NEXUS 101

An Academic Workflow Automation System

MARCH 20, 2018

Nexus 101: An Academic Workflow Automation System

Submitted to

SPL 2 coordinators

Institute of Information Technology

University of Dhaka

Submitted by

Md. Rafiul Islam BSSE0831

Md. Jewel Rana BSSE0842

Submission date: March 20, 2018

Acknowledgement

We are highly indebted for getting such a tremendous opportunity to prepare the report on this academic workflow automation system. We would like to thank whole-heartedly our supervisor, Nadia Nahar, Lecturer, Institute of Information Technology, University of Dhaka, and our SPL 2 coordinators for giving us guidelines about how we can prepare this report. In completing this report, we have collected various important data and information from students and teachers of various departments of University of Dhaka. We are thankful to all of the works cited.

Abstract

This study is made for Nexus 101: An Academic Workflow Automation System. The scope of the study is to analyze the existing workflows of various academic institutions and to know the functions and drawbacks and design the SRS (software requirements and specification) of the system. The object of the study is to develop an SRS of an existing academic workflow.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter is a part of the software requirement specification for the project ‘Nexus 101’. In this chapter intended audience for the project are focused on.

1.1 Purpose

This document briefly describes the Software Requirement Analysis of the academic automation system Nexus 101. It contains functional, non-functional, and supporting requirements and establishes a requirement baseline for the development of the system. The requirements specified in the SRS are independent, uniquely numbered and organized by topic. The SRS serves as an official mean of communicating user requirements to the developer and provides a common reference point for both the developer team and the 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 in intended for several audiences including the faculty members, the students and the program officer.

- The faculty members, the students and the program officer will use this SRS to be sure about the requirements they have given.
- The designers will use this SRS as a basis for creating the system’s design. The designers 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 they have created a software that will fulfill all 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 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

1.3 Conclusion

This analysis of the audience helped us to focus on the users who will be using our analysis. This document will help each and every person related to this project to have a better idea about the project.

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 a software project gets started and what the scope and nature of the problem to be solved is. 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 established a basic understanding of the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.

To establish the groundwork, the following factors have been worked on to the inception phase:

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

2.1.1 List of Stakeholders

Stakeholders 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. The initial list will grow as stakeholders are contacted because every stakeholder will be asked: “whom else do you think I should talk to?”

The following stakeholders were identified for Nexus 101.

Faculty Members:

Teachers are assigned for many courses based on semester/year. They take classes and give study materials for the course they are taking. They provide the study materials to the class representative and class representative provide the files to the whole class. They take attendance after every class.

Students:

Students collect study materials from the class representative. Students have their Facebook group or email to transfer or share study materials.

Program Officer

Program Officer notify students about academic notices.

Software Developer

A software developer is concerned with facets of the software development process, including the research, design, programming, maintenance and testing of computer software. He will be responsible for the outcomes of the software.

2.1.2 Recognizing Multiple Viewpoints

Different stakeholders demand different features from the software. To satisfy the stakeholders, most of these features should be included in the software.

Administrator’s viewpoint

- Error free system

- User friendly
- Strong authentication system

Faculty members' viewpoint

- User friendly
- Error free system
- File management through website
- flexibility in taking attendance
- search out students from attendance list
- send mail or call any student directly from the app
- upcoming class reminder

Students' viewpoint

- Error free system
- User friendly
- Easy access
- Smartphone based system
- well organized file system
- effective notification system
- search option

Developer's viewpoint

- Easy to built
- Error free effective software
- No ambiguous requirement
- Getting a decent amount of money for project budget

2.1.3 Working towards collaboration

While working with different stakeholders, some conflicting and common viewpoints can be noticed. For this reason, final requirements can be gotten by collaborating the viewpoints.

Common viewpoints

- Error free system
- User friendly
- Easy access
- effective notification system
- organized file system
- search option

Conflicting viewpoints

- Device

Final Requirements

- Error free system
- Strong authentication system
- User friendly
- Easy access
- effective notification system
- organized file system
- search option
- flexibility in taking attendance
- send mail or call directly from the app using system dialer

At first some context free questions were asked for identifying the stakeholders. Context free questions are helpful to identify some stakeholders who cannot be identified by structural questions. Then questions regarding the software were regarding to know their demands. The questionnaires are included in the appendix section.

2.2 Conclusion

In this inception phase, a basic understanding of the problem was developed and a preliminary nature of the solution was obtained. The requirements which are identified in this phase, will be used later for further steps of requirement engineering.

Chapter 3

Elicitation

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. Many difficulties were faced, like understanding the problems, making questions for the stakeholders, limited communication with the stakeholders due to a short amount of time and volatility. Though it is not easy to gather requirements within a short time, these problems have been surpassed in an organized and systematic manner.

3.2 Eliciting Requirements

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. The following tasks were done for eliciting requirements-

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

3.2.1 Collaborative Requirements Gathering

The meetings with the stakeholders created an indecisive state to elicit the requirements. To solve this problem, more than one meeting was held with the stakeholders. A slightly different scenario from these approaches has been found following activities have been completed to accomplish this task.

- The meetings were conducted with the faculty members, students and program officer. They were questioned about their requirements and expectations from Nexus 101.
- They were asked about the problems they were facing with the current manual system.
- Lastly final requirement list was selected from the meetings.

3.2.2 Quality Function Deployment

Quality Function Deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. Ultimately the goal of QFD is to translate 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 methodology that concentrates on maximizing customer satisfaction from the software engineering process. The requirements, which are given below, are identified successfully by the QFD.

3.2.2.1 Normal Requirements

The normal requirements are generally the objectives and goals that are stated for a product or system during meetings with the clients. These requirements fulfill customers' satisfaction. These are the normal requirements for the project.

- effective notification system
- organized file system
- search option
- flexibility in taking attendance
- send mail or call directly from the app using system dialer

3.2.2.2 Expected Requirements

These requirements are intrinsic to the product or system and may be so elementary that the clients does not explicitly state them. Their absence will be a cause for significant dissatisfaction. Below the expected requirements for our project are briefly described.

- Error free software
- Strong authentication system
- User friendly
- Effective system
- No ambiguous feature
- Data backup

3.2.2.3 Exciting Requirements

These requirements are for features that go beyond the clients' expectations and prove to be very satisfying when present. Following are some exciting requirements of this project.

- Notification via SMS
- File management through web client

3.2.3 Usage scenario

Authentication

The system will have an authentication system for users (student, teacher and administrator). The administrator will create accounts for newly admitted students with the email id they are provided by the institution and password. So there will be no need of registration or sign up. Everyone with an account will log in to the system using their username and password. If anyone forgets their username or password, they will get their needed information through the provided email id. Anyone can change their username and password whenever they want but to change any other information in the profile they will need administrator’s approval.

Administrator

There will be an administrator who will look after the activities in the system. One of their major tasks is to create the groups based on semesters or years and distribute the students in the groups. Updating information in students’ and faculty members’ profile is also a task of the administrator. Administrator will assign students and teacher in the group they need to be assigned. After a Class representative is selected in a batch, the students will let the administrator know and s/he will update the info in the system. Thus, the class representative will get the permission to access and do the things that are only permitted to them. Uploading files is one of them. Only the administrator will have full access to the database.

Student

Every student will have a unique student profile that contains name, personal information (Email Address, Contact, Address, Date of Birth, Blood Group) and academic information (Registration Number, Roll Number, Session, Attached Hall, Year/Semester). As students are sorted based on semester or year, they will have groups consisting of students who are in the same semester or year. Study materials (slides, PDFs, images, etc.) will be stored in that group based on courses. Every study material will have name, upload date and group name. Every course will have names and course codes. Students will be able to download study materials. Students will get push notification for any kind of activity regarding study materials (e.g. when new file is uploaded) and class schedules (changed or canceled). Only the class representatives will have the permission to upload study materials.

Teacher

Every teacher will also have a unique teacher profile that contains academic information (Name, Designation, Contacts, Email Address, Address). Teachers will be able to upload and download study materials in the group they are assigned. Teacher will also notify the students using push notification and SMS if they cancel a scheduled class or change the time. Teachers will take attendance using this application. They will pick the date and group, take the attendance and submit it. Teachers can edit the attendance data any time until they submit the marks of continuous evaluation to the office.

3.2.4 Elicitation Work Product

At first, it has to be known whether the output of the Elicitation task may vary because of the dependency on the size of the system or the product to be built. Here, the Elicitation work product includes

- Making a statement of our requirements for Nexus 101
- Making a bounded statement of scope for the system.
- Making a list of stakeholders who participated in the requirements elicitation.
- Making a list of requirements that are organized by function and domain constraints that apply to each other.
- A set of usage scenarios that provide insight into the use of the system.
- Description of the system’s technical environment.

Chapter 4

Scenario Based Modeling

This chapter describes the scenario-based model for Nexus 101.

4.1 Introduction

Although the success of a computer-based system or product is measured in many ways, user satisfaction resides at the top of the list. If the software developer team understands how end users (and other actors) want to interact with a system, they will be better able to properly characterize requirements and build meaningful analysis and design models. 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 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 roles 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.

4.3 Use Case Diagrams

Use Case diagrams give the non-technical view of system.

4.3.1 Level 0 – Nexus 101

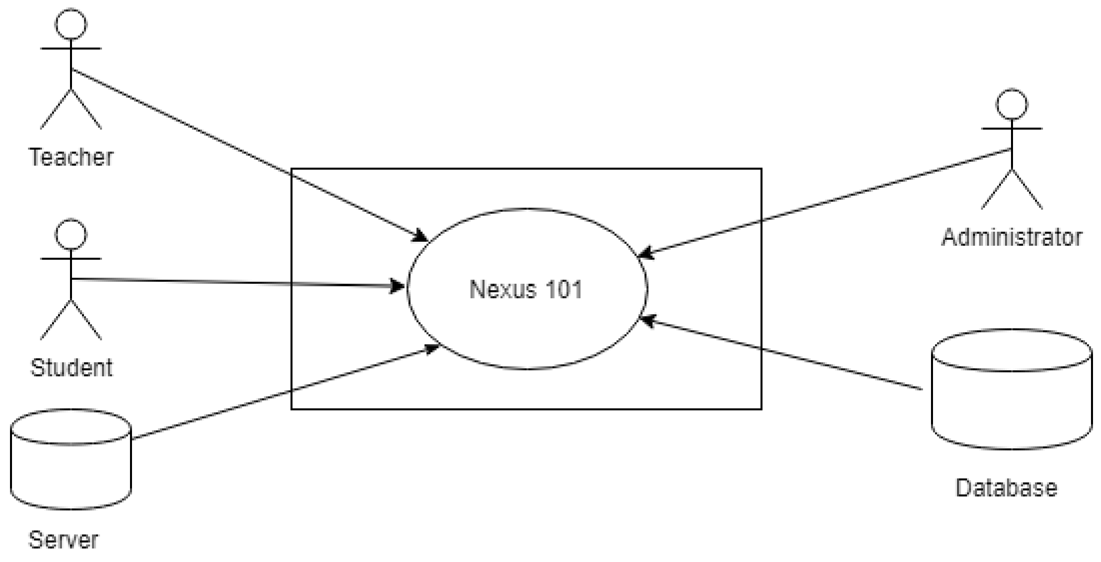

Figure 1: level 0 – Nexus 101

Actors

1. Administrator
2. Teachers
3. Students
4. Database
5. File server

Description

There are four actors in this system. Primary actors are those who play action and get replies from the system. Secondary actors only produce or consume information.

4.3.2 Level 1 – Subsystems

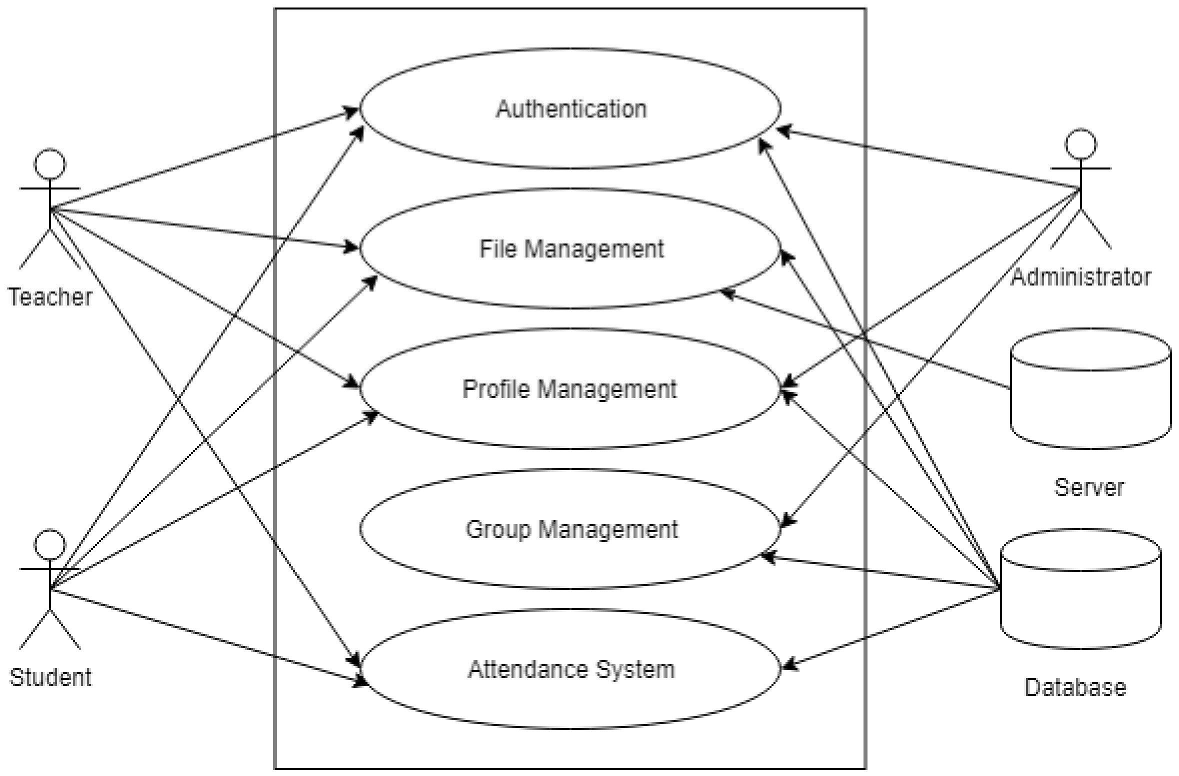

Figure 2: level 1 - subsystems

Actors

1. Administrator
2. Teachers
3. Students
4. Database
5. File server

Description

There are 5 subsystems in Nexus 101. They are:

1. Authentication
2. File management
3. Profile management
4. Group management
5. Attendance system

Authentication

This module has 3 subsystems:

1. Sign up
2. Sign in
3. Log in

File management

Teachers will be able to upload, download and remove files. Uploaded files will be stored in the file server. Students will only have the permission to download files. Only the class representative gets the permission to upload files

Action reply

Action: teacher uploads new file in under course

Reply: new file uploaded in file server

Action: teacher/student downloads file

Reply: file downloaded

Action: teacher deletes file

Reply: selected file deleted from server

Profile management

This module lets user do all the editing, changing and deleting in their profiles. Any user can change their information in the profiles. But if the user is a teacher or a student, they need the approval of the administrator to make the changes happen in the profiles.

