6
Profile_management	S	3,6
Search	S	3,5
Communication	P	
Notification	S	3
To-Do	S	2,6
In-Progress	S	2,6

completed	S	2,6
Issues	S	3
Comment_description	S	
Issues_description	S	
Issue_time	S	
Comment_time	S	
Version_number	S	
Version_file_name	S	
Version_description	S	
Database	S	1,6
Validation	S	3,5

6.3 Selection criteria

The potential classes were then selected as classes by six Selection Criteria. A potential class becomes a class when it fulfills all six characteristics.

1. Retained Information
2. Needed Services
3. Multiple Attributes
4. Common attributes
5. Common operations
6. Essential requirements

Table 19: Selection Criteria

Potential Classes	Accepted Criteria
Users	1,2,3,4,5
Recovery	6
File	3,4,6
File_uploader	2,3,4,5
Folder	3,4

Project	3,4
Repository	3,4
Developer	1,2,3,4,5
Version	2,3,4
Download	2
Release	2
Rebase	2
Master	1,2,3,4,5
Authentication	3,4,5
Search	3,4,5
Notification	3,4,5
To-Do	3
Done	3
In-Progress	3
Issues	
Database	6
Project_management	3,4,5
Validation	3,4,5
Profile_management	3,4,5
File_management	3,4,5
Version_management	3,4,5

6.4 Associate noun and verb identification

In this section, we identified the nouns and verbs associated with the potential classes to have a clear view of the methods and attributes of each class.

Table 20: Associated Verb and Noun Identification

No	Potential Class	Nouns	Verbs
1	Authentication	Name, Username, Password, email	Signup, signin, signout, recover account
2	Search	Filename, username, name, projectname	Search by name, search by username, search by project name,
3	System	text pattern username email notificationLink notificationContent	Check, verify, validate, notify, send email
4	Profile Management	name username password email biography skillTag profilePicture	Change password, change profile picture, change information
5	Project Management	projectTag projectName issueNumber projectAccessType issueTime commentTime	Create project, create project progress, create project issue, comment on project, release project
6	File Management	fileName fileSize fileType fileUploader folderName modificationTime folderCreatedAt	upload file and folder, download file and folder, rebase file, delete file or folder, create file issue, comment on file
7	Version Management	versionNumber versionFilename versionDescription	Version manage, retrieve version detect conflicts, store version
8	Collaborator	-	Manage profile, manage file, search, manage file

6.5 Attribute Selection

Table 21: Attribute Selection

No.	Class	Attribute
1	Authentication	name username email password
2	Search	filename name projectName
3	System	text pattern username email notificationLink notificationContent
4	Profile Management	name username password email biography skillTag profilePicture
5	Project Management	projectTag projectName issueNumber projectAccessType issueTime commentTime
6	File Management	fileName fileSize fileType fileUploader folderName modificationTime folderCreatedAt
7	Version Management	versionNumber versionFilename versionDescription
8	Collaborator	-

6.6 Method Identification

Table 22: Method Identification

No.	Class	Methods
1	Authentication	signUp() signIn() recovery() signOut()
2	Search	searchByName() searchByProjectName() searchByFile()
3	System	validate() verify() notify() sendEmail()
4	Profile Management	uploadProfilePicture() updatePassword() updateInformation()
5	Project Management	createProject() createProjectProgress() createProjectIssue() commentOnProject() release()
6	File Management	upload() download() rebase() delete() createFileIssue() commentOnFile()
7	Version Management	manageVersion() retrieveVersion() detectConfliction() storeVersion()

8	Collaborator	manageProfile() manageProject() search() fileManage()

6.7 Class Cards

Table 23: Class Card for Authentication

Authentication	
Attributes	**Methods**
name username password email	signin() signup() recovery() signout()
Responsibilities	**Collaborative class**
User registration	System
Recover account	System
User login	System
User logout	-

Table 24: Class Card for Search

Search	
Attributes	**Methods**
name projectName fileName	searchByName() searchByProjectName() searchByFile()
Responsibilities	**Collaborative class**
Providing information based on search	-

Table 25: Class Card for Project Management

Project Management	
Attributes	**Methods**
projectTag projectName issueNumber projectAccessType issueTime commentTime	createProject() createProjectProgress() createProjectIssue() commentOnProject() release()
Responsibilities	**Collaborative class**
Storing all project information	
Release project and notify users	System

Table 26: Class Card for File Management

File Management	
Attributes	**Methods**
fileName fileSize fileType fileUploader folderName modificationTime folderCreatedAt	upload() download() rebase() delete() createFileIssue() commentOnFile()
Responsibilities	**Collaborative class**
Uploading file and folder	Version Management, System
Downloading file and folder	-
Rebase file	Version management, System
Delete file	-

Table 27: Class Card for Version Management

Version Management	
Attributes	**Methods**
versionNumber versionFilename versionDescription	manageVersion() retrieveVersion() detectConfliction() storeVersion()
Responsibilities	**Collaborative class**
Storing versions of file	
Detecting conflictions	System

Table 28: Class Card for Profile Management

Profile Management	
Attributes	**Methods**
name username password email biography skillTag profilePicture	uploadProfilePicture() updatePassword() updateInformation()
Responsibilities	**Collaborative class**
Retrieving all user information	System
Storing all updates	-

Table 29: Class Card for Collaborator

Collaborator	
Attributes	**Methods**
-	manageProfile() manageProject() search() fileManage()

Responsibilities	Collaborative class
Manages profile	Profile Management
Manages project	Project Management
Search information	Search
Manages files and folders	File Management

Table 30: Class Card for System

System	
Attributes	**Methods**
text pattern username email notificationLink notificationContent	validate() verify() notify() sendEmail()
Responsibilities	**Collaborative class**
Input format validation and verification	-
Notifying users	-

6.9 Class Diagram

Class diagram is a diagram where dynamics of object interaction and collaboration are represented through UML diagrams and their networks. Here composition, association and inheritance of the classes are shown in the diagram.

