SE__Group1_Project_-_Report_1_

Submission date: 4th November, 2021.

Proposed to: Mr. Daniel Chaytor

Submitted by: Group 1

Project URL: https://github.com/amaduQuincy/Student-Database-Management-System.git

Team members:

ID	NAME	EMAIL
24903	Amadu Abubakarr Bah (Team Leader)	amaduabah044@gmail.com
24498	Augustina Hawanatu Brima	augustinahbrima@gmail.com
25332	Charles Thomas	fengehunthomas@gmail.com
24419	Abubakarr Turay	abturay29@gmail.com

INDIVIDUAL WORK BREAKDOWN

AMADU ABUBAKARR BAH	login management	Admin login
		Staff login
		Student login
		Logout funciton
	dashboards	Admin dashboard
		Staff dashboard
		Student dashboard
	Staff management	Add, edit, and delete staff record
AUGUSTINA HAWANATU BRIMA	Attendance management	Staff record attendance
		Student view attendance
		Admin view attendance
	Result management	Staff record a result
		Student view result
ABUBAKARR TURAY	Student management	Add, edit, and delete student
	Course management	Add, edit, and delete course
	Subject management	Add, edit, and delete subject
	Session management	Add, edit, and delete session
CHARLES THOMAS	Leave management	Student apply for leave
		Staff apply for leave
		Admin respond to leave
		Leave status
	Feedback management	Student send feedback
		Staffs send feedback
		Admin reply feedback
		Reply notification

RESPONSIBILITY MATRIX

DOMAIN CONCEPTS													
USE CASE	PW	INTERFACE PAGE	CONTROLLER	URLCONFIG	LOGIN CHECKER	DATABASE CONNECTION	DASHBOARD MAKER	PAGE MAKER	REQUEST-X	STAFFS VIEW	ADMIN VIEW	STUDENTS VIEW	FORM VALIDATOR
UC-1	8	x	x	x	x	x		x	x				x
UC-2	24	x	x	x	x	x	x	x	x				x
UC-3	5	x	x	x	x	x		x	x	X			x
UC-4	5	x	x	x	x	x		x	x	X			x

UC-5	10	x	x	x	x	x		x	x	X		x	x
UC-6	4	x	x	x	x	x		x	x	x		x	x
UC-7	10	x	x	x	x	x		x	x		x	X	x
UC-8	5	x	x	x	x	x		x	x			x	x
UC-9	4	x	x	x	x	x		x	x		x		x
UC-10	4	x	x	x	x	x		x	x		x		x
UC-11	4	x	x	x	x	x		x	x		x		x
UC-12	4	x	x	x	x	x		x	x		x		x
UC-13	4	x	x	x	x	x		x	x		x		x
UC-14	2	x	x	x	x	x		x	x		x		x
UC-15	2	x	x	x	x	x		x	x		x		x
UC-16	16	x	x	x	x	x		x	x	x	x	x	x
MAX PW		24	24	24	24	24	24	24	24	16	16	16	24
TOTAL PW		111	111	111	111	111	24	111	111	40	49	29	111

Table of Contents

Part 1:

Part 2:

Part 3:

Project Management (described in Section 7 below) and References (described in Section 8 below)

References

- https://www.brainkart.com/article/UML-Operation-Contract_9993/ (Friday, 4 November 2021)
- https://static.packt-cdn.com/products/9781783986644/graphics/6644OS_01_01.jpg (Accessed: 1 november 2021)
- https://docs.djangoproject.com/en/3.2/ (1 November 2021)
- https://i.ytimg.com/vi/bgSToGS5J1E/maxresdefault.jpg Thursday, 4 November 2021)
- https://www.lucidchart.com/pages/ (Accessed: October 29 2021)

Part 1:

Customer Problem Statement

a. Problem Statement

The school administrators who school related records find it cumbersome to add, maintain, store and retrieve the records.

Many schools/universities are operated using a pen and paper work. The system starts with the registration of a new student. When a student enters the school for registration he/she is greeted with a long queue, where registering students are aligned. The registrar uses a pen and a book to collect students' information. Such as, their name, date of birth, sex, attendance, results etc. after registration the books containing students' information is placed in a shelf for record keeping. The students whose information these shelves are containing exceed five thousand. To search for a particular student information, it involves a hired staff(s) to locate the particular shelf containing the book containing the student information. The staff would have to traverse each record to locate the information of the desired student. To maintain a student record, that is to update or delete, involves locating and scratching out the whole record and rewriting it in a new book. The current system is unreliable. As staffs and administrators maintain their own separate data.

Staffs also find it challenging to record and main the attendance records and grades of student. Students also find it challenging to view their grades as the school system is slow and would have to wait for staffs to accumulate their grades. In case of having to apply for a leave, students would have to come to the school and fill in a printed form, and take it to the administrator, who, to reply the leave application would have to write a letter and contact students to collect the letters.

PROBLEMS IN THE EXISTING SYSTEM:

Students have to stand in a queue waiting to be attended to. The existing system require more staff as there is a lot of paper work involved. Present System is time-consuming and also results in lack of getting inefficient results. Storing and accessing the data in the form of account books is a tedious work. It requires a lot of laborious work. It may often yield undesired results. Maintaining these records as piles may turn out to be a costlier task than any other of the colleges and institutions. Records are unreliable as data is redundant.

