Institute Of Information Technology
University Of Dhaka

Ekushey Online Code Editor

Software Requirements Specification Documentation

Ekushey Online Code Editor

Course: SE505 Software Project Lab-II

Submitted by

Md. Nazmul Haque

BSSE-0635

Submitted to

Asif Imran Anik

Lecturer

Institute of Information Technology (IIT)

University of Dhaka

Supervised by

Md. Saeed Siddik

Lecturer

Institute of Information Technology (IIT)

University of Dhaka

Date:13.04.2016

Letter of Transmittal

6th April, 2016
Asif Imran Anik
Lecturer
Institute of Information Technology
University of Dhaka

Subject: Submission of term report on "Ekushey Online Code Editor".

Dear Sir,

With due respect, we are submitting the report on the above topic you assigned to us. In this report, we have to give our best effort albeit there might be some shortcomings. We would be highly obliged if you consider those from excusable point.

Yours sincerely

MD. Nazmul Haque (BSSE 0635)
3rd Year, 5th Semester, 6th Batch
Institute of Information Technology
University of Dhaka
Session: 2013-14

Acknowledgement

I am highly indebted for getting such a tremendous opportunity to prepare the report on **Ekushey Online Code Editor.** I would like to thank whole-heartedly my teacher, Md. Saeed Siddik, Lecturer, Institute of Information Technology, University of Dhaka, for giving me guideline about how can I prepare this report. In completing this paper I have collected various important data and information from students, Institute of Information Technology. I am thankful to all of the works sited.

Abstract

The study is made for Ekushey Online Code Editor. The object of this study is to develop a SRS (software requirements and specification) of the Ekushey Online Code Editor.

Table of Contents

Chapter 1: Introduction 9

1.1 Purpose 10

1.2 Intended Audience 10

Chapter 2: Inception 11

2.1 Introduction 12

2.1.1 List of Stakeholders 12

2.1.2 Recognizing multiple view point 13

2.1.3 Working towards collaboration 13

2.1.4 Requirements questionnaire 14

2.2 Conclusion 14

Chapter 3: Elicitation 15

3.1 Introduction 16

3.2 Eliciting requirements 16

3.2.1 Collaborative Requirements Gathering 16

3.2.2 Quality Function Deployment (QFD) 16

3.2.3 Usage Scenario 18

3.2.4 Elicitation Work Product 20

3.3 Conclusion 20

Chapter 4: Scenario Based Modeling 21

4.1 Definition of Use case 22

4.2 Use Case Diagrams 22

4.2.1 System Description from Level-0 use case: 22

4.2.2 System description from level-1 use case diagram: 24

4.3 Activity & Swim lane Diagrams 32

Chapter 5: Data Model 53
5.1 Data Modeling Concept 54
5.2 Data Objects 54
5.2.1 Grammatical parsing (Noun identify) 54
5.2.2 Identify Data Objects 56
5.2.3 Data Object Relation 57
5.2.4 E-R Diagram 58
5.3.5 Schema Form (Tables) 59
5.3.5 Data Tables 60
Chapter 6: Class Based Model 61
6.1 Class Based Modeling Concept 62
6.1.1 Identifying Analysis Class 62
6.1.2 Class Responsibility Collaboration (CRC) 67
Chapter 7: Flow Oriented Model 68
7.1 Introduction 69
7.2 Data Flow Diagram 69
Chapter 8: Behavioral Model 72
8.1 Introduction 73
8.2 Identifying Events 73
8.3 State Transition Diagram 74
8.3 Sequence Diagram 77
Chapter 9: Conclusion 79
Appendix 80

Index of Figures

Figure 1: Level-0 Use case diagram 23
Figure 2: Level-1 Use case diagram 26
Figure 3: Level-1.1 Use Case Diagram 27
Figure 4: Level-1.2 Use Case Diagram 28
Figure 5: Level-1.3 Use Case Diagram……29
Figure 6: Level-1.4 Use Case Diagram……30
Figure 7: Level-1.3.2 Use Case Diagram……31
Figure 8: Activity diagram for Sign Up……..33
Figure 9: Swim Lane for Sign Up……34
Figure 10: Activity diagram for Sign In……..35
Figure 11: Swim Lane for Sign in……36
Figure 12: Activity diagram for change profile……...37
Figure 13: Swim Lane for change profile……38
Figure 14: Activity diagram for create user……..39
Figure 15: Swim Lane for create user……..40
Figure 16: Activity diagram for remove user……..41
Figure 17: Swim Lane for remove user……42
Figure 18: Activity diagram for file operation……43
Figure 19: Swim Lane for file operation……44
Figure 20: Activity diagram for text field……45
Figure 21: Swim Lane for text field……46
Figure 22: Activity diagram for Template……..47

Figure 23: Swim Lane for Template……48
Figure 24: Activity diagram for Snippet……..49
Figure 25: Swim Lane for Snippet……..50
Figure 26: Activity diagram for Console…….51
Figure 27: Swim Lane for Console……..52
Figure 28: E-R diagram……..58
Figure 29: Data Scema……59
Figure 30: CRC diagram…….67
Figure 31: Level-0 DFD……..69
Figure 32: Level-1 DFD……..70
Figure 33: Level-1.1 DFD for authentication……..71
Figure 34: State transition diagram for user……74
Figure 35: State transition diagram for database…….75
Figure 36: State transition diagram for Admin..……..76
Figure 37: Sequence diagram……..78

Chapter 1: Introduction

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

1.1 Purpose

This document is the Software Requirement Specification (SRS) for the **Ekushey Online Code Editor** (EOCE). It contains functional, non-functional and support requirements and establishes a requirements baseline for the development of the system. The requirements contained in the SRS are independent, uniquely numbered, and organized by topic. The SRS serves as official means of communicating user requirements to the developer and provides a common reference point for both the developer team and stakeholder community. The SRS will evolve over time as users and developers work together to validate, clarify and expand its contents.

1.2 Intended Audience

This SRS is intended for several audiences including the customers as well as the project managers, designers, developers, and testers.

- The customer will use this SRS to verify that the developer team has created a product that is acceptable to the customer.
- 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 fulfill the customer's needs.
- 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 customer's documented requirements.
- The testers will use this SRS to derive test plans and test cases for each documented requirement. When portions of the software are complete, the testers will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The testers will again run their tests on the entire system when it is complete and ensure that all requirements documented in this SRS have been fulfilled.

Chapter 2: Inception

In this chapter, the Inception part of the SRS will be discussed briefly.

2.1 Introduction

Inception is the beginning phase of requirements engineering. It defines how does a software project get started and what is the scope and nature of the problem to be solved. The goal of the inception phase is to identify concurrence needs and conflict 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 the other stakeholders and the software team.

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

Stakeholder refers to any person or group who will be affected by the system directly or indirectly. Stakeholders include end-users who interact with the system and everyone else in an organization that may be affected by its installation. At inception, a list of people who will contribute input as requirements are elicited (Chapter 3) is created. The initial list will grow as stakeholders are contacted because every stakeholder will be asked: "Whom else do you think I should talk to?"

To identify the stakeholders we consulted with Programmers and asked those following questions:

- Who is paying for the project?
- Who will be using the project outcomes?
- Who gets to make the decisions about the project (if this is different from the money source)?
- Who has resources I need to get the project done?
- Whose work will my project affect? (During the project and also once the project is completed).

Concluding thoughts on Stakeholders thoughts I identified the following stakeholders for Ekushey Online Code Editor.

Programmer/user: Programmers use this system through sign in. He/she can create file, operate different types of operation. They can see their own profile.

Admin: Admin has a super administrative power all over the system. He can see the overall system performance.

2.1.2 Recognizing multiple view point

Different stakeholders achieve different benefits. 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:

Programmers view point:

- User friendly and efficient system
- Error free system
- The system can be accessed from any computer that has Internet access.
- Maintain a database
- Strong authentication

Users view point:

- User friendly
- Easy access
- Provide better services
- Provide suggestion

2.1.3 Working towards collaboration

Every stakeholder has their own requirements. We followed following steps to merge these requirements we-

- identify the common and conflicting requirements
- Categorize the requirements
- Take priority points for each requirements from stakeholders and on the basis of this voting prioritize the requirements
- Make final decision about the requirements

Common requirements:

- User friendly and efficient system
- Easy to operate
- Authentication
- Database containing detailed information about users

Conflicting requirements: We found some requirements conflicting each other. We had to trade-off between the requirements.

- Limited budget
- Availability of all requirements within the budget
- No ambiguous requirement
- Easy access
- Strong authentication and high security

Final requirements: We finalized following requirements for the system by categorizing and prioritizing the requirements.

- Error free system (Maximum 5% error may be considerable)
- Allow the users login and logout
- Restrict access to functionality of the system based upon user roles
- Allow administrators of the system to change user types and configure parameters of the system
- User friendly and efficient system
- Central database contain all result related information about every users

2.1.4 Requirements questionnaire

I set my first set of context-free questions focuses on the customer and other stakeholders, overall project goals and benefits. The questions are mentioned above. These questions helped us to identify all stakeholders, measurable benefit of the successful implementation and possible alternatives to custom software development. Next set of question helped us to gain a better understanding of problem and allows the customer to voice his or her perception about the solution. The final set of question focused on the effectiveness of the communication activity itself.

2.2 Conclusion

Inception phase helped us to establish basic understanding about the **Ekushey Online Code Editor** system establish a preliminary communication with our stakeholders.

In my project, I have established a basic understanding of the problem, the nature of the solution that is desired and the effectiveness of preliminary communication and collaboration between the stake-holders and the software team. More studies and communication will help both side (developer and client) to understand the future prospect of the project. I believe that the full functioning document will help me to define that future prospect.

Chapter 3: Elicitation

After discussing on inception part, I need to keep focus on the elicitation part. So this chapter specifies the elicitation part.

3.1 Introduction

Requirements elicitation is a part of requirement engineering that is the practice of gathering requirements from the users, customers, and other stakeholders. I have faced many problems like understanding the problems, problems of making questions for the stakeholders, problems of less communication with the stakeholders for time limitation, problems of volatility. Though it is not too easy to gather requirements within a very short time, I have surpassed these problems in an organized and systematic manner.

3.2 Eliciting requirements

I have seen Question and Answer (Q&A) approach in the previous chapter where the inception phase of requirement engineering has been described. The main task of this phase is to combine the elements of 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

3.2.1 Collaborative Requirements Gathering

Actually, I met with many stakeholders in the inception phase such as students, programmers, teachers etc. These meetings created an indecisive state for me to elicit the requirements. To solve this problem I have met with the stakeholders (who are acting a vital rule) again to elicit the requirements. A slightly different scenario from these approaches has been found. I had to complete the following steps to do it.

- They were asked about the problems they are facing with the current manual system.
- At last I selected my final requirement list.

3.2.2 Quality Function Deployment (QFD)

Quality Function Deployment (QFD) is the methods or quality management technique that translates the needs of the customer into technical requirements for the software. Ultimately the goal of QFD is to translate often subjective quality criteria into objective ones that can be quantified and measured and which can then be used to design and manufacture the product. It is a method for 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-

3.2.2.1 Normal requirements

Normal requirements consist of objectives and goals that are stated during the meeting with the customers. Normal requirements of our project are-

- Allow valid user to sign in and sign out
- Check user validity
- Help feature to explain, what they are looking for
- Efficient and user friendly
- The user interface of the system would be easy
- Security issue
- Error free activity
- Proper use of resource

3.2.2.2 Expected requirements

These requirements are implicit to the system and may be so fundamental that the customer does not explicitly state them .Their absence will be a cause for dissatisfaction.

- Maintain a databases of all items to keep the information of the users
- Security issue will be ensured
- The system would be allow the user to login based on assigned email and password
- If anyone forget password, show the recovery procedure
- Operational correction
- The user interface of the system shall be easy to use and shall make use of drop-down boxes, radio buttons, and other selectable fields wherever possible instead of fields that require the user to type in data

3.2.2.3 Exciting requirements

These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present.

- Good graphical presentation
- The user interface should provide appropriate error massage for invalid input

Because of time limitations I could not add exciting requirements in the system.

3.2.3 Usage Scenario

An Ekushey online code editor is a text editor program designed for editing code, includes powerful C programming language of computer programs by programmers.

Authentication:

The users authentication of Ekushey Online Code Editor can be separated into three parts such as sign up, sign in and sign out. Although there is a sign up system, verified users are the only active users. Sign up form includes user name, country, email address, password. When a user fills up the sign up form, a verification request is sent to his/her giving email. Verification request includes all the information of the student except the password. When users verify their email, the account becomes active. Signing in, users can get into this system through email address and password. Users can change their information. Users can only review their own activities and that requires the user to be signed in. Users can exit from the system through sign out.

New file creation:

When programmers want to write a code, firstly they have to create a new file and add an extension like .c extension. Programmers create their file through this system.

Save/Save as:

If programmers create a new file, then they can save the file as his/her preferable name and add an extension on that file like .c. By saving the file, programmers use that code next time.

Open file:

By this program, programmers open an existing file that saved their storage. Programmer creates a file, save that file and write some codes, then close the code. After a while If programmer wants to write that existing code that he/she worked previous time, he/she will reopen that file by this program.

Edit file:

Suppose, a programmer writes a code and save that. After a while, he/she realize that that code has some wrong instructions. By this program, programmers easily edit his/her files.

Cut:

Sometimes programmers cut some selected codes that are not useful on that place. By the cut option, programmers cut that selected items.

Copy:

Programmers copy specific lines of code that are frequently used or wanted through this system.

Paste:

What articles a programmers cut or copy, if they want to paste that content some specific positions, they can easily paste that articles.

Replace:

Sometimes programmers write some lines of codes and after a while he/she realize that some words want to be replaced by another words. By this option, programmers easily replace that words.

Select all:

Generally, programmers write many lines of code, create many files. Sometimes programmers want to copy all lines of codes on a specific file. By this program, programmers select all lines of codes.

Delete:

This system allows programmers to delete some specific lines of code and file that they want.

Exit:

When programmers want to quit the program, he/she can exit the program.

Templates:

Every programming language, some contents are same. Suppose we want to write a code in C language that have some header file, a main function, return type etc. By this application, programmers add templates what he/she preferred. Templates will be suggested when users create a file.

Snippets:

Snippet is a programming term for a small region of source code or text. Along the entire process of coding, programmers tend to rewrite or reuse the same pieces of code over and over again. One way to eliminate this repetitive process is to keep codes programmers frequently use close by in the form of snippet. By this application, programmers keep their codes which they frequently use by some keyword. This keyword makes easy to retrieve and access them. When programmers write codes, they feel bored to write some algorithms as well as data types that are frequently used. By this application, programmers can add their preferable snippets and set a keyword and by that keyword he/she can get that contents.

Console:

Programmers run their code (third party).

3.2.4 Elicitation Work Product

At first we have to know that the output of the Elicitation task may vary because of the dependency on size of the system or product to be built. Here, the Elicitation work product includes:

- Making a statement of our requirements for the **Ekushey Online Code Editor**
- Making a bounded statement of scope for my system.
- Making a list of users and other stakeholders who participated in the requirements elicitation.
- Making a list of requirements that are organized by function and domain constraints that apply to each.
- A set of usage scenarios that provide insight into the use of the system.
- Description of the system's technical environment.

3.3 Conclusion

Elicitation phase helped us to understand about the problems of our scopes of the system. This phase also helped us to identify the requirements, negotiate different approaches and specify a preliminary set of solution requirements in an atmosphere that is conducive to the accomplishment of the goal.

Chapter 4: Scenario Based Modeling

This chapter describes the scenario based model for **Ekushey Online Code Editor**.

4.1 Definition of Use case

A use case captures a contract that describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. In essence, a use case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A use case diagram simply describes a story using corresponding actors, who perform important role in the story and makes the story understandable for the users.

The first step in writing a use case is to define that set of "actors" that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using system.

Primary Actor:

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor:

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

4.2 Use Case Diagrams

Use case diagrams give the non-technical view of overall system.

4.2.1 System Description from Level-0 use case:

After analyzing the user story we found three actors who will directly use the system as a system operator. Primary actors are those who will play action and get a reply from the system whereas secondary actors only produce or consume information.

Following are the actors of **Ekushey Online Code Editor** –

- New user (primary)
- Verified user (primary)
- Admin (primary)

4.2.1.1 Level-0 Use Case Diagram

In this level of use case diagram describes the overall system and the actors interacting with the system. Here in my Ekushey Online Code Editor system that have three actors interacting with the system.