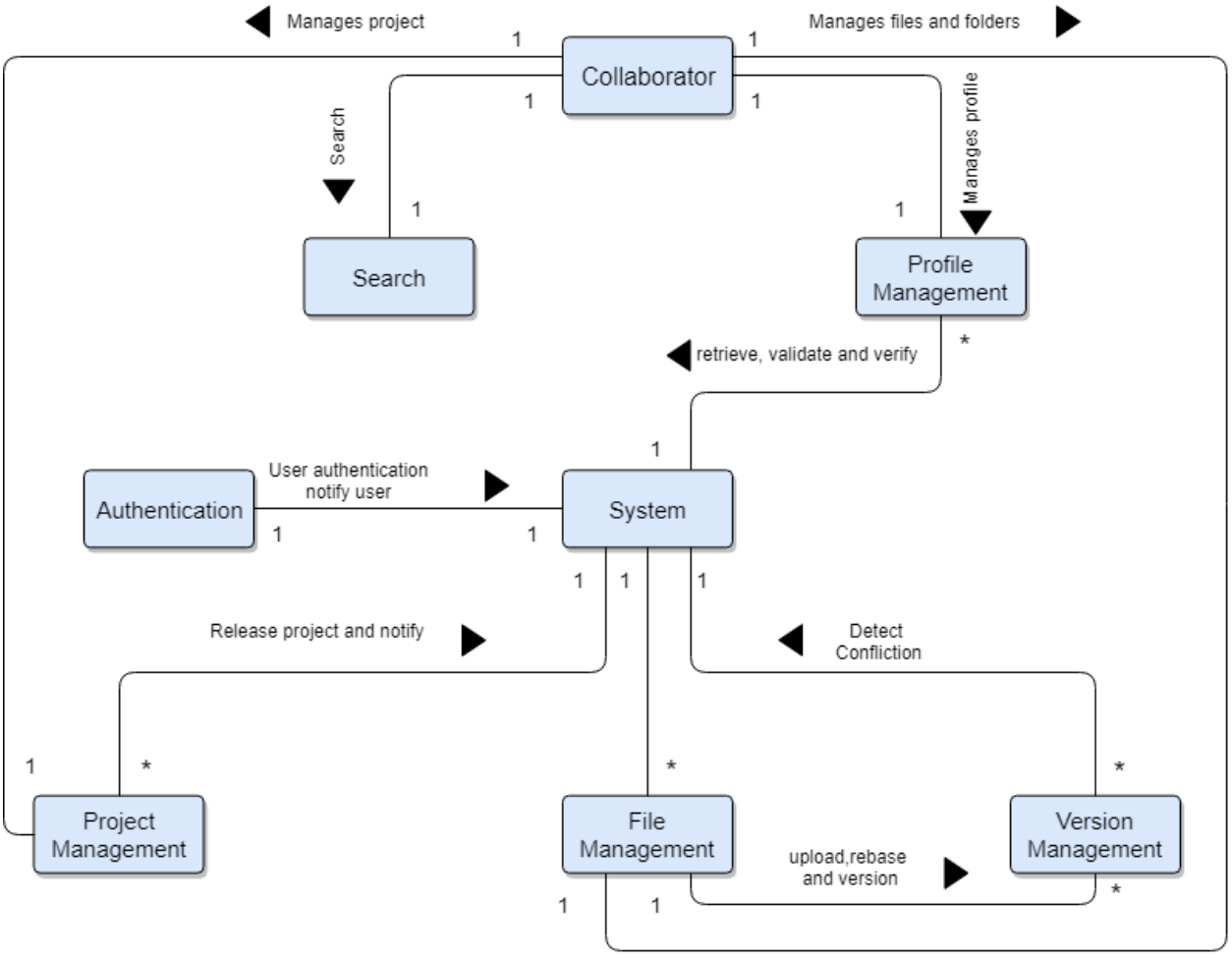


Figure 35: CRC Diagram

CHAPTER 7: BEHAVIORAL MODEL

The behavioral model represents response to external events. Two different behavioral representations are discussed in this chapter. The first indicates how individual class changes state based on external events and the second shows the behavior of the software as a function of time.

7.1 Event Identification

Table 30: Event Identification

No	Events	Primary Objects	Collaborator	Invoked Methods
1	Validates, verifies and signing up into the system, sends confirmation email	Authentication	System	**Authentication**: signUp() **System**: validate() verify() sendEmail()
2	Verifies and sign in into the system	Authentication	System	**Authentication**: signIn() **System**: verify()
3	Verifies and send username and password to user	Authentication	System	**Authentication**: recovery() **System**: verify() sendEmail()

4	Sign-out from the system	Authentication	-	**Authentication**: signOut()
5	Manages profile	Collaborator	Profile Management	**Collaborator**: manageProfile() **Profile Management**: uploadProfilePicture() updatePassword() updateInfromation()
6	Uploads Profile picture	Profile Management	System	**Profile Management**: uploadProfilePicture() **System":** validate()
7	Updates Password	Profile Management	System	**Profile Management**: updatePassword() **System**: validate() verify()
8	Updates Information	Profile Management	-	**Profile Management**: updateInfromation()
9	Uploads file and folder and manage version	File Management	Version Management	**Project Management**: upload() **Version Management**:

				manageVersion()
10	Download file and folder	File Management	-	**File Management**: download()
11	Validates credentials	File Management	System	**System**: validate()
12	Rebas file	File Management	-	**File Management:** rebase()
13	Crates issue on file	File Management	-	**File Management:** createFileIssue()
14	Releases project and notifies users	Project Management	System	**Project Management:** release() **System**: notify() sendEmail()
15	Deletes file, folder and project	File Management	-	**Fie Management**: delete()
16	Manages Version	Version Management	-	**Version Mangement:** ManageVersion()

17	Creates project progress	Project Management	-	**Project Managemen**t: createProjectProgress()
18	Crates issue on project	Project Management	-	**Project Management:** createProjectIssue()
19	Create Project	Project Management	-	**Project Management:** createProject()
20	Search Information	Collaborator	Search	**Collaborator:** search() **Search:** searchByName() searchByProjectName() searchByFile()
21	Search by name	Search	-	**Search**: searchByName()
22	Search by project	Search	-	**Search**: searchByProjectName()
23	Search by file	Search	-	**Search**: searchByFile()

24	Makes comments on file	File Management	-	**File Management**: commentOnFile()
25	Makes comments on project	Project Management	-	**Project Management**: commentOnProject()
26	Manages version, retrieve version and store version	Version Management	-	**Version Management**: manageVersion() retrieveVersion() storeVersion()
27	Detects confliction while uploading	Version Management		**Version Management**: detectConfliction()
28	Validates Input	System	-	**System**: validate()
29	Verifies inputs	System	-	**System:** verifies()

7.2 State Transition Diagram

State-transition diagrams describe all of the states that an object can have, the events under which an object changes state (transitions), the conditions that must be fulfilled before the transition will occur (guards), and the activities undertaken during the life of an object (actions).

AUTHENTICATION:

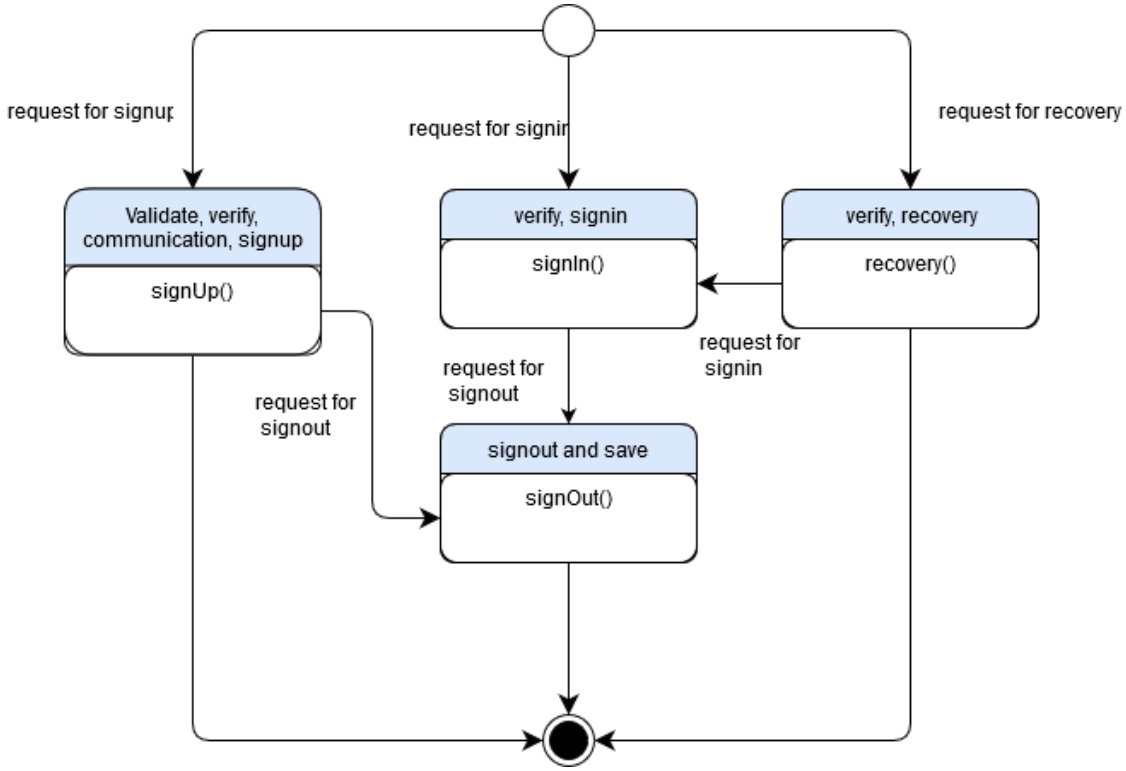


Figure 36: State transition diagram Authentication

COLLABORATOR:

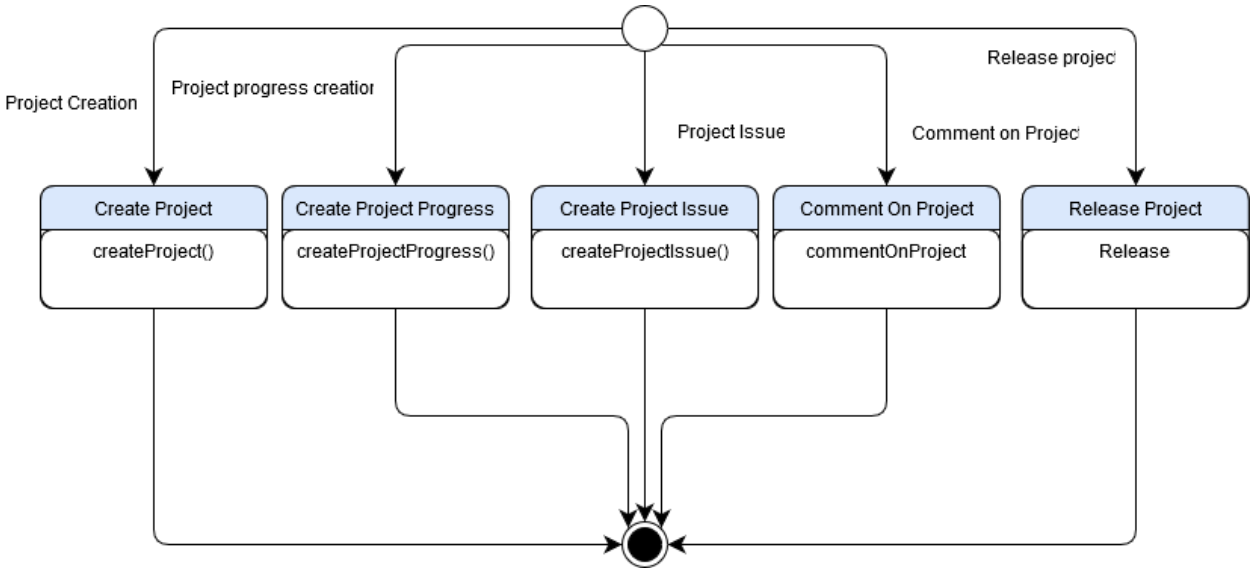


Figure 37: State transition diagram Collaborator

FILE MANAGEMENT:

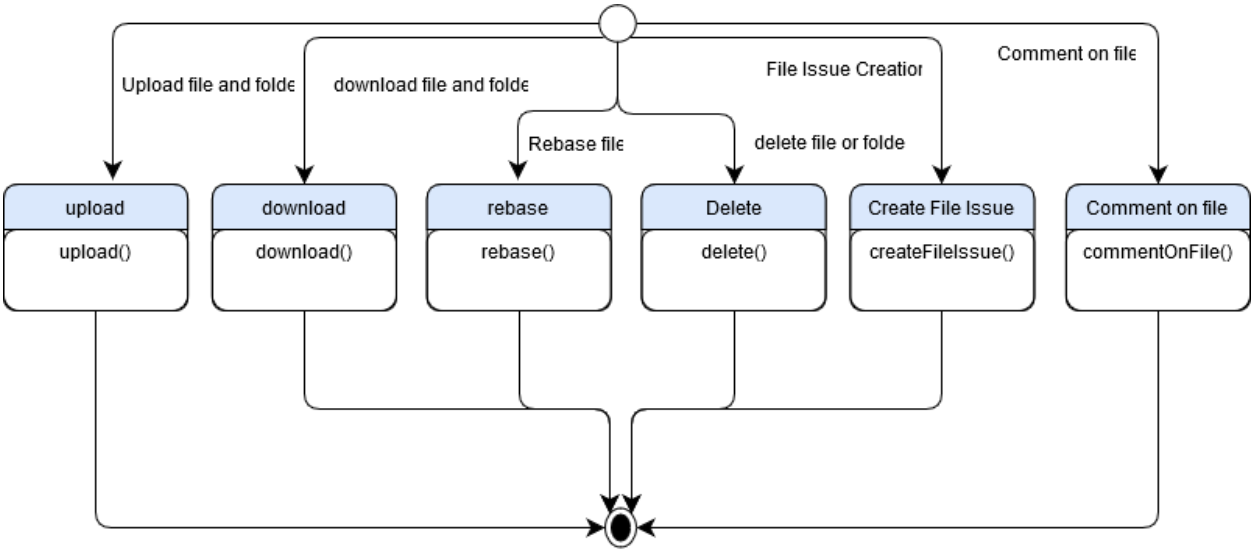


Figure 38: State transition diagram File management

PROFILE MANAGEMENT:

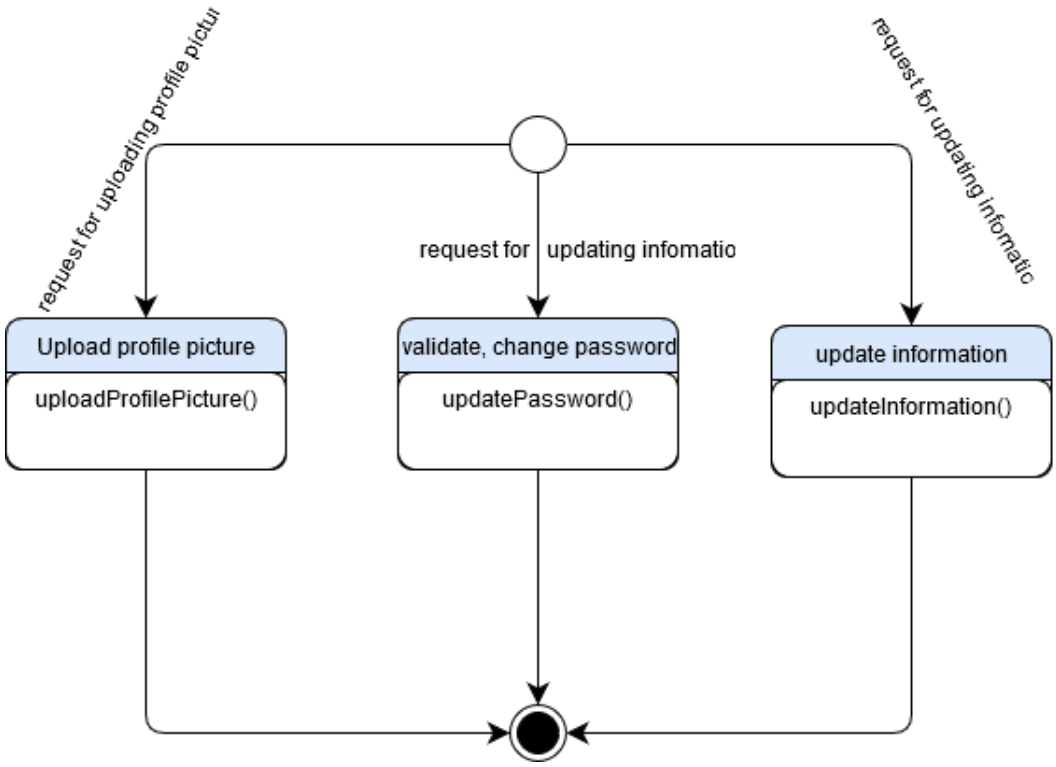


Figure 39: State transition diagram Profile management

PROJECT MANAGEMENT:

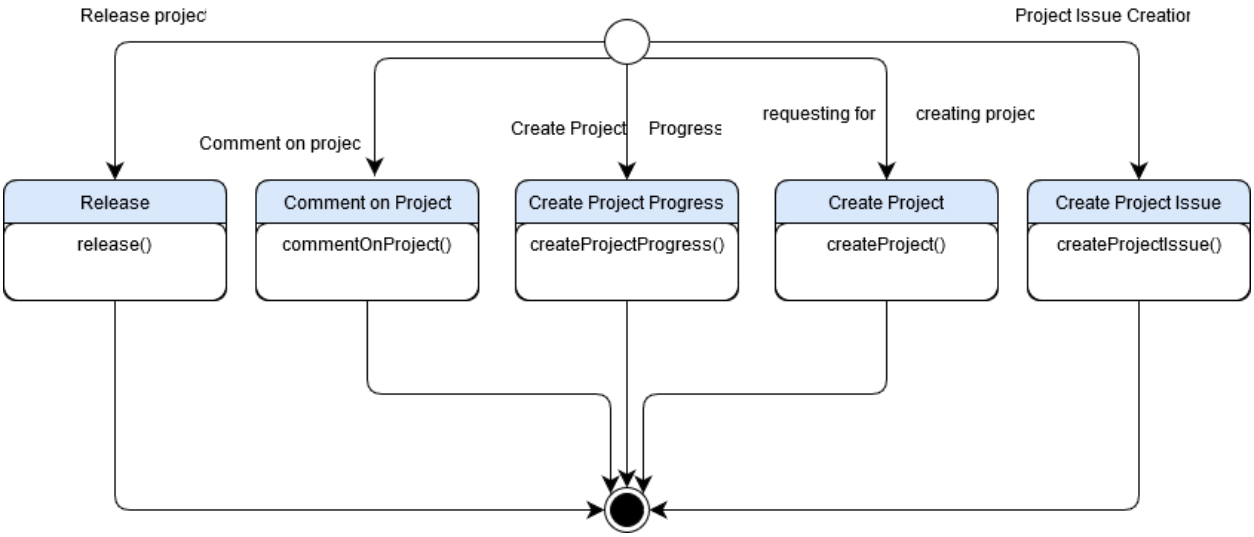


Figure 40: State transition diagram project management

SEARCH:

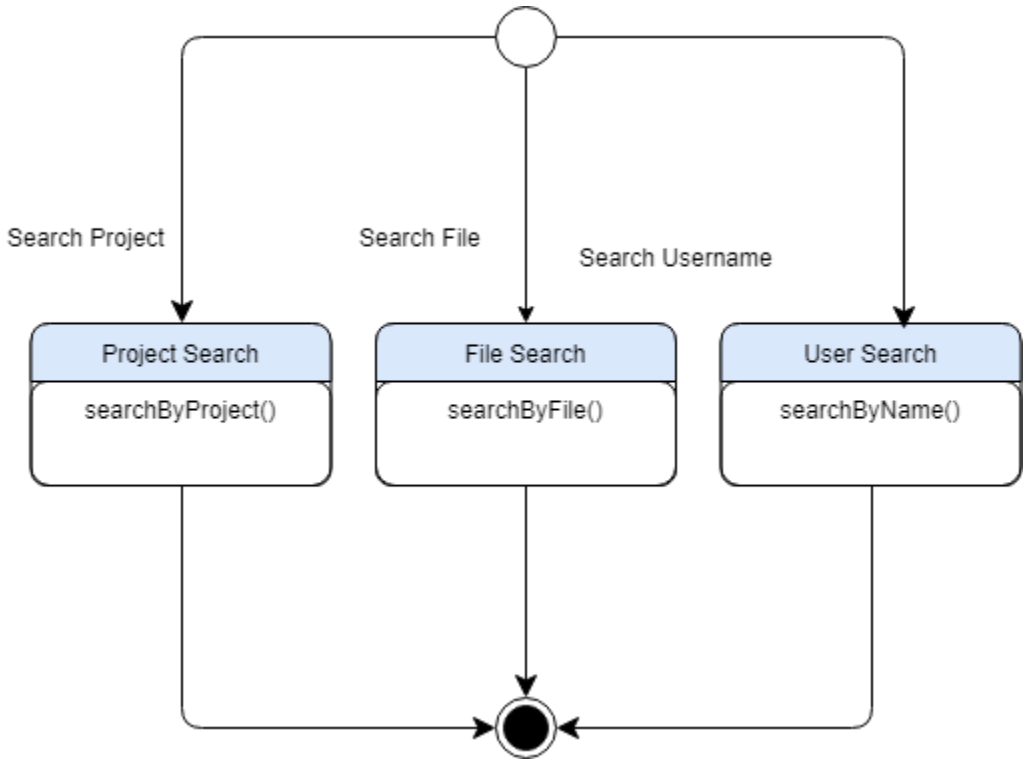


Figure 41: State transition diagram Search

SYSTEM:

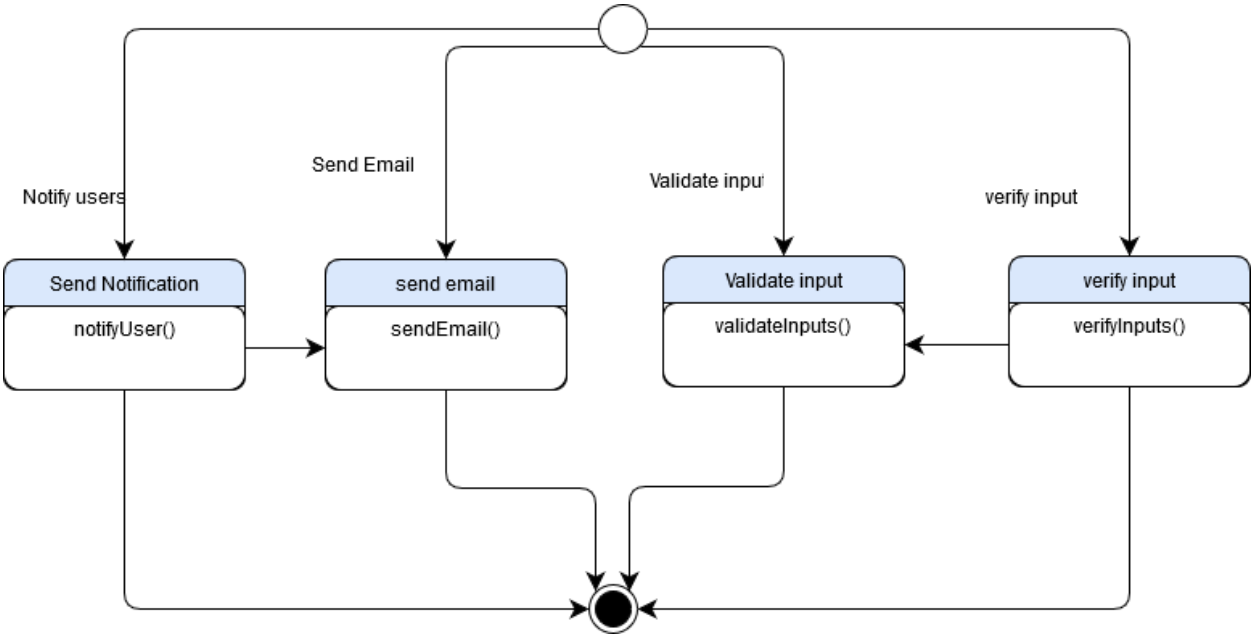


Figure 42: State transition diagram system

VERSION MANAGEMENT:

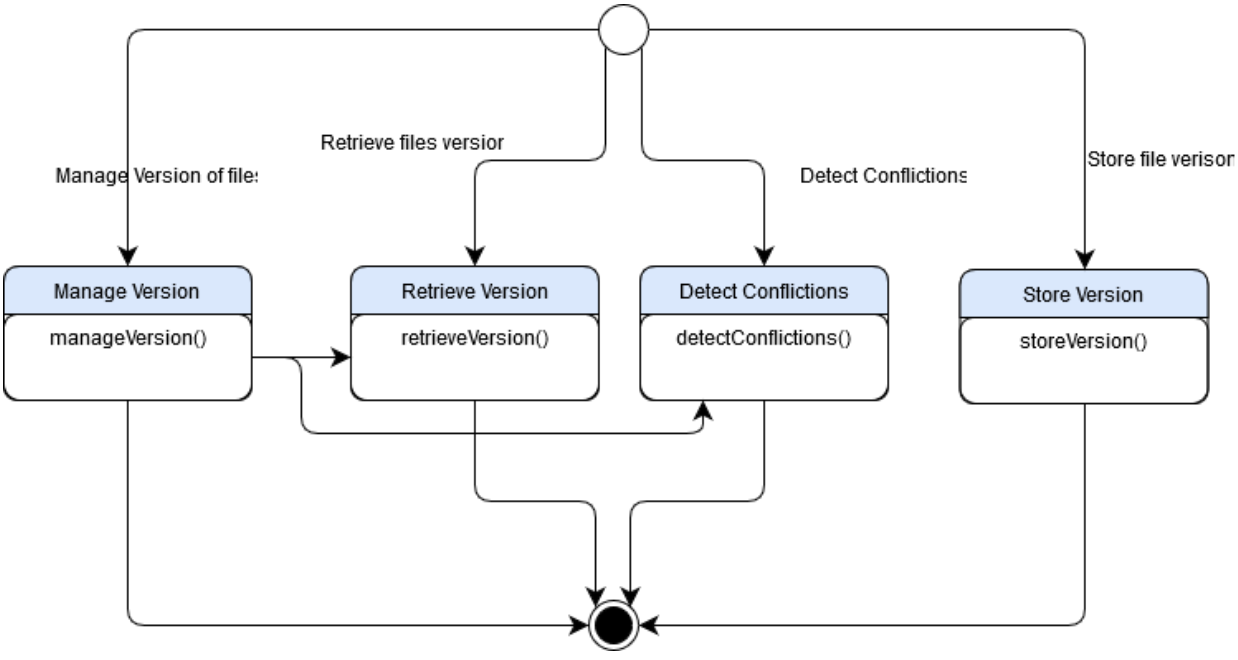
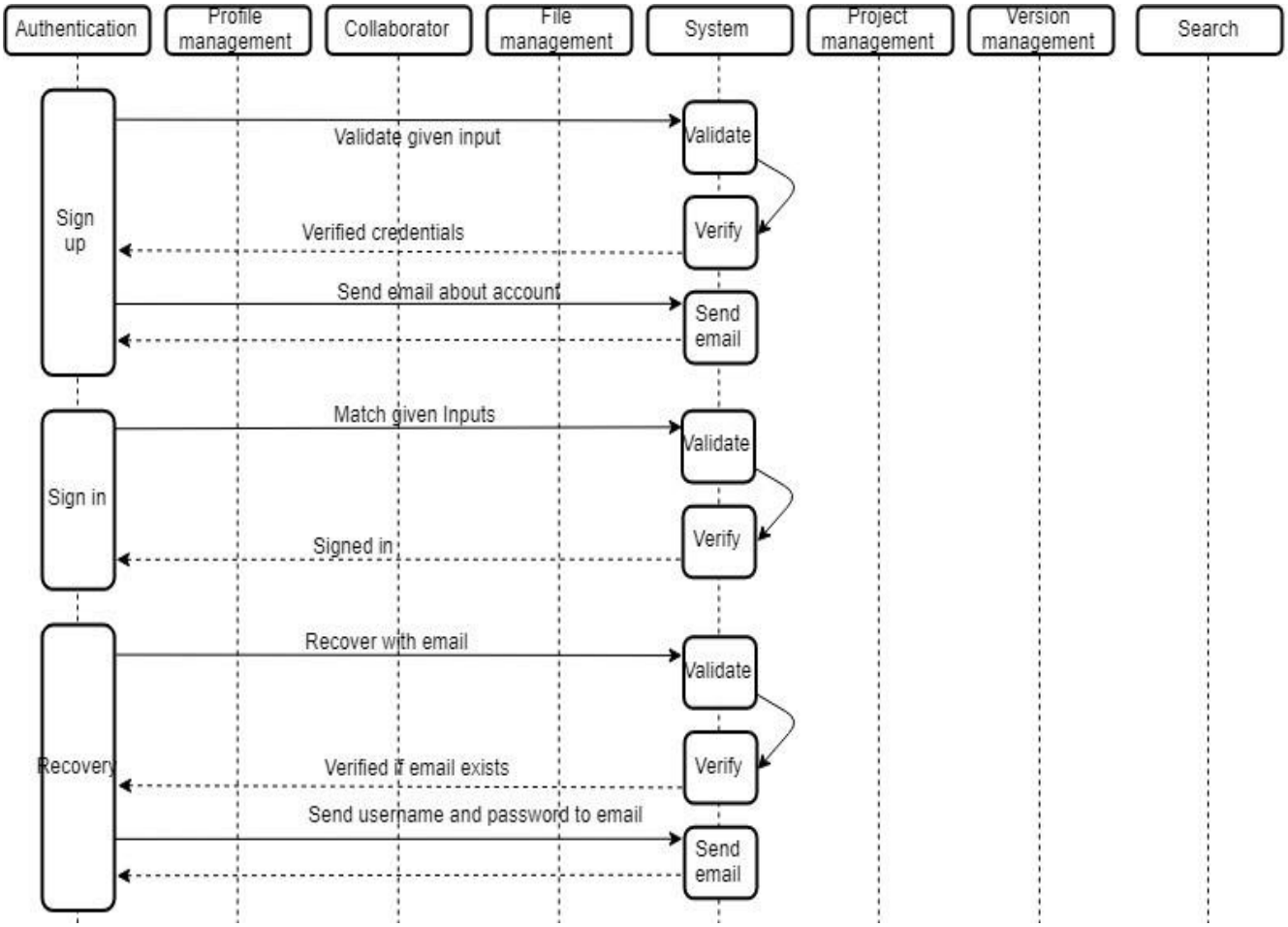
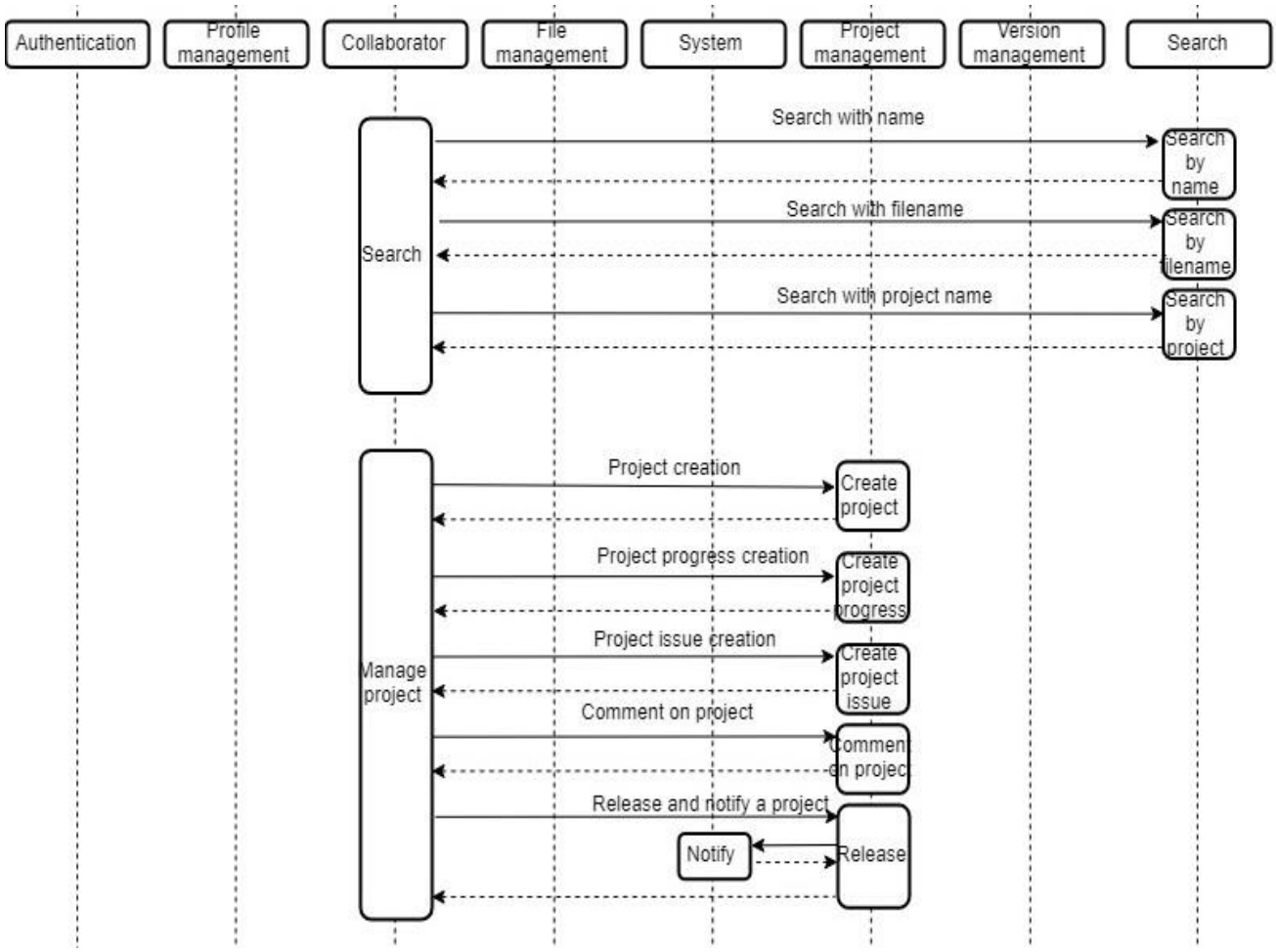