RISKS INVOLVED IN EXISTING SYSTEM:

Some of the risks involved in the present system are:

During the manual entrance of student information, if any mistake is done at a point, then this becomes cumulative and leads to adverse consequences. If there is any need to retrieve records it may seem to be difficult to search. In case of fire outbreak, there is no hope of retrieving lost student information, as it is not feasible to replicate students' records as backup.

b. Glossary of Terms

SIMS	Student information management system
Admin	Administrator
Self	initiating actor
REQ-x	requirement number
UC-x	use case number
PW	priority weight
WSGI SERVER	WSGI servers handle processing requests from the web server and deciding how to communicate those requests to an application framework's process.
view	A view function, or view for short, is a Python function that takes a Web request and returns a Web response.
URLconf	URLconf is a set of patterns that Django will try to match the requested URL to find the correct view
template	Provides a convenient way to generate dynamic HTML pages by using its template system. A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.
model	A model is a Python class that inherits from the Model class. The model class defines a new Kind of datastore entity and the properties the Kind is expected to take.
migrations	Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema.

2. System Requirements

a. Enumerated Functional Requirements

REQ-X	**Priority weight**	**description**
REQ-1	8	As a lecturer, I can See the Overall Summary Charts related to their students, their subjects, leave status online and more, so that I can see all information relating to the students and I.
REQ-2	5	As a lecturer, I can Take/Update Students Attendance
REQ-3	5	As a lecturer, I can Add/Update Result online
REQ-4	2	As a lecturer, I can Apply for Leave online

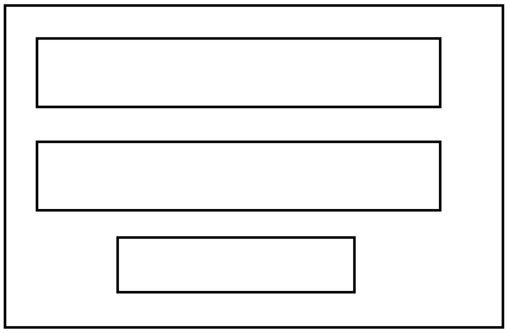
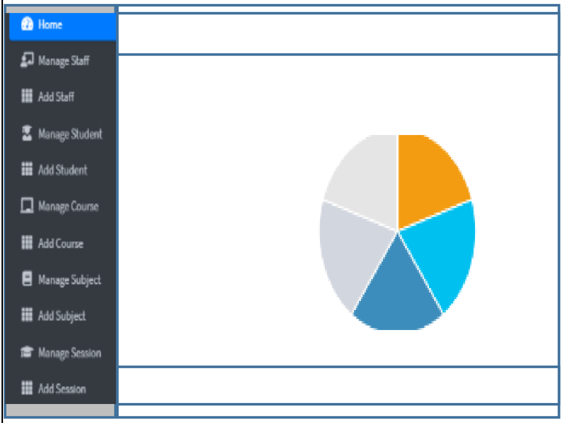
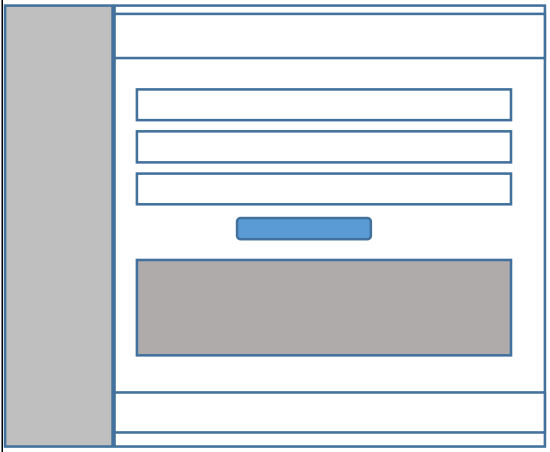
REQ-5	2	As a lecturer, I can Send Feedback to the administrator online, so that i can simply suggest improvements for the system and the school
REQ-6	8	As a student, I can see the Overall Summary Charts related to their attendance, their subjects, leave status online, so that I can simply see all information relating to me
REQ-7	5	As a student, I can view my attendance online
REQ-8	5	As a student, I can view my result online
REQ-9	2	As a student, I can apply for leave online
REQ-10	2	As a student, I can Send Feedback to the administrator online, so that i can simply suggest improvements for the system and the school
REQ-11	8	As an admin, I can See Overall Summary Charts of Students Performance, Staffs Performances, Courses, Subjects, Leave, online and more, so that I can simply see information relating to the student and lecturers
REQ-12	4	As an admin, I can Manage Staffs (Add, Update and Delete) records online
REQ-13	4	As an admin, I can Manage Students (Add, Update and Delete) records online
REQ-14	4	As an admin, I can Manage Course (Add, Update and Delete) online
REQ-15	4	As an admin, I can Manage Subjects (Add, Update and Delete) online
REQ-16	4	As an admin, I can Manage Sessions (Add, Update and Delete) online
REQ-17	5	As an admin, I can View Student Attendance online
REQ-18	2	As an admin, I can Review and Reply Student/Staff Feedbacks online
REQ-19	2	As an admin, I can Review (Approve/Reject) Student/Staff Leave online

b. Enumerated Non-functional Requirements

REQ-X	Priority weight	description
REQ-20	8	As a customer I want the system to perform mentioned features, so that I can easily manage information relating to students
REQ-21	8	As a user, I want the system to be efficient and effective so that I can simply perform tasks online
REQ-22	8	As a user I want the system to have high probability of failure-free when introduced to operational environment
REQ-23	8	As a user, under certain predicaments. I want the system to be able to perform its functions in less time with less resources
REQ-24	8	As a customer I want the system to be easy to maintain so that I can scale up, and solve new real world problems
REQ-25	8	As a customer I want the system to be secured so that the school can maintain its integrity