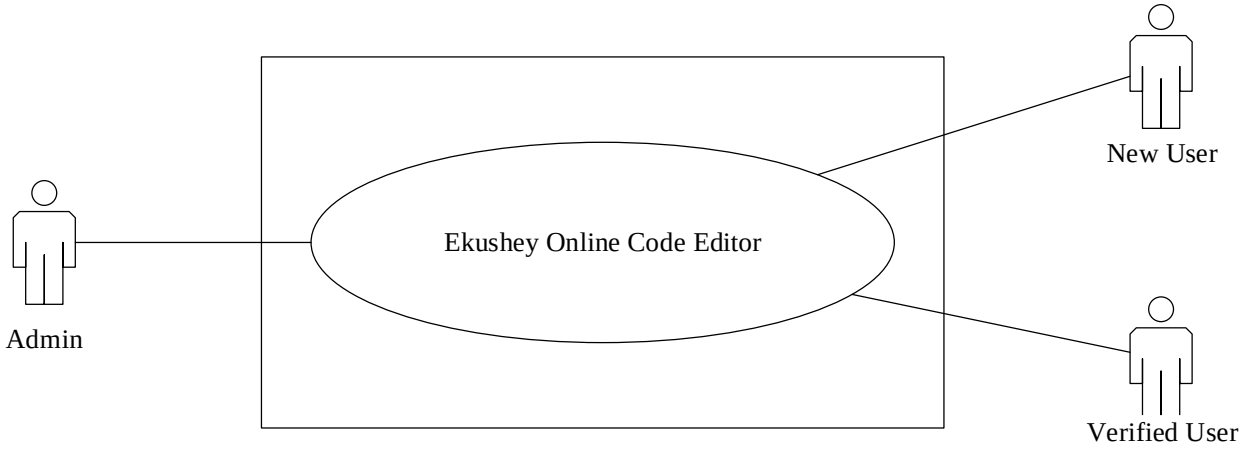


Figure 1: Level-0 Use case diagram

4.2.2 System description from level-1 use case diagram:

The actors of Ekushey Online Code Editor system have to play different actions and system will reply according to these actions –

Admin:

Action1: Enters signup.

Reply1: Please fill up the required information.

Action2: Enters the information.

Reply2: Registration successful.

Action3: Enters email and password.

Reply3: Login successful.

Action4: Send notification to user email address.

Reply4: Notification sent.

Action5: verify user.

Reply5: verified.

New User:

Action1: Enters signup.

Reply1: Please fill up the required information.

Action2: Enters the information.

Reply2: Registration successful.

Verified User:

Action1: Enters signup.

Reply1: Please fill up the required information.

Action2: Enters the information.

Reply2: Registration successful.

Action3: Enters email and password.

Reply3: Login successful.

Action4: Create file.

Reply4: File created.

Action5: Write Codes.

Reply5: Provide proper suggestion.

Action6: type word.

Reply6: show corresponding text.

Action7: Open new file

Reply7: Provide template.

Action8: Open file.

Reply8: Provide different types of operation.

4.2.2.1 Level-1 Use Case Diagram

Level-1 use case where total system is divided into its subsystems which elaborately described in section 4.2.2.

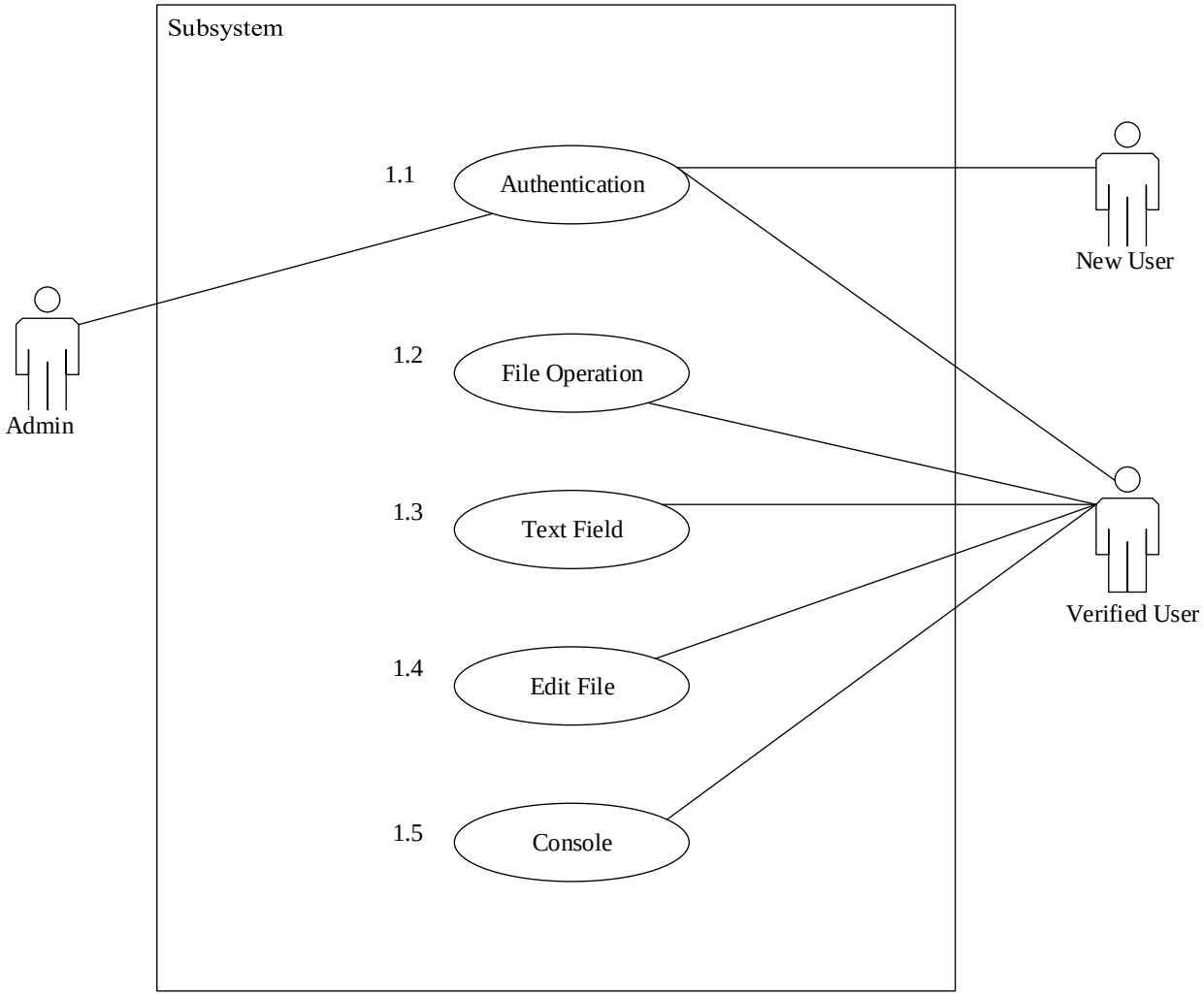


Figure 2: Level-1 Use case diagram

4.2.2.1.1 Subsystems of Level-1.1 Use Case Diagram

Subsystems of subsystem 1.1 of level-1 use case diagram. System description described in the section 4.2.2.

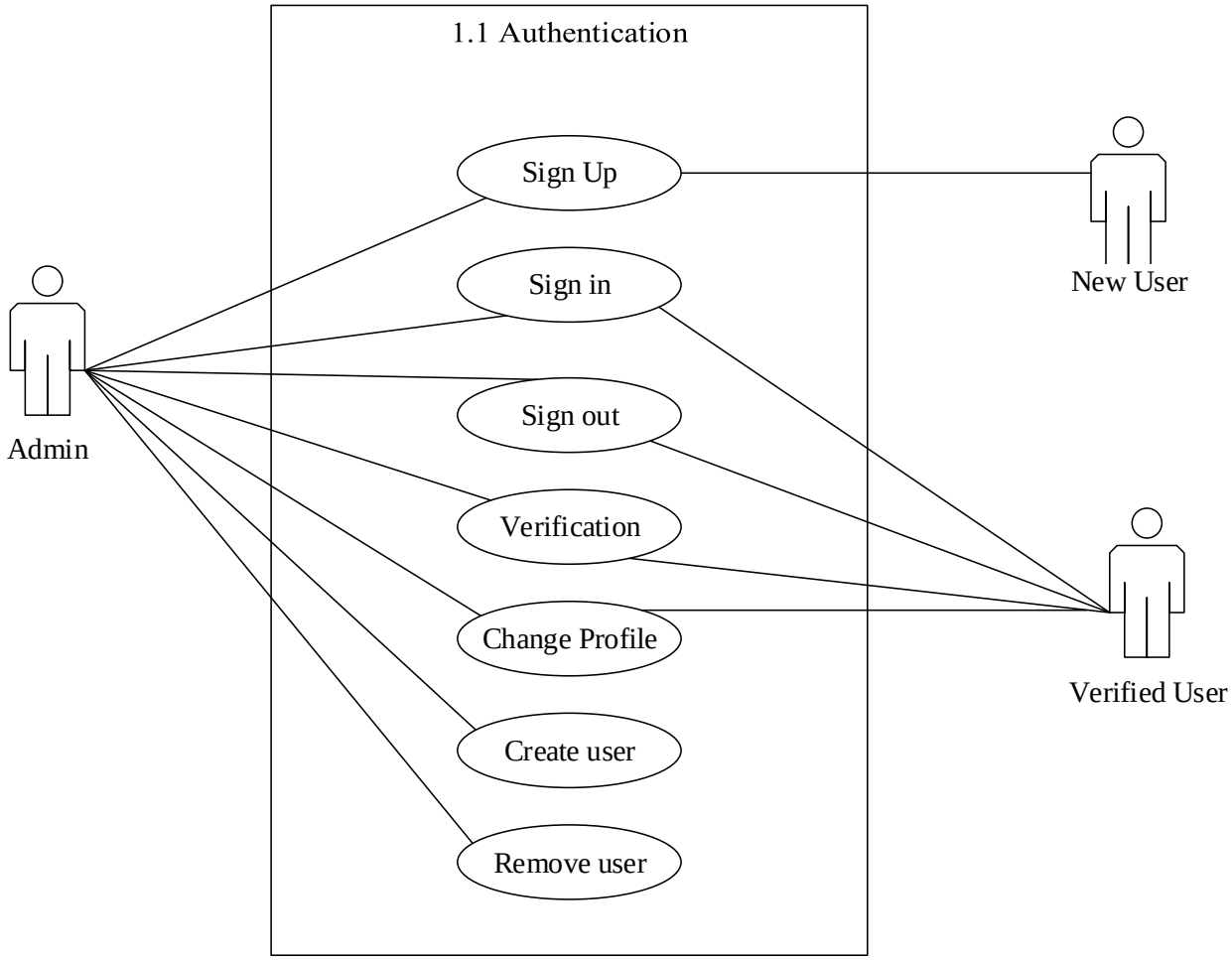


Figure 3: Level-1.1 Use Case Diagram

4.2.2.1.2 Subsystems of Level-1.2 Use Case Diagram

Subsystems of subsystem 1.2 of level-1 use case diagram. System description described in the section 4.2.2.

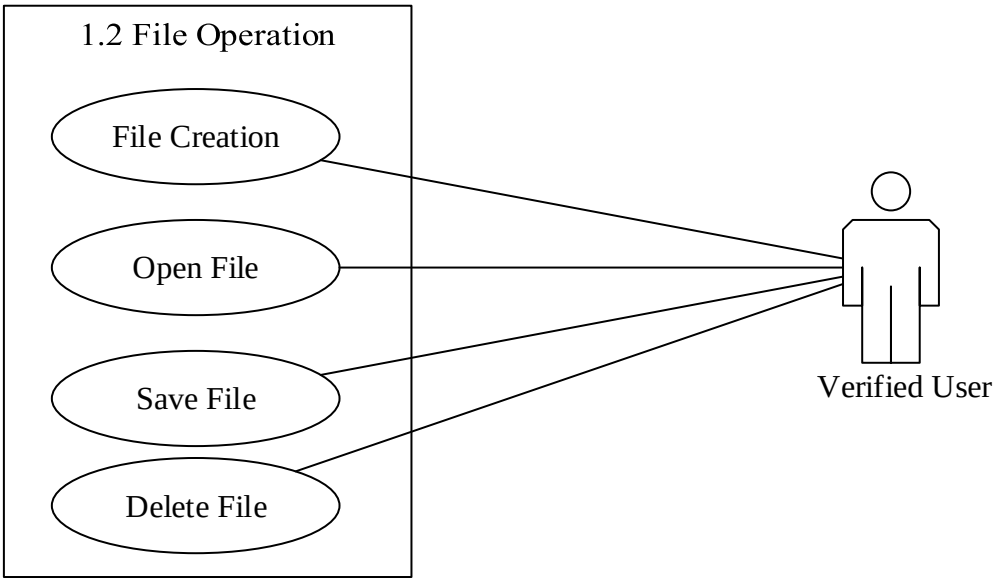


Figure 4: Level-1.2 Use Case Diagram

4.2.2.1.3 Subsystems of Level-1.3 Use Case Diagram

Subsystems of subsystem 1.3 of level-1 use case diagram. System description described in the section 4.2.2.

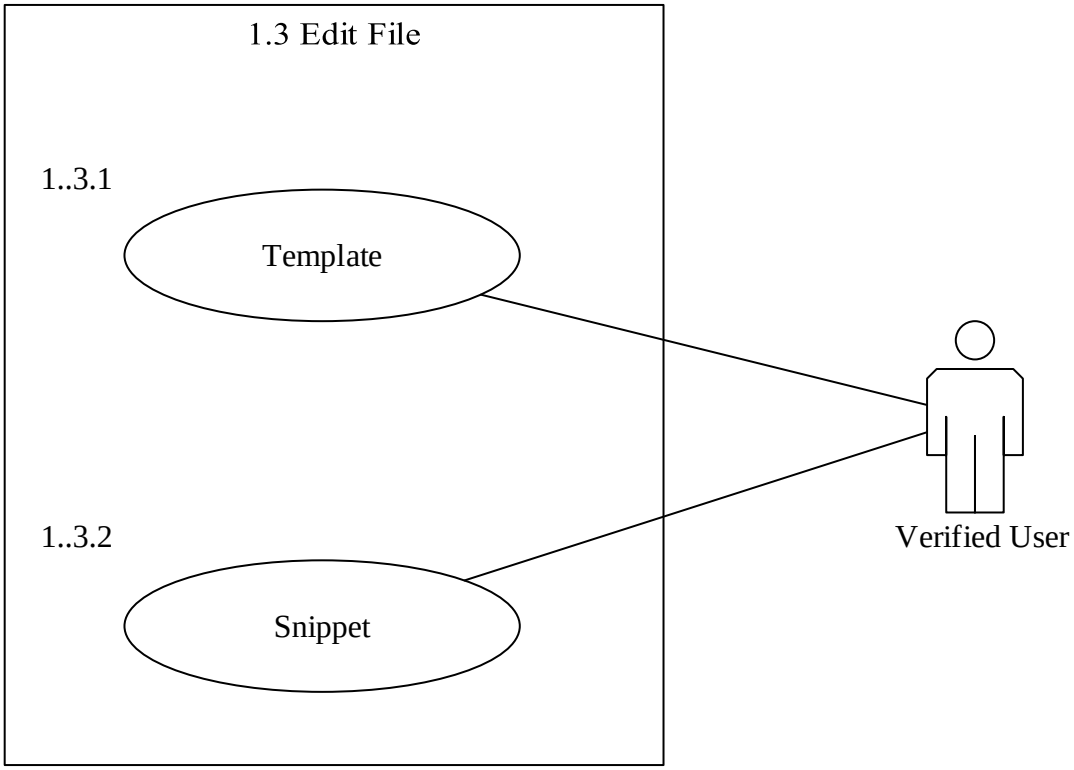


Figure 5: Level-1.3 Use Case Diagram

4.2.2.1.4 Subsystems of Level-1.4 Use Case Diagram

Subsystems of subsystem 1.4 of level-1 use case diagram. System description described in the section 4.2.2.

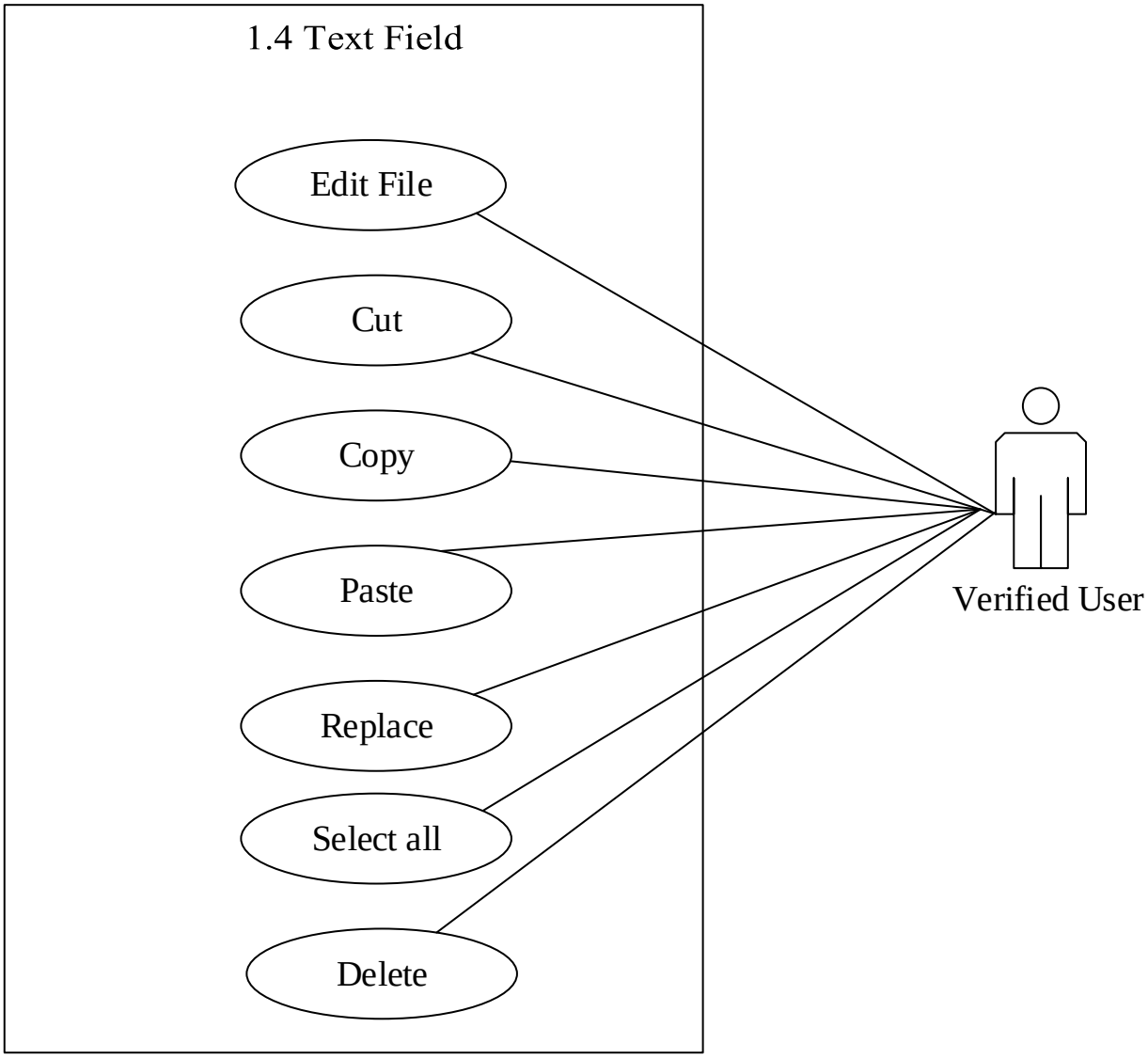


Figure 6: Level-1.4 Use Case Diagram

4.2.2.2.1 Subsystems of Level-1.3.2 Use Case Diagram

Subsystems of subsystem 1.3.3 of level-1.3 use case diagram. System description described in the section 4.2.2.