Figure 43: State transition diagram version management

7.3 Sequence Diagram of BYTE PRO

Sequence diagrams describe interactions among classes. It also helps visualize and validate various runtime scenarios.




Authentication
Profile management
Collaborator
File management
System
Project management
Version management
Search
Search with name
Search by name
Search with filename
Search by filename
Search
Search with project name
Search by project
Project creation
Create project
Project progress creation
Create project progress
Project issue creation
Create project issue
Manage project
Comment on project
Commen on projec
Release and notify a project
Notify
Release

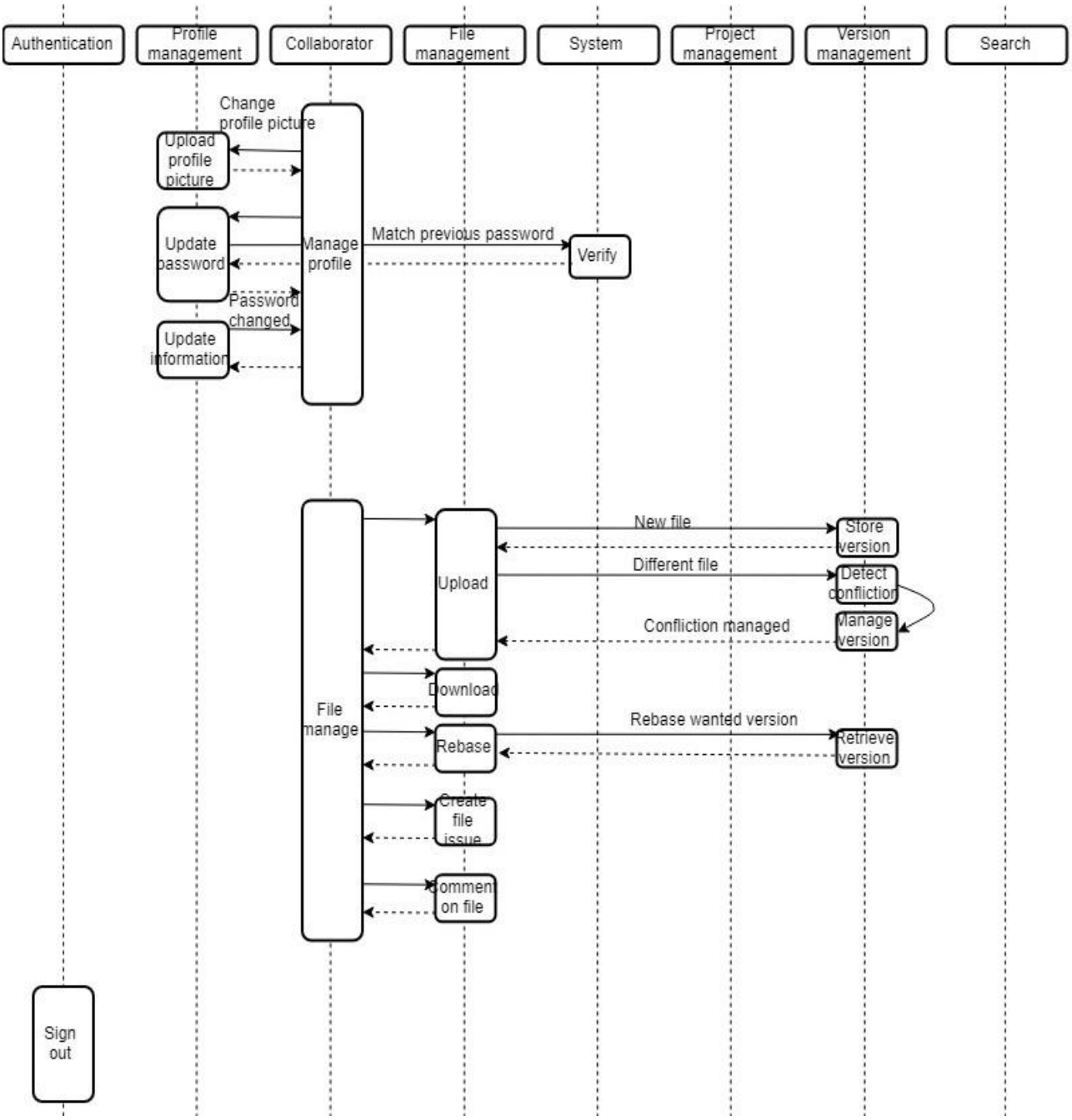


Figure 44: Sequence Diagram of BYTE PRO

CHAPTER 8: DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. DFD graphically represents the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. There are four components of DFD.