c. User Interface Requirements

REQ-X	priority	description	Graphical illustration
REQ-26	8	**Login form** that collects data from users in order to authenticate and	

		specify user type. It contains: a field for user name and a field for password. With a sign in button to send request	
REQ-27	8	**Admin, staffs and students dashboard** containing overall summary of related data and charts and graphs. It contains header and footer. And navigation panel	 Home Manage Staff Add Staff Manage Student Add Student Manage Course Add Course Manage Subject Add Subject Manage Session Add Session
REQ-28	8	Page that enables user to send request and view response	
REQ-29	8	**Appearance:** The system should be attractive according to the administrators, lecturers and students. The design and the color should make users feel comfortable when using the system instead of flashing useless colors on the screen. The design	

		should also reflect the seriousness of the school environment.	
REQ-30	8	**Style:** The overall style should be built up easily in order for users to use it easily and efficiently. After accessing the system, the users should feel comfortable while looking at it and browsing through it. The design should not be too colorful to maintain a certain seriousness of the design of the school but at the same time it should not be too boring for the eye, so that it can appear pleasant to use.	
REQ-31	8	**Ease of Use:** The system should have an easily understandable design in order for users to use it. It should provide the necessary information when the user commits possible errors. It should indicate the several possibilities that the user has to go on in using the system. The user will be allowed to undo any of the operation computed or, for irreversible operation, will always be asked to double-check their choice in case they misunderstood the option or clicked on a button by accident.	
REQ-32	8	**Personalization and Internationalization:** the SIMS should be presented in different languages in case of its use in different countries.	
REQ-33	8	**Learning:** It should not require a massive amount of time learning how to use the system. The goal is to create a self-explanatory system that does not ideally need any tutorial section.	
REQ-34	8	**Understandability and Politeness:** The system will not use any term that might be	

		incomprehensible or offensive to users.	
REQ-35	8	**Accessibility:** The system should also consider people with common disabilities and should make possible access to SIMS. For example, since approximately 20% of males are red-green colorblind, the system should be designed in different colors avoiding red and green. Also, all the buttons that need to be clicked should be big enough to be clearly distinguished also by people who have sight issues.	

Part 2:

3. Functional Requirements Specification

UC-X	NAME
UC-1	login
UC-2	ViewOverallSummary
UC-3	recordAttendance
UC-4	addResults
UC-5	applyForLeave
UC-6	giveFeedBack
UC-7	viewAttendance
UC-8	ViewResult
UC-9	ManageStaffs
UC-10	ManageStudents
UC-11	ManageCourse
UC-12	ManageSubjects
UC-13	ManageSessions
UC-14	ManageLeaveApplications
UC-15	ManageFeedBack
UC-16	search

a. Stakeholders

-school

-administrators

-staffs

-students

b. Actors and Goals

Actor	Role	Type	goal
Administrator	Manage school related administrative tasks	Initiating	Easy online management of school and system related administrative tasks
Lecturer/staff	Manage students related information and participate in enhancing the school system	Initiating	Easy online management of students related information
Student	View school related information and participate in enhancing the school system	Initiating	Easy online viewing of school related information
Database	Provide the platform of where school related information can safely reside	participating	