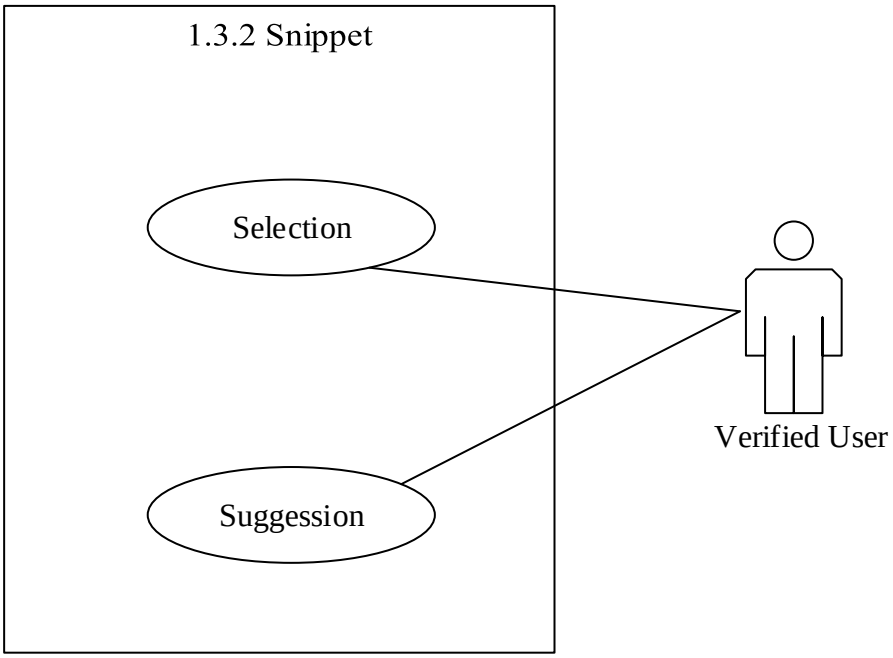


Figure 7: Level-1.3.2 Use Case Diagram

4.3 Activity & Swim lane Diagrams

Activity diagram shows the technical view of the system for every use case from which we can understanst how the system actualy works and how the actors interact with the system.

Swim lane diagram of a specific activity diagram shows the responisibilites of each actor dividing them into lanes. From this diagram we can improve our understanding about how the system works and which actors play what role.

Following is the Activity diagram of use case Sign Up.

Use Case: Sign up

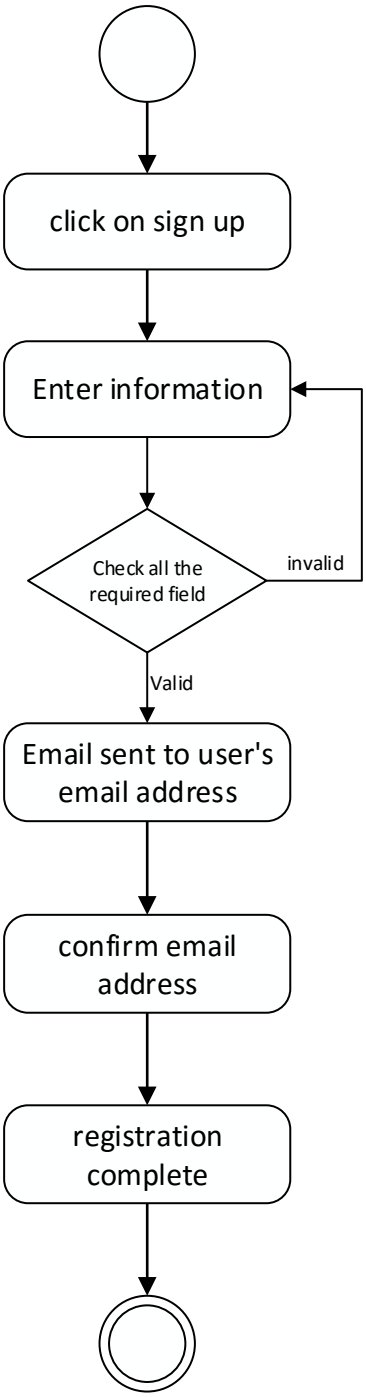


Figure 8:Activity diagram for Sign Up

Following is the swim lane of use case Sign Up.

Swimlane Diagram: Sign Up

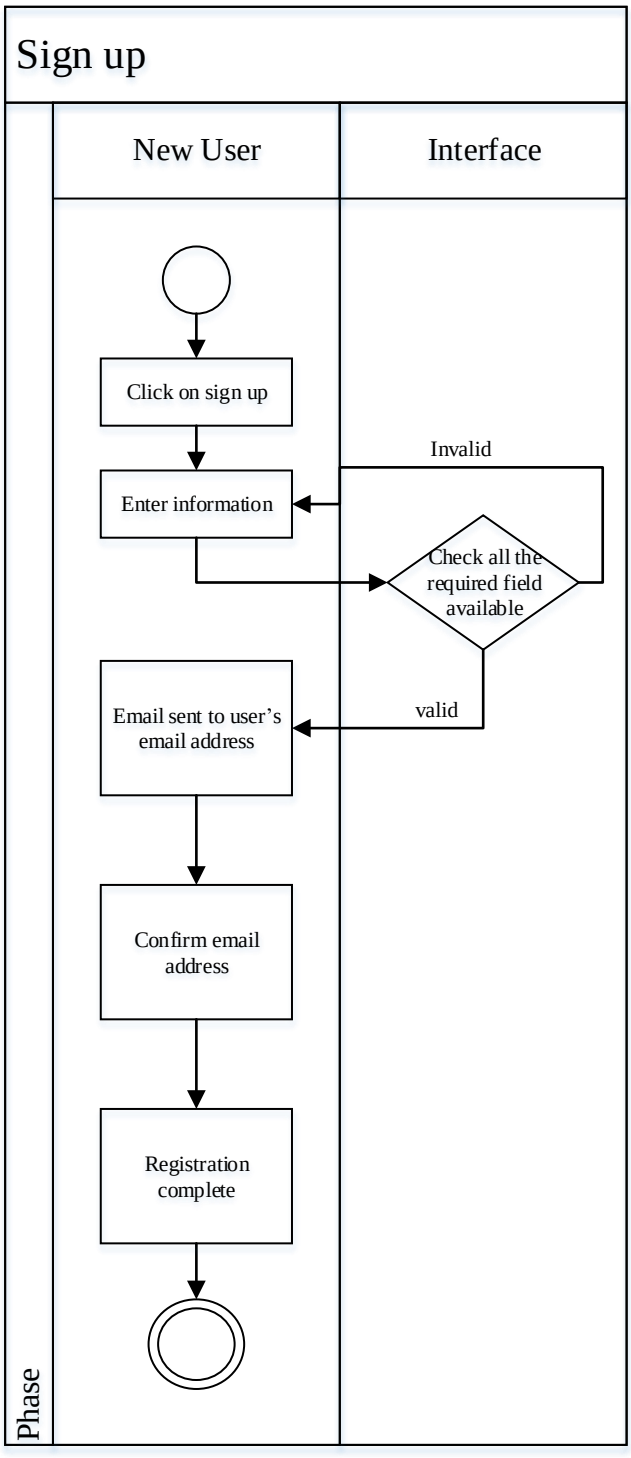


Figure 9: Swim Lane for Sign Up

Following is the activity diagram of use case Sign In.

Use Case: Sign in

Activity Diagram:

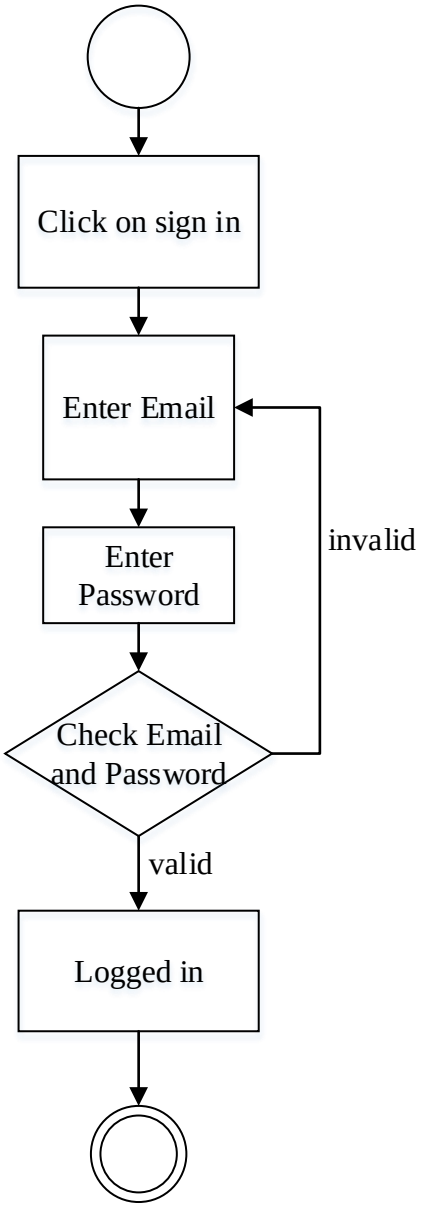


Figure 10: Activity for Sign In

Following is the swim lane diagram of use case Sign In.

Swimlane Diagram: Sign In

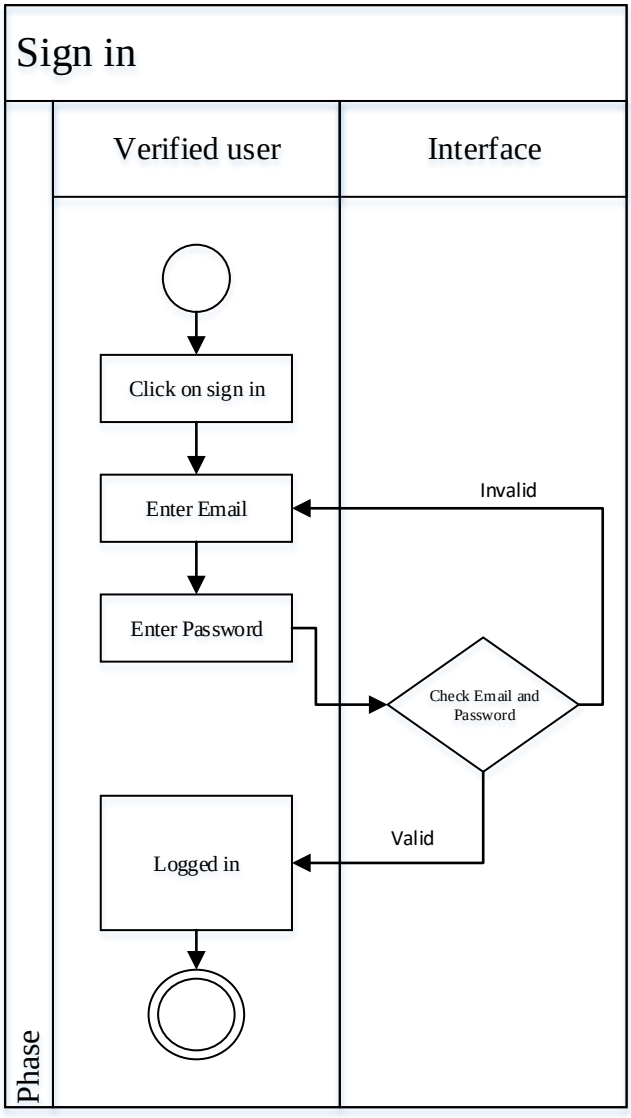


Figure 11: Swim Lane Sign In

Following is the activity diagram of use case Change Profile.

Use Case: Change Profile

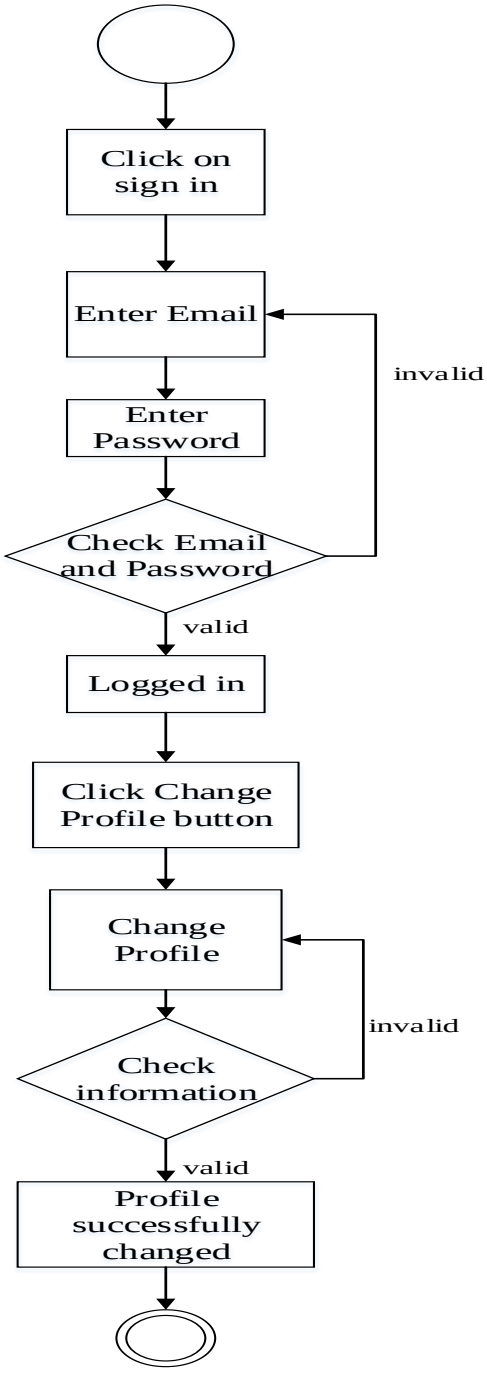


Figure 12: Activity diagram change Profile

Following is the swim lane diagram of use case Change Profile.

Swim lane: Change Profile

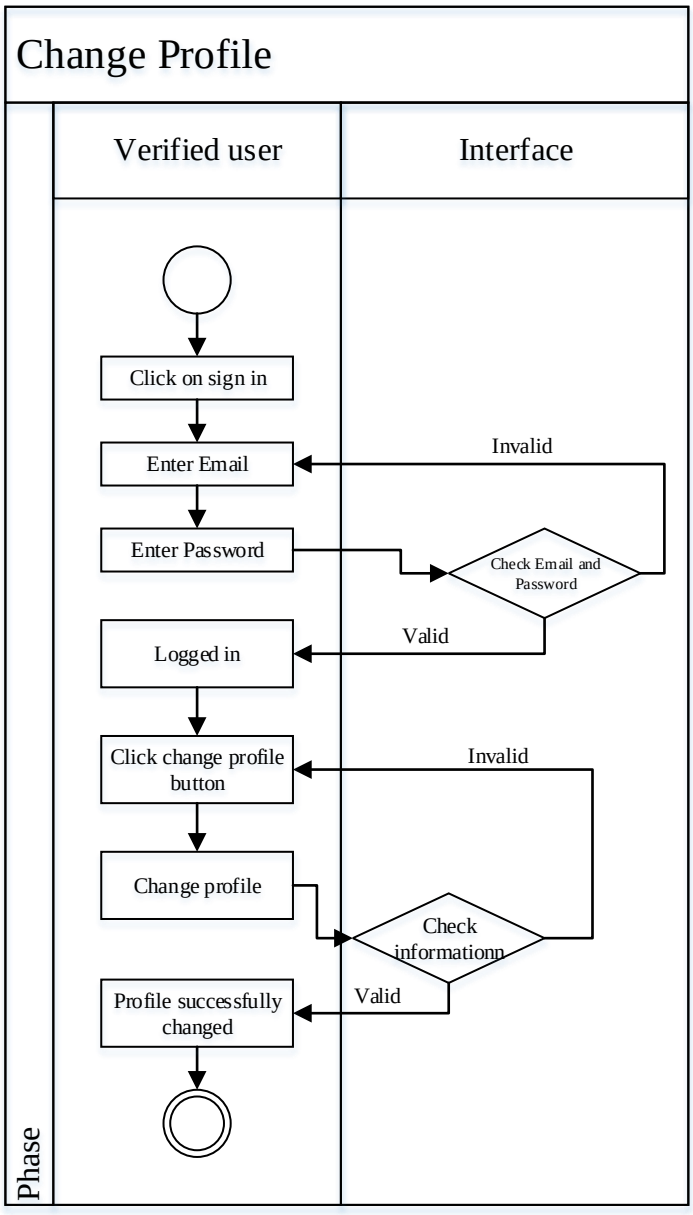


Figure 13: Swim Lane change Profile

Following is the activity diagram of use case Create user.