Action reply

Action: user inputs new information

Reply: administrator approves change, changes in database and user gets notification

Action: user deletes information

Reply: administrator approves change, deletes from database and user gets notification

Group management

Administrator will create groups. Every group will have courses under them and courses will have files. There will be teachers assigned to groups. The groups will be sorted based on year/semester. The administrator will distribute students under the groups.

Action reply

Action: administrator enters group name, year

Reply: new group created and database updated

Action: administrator selects teacher, student, group name

Reply: selected teachers and students assigned under the selected group

Attendance system

Teachers will be able to take attendance with this system. They have to pick the date and select the course. The list of students under the selected course will be shown and then the teacher will select the students who are absent and submit the list. If there is a mistake, the teacher will be able to edit the attendance.

Editing the attendance data will be permitted until the marks of continuous evaluation is submitted to program office. Students can only view their attendance.

Action reply

Action: teachers select course and date

Reply: list of student shows

Action: teacher selects absent students and submit

Reply: database updates

Action: teacher edits information in attendance data

Reply: database updates

Action: teachers select course and date

Reply: attendance shows

4.3.3 Level 1.1 - Authentication

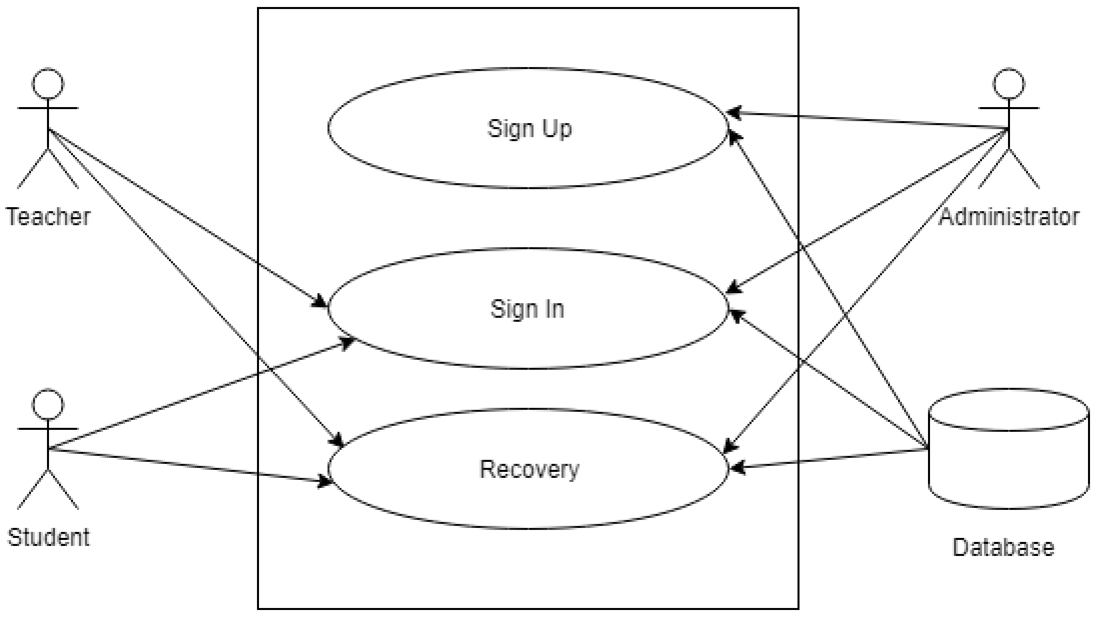

Figure 3: Authentication

Actors

1. Administrator
2. Teachers
3. Students
4. Database

Description

Sign up

In this system, sign up module will only be used by the administrator. The students and teachers will not need to create their own accounts. The administrator will do it for them. With the email id provided by the institute, the administrator will create user accounts with a default password. The users can change the password later.

Sign in

Any user with an account can sign in to the system. To sign in they need to input email id and password. The email id and password will be verified. User can log in to the system if the verification is successful. If the input is wrong, there will be retry option.

Recovery

User will be able to recover their password. All they have to do is provide the email id and the password will be sent to the id if the id is valid.

Action reply

Action: administrator enters email id and password to create new account for users

Reply: new account created and stored in database

Action: user enters email id and password

Reply: logs in if input is valid

Action: users enter email id to recover password

Reply: password sent to email id if the id is valid

4.4 Activity Diagram

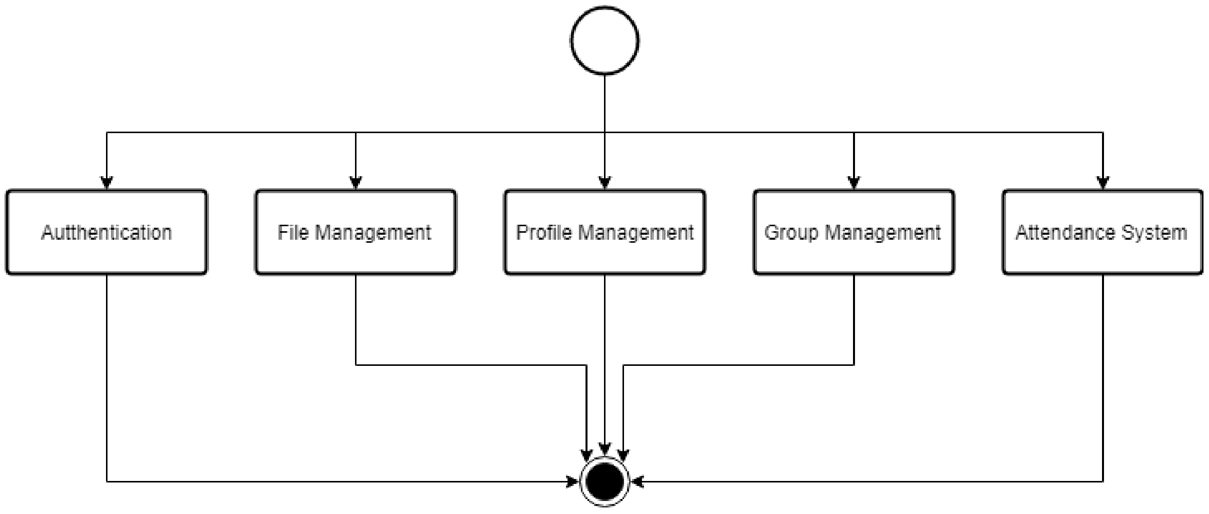

Figure 4: Nexus 101

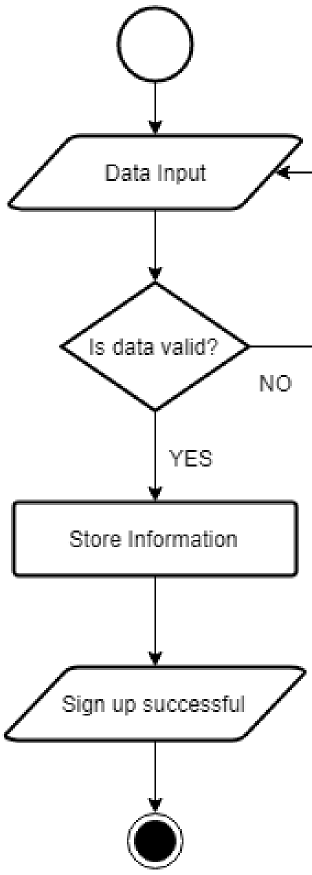

Figure 5: Sign up

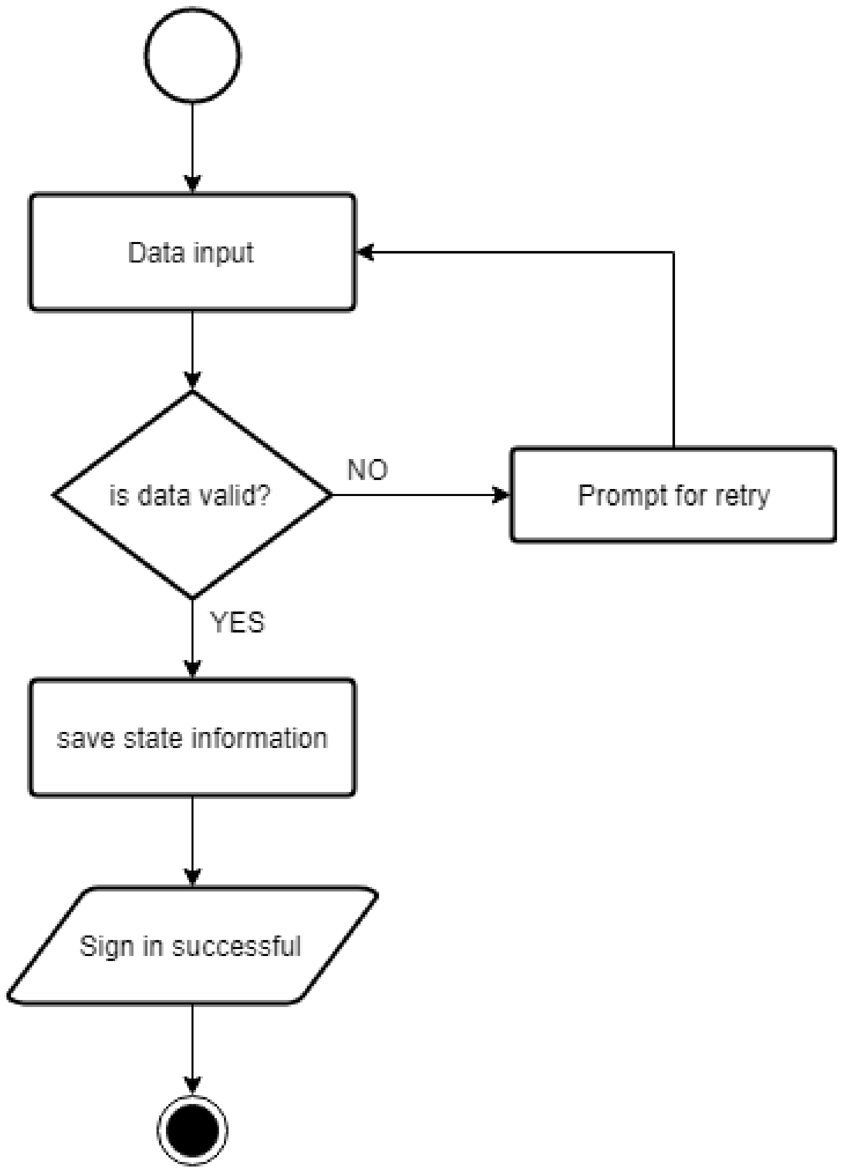

Figure 6: Sign in

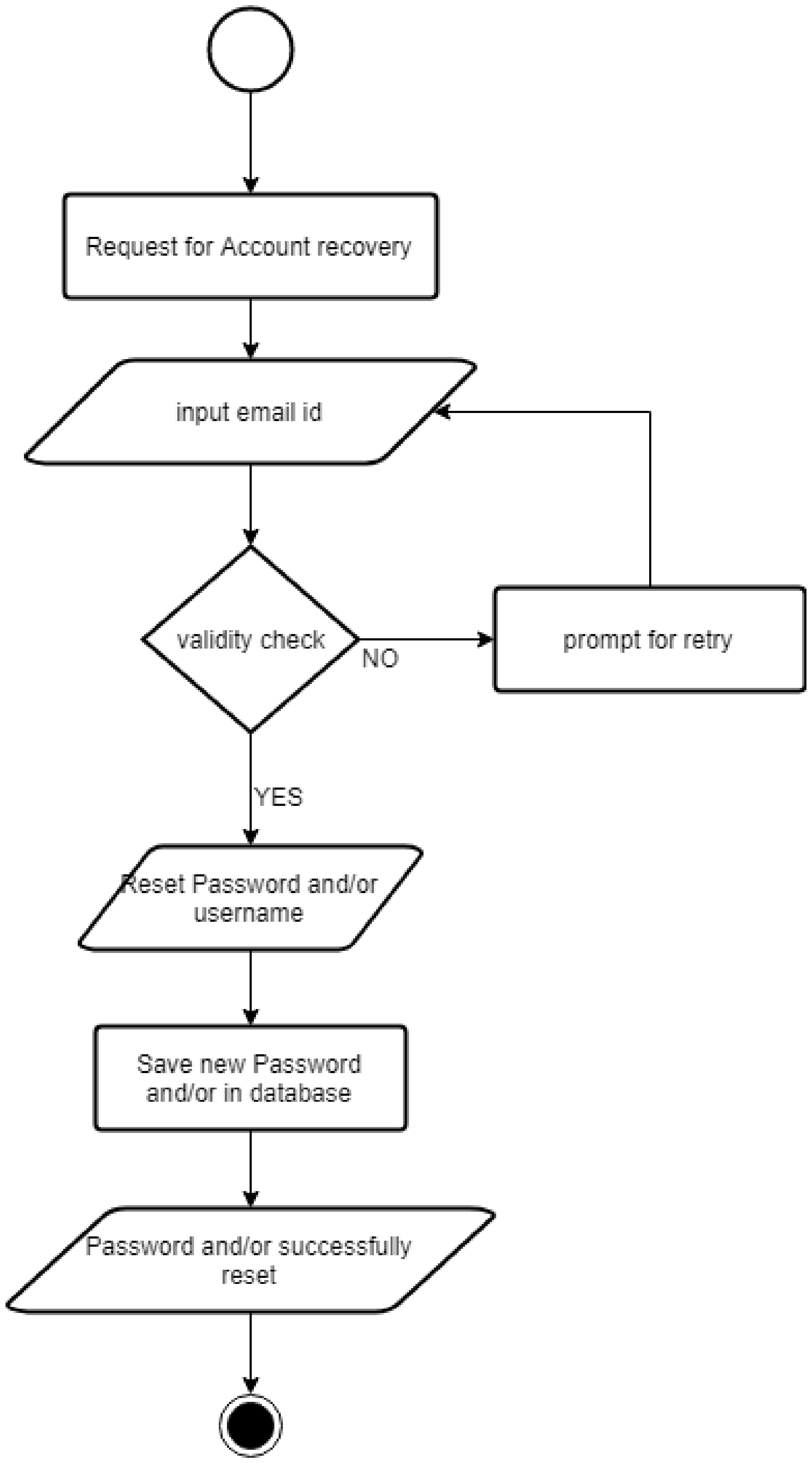
Request for Account recovery
input email id
validity check
NO
prompt for retry
YES
Reset Password and/or username
Save new Password and/or in database
Password and/or successfully reset