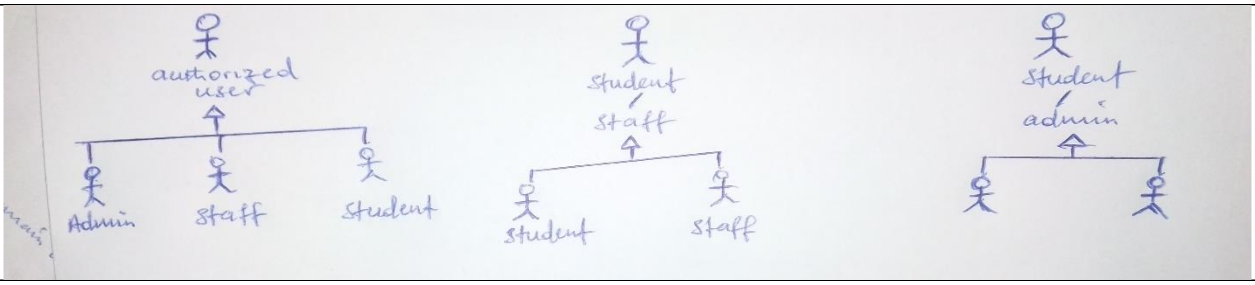
c. Use Cases

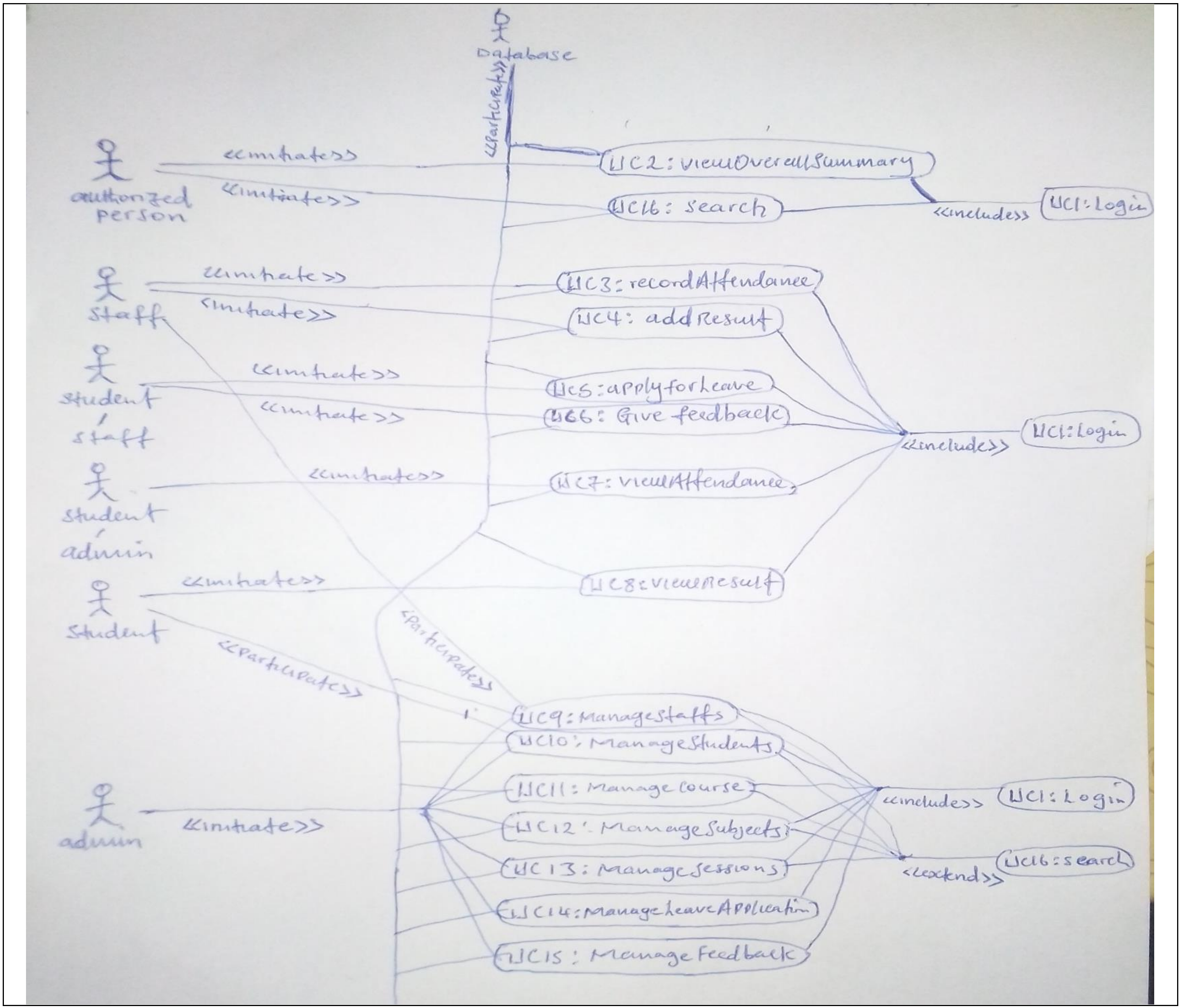
i. Casual Description

Use case	Casual description	Related requirements
login (UC-1)	A usage scenario of when one of the initiating actors wants to gain access to 'self' dashboard to perform tasks. 'self' is the initiating actor (admin/staff/student).	REQ-25
ViewOverallSummary (UC-2)	A usage scenario of when one of the initiating actors gets to view an overall summary relating to 'self'	REQ-1, REQ-6, REQ-11
recordAttendance (UC-3)	A usage scenario of when a lecturer/staff records a new attendance or modify an existing one	REQ-2
addResults (UC-4)	A usage scenario of when a lecturer/staff records the result of a student on a subject in a course or modify an existing one	REQ-3
applyForLeave (UC-5)	A usage scenario of when a student/lecturer fill in a form to apply for a leave	REQ-4 and REQ-9
giveFeedBack (UC-6)	A usage scenario of when a student/lecturer fill in a form to give feedback about the school activities or the system	REQ-5 and REQ-10
viewAttendance (UC-7)	A usage scenario of when a/an admin/student view the attendance recorded by a particular lecturer	REQ-7 and REQ-17
ViewResult (UC-8)	A usage scenario of when a student gets to view his result on a particular subject	REQ-8

ManageStaffs (UC-9)	A usage scenario of when an administrator gets to fill in a form to add/edit the record of a staff or delete the record of a staff	REQ-12
ManageStudents (UC-10)	A usage scenario of when an administrator gets to fill in a form to add/edit the record of a student or delete the record of a student	REQ-13
ManageCourse (UC-11)	A usage scenario of when an administrator gets to fill in a form to add/edit a record of a course or delete the record of a course	REQ-14
ManageSubjects (UC-12)	A usage scenario of when an administrator gets to fill in a form to add/edit the record of a subject or delete the record of the subject	REQ-15
ManageSessions (UC-13)	A usage scenario of when an administrator gets to fill in a form to add/edit the record of a session year or delete the record of a session year	REQ-16
ManageLeaveApplications (UC-14)	A usage scenario of when an administrator gets to approve or reject the leave applications of a student and staffs	REQ-19
ManageFeedBack (UC-15)	A usage scenario of when an administrator gets to view and reply the feedbacks given by students and staffs	REQ-18
search (UC-16)	A usage scenario of when a user gets to search for a particular record	REQ-21 and REQ-23

ii. Use Case Diagram

USE CASE DIAGRAM
 authorized user Admin staff student student / staff student staff student / admin



iii. Traceability Matrix

REQ-x	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15	UC-16
REQ-1	8		x														
REQ-2	5			x													
REQ-3	5				x												
REQ-4	2					x											
REQ-5	2						x										
REQ-6	8		x			x											