Use Case: Create user

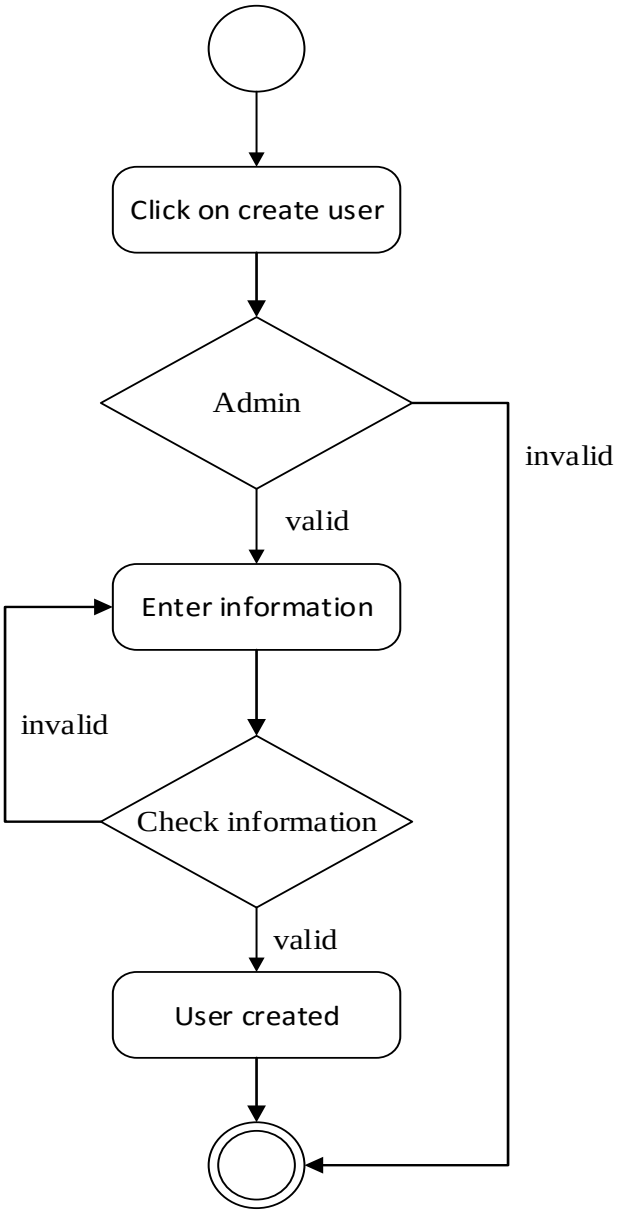


Figure 14: Activity diagram create user

Following is the swim lane diagram of use case Create User.

Swim lane: Create User

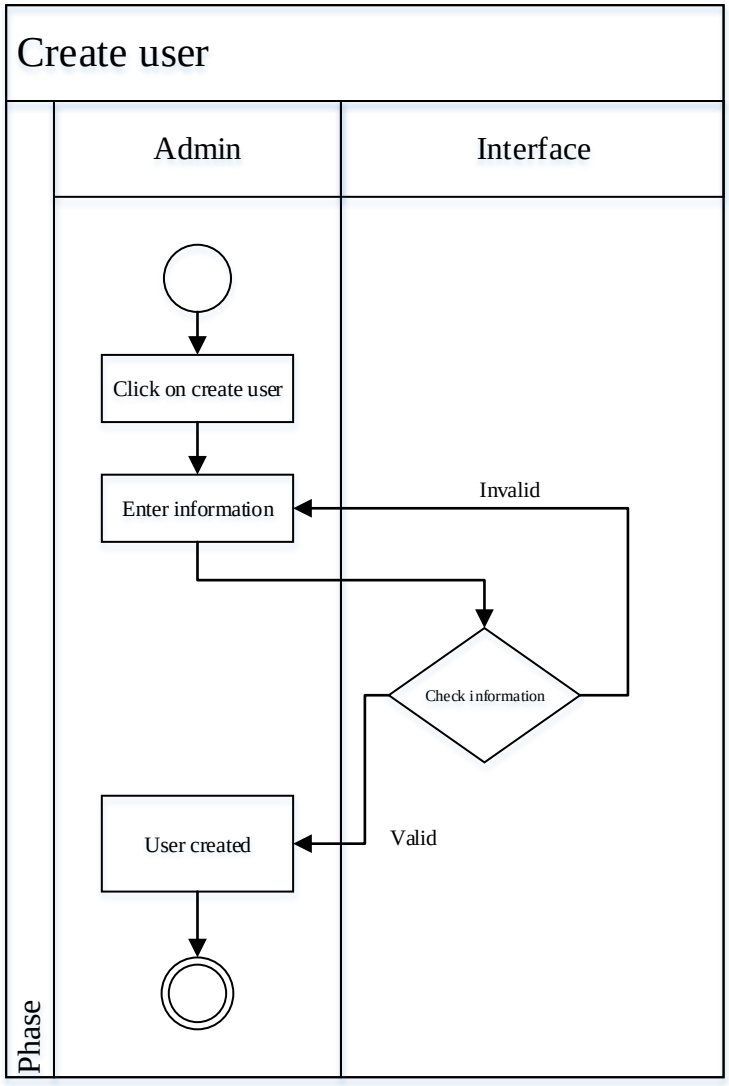


Figure 15: Swim Lane create user

Following is the activity diagram of use case Remove user.

Use Case: Remove user

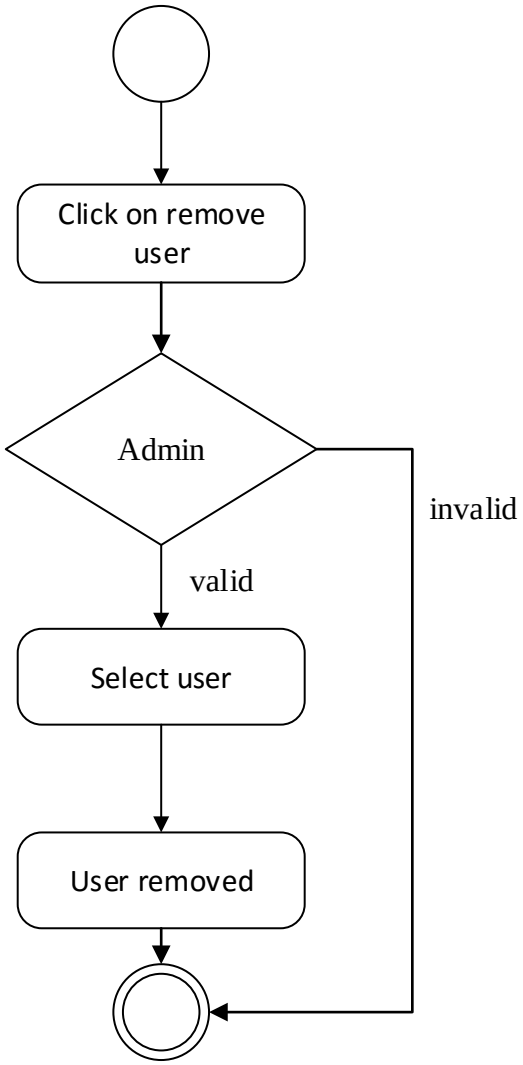


Figure 16: Activity diagram remove user

Following is the swim lane diagram of use case Remove User.

Swim lane: Remove User

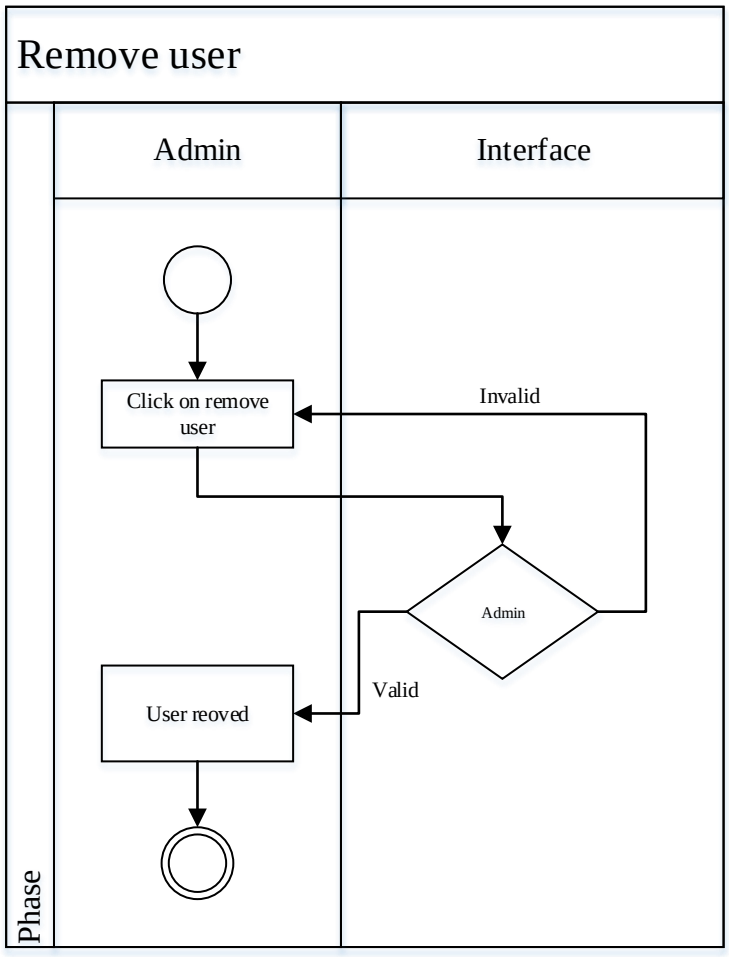


Figure 17: Swim Lane remove user

Following is the activity diagram of use case File Operation.

Use case: File Operation

Activity diagram:

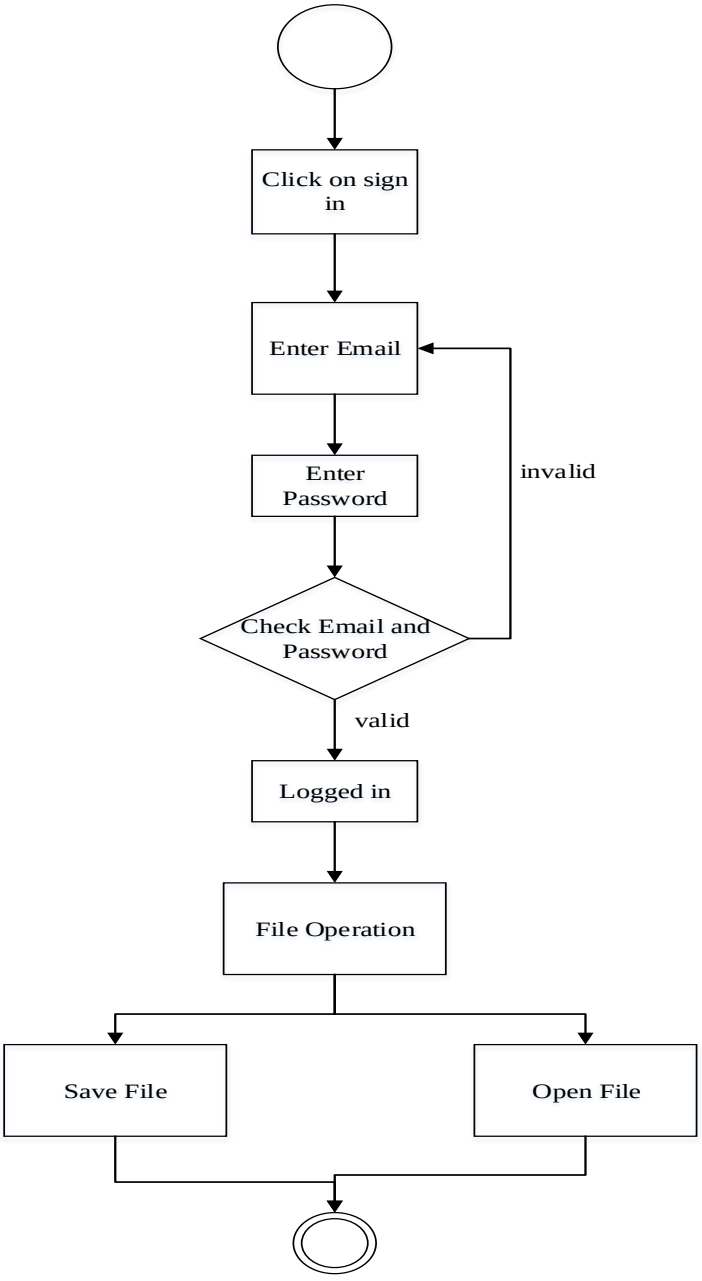


Figure 18: Activity diagram File Operation

Following is the swim lane diagram of use case File Operation.

Swimlane: File Operation

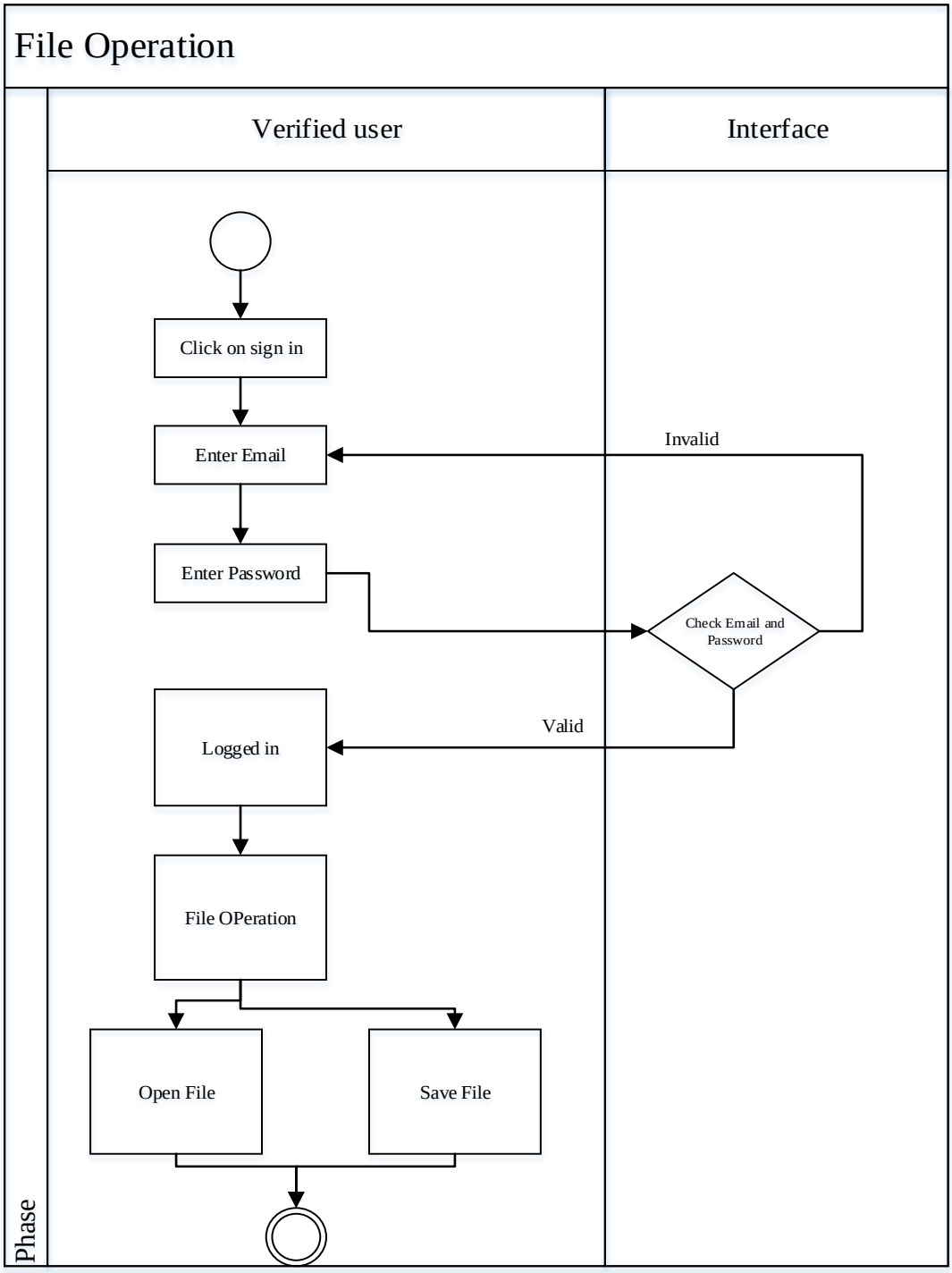


Figure 19: Swim Lane File Operation

Following is the activity diagram of use case Text Field.