Figure 7: Recover password

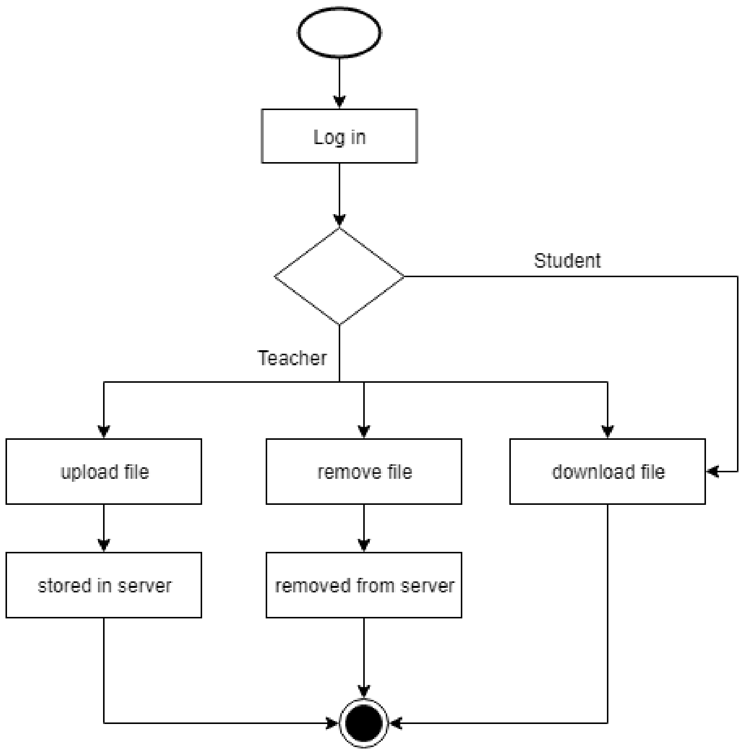

Figure 8: File management

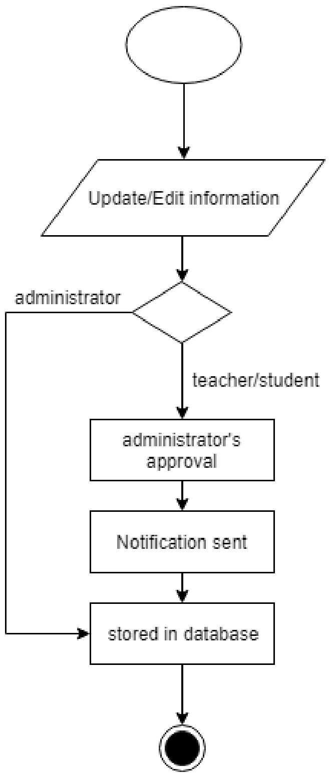

Figure 9: Profile management

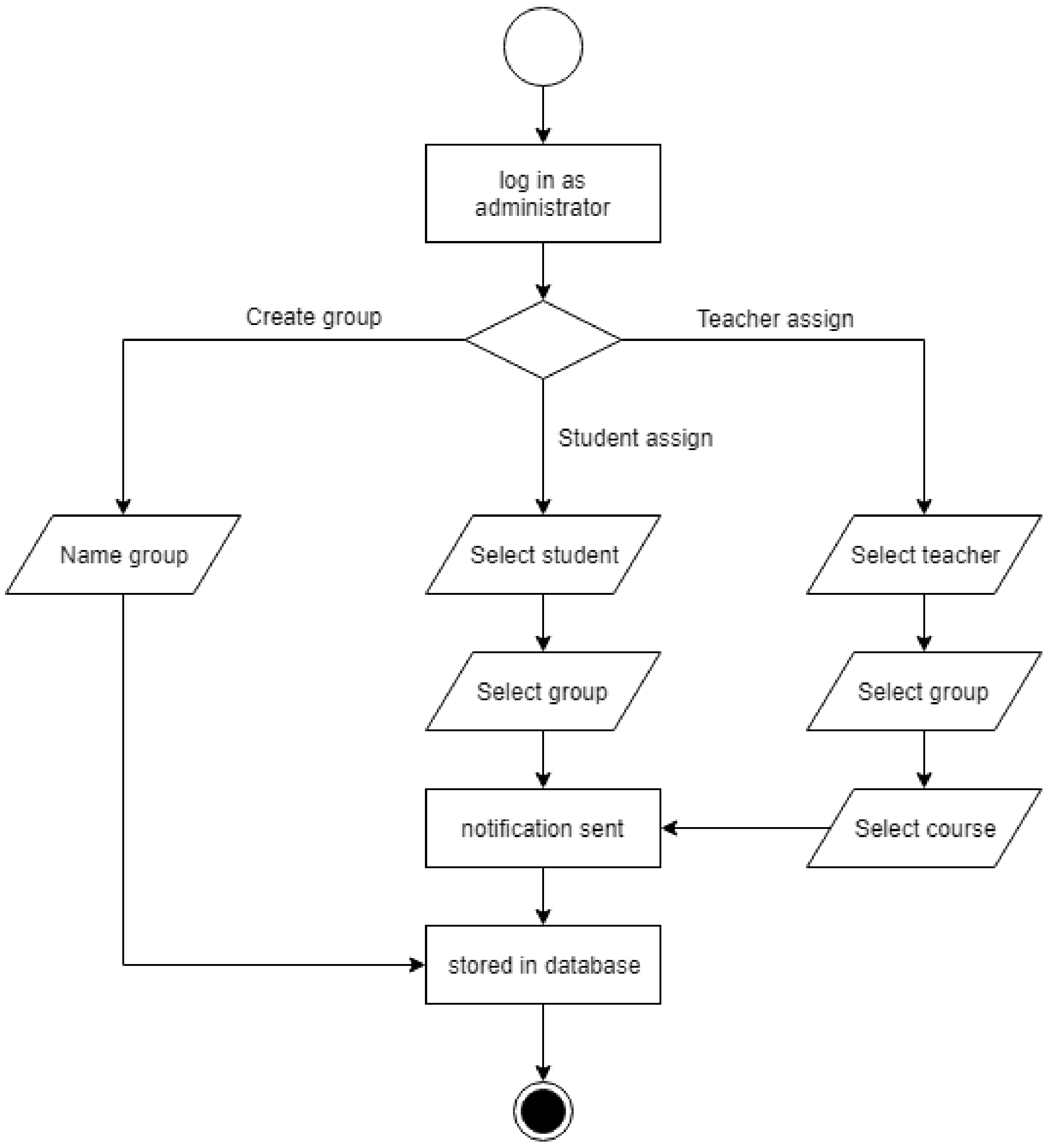

Figure 10: Group Management

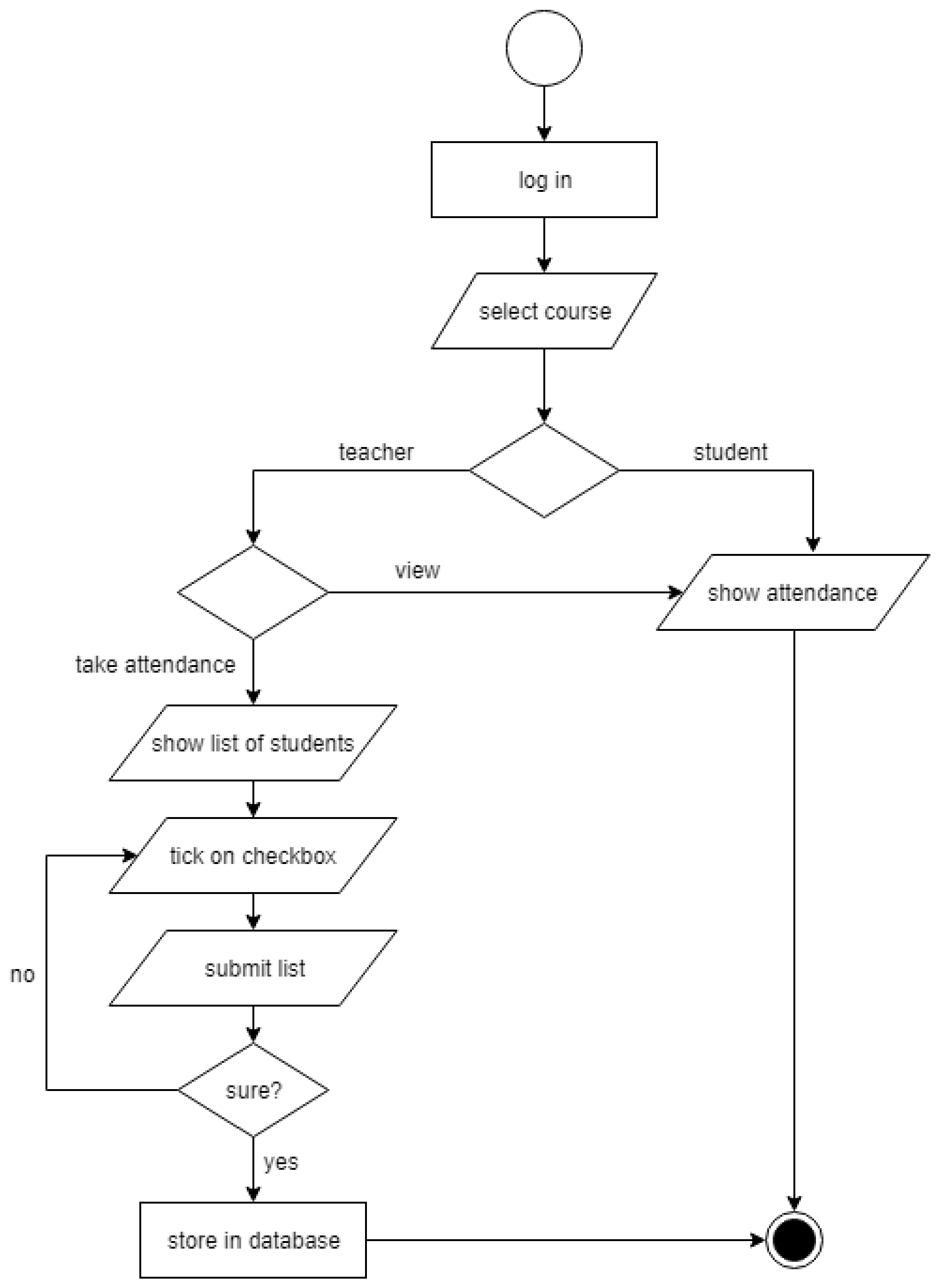

Figure 11: Attendance

4.5 Swim Lane Diagram

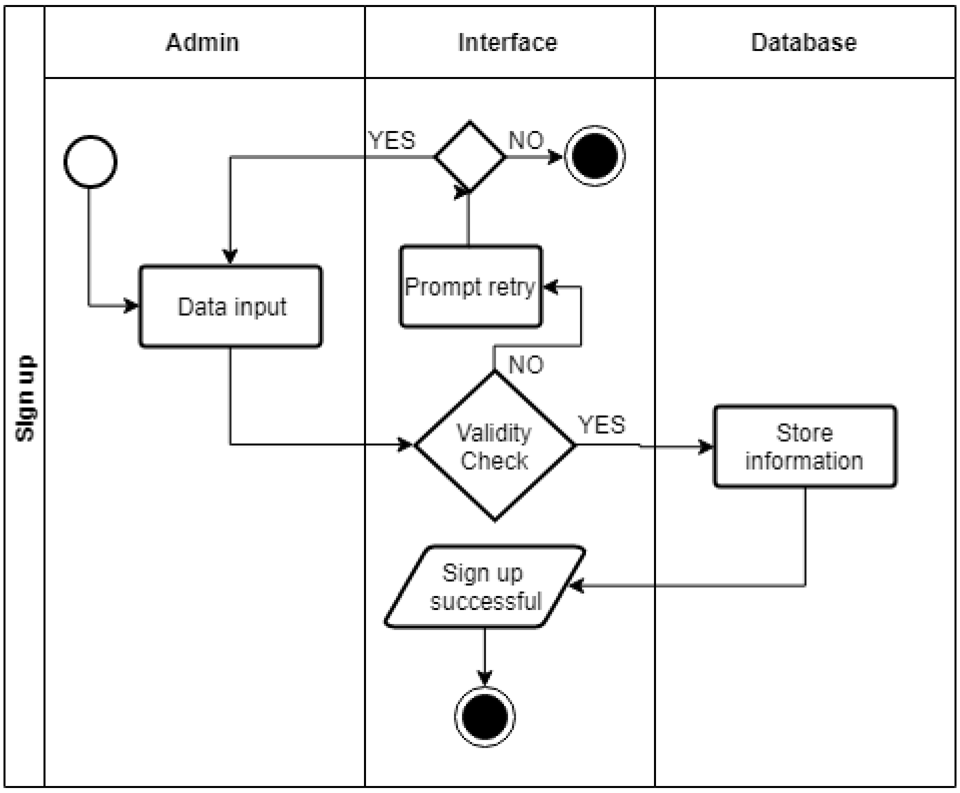

Figure 12: Sign up

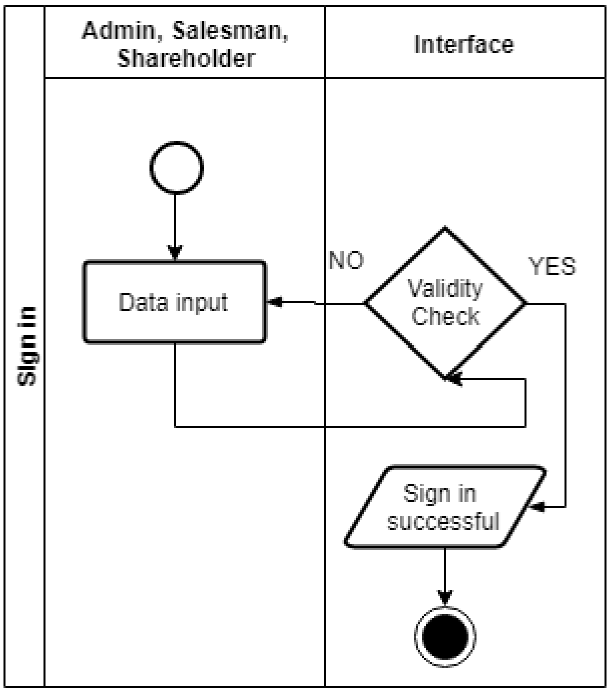

Figure 13: Sign in

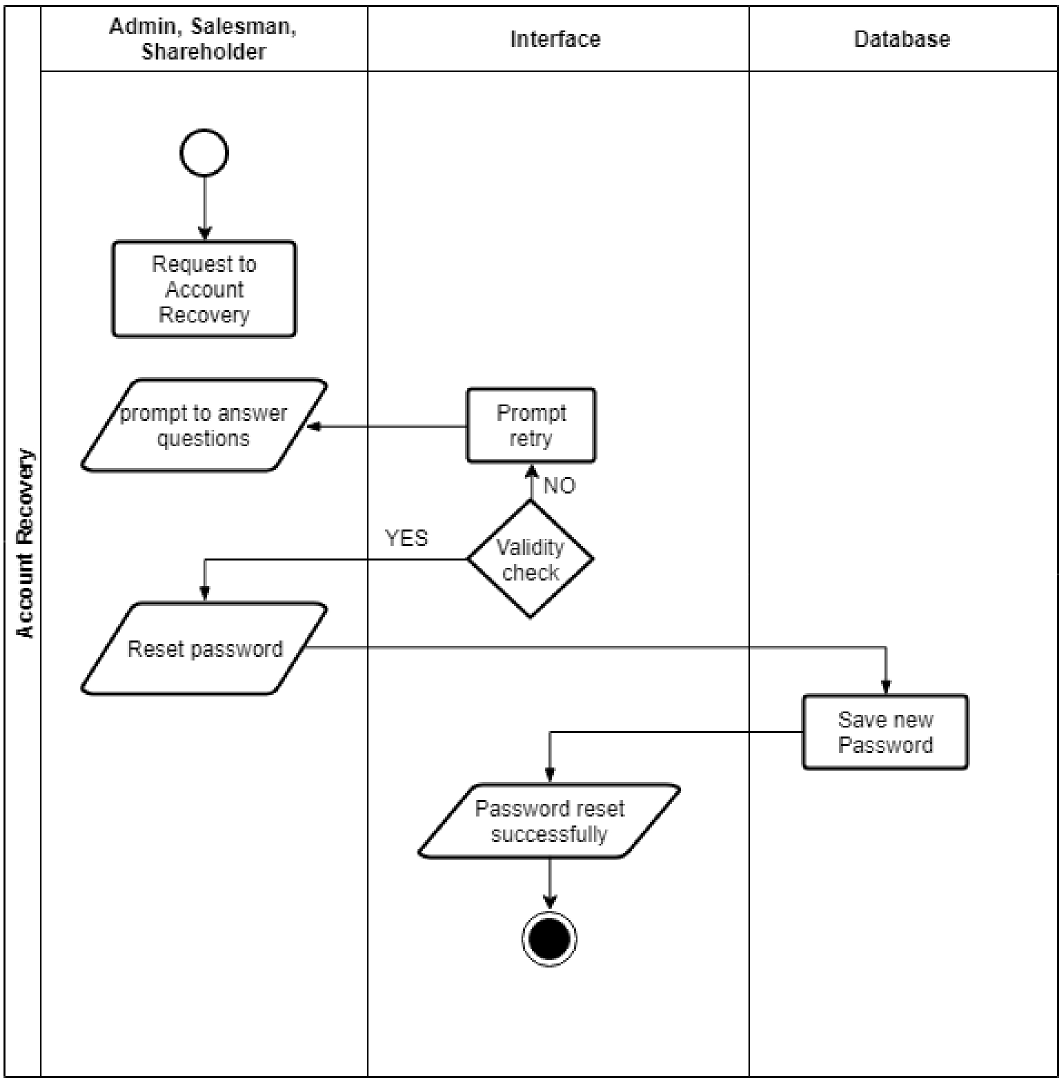
Account Recovery
Admin, Salesman, Shareholder
Interface
Database
Request to Account Recovery
prompt to answer questions
Prompt retry
NO
YES
Validity check