REQ-7	5							x									
REQ-8	5								x								
REQ-9	2																
REQ-10	2						x										
REQ-11	8		x														
REQ-12	4									x							
REQ-13	4										x						
REQ-14	4											x					
REQ-15	4												x				
REQ-16	4													x			
REQ-17	5							x									
REQ-18	2															x	
REQ-19	2														x		
REQ-21	8																x
REQ-23	8																x
REQ-25	8	x															
Max points		8	8	5	5	8	2	5	5	4	4	4	4	4	2	2	8
Total point		8	24	5	5	10	4	10	5	4	4	4	4	4	2	2	16

iv. Fully-Dressed Description

Use Case UC-1:	**login**
Related Requirements:	REQ-25
Initiating Actor:	Any of: admin, staff, student
Actor's Goal:	Gain access to admin/staff/student dashboard
Participating Actors:	database
Preconditions:	• The set of valid username and password stored in the system database is non-empty. • the system prompts the user to input username and password
Postconditions:	The website is navigated to the admin/staff/student dashboard
Flow of Events for Main Success Scenario:	

→	1.	The actor visit the homepage
←	2.	The system prompts the actor for his username and password
→	3.	The actor enters valid username and password and click login
←	4.	The system (a) verify key validity and (b) navigate to actors dashboard
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

<table>
<tr><td colspan="2"><b>Use Case :</b> (UC-2)</td><td>ViewOverallSummary</td></tr>
<tr><td colspan="2"><b>Related Requirements:</b></td><td>REQ-1, REQ-6, REQ-11</td></tr>
<tr><td colspan="2"><b>Initiating Actor:</b></td><td>Any of: admin/staff/student</td></tr>
<tr><td colspan="2"><b>Actor’s Goal:</b></td><td>View an overall summary or holistic view of related data</td></tr>
<tr><td colspan="2"><b>Participating Actors:</b></td><td>database</td></tr>
<tr><td colspan="2"><b>Preconditions:</b></td><td>• set of related data in system database is non-empty
• actor is logged in the system</td></tr>
<tr><td colspan="2"><b>Postconditions:</b></td><td>none</td></tr>
<tr><td colspan="3"><b>Flow of Events for Main Success Scenario:</b></td></tr>
<tr><td><b>--</b></td><td>1.</td><td><<include>> UC1: login</td></tr>
<tr><td>←</td><td>2.</td><td>The system (a)connect to database and retrieve related data of actor (b) sums the quantitative data and (c) displays it different charts and graphs</td></tr>
<tr><td colspan="3"><b>Flow of Events for Extensions (Alternate Scenarios):</b></td></tr>
<tr><td>←</td><td>2a.</td><td>System (a) detects database to be empty and (b) signals the actor</td></tr>
</table>

Use Case (UC-3) **:**	recordAttendance
Related Requirements:	REQ-2

Initiating Actor:	staff
Actor’s Goal:	Save a record of students that are present or absent in a subject session
Participating Actors:	database
Preconditions:	• a set of students&subjects&session year records are stored in system database • actor is logged in the system
Postconditions:	none

Flow of Events for Main Success Scenario:		
→	1.	Actor selects take attendance or update attendance function
←	2.	The system prompts the actor to select a subject and session year
→	3.	The actor selects subject and session year
←	4.	The system prompts for the date and displays all student in the subject with check boxes
→	5.	Actor uncheck students that are absent and select the ‘save ’ function
←	6.	System records data in appropriate relations and notify the actor of a successful record recorded

Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Use Case (UC-4) **:**	addResults
Related Requirements:	REQ-3
Initiating Actor:	staff
Actor’s Goal:	Save a record of a student’s result on a particular subject, in a particular session year
Participating Actors:	database
Preconditions:	• a set of students&subjects&session year records are stored in system database

	• actor is logged in the system	
Postconditions:	none	
Flow of Events for Main Success Scenario:		
→	1.	Actor selects add result function
←	2.	The system prompts the actor to select a subject and session year
→	3.	The actor selects subject and session year
←	4.	The system prompts to select a student and enter the assignment and exam result
→	5.	Actor submit required data
←	6.	System records data in appropriate relations and notify the actor of a successful record recorded
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Use Case (UC-5) **:**	applyForLeave	
Related Requirements:	REQ-4 and REQ-9	
Initiating Actor:	Any of: staff/student	
Actor's Goal:	Send a leave application to the administrator	
Participating Actors:	• Database • Administrator	
Preconditions:	• actor is logged in the system	
Postconditions:	none	
Flow of Events for Main Success Scenario:		
→	1.	Actor selects apply for leave function
←	2.	The system prompts the actor to enter a date and leave reason
→	3.	The actor enters data and select apply for leave function

←	4.	System (a) records data in appropriate relations and notify the actor of a successful leave application and (b) displays the leave application status
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Use Case (UC-6) **:**	giveFeedBack
Related Requirements:	REQ-5 and REQ-10
Initiating Actor:	Any of: staff/student
Actor's Goal:	Give a feedback about the system
Participating Actors:	database
Preconditions:	• actor is logged in the system
Postconditions:	none

Flow of Events for Main Success Scenario:		
→	1.	Actor selects feedback function
←	2.	The system prompts the actor to enter feedback message
→	3.	The actor enters data and select send feedback function
←	4.	System (a) records data in appropriate relations and notify the actor of a successful feedback submitted and (b) displays the feedback submission status
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Use Case (UC-7) **:**	viewAttendance