Use case: Text Field

Activity diagram:

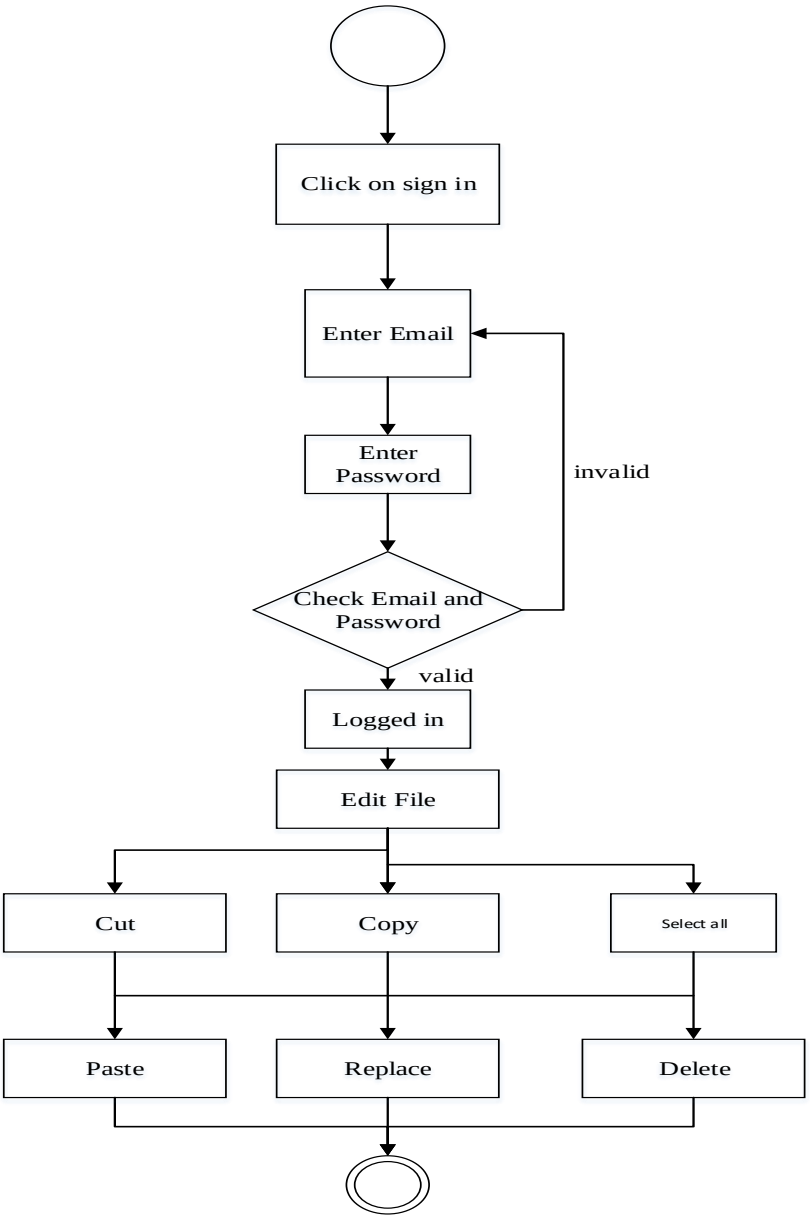


Figure 20: Activity diagram Text Field

Following is the swim lane diagram of use case Text Field.

Swim lane: Text Field

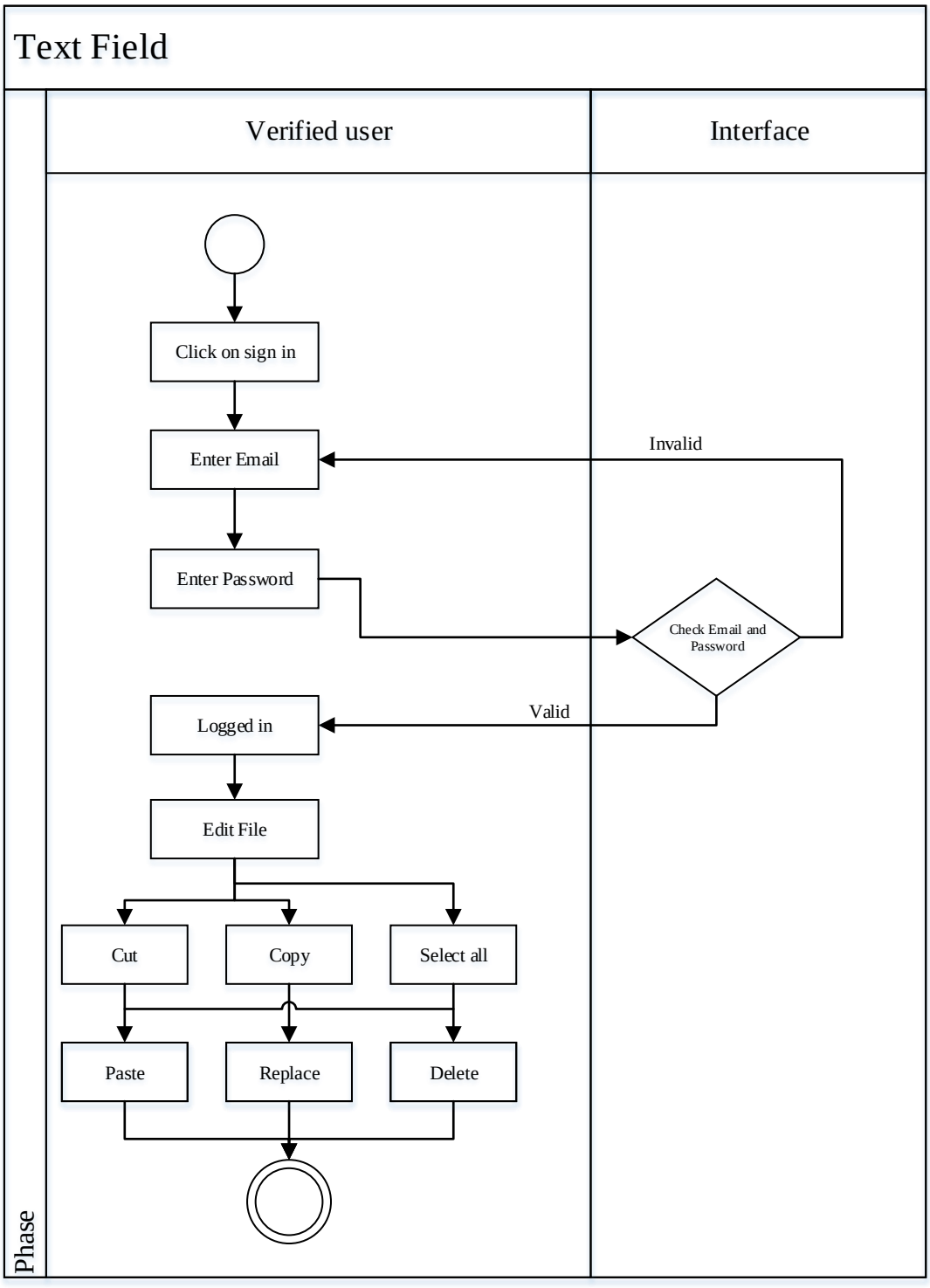


Figure 21: Swim Lane Text Field

Following is the activity diagram of use case Template.

Use case: Template

Activity diagram:

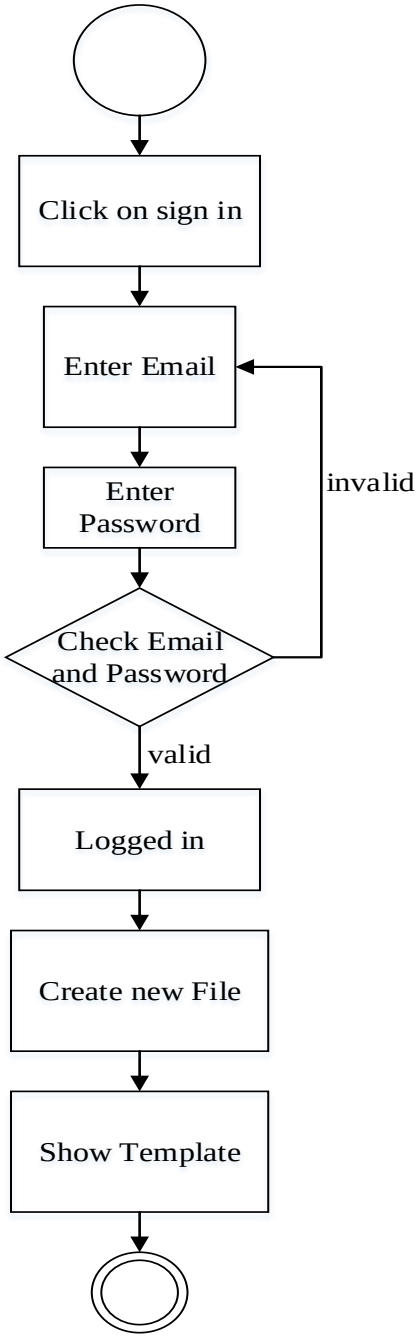


Figure 22: Activity diagram Template

Following is the swim lane diagram of use case Template.

Swim lane: Template

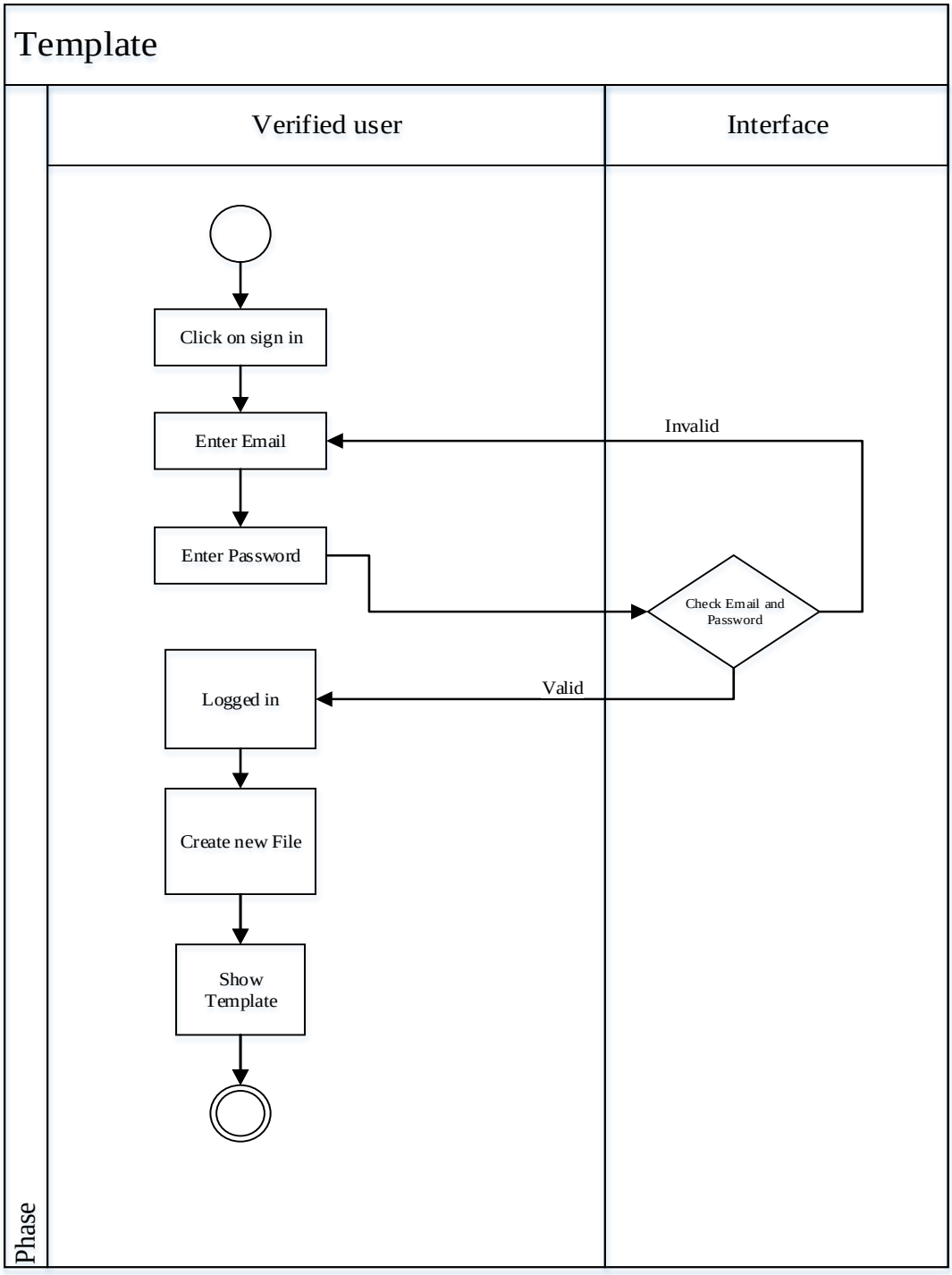


Figure 23: Swim Lane Template

Following is the activity diagram of use case Snippet.

Use case: Snippet

Activity diagram:

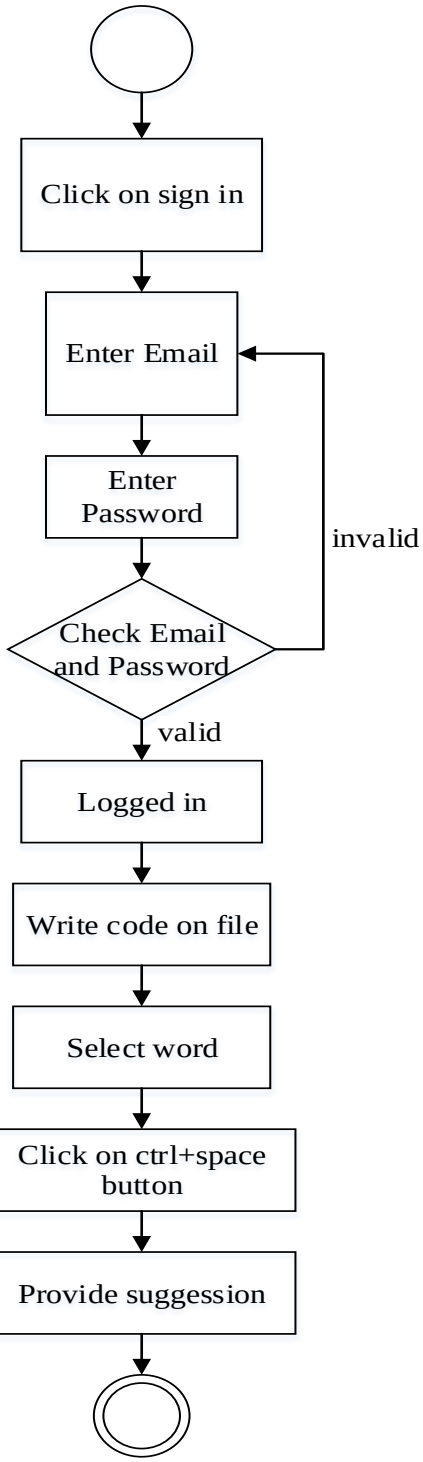


Figure 24: Activity diagram Snippet

Following is the swim lane diagram of use case Snippet.