Reset password
Save new Password
Password reset successfully

Figure 14: Account recovery

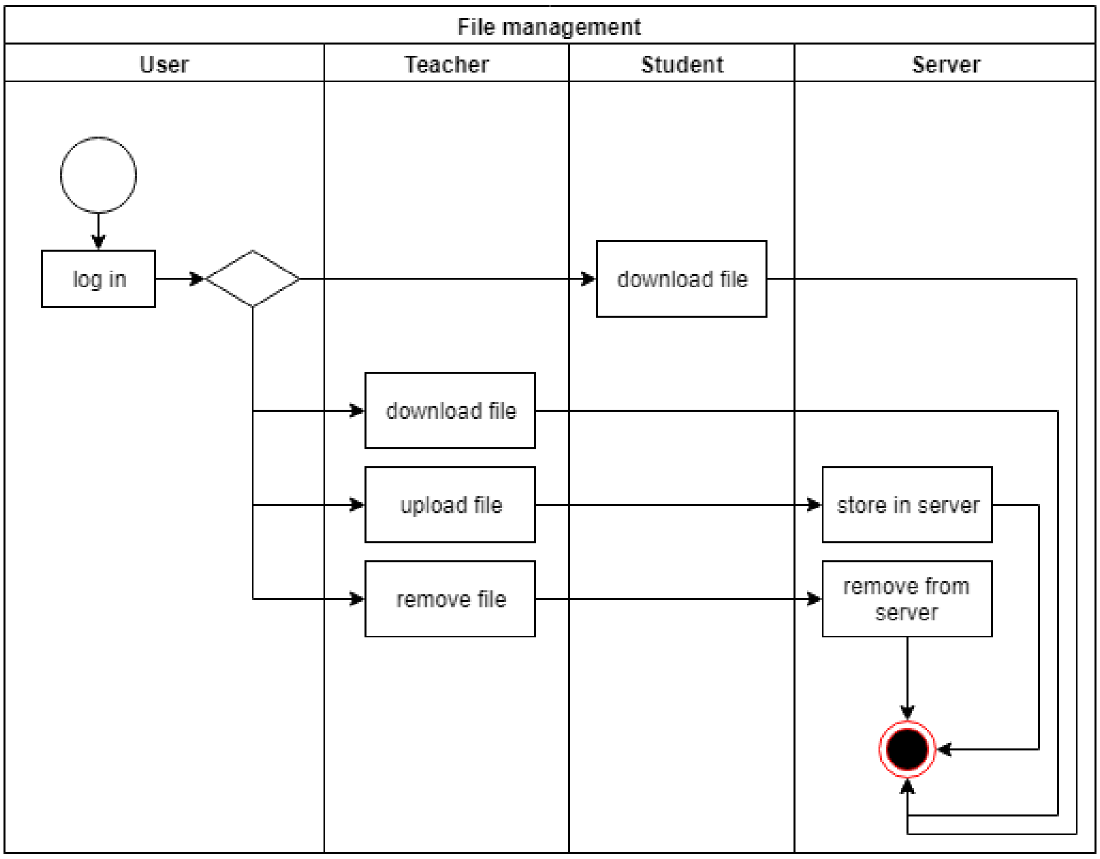

Figure 15: File management

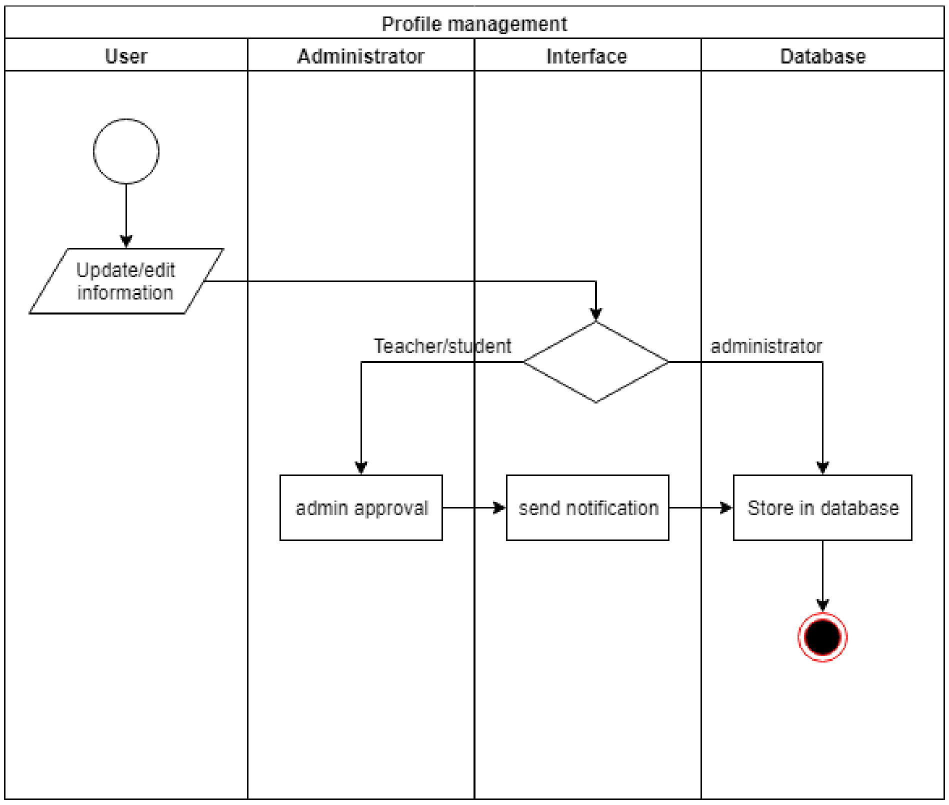

Figure 16: Profile management

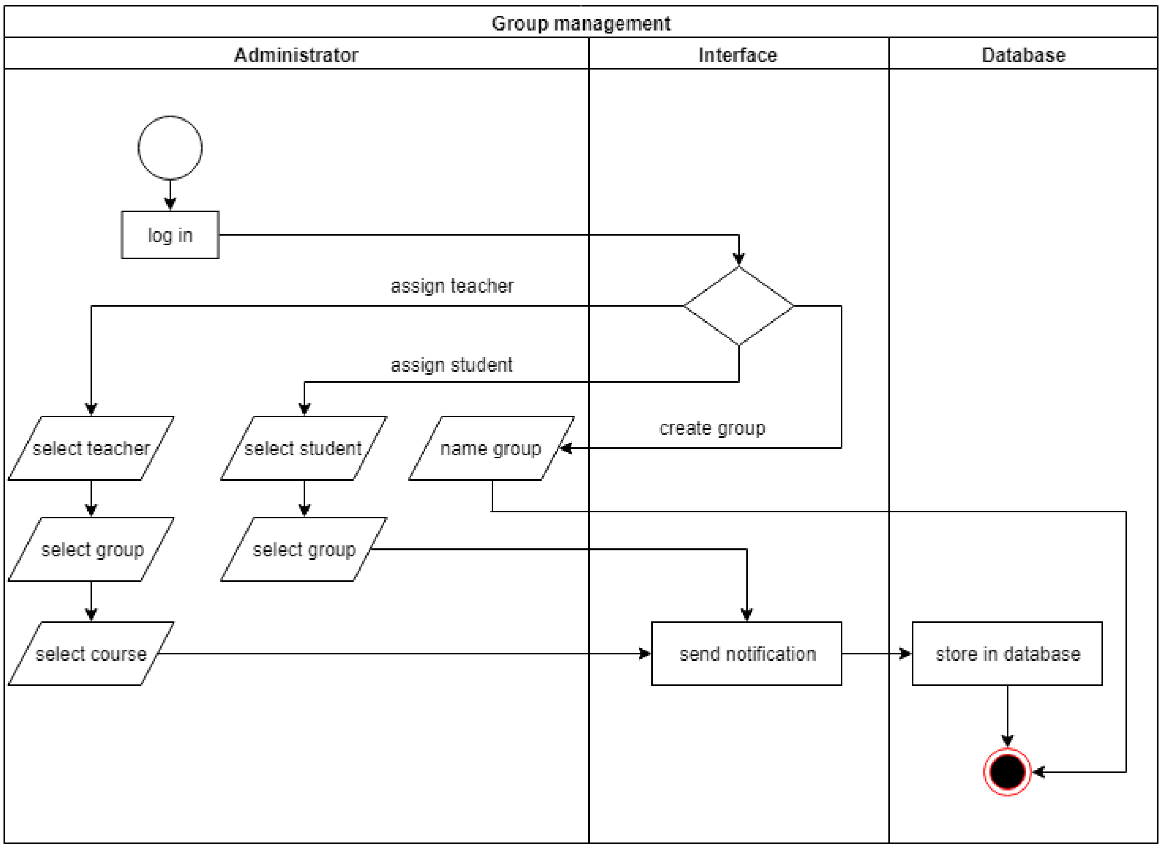

Figure 17: Group management

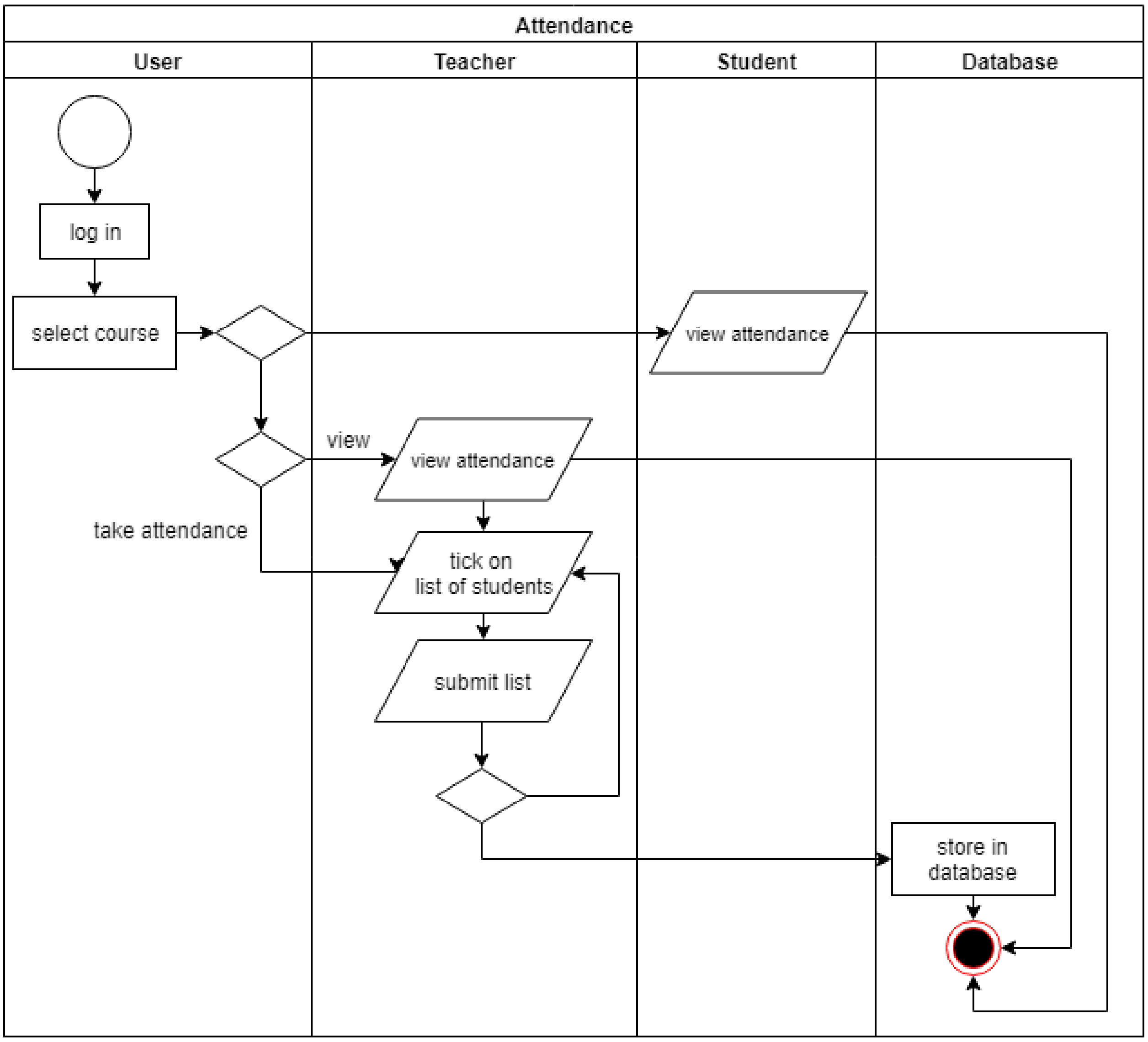

Figure 18: Attendance

Chapter 5

Data Modeling

5.1 Data modeling concept

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-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

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

All the nouns in the scenario were identified.