1. External entity

2. Process

3. Data store

4. Data flow

8.1 Data Flow Diagram

Level 0 Data flow diagram

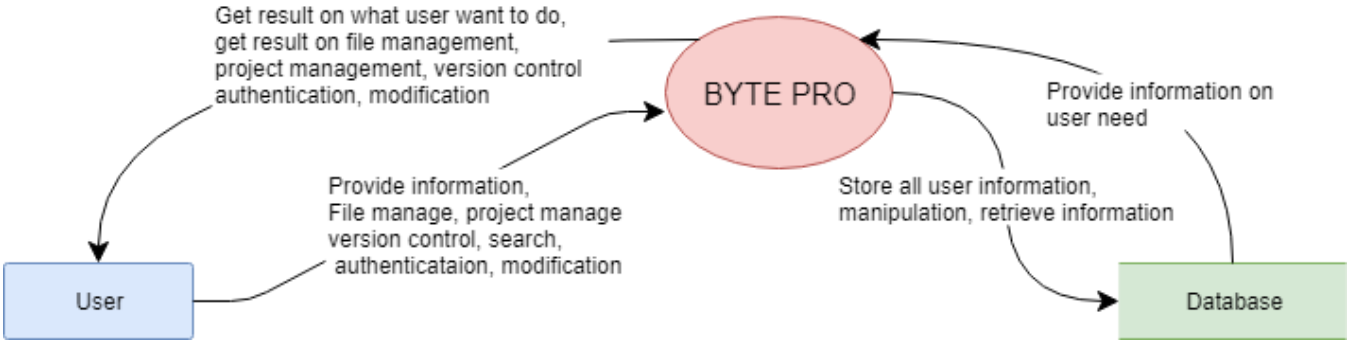


Figure 45: Level 0 Data flow diagram

Level 1 Data flow diagram

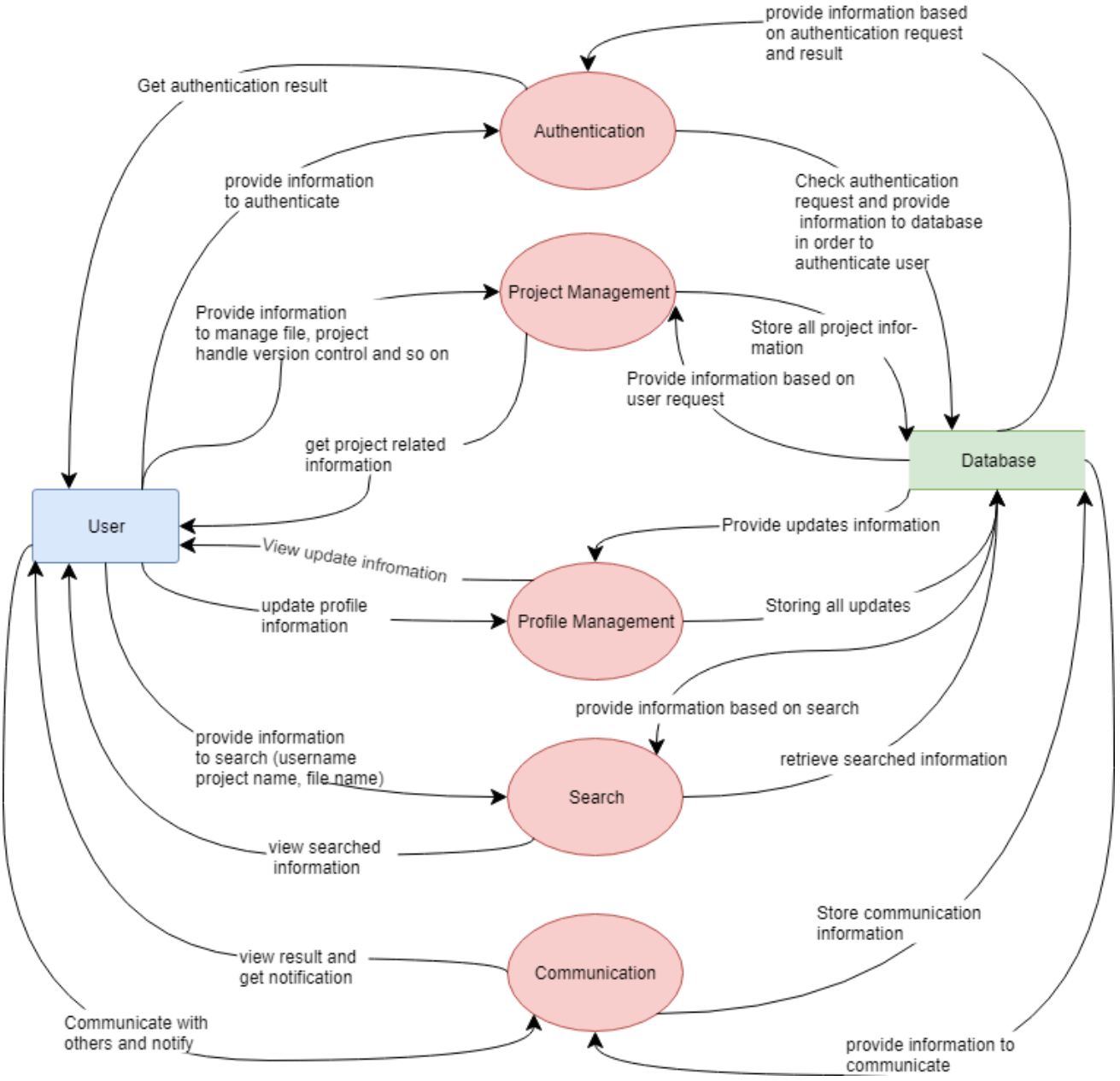


Figure 46: Level 0 Data flow diagram

Level 1.1 Data flow diagram

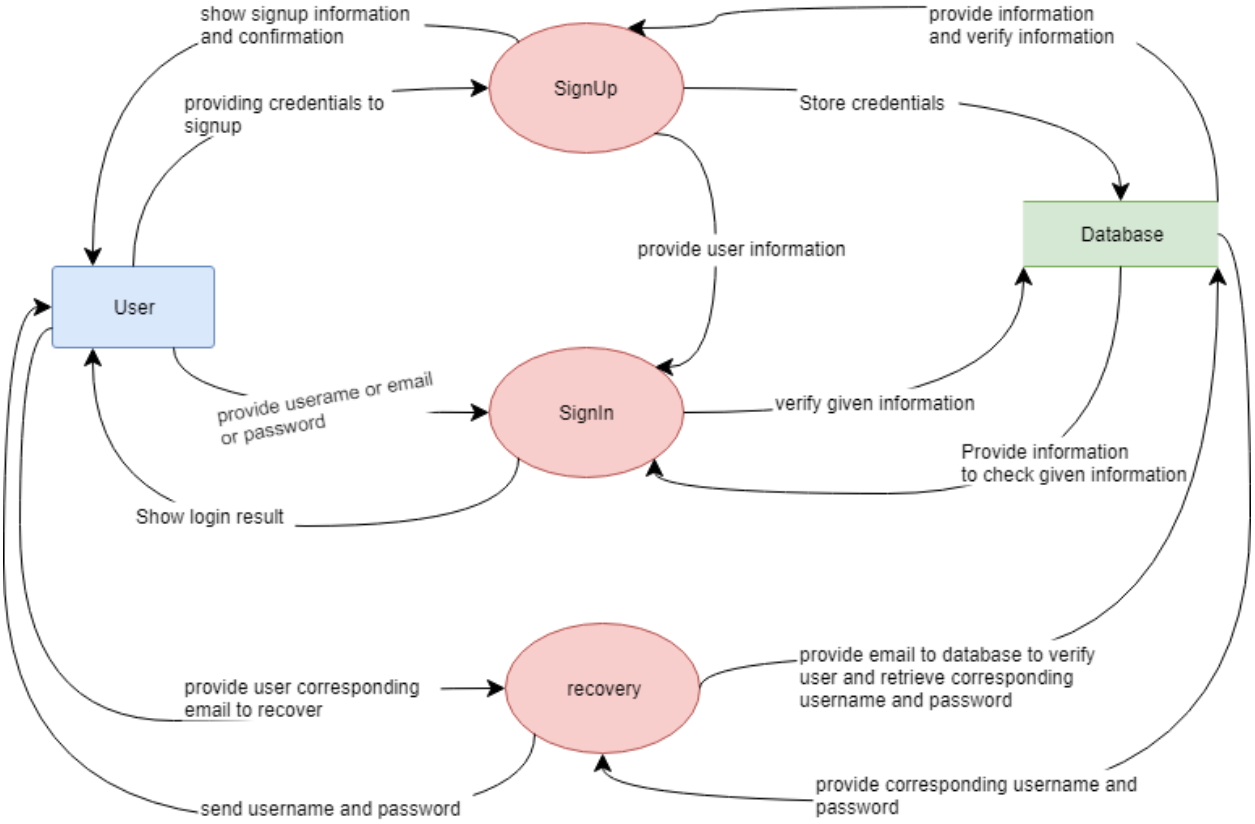


Figure 47: Level 0 Data flow diagram

Level 1.2 Data flow diagram

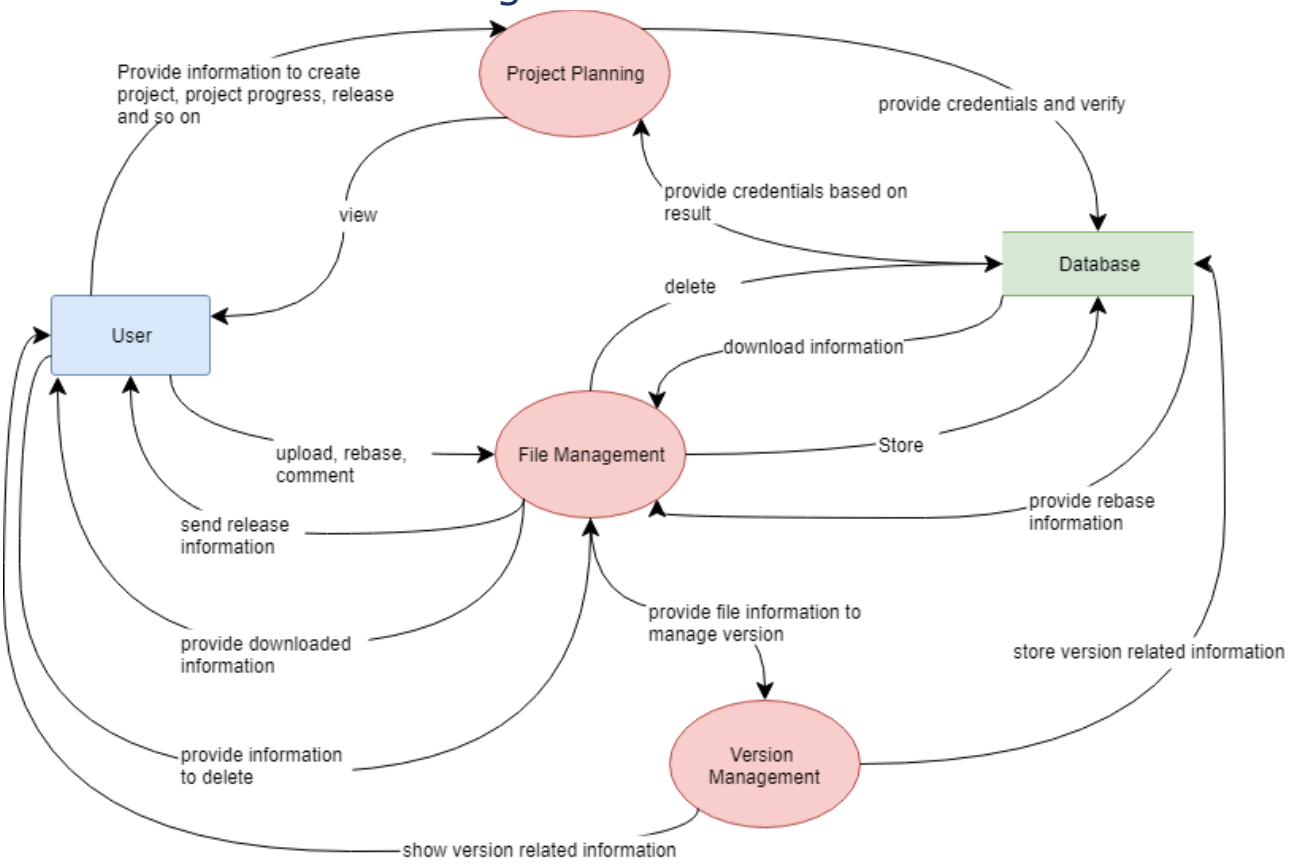


Figure 48: Level 0 Data flow diagram

CHAPTER 9: CONCLUSION

The purpose of this document is to give a detailed description of the requirements for the BYTE PRO software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a user for its approval and a reference for developing the first version of the system for the development team.

It will be very easy to conduct the whole project using this document.

APPENDIX

Group Meetings:

1.

Date: 10 January, 2018

Place: IIT, DU

Subject: Discussing about the group

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

2.

Date: 12 January, 2018

Place: IIT, DU

Subject: Discussing about the project and its feasibility

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

3.

Date: 13 January, 2018

Place: NSU, Jantrik telecom

Subject: Meeting with stakeholders

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

4.

Date: 22 January, 2018

Place: IIT, DU

Subject: Identifying some requirements and discussing with supervisor

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

5.

Date: 25 January, 2018

Place: IIT, DU

Subject: Finalizing requirements and completing elicitation process

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

5.

Date: 30 January, 2018

Place: IIT, DU

Subject: Discussing about sub-system and use case

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

6.

Date: 2 February, 2018

Place: IIT, DU

Subject: Discussing about activities and swimlane

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

7.

Date: 10 February, 2018

Place: IIT, DU

Subject: Discussing about data modeling of the project and noun parsing

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

8.

Date: 20 February, 2018

Place: IIT, DU

Subject: Completing ER and discussing with teachers and class based modeling

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829

9.

Date: 5 March, 2018

Place: IIT, DU

Subject: Discussing about behavioral modeling, sequence diagram

Group Members:

Chinmoy Acharjee – BSSE0819

Sabik Abtahee – BSSE0829