Swim lane: Snippet

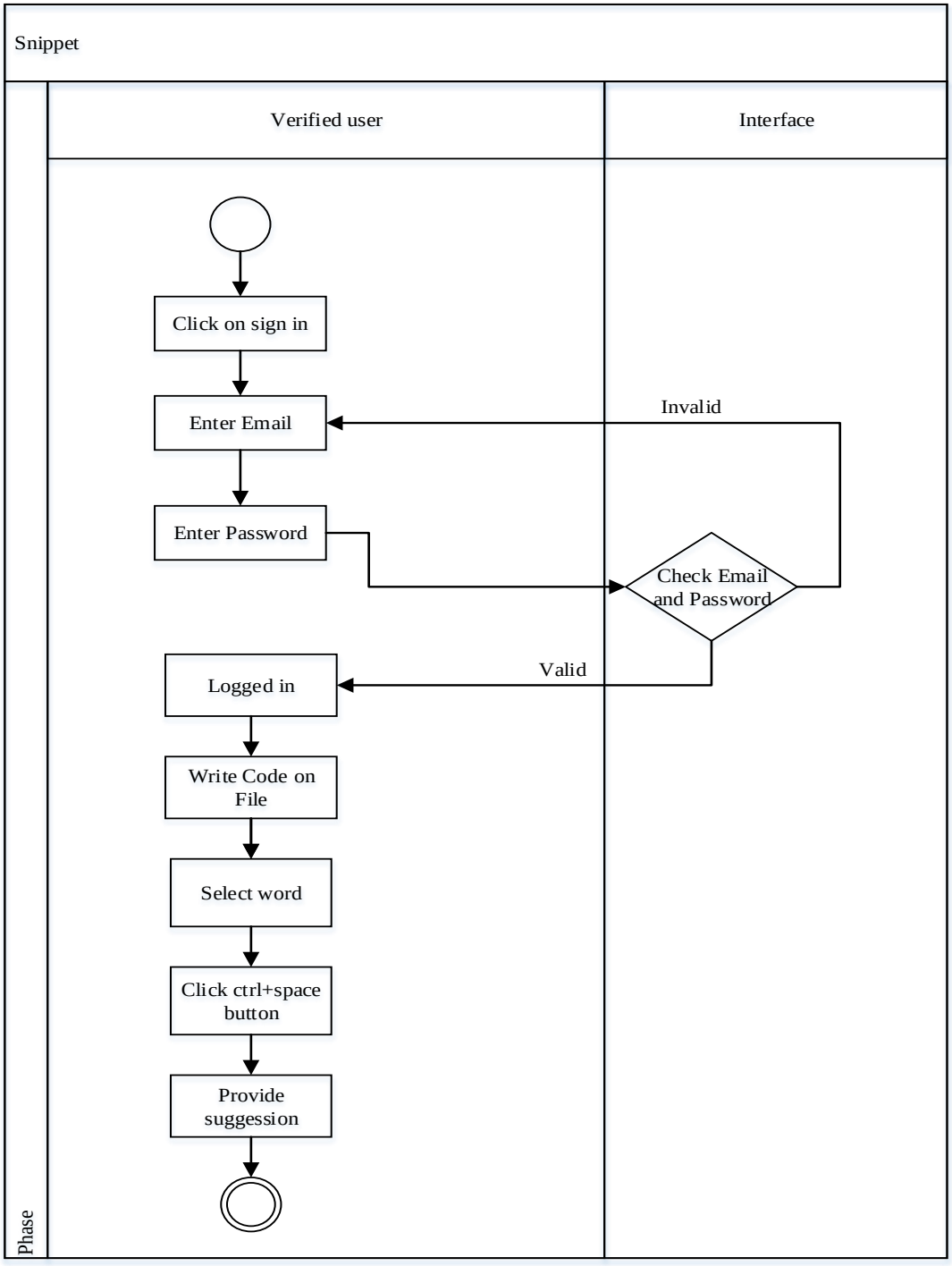


Figure 25: Swim Lane Snippet

Following is the activity diagram of use case Console.

Use case: Console

Activity diagram:

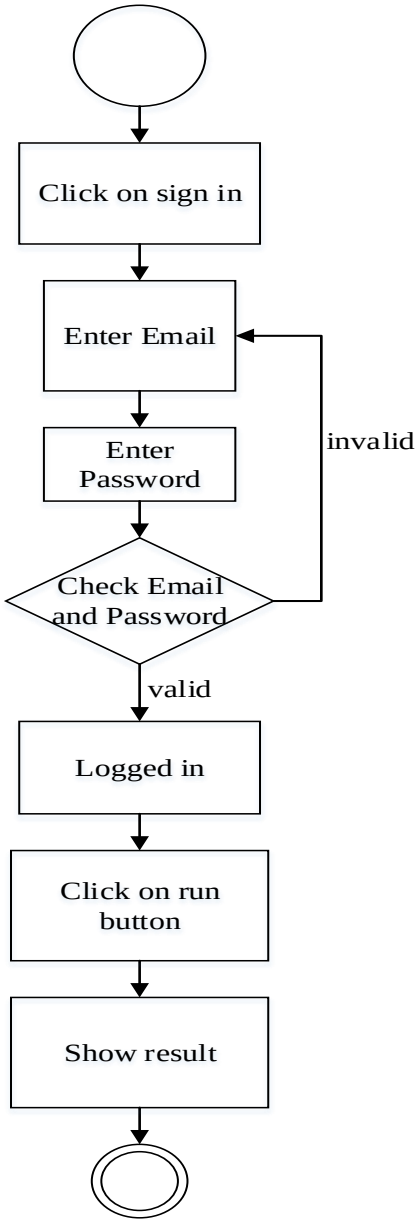


Figure 26: Activity diagram Console

Following is the swim lane diagram of use case Console.

Swim lane: Console

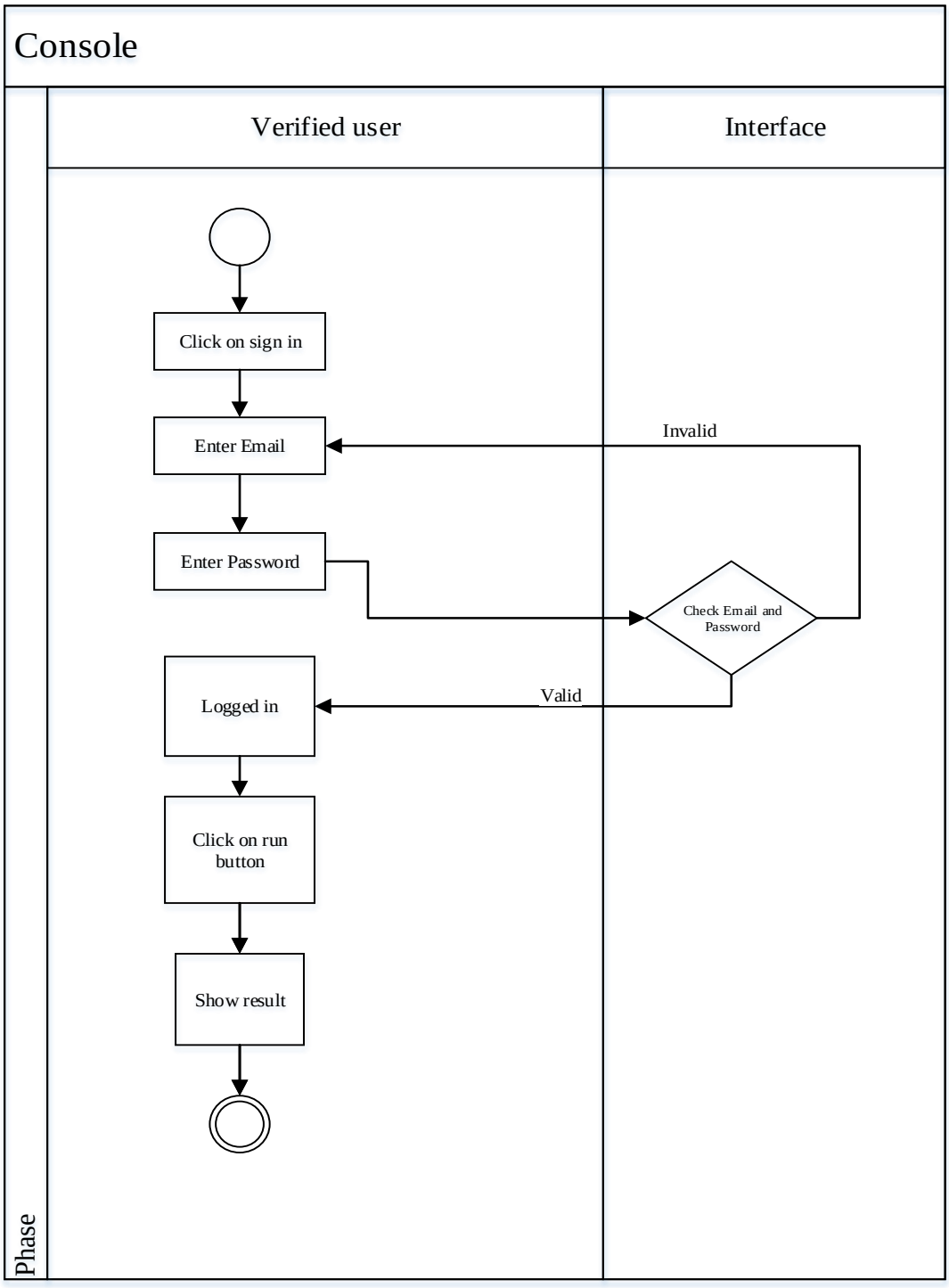


Figure 27: Swim Lane Console

Chapter 5: Data Model

In this chapter we will discuss about the data models of our system.

5.1 Data Modeling Concept

If software requirements include the need to create, extend or interface with a data base or if complex data structures must be constructed and manipulated, the software team choose to create data model as part of overall requirements modeling. The entity-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information that how the data objects are entered, stored, transformed and produced within the system.

5.2 Data Objects

A data object is representation of composite information that must be understood by software. Here, composite information means 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 Grammatical parsing (Noun identify)

We identified all the nouns whether they are in problem space or in solution space from our usage scenario and categorized them according to their attributes.

Nouns	**Problem/Solution Space (P/S)**	**Attributes**
1. Editor	P	
2. C language	P	
3. Programmer/user	S	9,10,11,12
4. Authentication	P	
5. Sign up	P	
6. Sign in	P	
7. Sign out	P	
8. Verified user	S	
9. User Name	S	
10. Country	S	
11. Email	S	
12. Password	S	
13. Admin	S	9,10,11,12

14. Review	S	
15. File	S	16,17
16. File name	S	
17. File extension	S	
18. Storage	P	
19. Instruction	P	
20. Item	P	
21. Line of code	P	
22. Articles	S	
23. Position	P	
24. Word	P	
25. Error	P	
26. Content	S	
27. Header	S	
28. Main function	S	
29. Return type	S	
30. Template	S	22,26,27,28,29
31. Snippet	S	32,33
32. Text	S	
33. Keyword	S	
34. Algorithm	P	
35. Data type	P	

5.2.2 Identify Data Objects

Nouns having attributes are selected as data object. Those who doesn't have any attributes have covered under the data objects.

Data Object: User

Attributes:

- User Name
- Password
- Country
- Email

Data Object: Admin

Attributes:

- User Name
- Password
- Email

Data Object: File

Attributes:

- File Name
- File extension

Data Object: Template

Attributes:

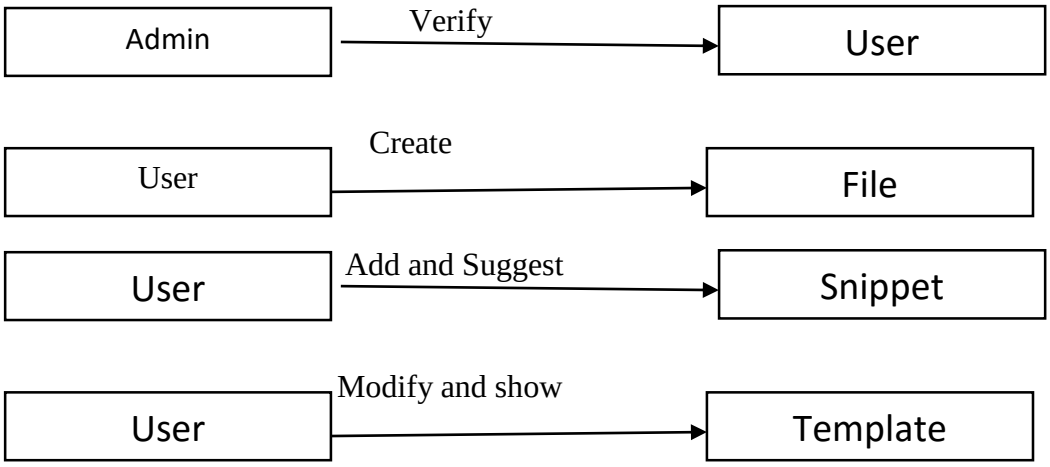
- Articles

Data Object: Snippet

Attributes:

- Keyword
- Text

5.2.3 Data Object Relation



5.2.4 E-R Diagram

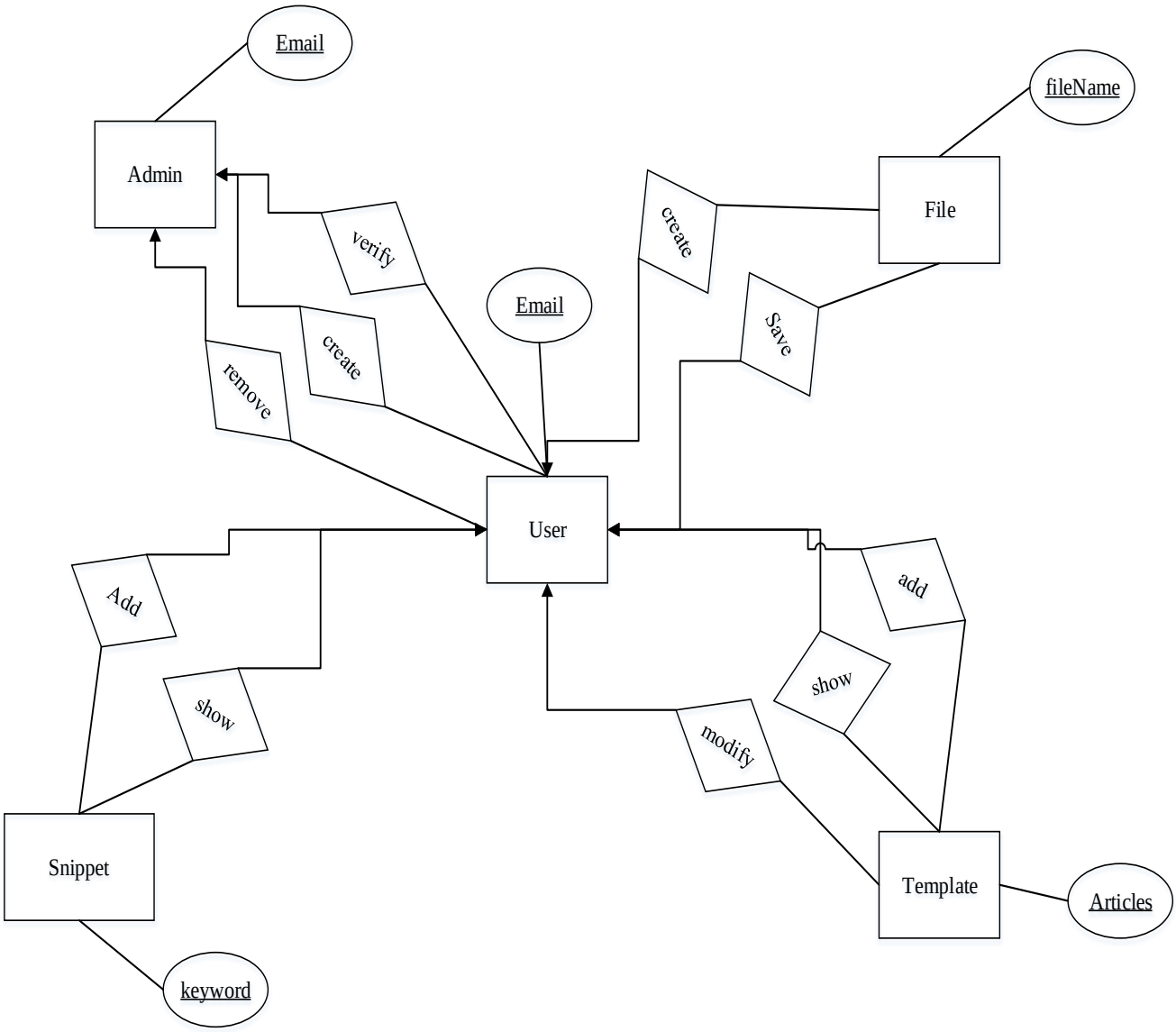


Figure 28: E-R Diagram

5.3.5 Schema Form (Tables)

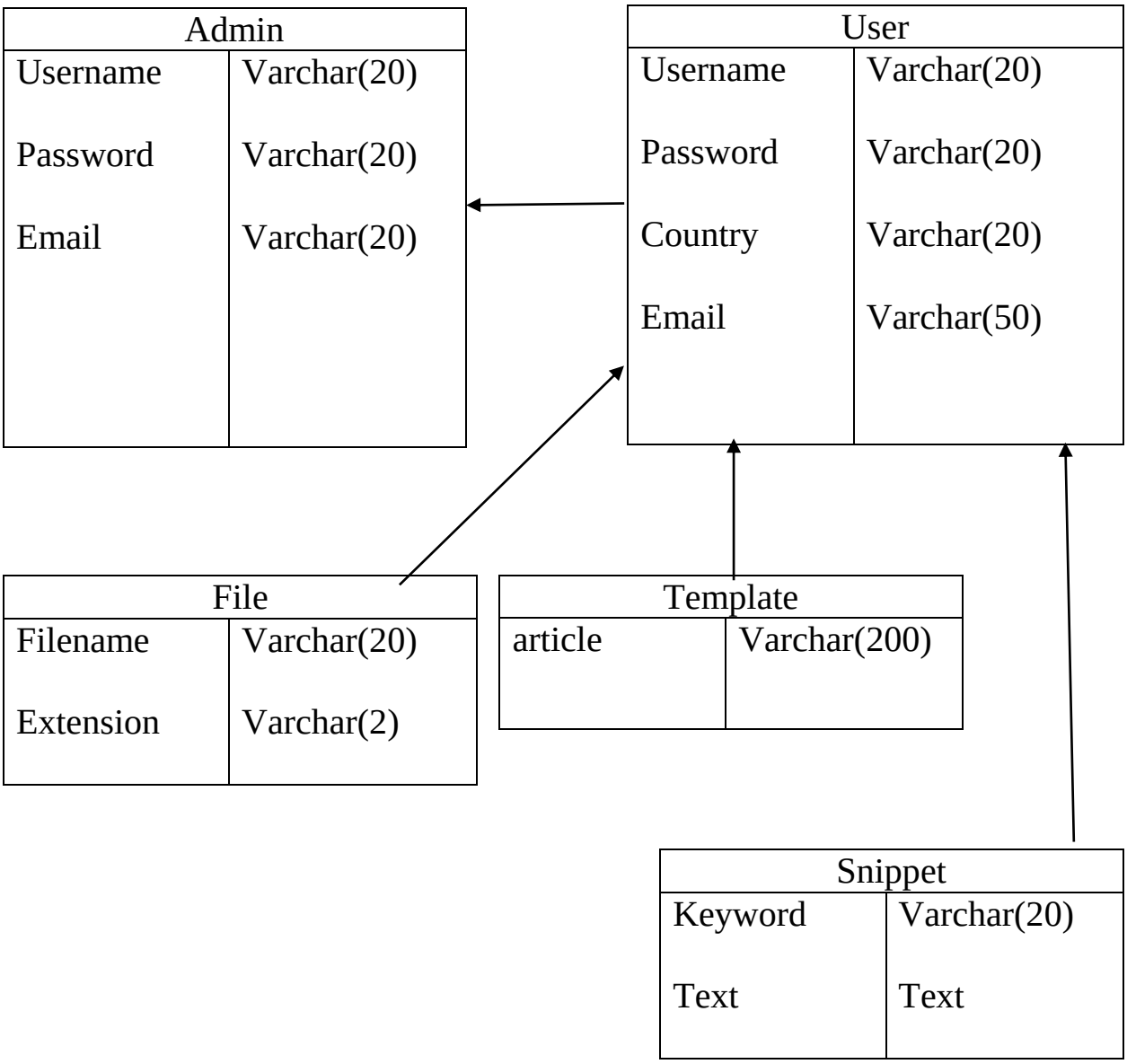


Figure 29: Data Schema

5.3.5 Data Tables

Table 1:

Admin					
Field name	Type	Null	Key	Length	Description
A_Name	Varchar2	No	-	50	Admin Name
Password	Varchar2	No	-	50	-
E_mail	Varchar2	No	Key	50	e-mail address

Table 2:

User					
Field name	Type	Null	Key	Length	Description
U_Name	Varchar2	No	-	50	Admin Name
Password	Varchar2	No	-	50	-
E_mail	Varchar2	No	Key	50	e-mail address
Country	Varchar2	No	-	50	Country name

Table 3:

Template					
Field name	Type	Null	Key	Length	Description
Content	Varchar2	No	Key	2000	User's template

Table 4:

Snippet					
Field name	Type	Null	Key	Length	Description
Keyword	Varchar2	No	Key	50	Keyword
Text	Varchar2	No	-	2000	contents

Chapter 6: Class Based Model

This Chapter is intended to describe class based modeling of post examination control system.

6.1 Class Based Modeling Concept

Class-based modeling represents the objects that the system will manipulate, the operations that will applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

6.1.1 Identifying Analysis Class

To identify our analysis class we firstly grammatically parsed all the nouns and then categorized them according to general classification and selection criteria.

Following are the steps we used to analysis the classes for our system.

Step-1: Grammatical parsing (noun identifying) and categorizing using general classification:

Table 5:

External entities	Database
Things	File, text Field, Edit
Occurrence or events	-
Roles	Admin, User
Organizational units	-
Places	-
Structure	-

Step-2: Selection Criteria:

1. Retained information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Table 6:

Potential class	**Characteristic number that applies**	**Remarks**
Database	6	accepted
Admin	1,3,6	Accepted
User	1,3,6	Accepted

Here database holds only one selection criteria but as this is important for our system we took this as our class.

Nouns that hold essential requirement we took them essential for our system and include them in our analysis classes.

Preliminary Classes:

1. Database
2. Admin
3. User

Attributes Selection:

Table 7:

Class Name	**Attribute**
Database	All attributes
Admin	username, Email, Password, Country
User	username, Email, Password

Step-3: Method Identification:

Verb Detection:

Table 8:

No	Verb	Remarks
1	Designed	Out of scope
2	Separated	Out of scope
3	Enter information	Yes
4	Sent request	Yes
5	Verify	Yes
6	Change profile	Yes
7	Review	Yes
8	Exit	Yes
9	Create file	Yes
10	Save File	Yes
11	Open file	Yes
12	Write code	Out of scope
13	Edit file	Yes
14	Cut	Yes
15	Copy	Yes
16	Paste	Yes
17	Replace	Yes
18	Select all	Yes
19	Want to	Out of scope
20	Delete	Yes
21	Add template	Yes
22	Rewrite	Out of scope
23	Keep code	Out of scope
24	Add snippet	Yes
25	Run code	Yes

Class Name	**Methods**
Admin	Sending request() Verify()
User	CreateFile() OpenFile() SaveFile() Cut() Copy() SelectAll() Paste() Replace() Delete() AddTemplate() AddSnippet()
Database	Create() insert(), update() delete() view()

Class Card:

Admin	
Attributes	**Methods**
username Email Password	SendingRequest() Verify()
Responsibilities	**Collaborative Class**
Providing access to system	database

User	
Attributes	**Methods**
UserName Email Password Country	CreateFile() OpenFile() SaveFile() Cut() Copy() SelectAll() Paste() Replace() Delete() AddTemplate() AddSnippet()
Responsibilities	**Collaborative Class**
Providing access to system	database

Database	
Attributes	**Methods**
All attributes	Create() insert(), update() delete() view()
Responsibilities	**Collaborative Class**
Creating new database	None
Inserting data	None

Updating data	None
Deleting data	None
View data	None

6.1.2 Class Responsibility Collaboration (CRC)

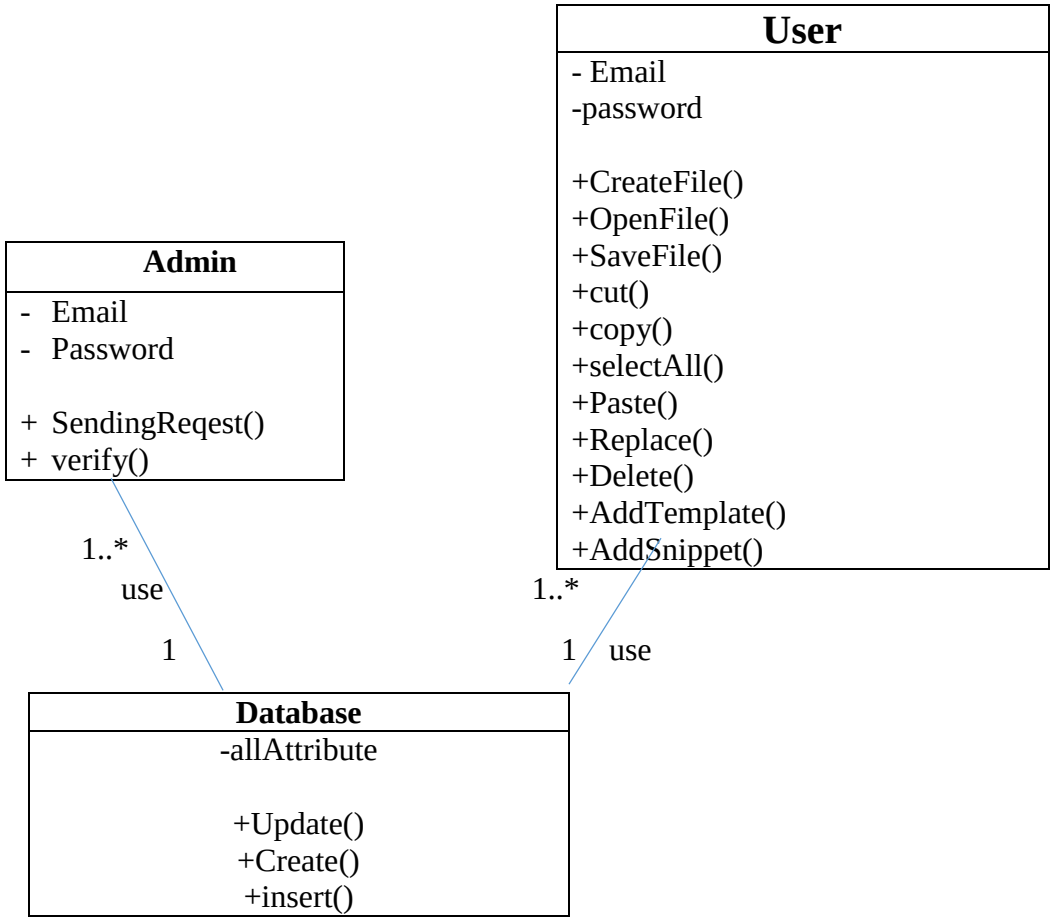


Figure 30: CRC diagram

Chapter 7: Flow Oriented Model

This chapter focuses on the flow oriented modeling.

7.1 Introduction

Although data flow-oriented modeling is perceived as an outdated technique by some software engineers, it continues to be one of the most widely used requirements analysis notations in use today. It provides additional insight into system requirements and flow.

7.2 Data Flow Diagram

The DFD takes an input-process-output view of a system. In the figures, data objects are represented by labeled arrows and transformations are represented by circles.

Level-0 DFD: Level-0 DFD describing the overall system's input and output data.

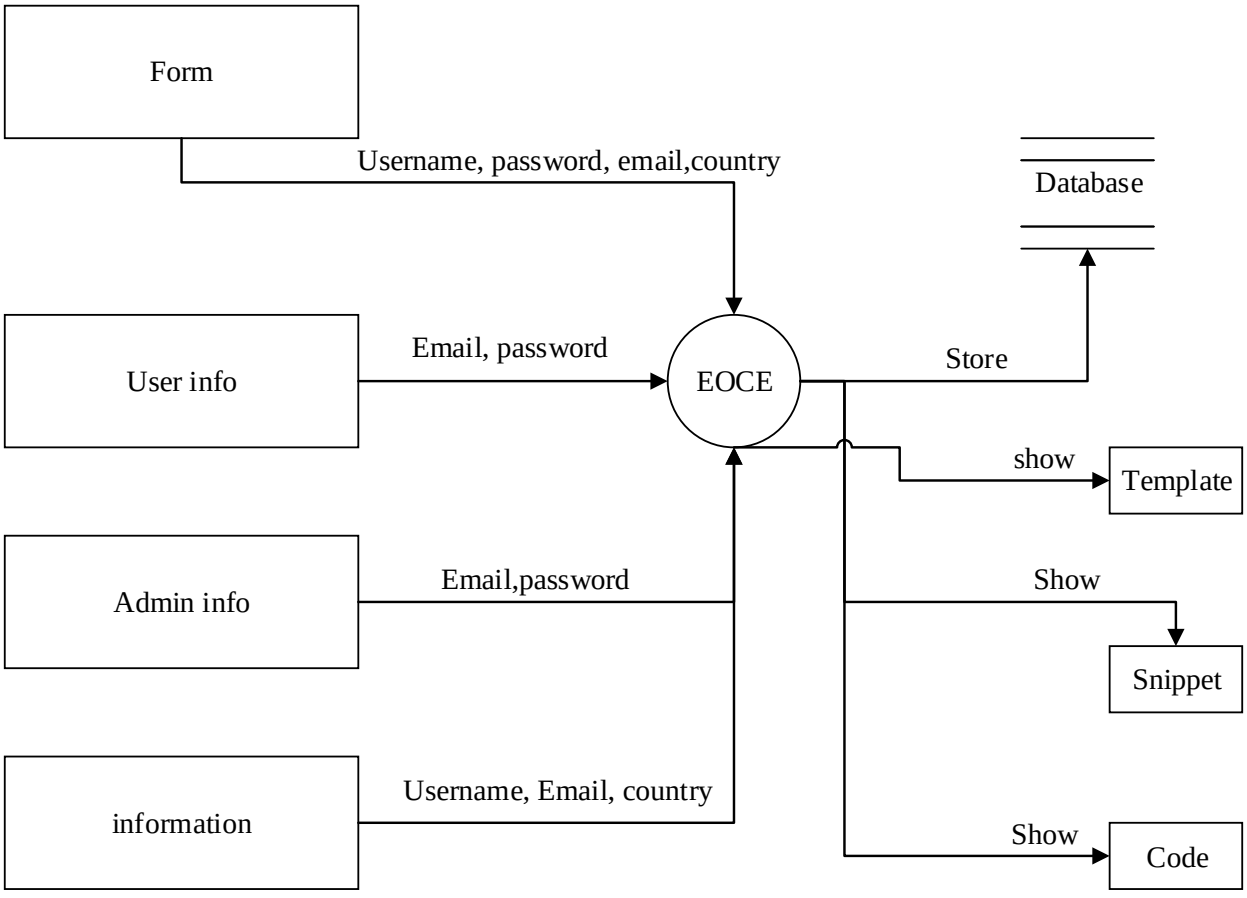


Figure 31: Level-0 DFD

Level-1 DFD: This level-1 DFD derived from level-0 DFD.

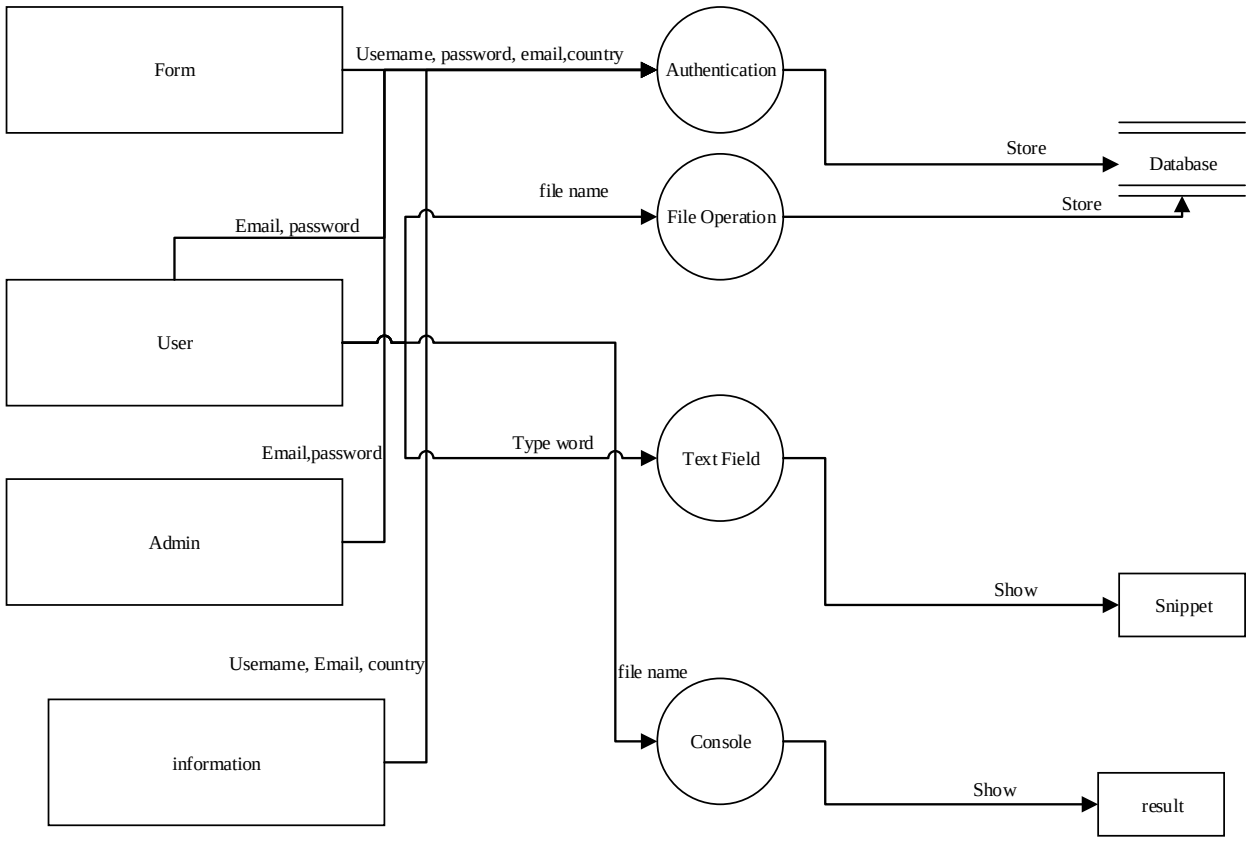


Figure 32: Level-1 DFD

Level-2.1 DFD: Every process of data flow can be described into many sub process. And thus we got level-2.1 DFD from the process Authentication in level-1 DFD.