Table 1: Noun identification

No	Noun	P/S	Attributes of
1	System	P	
2	Authentication	S	
3	User	S	

4	Student	S	
5	Teacher	S	
6	Administrator	S	
7	Account	S	
8	Email id	S	3, 4, 5, 6
9	Institution	P	
10	Password	S	3, 4, 5, 6
11	Registration	S	
12	Sign up	S	
13	Username	S	3, 4, 5, 6
14	Information	P	
15	Profile	S	3, 4, 5
16	Group	S	4, 5, 41
17	Semester	S	4, 5, 16
18	Year	S	4, 5, 16
19	Class representative	S	
20	Batch	P	
21	Permission	S	
22	File	S	
23	Database	S	
24	Student profile	S	
25	Name	S	4, 5, 37, 42
26	Personal information	P	
27	Email address	S	4, 5
28	Contact no	S	4, 5
29	Address	S	4
30	Date of birth	S	4,
31	Blood group	S	4, 5

32	Academic information	P	
33	Registration no	S	4
34	Roll no	S	4
35	Session	S	4
36	Attached hall	S	4
37	Study materials	S	
38	Slides	S	
39	PDFs	S	
40	Images	S	
41	Upload date	S	37
42	Course	S	5, 16
43	Course code	S	42
44	Push notification	S	
45	Class schedules	S	
46	Teacher profile	S	
47	Designation	S	5
48	SMS	S	
49	Attendance	S	4
50	Data	P	
51	Time	P	
52	Marks	P	
53	Continuous evaluation	P	
54	Office	P	

5.2.2 Potential data objects

- User: 8, 10, 13, 15
- Student: 8, 10, 13, 15, 16, 17, 18, 25, 27, 28, 29, 30, 31, 33, 34, 35, 36, 49
- Teacher: 8, 10, 13, 15, 16, 17, 18, 25, 27, 28, 31, 42, 47
- Administrator: 8, 10, 13

- Group: 17, 18, 42
- Course: 25, 43
- Study materials: 25, 41

5.2.3 Analysis for finalizing Data objects

- Students, teachers and administrator are users of the system, Nexus 101. So, all three kinds of users can be merged into a user data object.
- All other data objects can be used as data objects as they have enough importance in the system.

5.2.4 Final Data objects

- **User**: Username, Password, Email ID
- **Student:** Email ID, Name, Address, contact no, Blood group, roll no, Reg no, Session, Attached hall, Semester, Year, Group
- **Teacher**: Email ID, Name, Address, Contact no, Blood Group, Semester, Year, Group, Course
- **Group**: Semester/Year, Course
- **Course**: Name, code
- **Study Materials**: File name, Upload date

5.2.5 Data Object Relations

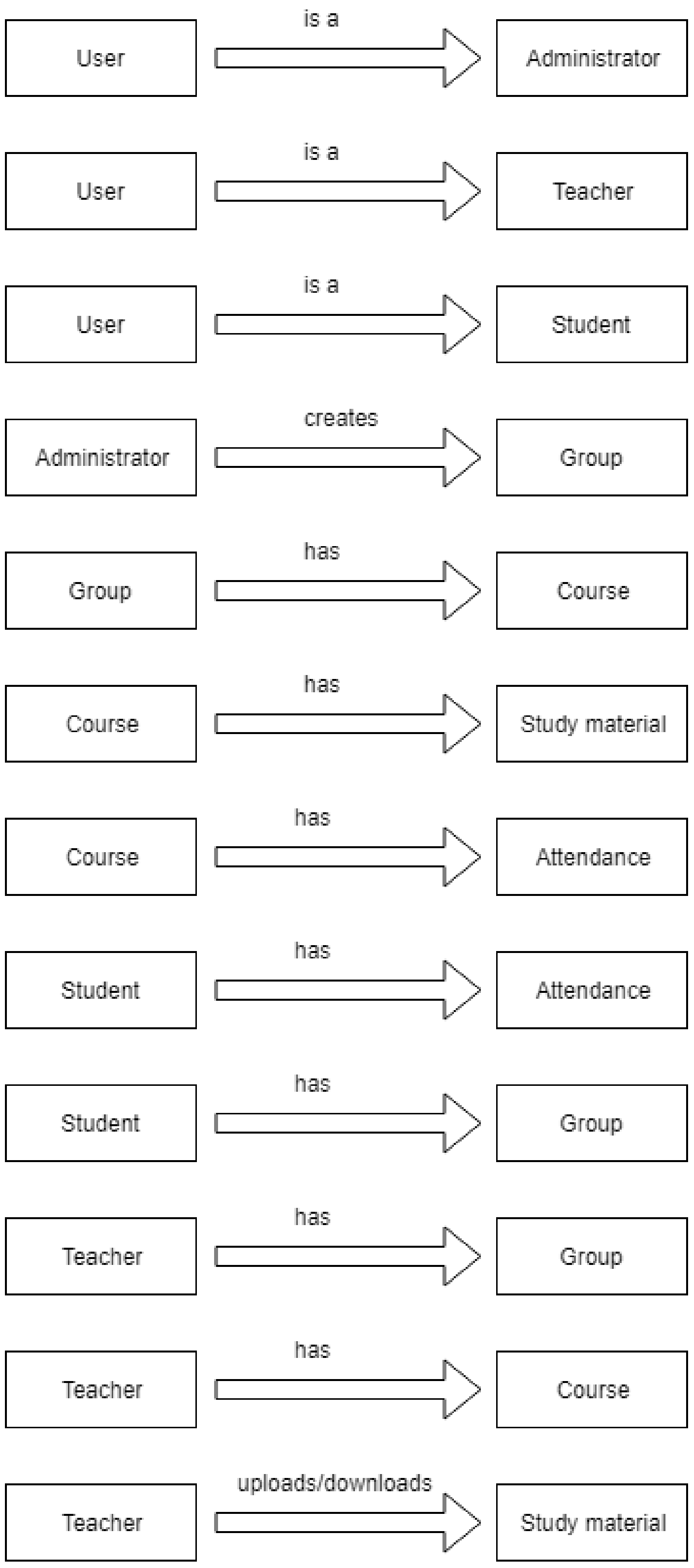

Figure 19: Data object relation

5.3 Entity Relationship Diagram

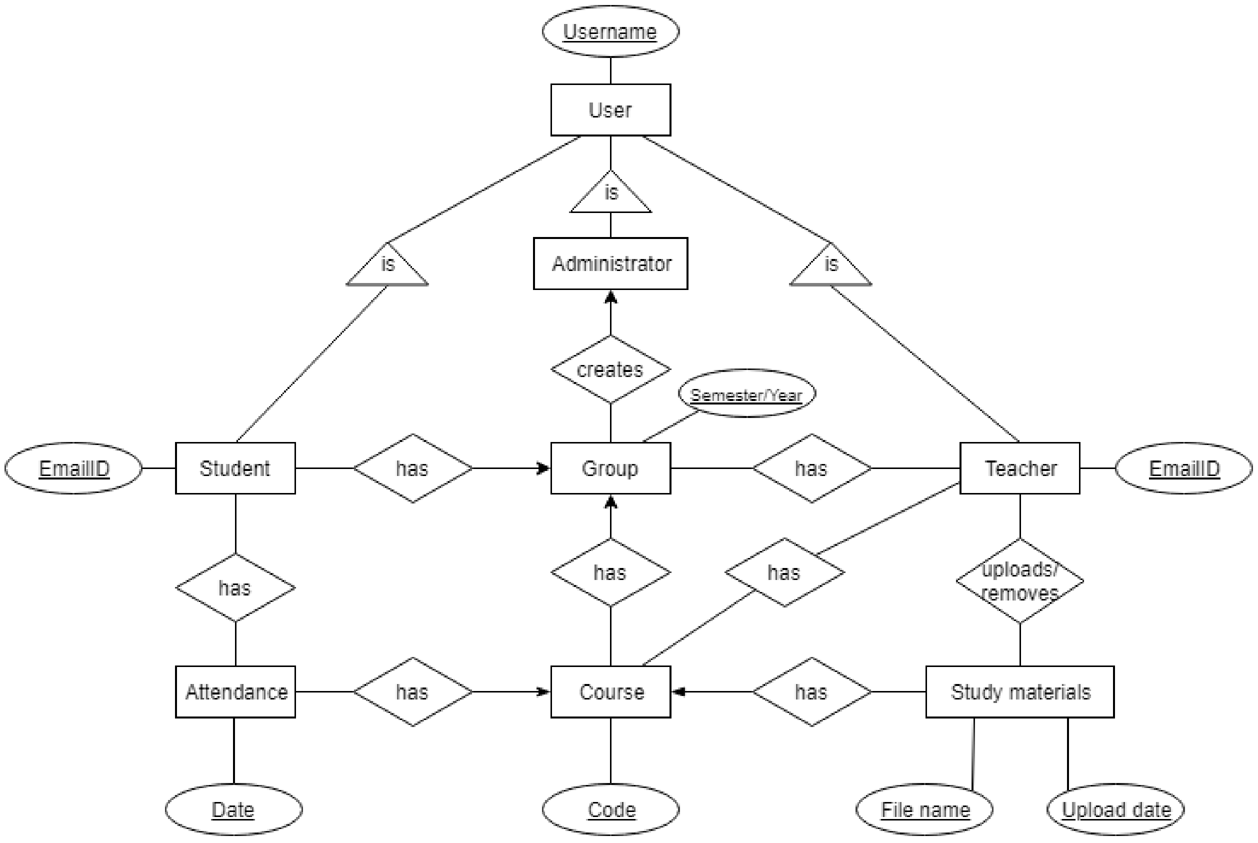

Figure 20: Entity Relationship Diagram

5.4 Schema Diagram

Table 2: User

User		
Attribute	**Type**	**Size**
<u>User Id</u>	VARCHAR2	30
First Name	VARCHAR2	30
Last Name	VARCHAR2	30
Email id	VARCHAR2	30
Contact no	VARCHAR2	11
Password	VARCHAR2	30
Group	VARCHAR2	10

Table 3: Administrator

Administrator		
Attribute	**Type**	**Size**
User Id	VARCHAR2	30

Table 4: Teacher

Teacher		
Attribute	**Type**	**Size**
User Id	VARCHAR2	30
Designation	VARCHAR2	20
Blood group	VARCHAR2	2
Course	VARCHAR2	10

Table 5: Student

Student		
Attribute	**Type**	**Size**
User Id	VARCHAR2	30
Blood group	VARCHAR2	2
Address	VARCHAR2	30
Date of birth	DATE	10
Registration number	VARCHAR2	20
Roll number	VARCHAR2	5
Session	VARCHAR2	7
Semester / year	VARCHAR2	5
Attached hall	VARCHAR2	10
Attendance	VARCHAR2	4

Table 6: Group

Group		
Attribute	**Type**	**Size**
<u>Year/Semester</u> User ID	VARCHAR2 VARCHAR2	5 30

Table 7: Course

Course		
Attribute	**Type**	**Size**
<u>Course name</u> <u>Course code</u> User ID	VARCHAR2 VARCHAR2 VARCHAR2	20 6 30

Table 8: Study material

Study material		
Attribute	**Type**	**Size**
<u>File name</u> Upload date User id Course code	VARCHAR2 VARCHAR2 VARCHAR2 VARCHAR2	20 10 30 6

Table 9: Attendance

Attendance		
Attribute	**Type**	**Size**
Date	DATE	10
User Id	VARCHAR2	30
Course code	VARCHAR2	6

Chapter 6

Class Based Modeling

This chapter is intended to describe class-based modeling of Nexus 101.

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.

6.2 General Classification

To identify the potential classes, nouns were selected 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 10: General classification

No	Noun	GC
1	Authentication	3, 6, 7
2	User	4, 5, 7
3	Student	4, 5, 7
4	Teacher	4, 5, 7
5	Administrator	4, 5, 7
6	Account	2
7	Email id	
8	Password	
9	Registration	3
10	Sign up	3
11	Username	
12	Profile	2,7
13	Group	2,7
14	Semester	
15	Year	
16	Class representative	2,4
17	Permission	3
18	File	2, 7
19	Database	1, 2, 6
20	Student profile	7
21	Name	

22	Email address	
23	Contact no	
24	Address	
25	Date of birth	
26	Blood group	
27	Registration no	
28	Roll no	
29	Session	
30	Attached hall	
31	Study materials	2, 5
32	Slides	2
33	PDFs	2
34	Images	2
35	Upload date	
36	Course	2, 5, 7
37	Course code	
38	Push notification	2, 3, 7
39	Class schedules	
40	Teacher profile	7
41	Designation	
42	SMS	2, 3, 7
43	Attendance	2, 5, 7

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 11: Selection criteria

No	Noun	SC
1	Authentication	2, 3, 6
2	User	1, 2, 3, 4, 5, 6
3	Student	1, 2, 3, 4, 5, 6
4	Teacher	1, 2, 3, 4, 5, 6
5	Administrator	1, 2, 3, 4, 5, 6
6	Profile	2, 3, 6
7	Group	1, 2, 3, 6
8	File	1, 2, 3, 6
9	Course	1, 2, 3, 6
10	Push notification	2, 3, 6
11	SMS	2, 3, 6
12	Attendance	2, 3, 6

6.4 Associate Noun and Verb Identification

The noun and the verbs associated with the potential classes are identified to find out the attributes and methods of every class.