Related Requirements:	REQ-7 and REQ-17	
Initiating Actor:	Admin/student	
Actor's Goal:	View an attendance record recorded by a staff	
Participating Actors:	database	
Preconditions:	• actor is logged in the system • attendance records are stored in database	
Postconditions:	none	
Flow of Events for Main Success Scenario:		
→	1.	Actor selects view attendance function
←	2.	The system prompts (a) a student for subject, start date and end date or (b) the admin for subject and session year
→	2.b.1	The admin selects the fetch attendance data
←	2.b.2	The system prompts the administrator to select attendance date
→	3.	The actor enters data and select fetch attendance data
←	4.	System retrieve attendance data from database and displays it to the actor
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Use Case (UC-8) **:**	ViewResult
Related Requirements:	REQ-8
Initiating Actor:	student
Actor's Goal:	View his or her result on a particular subject
Participating Actors:	database

Preconditions:		• actor is logged in the system • set of result records are present in database
Postconditions:		none
Flow of Events for Main Success Scenario:		
→	1.	Actor select view result function
←	2.	The system retrieve result records from database and display it to the actor

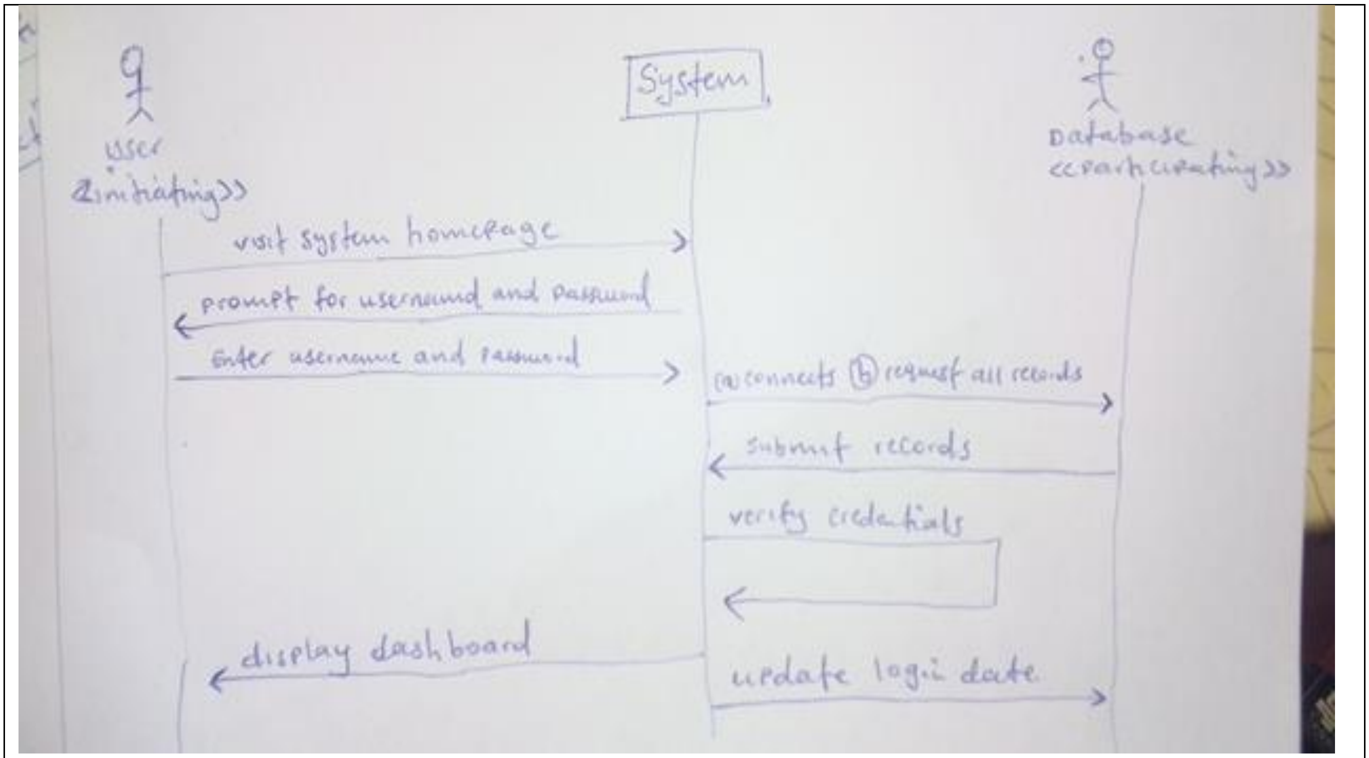
Use Case (UC-10) **:**		ManageStudents
Related Requirements:		REQ-13
Initiating Actor:		Admin
Actor's Goal:		Add/edit/delete student record
Participating Actors:		• database • student
Preconditions:		• actor is logged in the system
Postconditions:		none
Flow of Events for Main Success Scenario:		
→	1.	Actor selects manage students function
←	2.	The system displays current students details and provide the add/edit/delete functions
→	2.a.1	Actor selects the add/edit function
←	2.a.2	System prompts for student details including student profile picture
→	2.a.3	Actor enters data and select the add student function
←	2.a.4	System record data in database and notify the actor of a successful new record saved
→	2.b.1	The actor selects the function delete
←	2.b.2	System delete the record of the select student and notify the actor
Flow of Events for Extensions (Alternate Scenarios):		
→	2.a.3	The actor enters invalid data