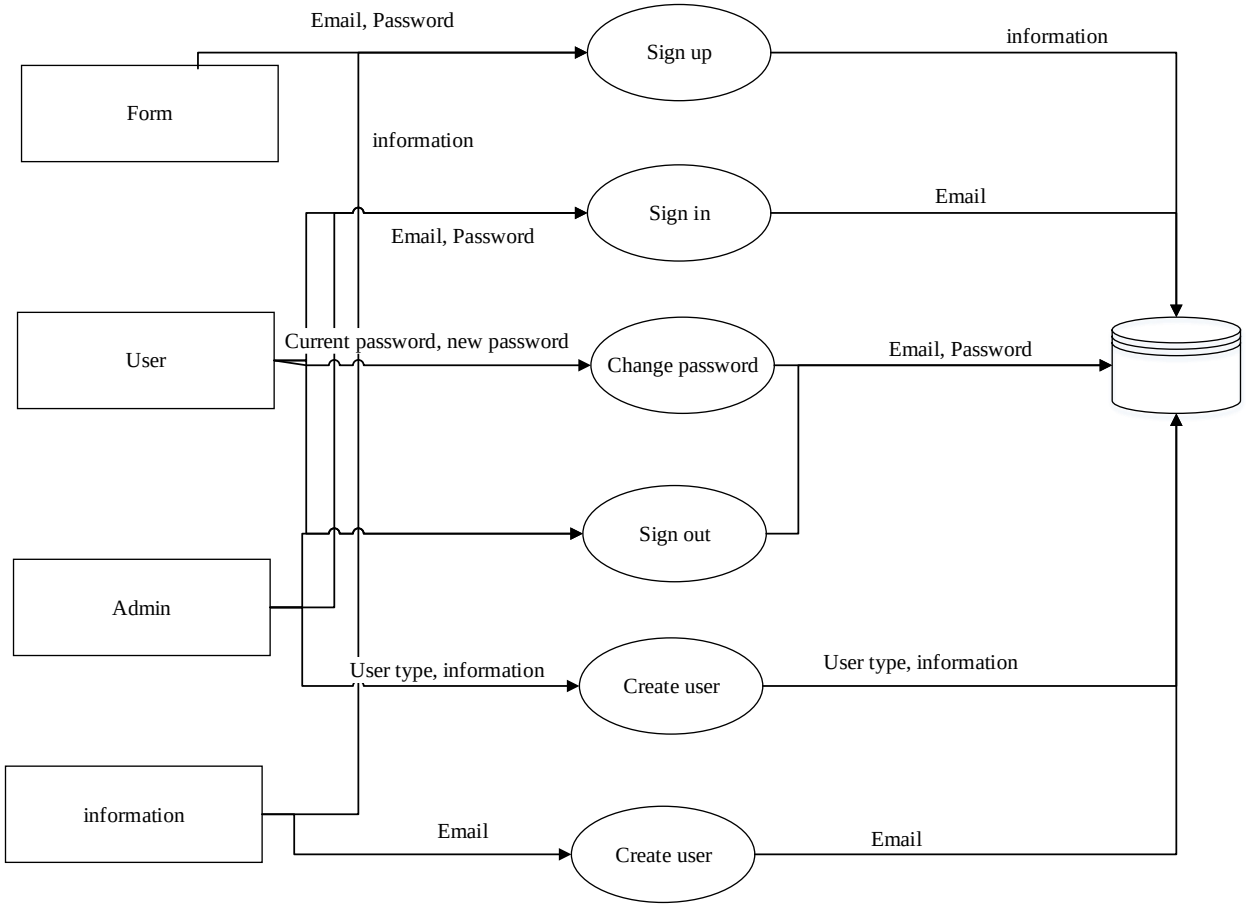


Figure 33: Level-1.1 DFD for Authentication

Chapter 8: Behavioral Model

The behavioral model indicates how software will respond to external events.

8.1 Introduction

Behavior modeling is also referred to as State modeling, State machines and State transition matrix. Behavior modeling is when one thinks of his ideas in terms of states and transitions. This requires both identifying all of the interesting states of being that software or its components are likely to be in. And also, at a high level abstracting what events are likely to cause software or its components to change between states of being.

8.2 Identifying Events

Here I have identified events from the Usage Scenario and listed their corresponding initiators & collaborators.

Table 9:

Event	Initiator	Collaborator
Sign up	Student	Admin, Database
Verify	Admin	User, Database
Create User	Admin	User, Database
Remove User	Admin	User, Database
Change password	User, Admin	Database
Sign in	User, Admin	Database
Sign out	User, Admin	Database
Create file	User	Database
Open file	User	Database
Delete file	User	Database
Edit file	User	Database
Add template	User	Database
Run code	User	Database

8.3 State Transition Diagram

State Transition Diagram represents active states for each class and the events (triggers) that cause changes between these active states. Here we have provided diagram for each of the actors.

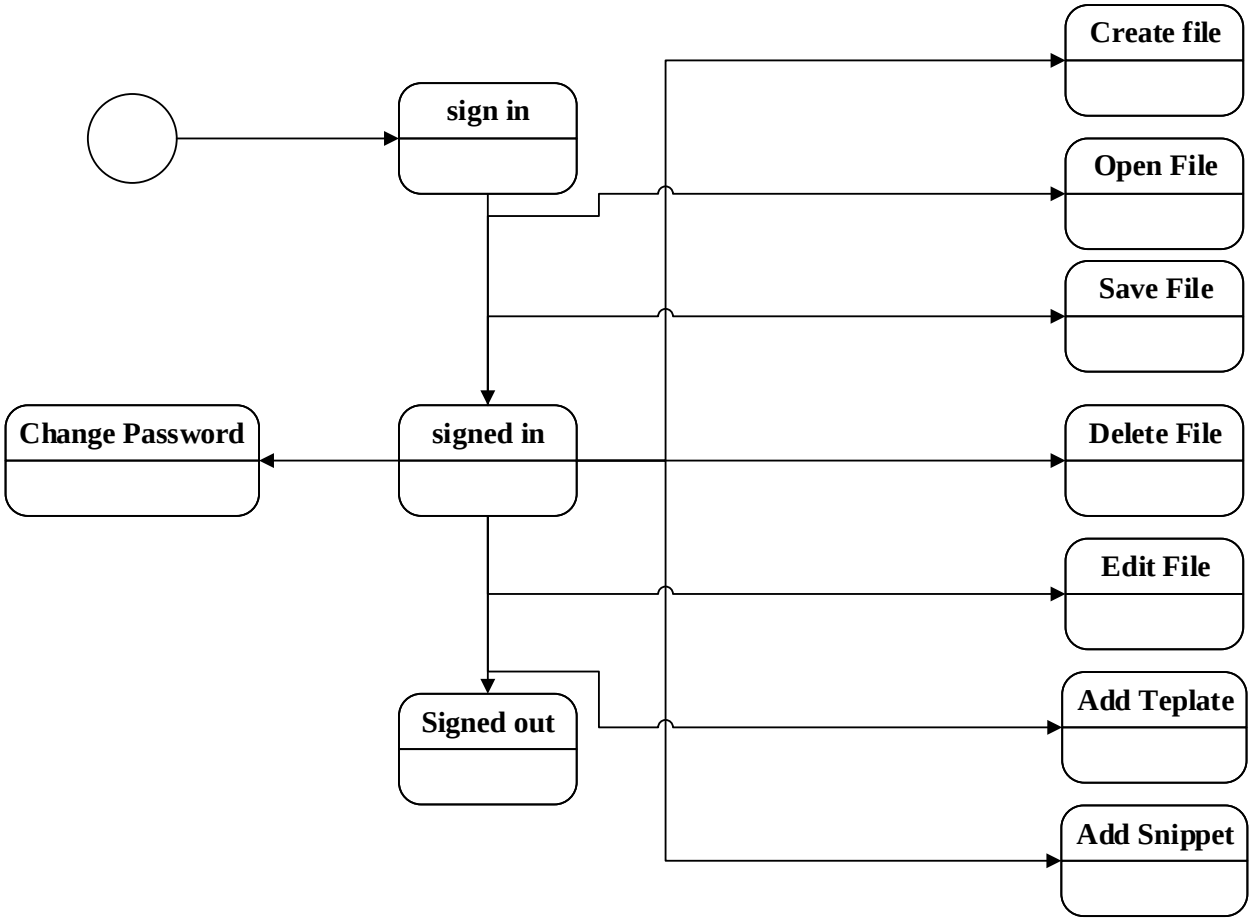


Figure 34: State transition diagram for User

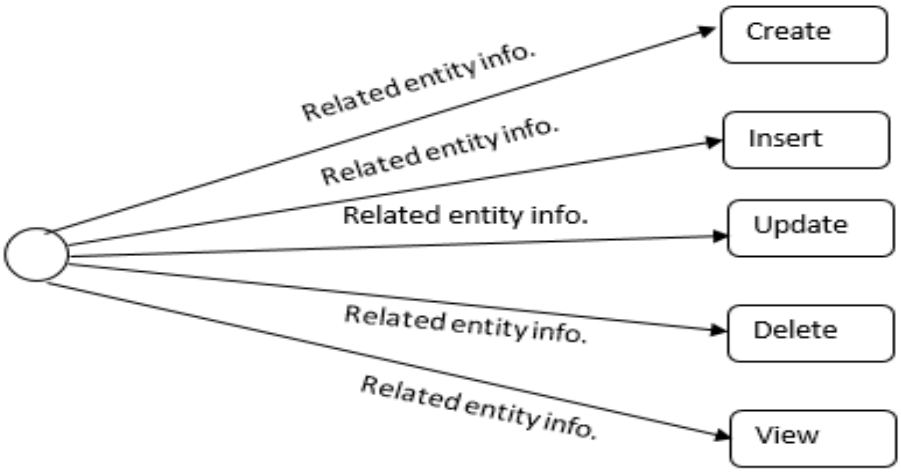


Figure 35: State transition diagram for Database

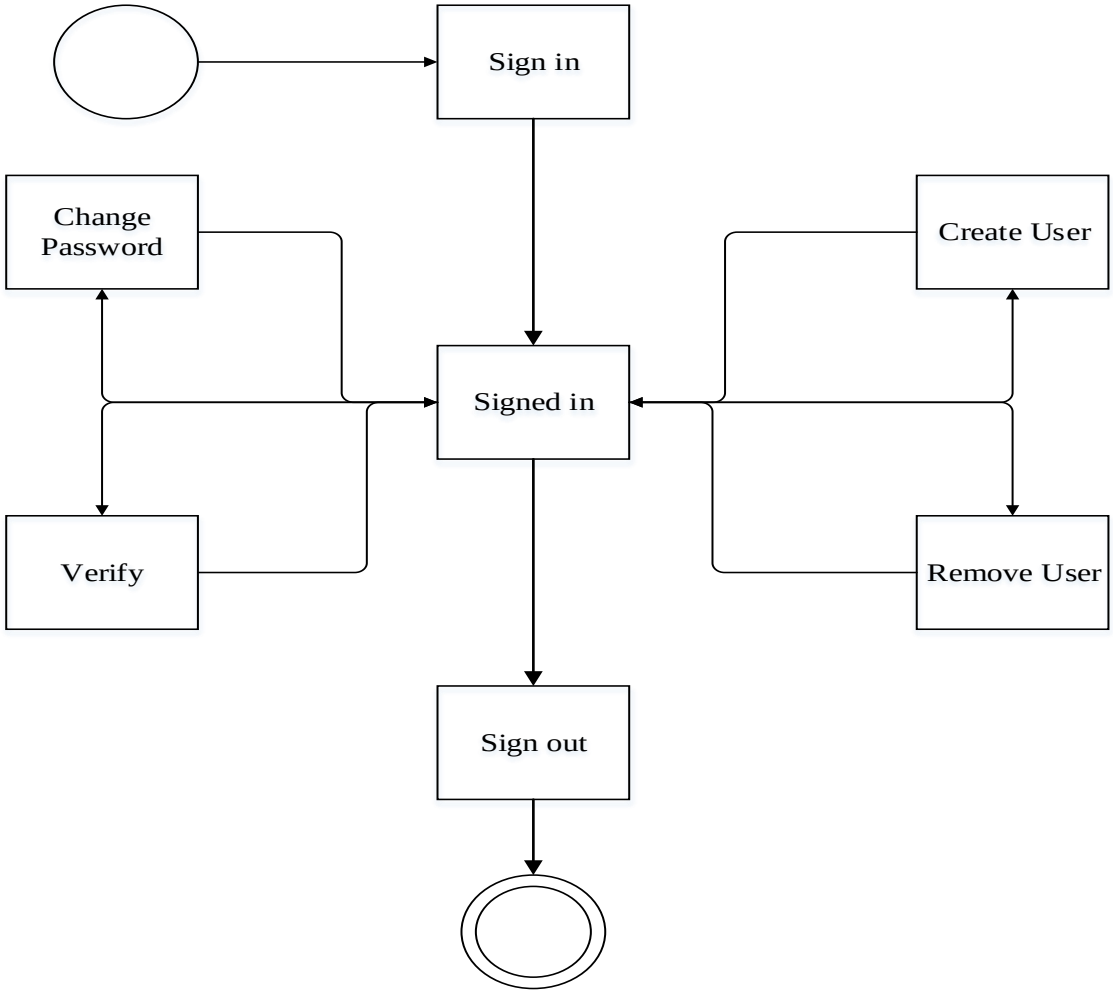


Figure 36: State transition diagram for Admin

8.3 Sequence Diagram

Sequence Diagram indicates how events cause transitions from object to object. It is actually a representation of how events cause flow from one object to another as a function of time.

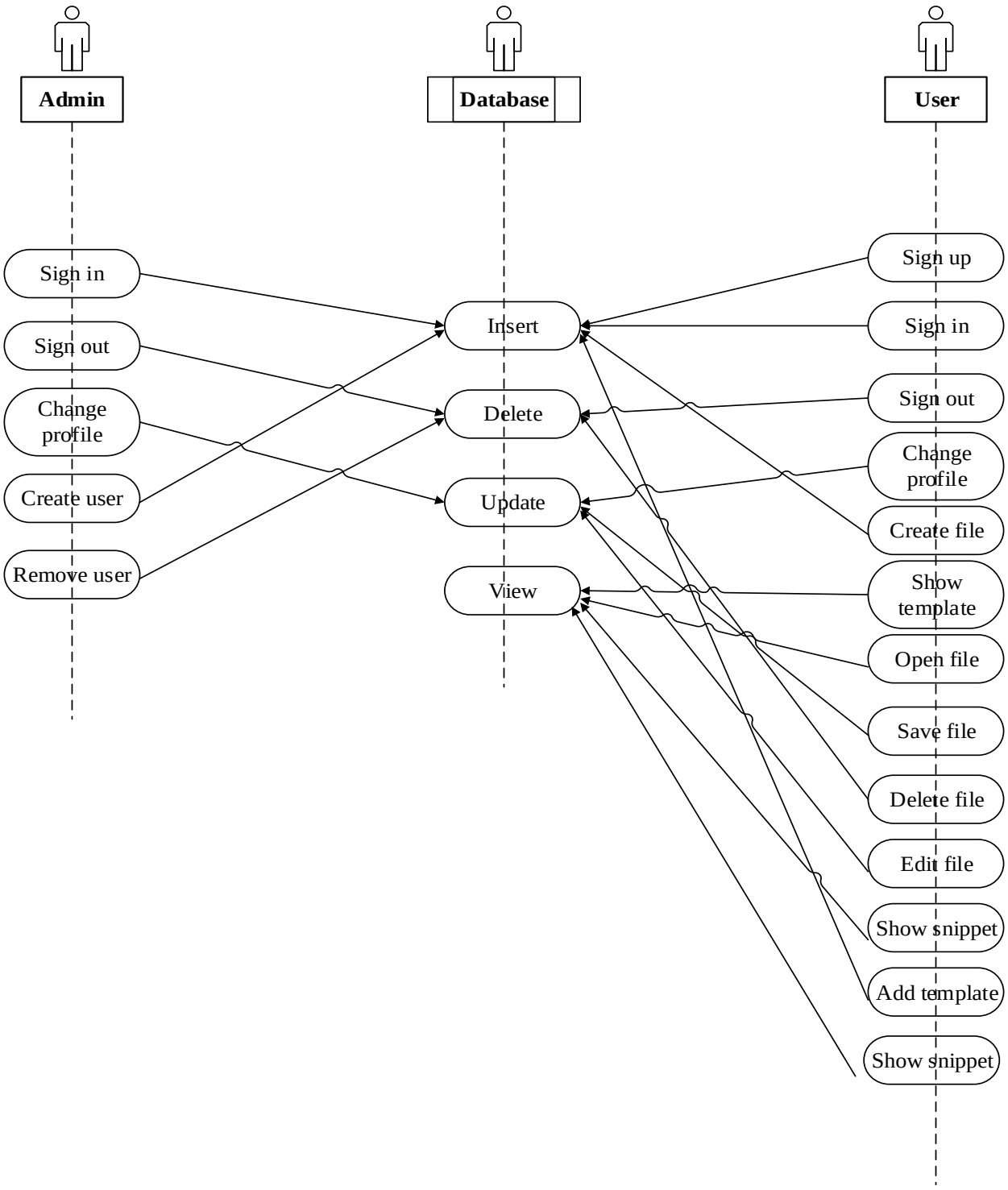


Figure-37: Sequence diagram of Ekushey Online Code Editor

Chapter 9: Conclusion

I am really very glad to submit the final SRS report on Ekushey Online Code Editor system. This SRS document can be used effectively to maintain software development cycle. It will be very easy to conduct the whole project using this SRS. Hopefully, this document can also help our junior BSSE batch students. I tried our best to remove all dependencies and make effective and fully designed SRS.I believe that reader will find it in order.

Appendix

References:

- Books
  - Pressman, Roger S. Software Engineering: A Practitioner's Approach (7th Ed.)

- URLs
  - https://en.wikipedia.org/wiki/Use_Case_Diagram
    (Last accessed: April 1, 2016)
  - https://en.wikipedia.org/wiki/Activity_diagram
    (Last accessed: April 5, 2016)
  - https://en.wikipedia.org/wiki/State_Transition_Diagram
    (Last accessed: April 7, 2016)
  - https://en.wikipedia.org/wiki/System_sequence_diagram
    (Last accessed: April 10, 2016)