Table 12: Associating nouns and verbs

No	Class name	Noun	Verb
1	Authentication	N/A	N/A
2	User	Email id, username, password	Log in, change user name/password, change information
3	Student	Name, email id, username, password, contact number, address, date of birth, blood group, registration number, roll number, session, attached hall, year/semester	Log in, change username/password, change information, download study materials, get push notification,
4	Teacher	Name, email id, username, password, designation, contacts number, year/semester, course code	Log in, change username/password, change information, upload, download and remove study materials, notify students, take attendance, edit attendance data
5	Administrator	Name, email id, username, password	Create account, change information, approves changes in accounts, create groups, distribute students in groups, updating information in student and teacher profile, assign teacher in groups and courses, notify users
6	Profile	Profile type	N/A
7	Group	Group name, year/semester	N/A

8	File	File name, file type, upload date, course code, group name	N/A
9	Course	Course name, course code	N/A
10	Notification	Notification type	N/A
`11	Attendance	Date, group, course code	N/A

6.5 Potential Classes

Table 13: Authentication

Authentication	
Attribute	**Methods**

	signUp()
	logIn()
	logOut()
	recoverAccount()
	validatyCheck()

Table 14: User

User	
Attribute	**Methods**

Email id
Username
Password
	logIn()
	logOut()
	recoverAccount()
	changeAccountInfo()
	sendNotification()

Table 15: Administrator

Administrator	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
	logIn()
	logOut()
	recoverAccount()
	createAccount()
	approveChangedInfo()
	createGroup()
	createCourse()
	assignStudent()
	assignTeacher()
	updateStudentInfo()
	updateTeacherInfo()
	sendNotification()

Table 16: Teacher

Teacher	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
Designation
Course code
year/semester
	logIn()
	logOut()
	recoverAccount()
	uploadFile()
	downloadFile()
	removeFile()
	takeAttendance()
	editAttendance()
	sendNotification()

Table 17: Student

Student	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
Address
	logIn()
	logOut()
	recoverAccount()
	uploadFile()
	downloadFile()
	removeFile()

Date of birth
Blood group
Registration no
Roll no
Session
Attached hall
year/semester
	receiveNotification()
	viewAttendance()

Table 18: Profile

Profile	
Attribute	**Methods**

Profile type	editInfo()
	removeInfo()

Table 19: Teacher Profile

TeacherProfile	
Attribute	**Methods**

Profile type	editTeacherInfo()
	removeTeacherInfo()

Table 20: Student Profile

StudentProfile	
Attribute	**Methods**

Profile type	editStudentInfo()
	removeStudentInfo()

Table 21: Group

Group	
Attribute	**Methods**

Group name
Year/semester
	createGroup()
	assignStudent()
	assignTeacher()

Table 22: File

File	
Attribute	**Methods**

File name
File type
Upload date
Course code
Group name
	uploadFile()
	downloadFile()
	removeFile()

Table 23: Course

Course	
Attribute	**Methods**

Course Name
Course code
	createCourse()
	assignStudent()
	assignTeacher()

Table 24: Notification

Notification	
Attribute	**Methods**
Notification type	generateNotification()

Table 25: Attendance

Attendance	
Attribute	**Methods**

Date
Group
Course
	takeAttendance()
	editAttendance()
	showAttendance()

6.6 Selected Classes

Table 26: Authentication

Authentication	
Attribute	**Methods**

	signUp()
	logIn()
	logOut()
	recoverAccount()
	validatyCheck()

Responsibilities	**Collaborative classes**
1. Sign up 2. Log in 3. Checks input validity 4. Log out 5. Recover account	User, Administrator, Teacher, Student

Table 27: User

User	
Attribute	**Methods**

Email id
Username
Password
	logIn()
	logOut()
	recoverAccount()
	changeAccountInfo()
	sendNotification()

Responsibilities	Collaborative classes
1. Recovers own account 2. Changes account information 3. Sends notification 4. Logs out	Authentication, Administrator, Teacher, Student, Notification, Profile

Table 28: Administrator

Administrator	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
	logIn()
	logOut()
	recoverAccount()
	createAccount()
	approveChangedInfo()
	createGroup()
	createCourse()
	assignStudent()
	assignTeacher()
	updateStudentInfo()
	updateTeacherInfo()
	sendNotification()

Responsibilities	**Collaborative classes**
1. Logs in 2. Creates account 3. Approves changes in accounts 4. Creates group 5. Creates course 6. Assigns students and teachers under groups and courses 7. Updates information of students and teachers 8. Sends notification 9. Signs out	Authentication, User, Teacher, Student, Group, Course, Notification, Profile

Table 29: Teacher

Teacher	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
Designation
Course code
year/semester
	logIn()
	logOut()
	recoverAccount()
	uploadFile()
	downloadFile()
	removeFile()
	takeAttendance()
	editAttendance()
	sendNotification()

Responsibilities	**Collaborative classes**
1. Logs in 2. Uploads file 3. Downloads file 4. Removes file 5. Takes attendance 6. Edits attendance 7. Sends notification 8. Signs out	Authentication, User, Administrator, Student, Profile, Group, File, Course, Notification, Attendance

Table 30: Student

Student	
Attribute	**Methods**

Name
Email id
Username
Password
Contact no
Address
Date of birth
Blood group
Registration no
Roll no
Session
Attached hall
year/semester
	logIn()
	logOut()
	recoverAccount()
	uploadFile()
	downloadFile()
	removeFile()
	receiveNotification()
	viewAttendance()

Responsibilities	Collaborative classes
1. Logs in 2. Uploads file 3. Downloads file 4. Removes file 5. Receives notification 6. Views attendance 7. Signs out	Authentication, Administrator, User, Teacher, Profile, Group, File, Course, Notification, Attendance

Table 31: Profile

Profile	
Attribute	**Methods**

Profile type	editInfo()
	removeInfo()

Responsibilities	**Collaborative classes**
1. Edits and removes information of profiles	User, Administrator, Teacher, Student

Table 32: Teacher profile

TeacherProfile	
Attribute	**Methods**

Profile type	editTeacherInfo()
	removeTeacherInfo()

Responsibilities	**Collaborative classes**
1. Edits information of teacher profiles	User, Administrator, Teacher

Table 33: Student profile

StudentProfile	
Attribute	**Methods**

Profile type	editStudentInfo()
	removeStudentInfo()

Responsibilities	Collaborative classes
1. Edits information of teacher profiles	User, Administrator, Student

Table 34: Group

Group	
Attribute	**Methods**

Group name
Year/semester
	createGroup()
	assignStudentIntoGroup()
	assignTeacherIntoGroup()

Responsibilities	**Collaborative classes**
1. Shows prompt to administrator to creates group 2. Shows prompt to administrator to assign students 3. Shows prompt to administrator to assign teachers	Administrator, Teacher, Student

Table 35: File

File	
Attribute	**Methods**

File name
File type
Upload date
Course code
Group name
	uploadFile()
	downloadFile()
	removeFile()

Responsibilities	**Collaborative classes**
1. Shows prompt to upload, download and remove file	Administrator, Teacher, Student

Table 36: Course

Course	
Attribute	**Methods**

Course Name
Course code
	createCourse()
	assignStudentIntoCourse()
	assignTeacherIntoCourse()

Responsibilities	**Collaborative classes**
1. Shows prompt to administrator to creates course 2. Shows prompt to administrator to assign students 3. Shows prompt to administrator to assign teachers	Administrator, Teacher, Student

Table 37: Notification

Notification	
Attribute	**Methods**
Notification type	generateNotification()
Responsibilities	**Collaborative classes**
1. Generates notification	Administrator, Teacher, Student

Table 38: Attendance

Attendance	
Attribute	**Methods**

Date
Group
Course
	takeAttendance()
	editAttendance()
	showAttendance()

Responsibilities	**Collaborative classes**
1. Shows list to teacher for taking attendance 2. Shows prompt to teacher for editing attendance 3. Shows attendance to students and teachers	Teacher, Student

6.6 Class Responsibility Collaboration (CRC) Diagram

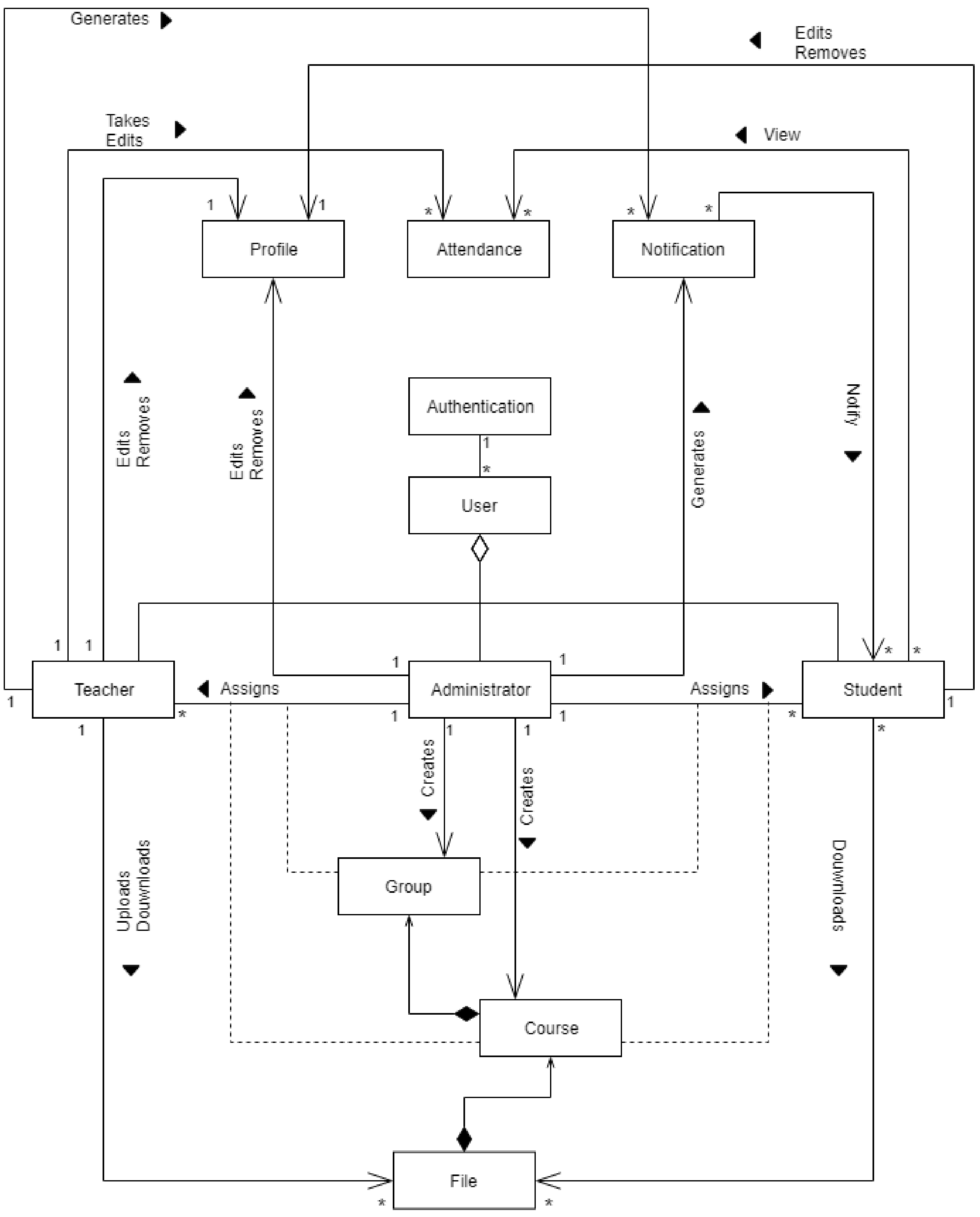

Figure 21: Class Responsibility Collaboration (CRC) Diagram

Chapter 7

Behavioral Modeling

7.1 State Transition Diagram

State diagram represents active states for each class the events (triggers). For this we identified all the events, their initiators and collaborators.

Identifying Events

Table 39: Event identification

Event	Primary object	Collaborator	Methods
Create account for teacher	Authentication	Administrator	signUp()
Create account for student	Authentication	Administrator	signUp()
Lets user log in	Authentication	Teacher, Student	logIn()
Allows to retry at wrong username and password	Authentication	Teacher, Student	logIn()
Sends username and password via email/SMS	Authentication	Teacher, Student	recoverAccount()
Validates input	Authentication	Teacher, Student, Administrator	validityCheck()
Allows to recover users account	Authentication	Teacher, Student	recoverAccount()
Allows to change account information	Authentication	Teacher, Student	recoverAccount()
Logs out of system	Authentication	Teacher, Student, Administrator	logOut()
Approves changes in user accounts	Authentication	Administrator	validatyCheck()
Creates groups	Group	Administrator	createGroup()
Assigns students under groups	Group	Administrator	assignStudentIntoGroup()

Assigns teachers under groups	Group	Administrator	assignTeacherIntoGroup()
Creates courses	Course	Administrator	createCourse()
Assigns students under courses	Course	Administrator	assignStudentIntoCourse()
Assigns teachers under courses	Course	Administrator	assignTeacherIntoCourse()
Sends notification	Notification	Teacher, Administrator	generateNotification() sendNotification()
Receives notification	Notification	Student	receiveNotification()
Allows administrator to edit information from teacher profile	TeacherProfile	Administrator	editTeacherInfo()
Allows teachers to edit information from teacher profile	TeacherProfile	Teacher	editTeacherInfo()
Allows administrator to remove information from teacher profile	TeacherProfile	Administrator	removeTeacherInfo()
Allows teachers to remove information from teacher profile	TeacherProfile	Teacher	removeTeacherInfo()
Allows administrator to edit information from student profile	StudentProfile	Administrator	editStudentInfo()
Allows students to edit information from student profile	StudentProfile	Student	editStudentInfo()
Allows administrator to remove information from student profile	StudentProfile	Administrator	removeStudentInfo()
Allows students to remove information from student profile	StudentProfile	Student	removeStudentInfo()
Allows user to upload files	File	Teacher, Student	uploadFile()
Allows user to download files	File	Teacher, Student	downloadFile()
Allows user to remove files	File	Teacher, Student	removeFile()

Allows teachers to take attendance	Attendance	Teacher	takeAttendance()
Allows teachers to edit attendance	Attendance	Teacher	editAttendance()
Allows teachers to view attendance data	Attendance	Teacher	viewAttendance()
Allows students to view attendance data	Attendance	Student	viewAttendance()