←	2.a.4	System (a) detects invalid credentials and (b) signals the actor

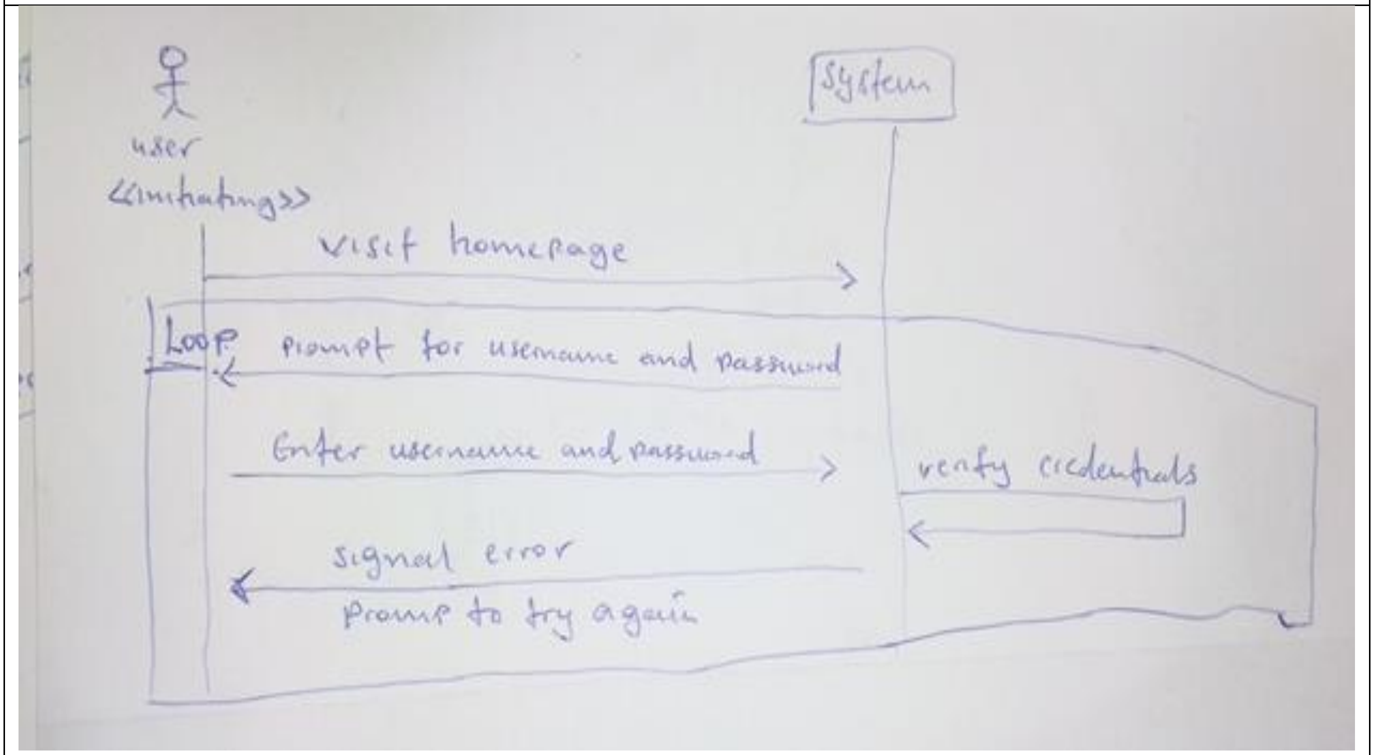
Use Case (UC-14) **:**		ManageLeaveApplications
Related Requirements:		REQ-19
Initiating Actor:		admin
Actor's Goal:		View, approve or reject leave applications submitted by students and staffs
Participating Actors:		• Database • Student • staff
Preconditions:		• actor is logged in the system • Set of leave applications are present in system database
Postconditions:		none
Flow of Events for Main Success Scenario:		
→	1.	Actor selects the student function or staff leave function
←	2.	The system (a) display details of current leave applications and (b) provide the functions: 'approve' and 'reject'
→	3.	Actor selects the approve/reject function
←	4.	System (a) records data in appropriate relations and displays to the actor of leave application response decision and (b) notify the student/staff of leave status
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