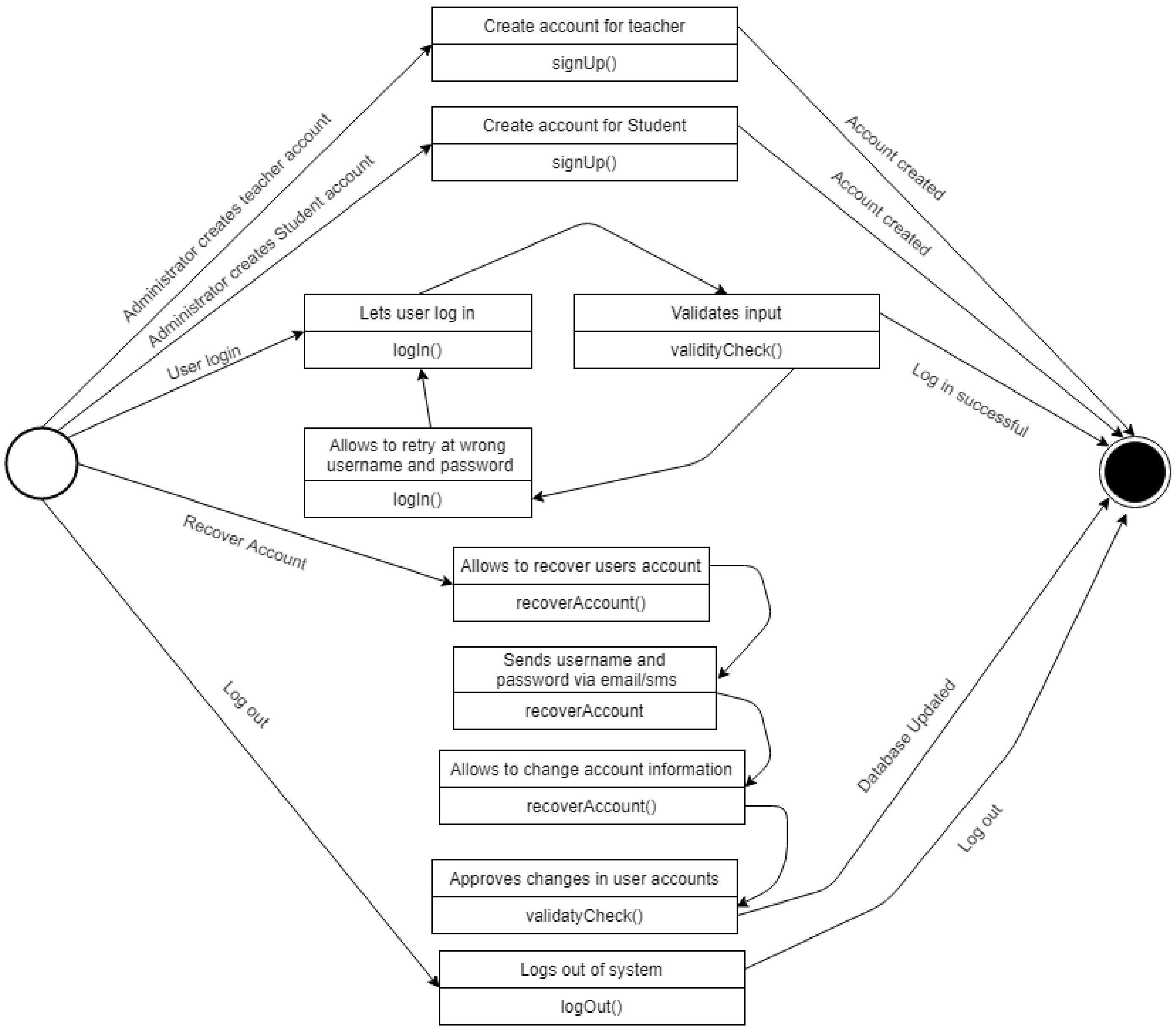
Create account for teacher
signUp()
Create account for Student
signUp()
Administrator creates teacher account
Administrator creates Student account
Account created
Account created
User login
Lets user log in
logIn()
Validates input
validityCheck()
Log in successful
Allows to retry at wrong username and password
logIn()
Recover Account
Allows to recover users account
recoverAccount()
Sends username and password via email/sms
recoverAccount
Allows to change account information
recoverAccount()
Approves changes in user accounts
validatyCheck()
Database Updated
Log out
Logs out of system
logOut()
Log out

Figure 22: Authentication

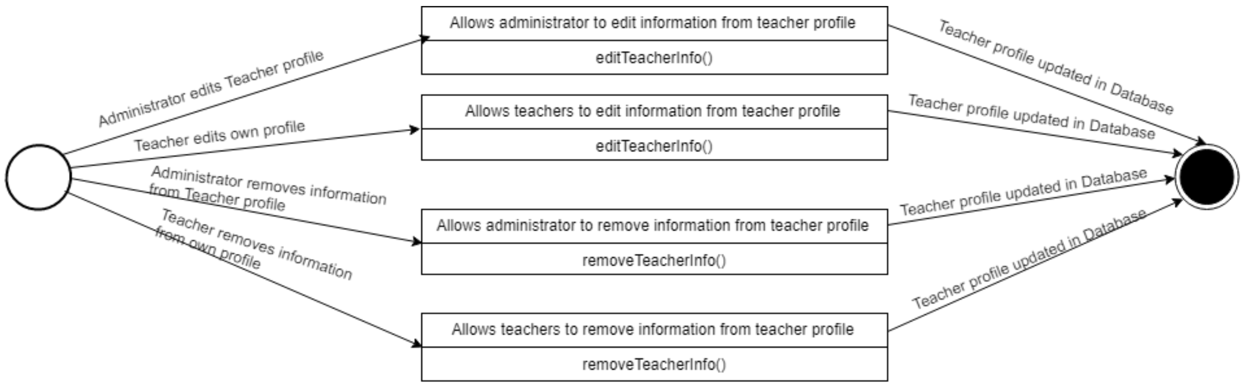
Allows administrator to edit information from teacher profile
editTeacherInfo()
Allows teachers to edit information from teacher profile
editTeacherInfo()
Allows administrator to remove information from teacher profile
removeTeacherInfo()
Allows teachers to remove information from teacher profile
removeTeacherInfo()
Administrator edits Teacher profile
Teacher edits own profile
Administrator removes information from Teacher profile
Teacher removes information from own profile
Teacher profile updated in Database
Teacher profile updated in Database
Teacher profile updated in Database
Teacher profile updated in Database

Figure 23: Teacher profile

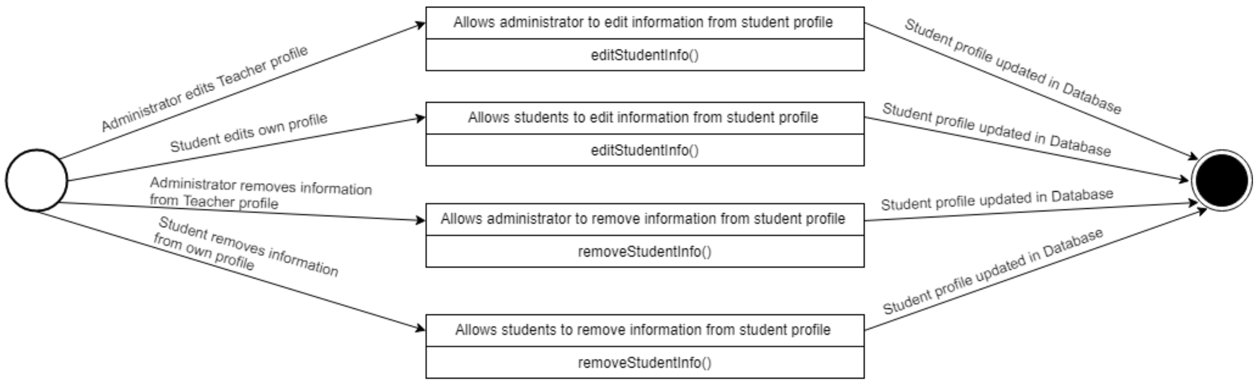
Allows administrator to edit information from student profile
editStudentInfo()
Allows students to edit information from student profile
editStudentInfo()
Allows administrator to remove information from student profile
removeStudentInfo()
Allows students to remove information from student profile
removeStudentInfo()
Administrator edits Teacher profile
Student edits own profile
Administrator removes information from Teacher profile
Student removes information from own profile
Student profile updated in Database
Student profile updated in Database
Student profile updated in Database
Student profile updated in Database

Figure 24: Student profile

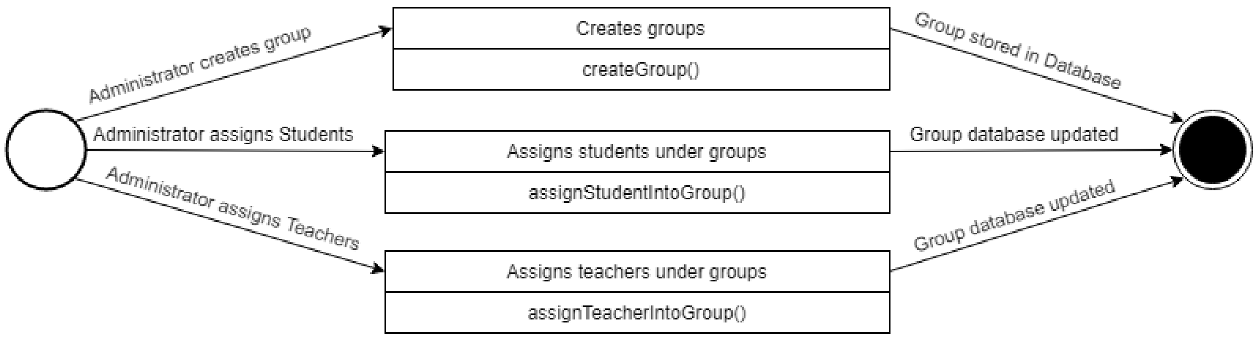
Creates groups
createGroup()
Assigns students under groups
assignStudentIntoGroup()
Assigns teachers under groups
assignTeacherIntoGroup()
Administrator creates group
Administrator assigns Students
Administrator assigns Teachers
Group stored in Database
Group database updated
Group database updated

Figure 25: Group

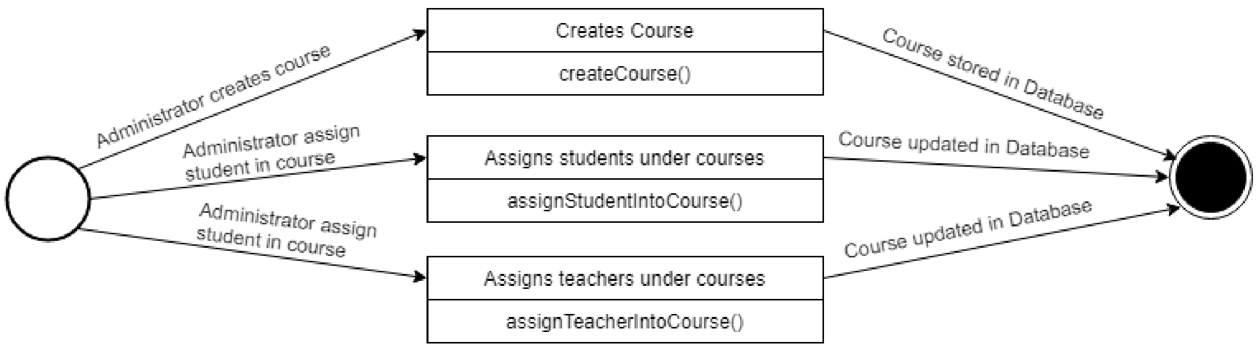
Creates Course
createCourse()
Assigns students under courses
assignStudentIntoCourse()
Assigns teachers under courses
assignTeacherIntoCourse()
Administrator creates course
Administrator assign student in course
Administrator assign student in course
Course stored in Database
Course updated in Database
Course updated in Database

Figure 26: Course

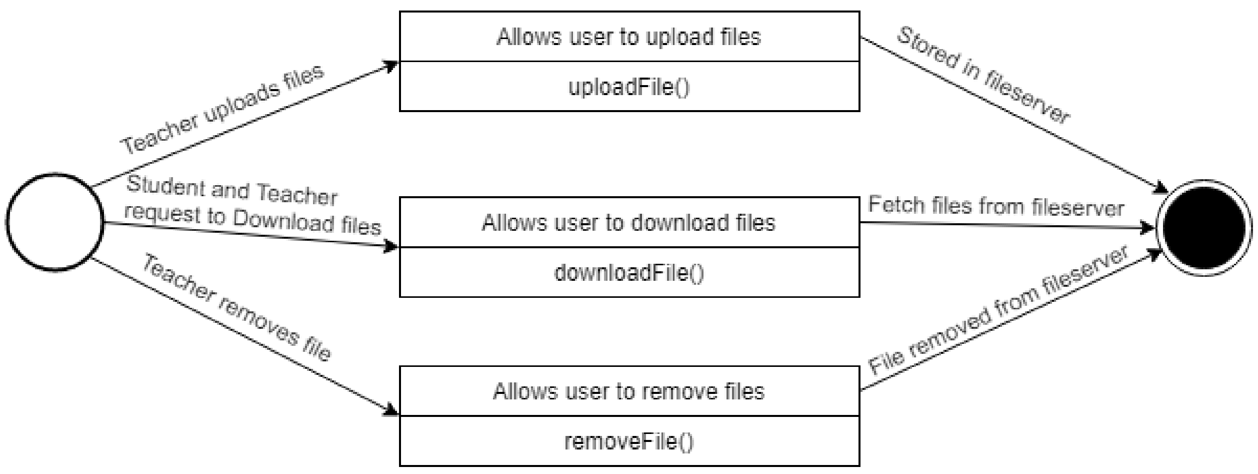
Allows user to upload files
uploadFile()
Allows user to download files
downloadFile()
Allows user to remove files
removeFile()
Teacher uploads files
Student and Teacher request to Download files
Teacher removes file
Stored in fileserver
Fetch files from fileserver
File removed from fileserver

Figure 27: File

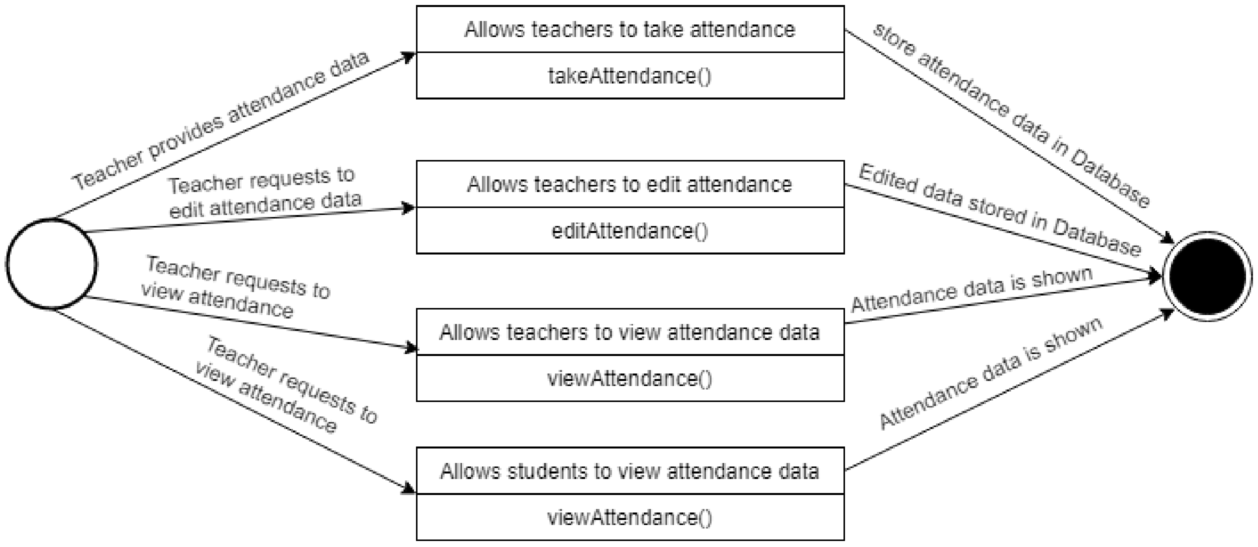
Allows teachers to take attendance
takeAttendance()
Allows teachers to edit attendance
editAttendance()
Allows teachers to view attendance data
viewAttendance()
Allows students to view attendance data
viewAttendance()
Teacher provides attendance data
Teacher requests to edit attendance data
Teacher requests to view attendance
Teacher requests to view attendance
store attendance data in Database
Edited data stored in Database
Attendance data is shown
Attendance data is shown

Figure 28: Attendance

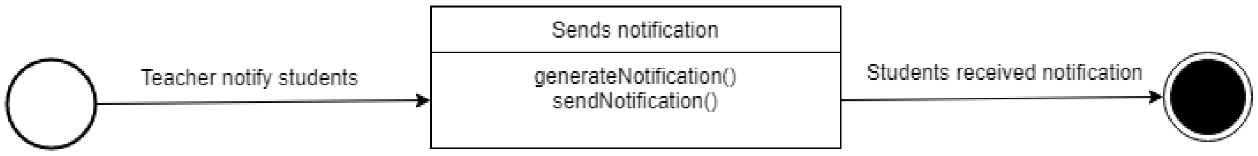

Figure 29: Notification

7.2 Sequence Diagram

7.3 Data Flow Diagram

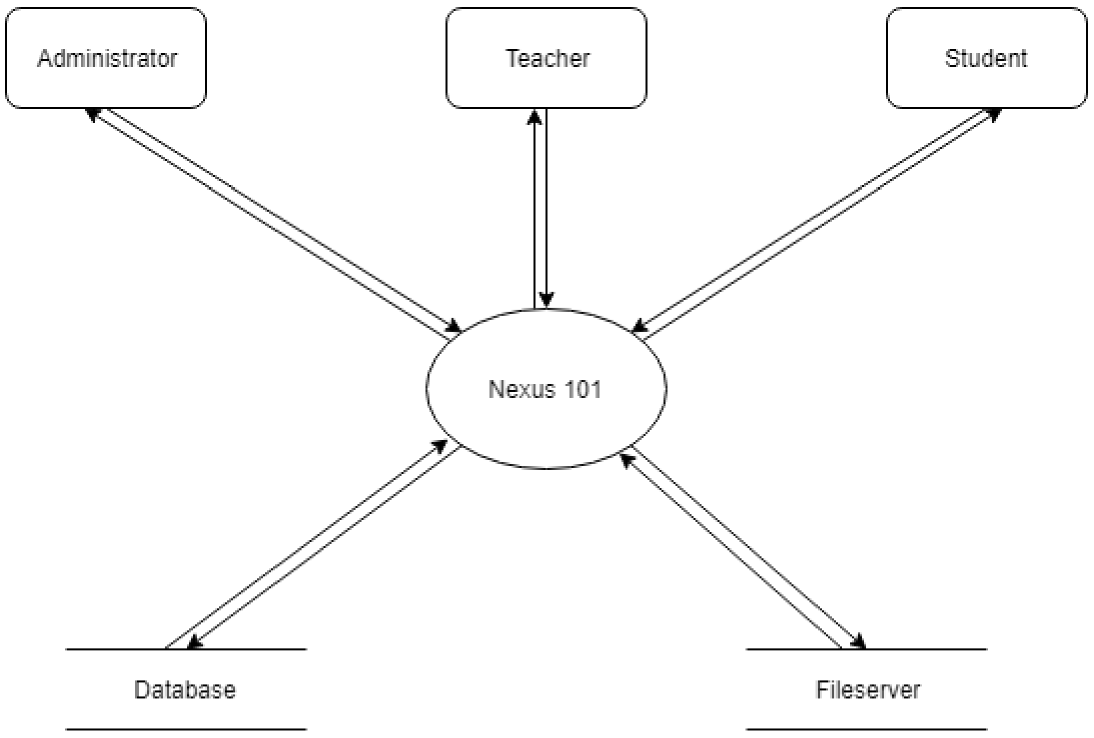

Figure 30: level 0

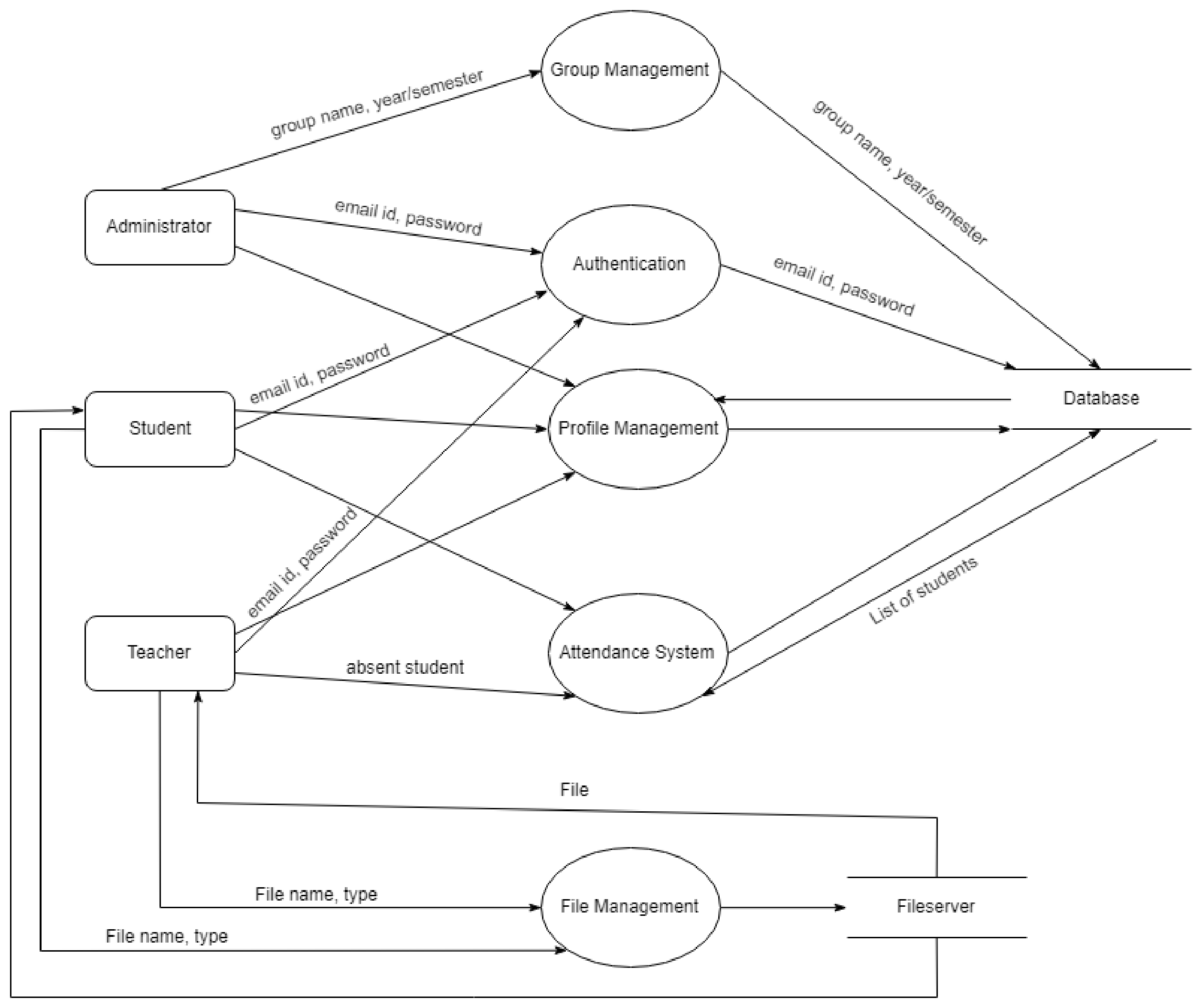

Figure 31: level 1

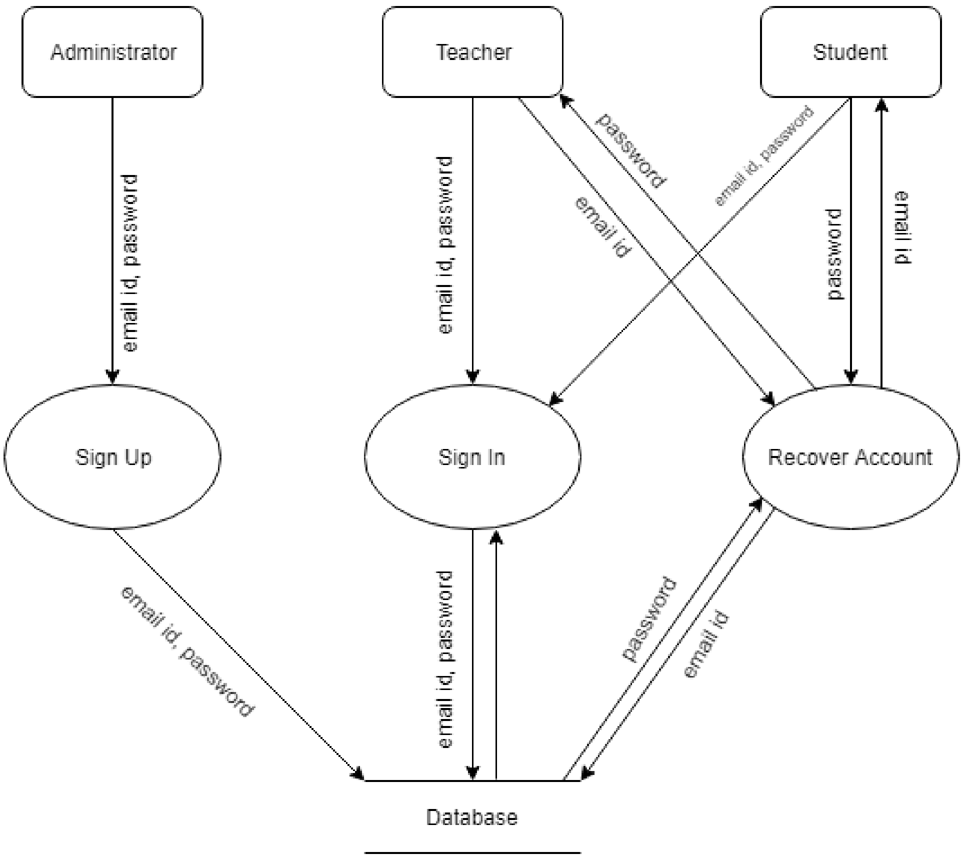

Figure 32: level 1.1

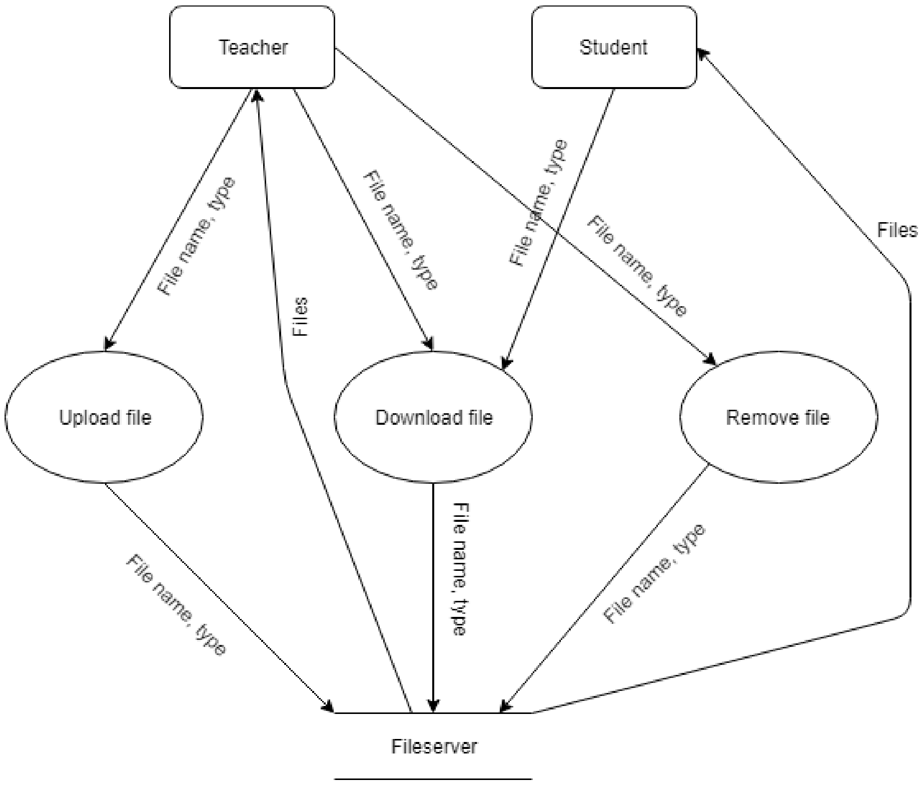

Figure 33: level 1.2

Chapter 8

Conclusion

From this SRS report on Nexus 101: An Academic Workflow Automation System, the readers will get a clear and easy view of the overall system of management system of the existing workflows. This SRS document can be used effectively to maintain the software development cycle. It will be very easy to conduct the whole project using SRS. We tried best to remove all dependencies and make an effective and fully designed SRS.