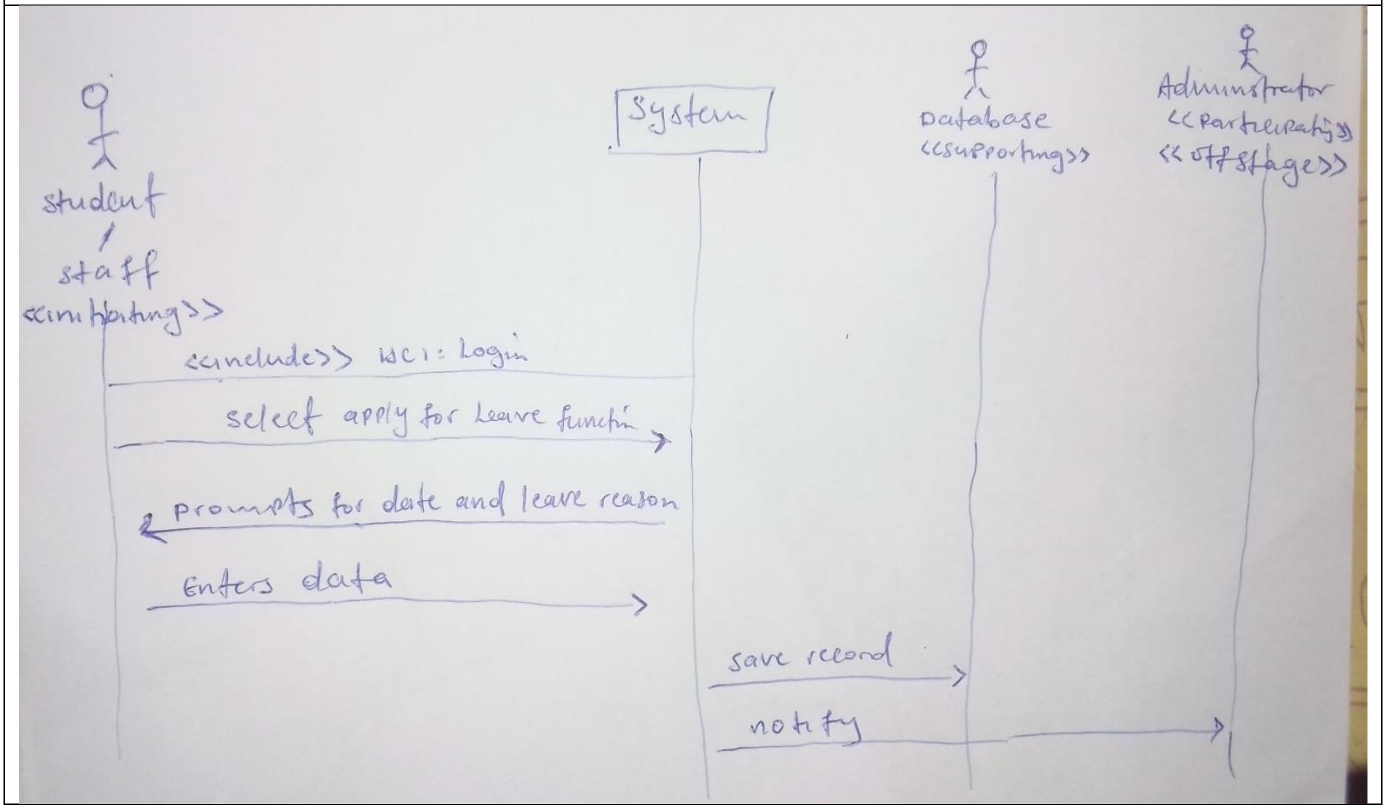
d. System Sequence Diagrams

Use case: **UC-1: login:** main success scenario

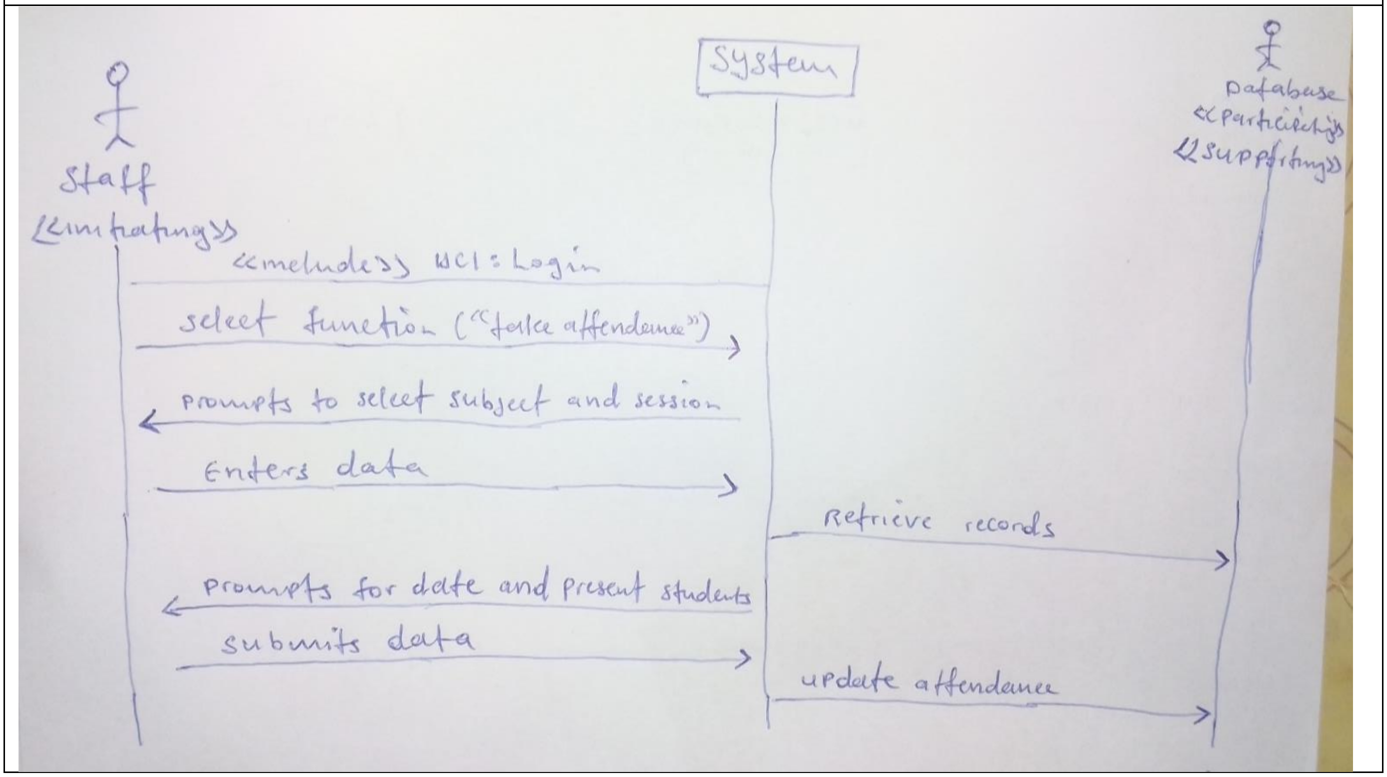


Use case: **UC-1: login:** alternate scenario

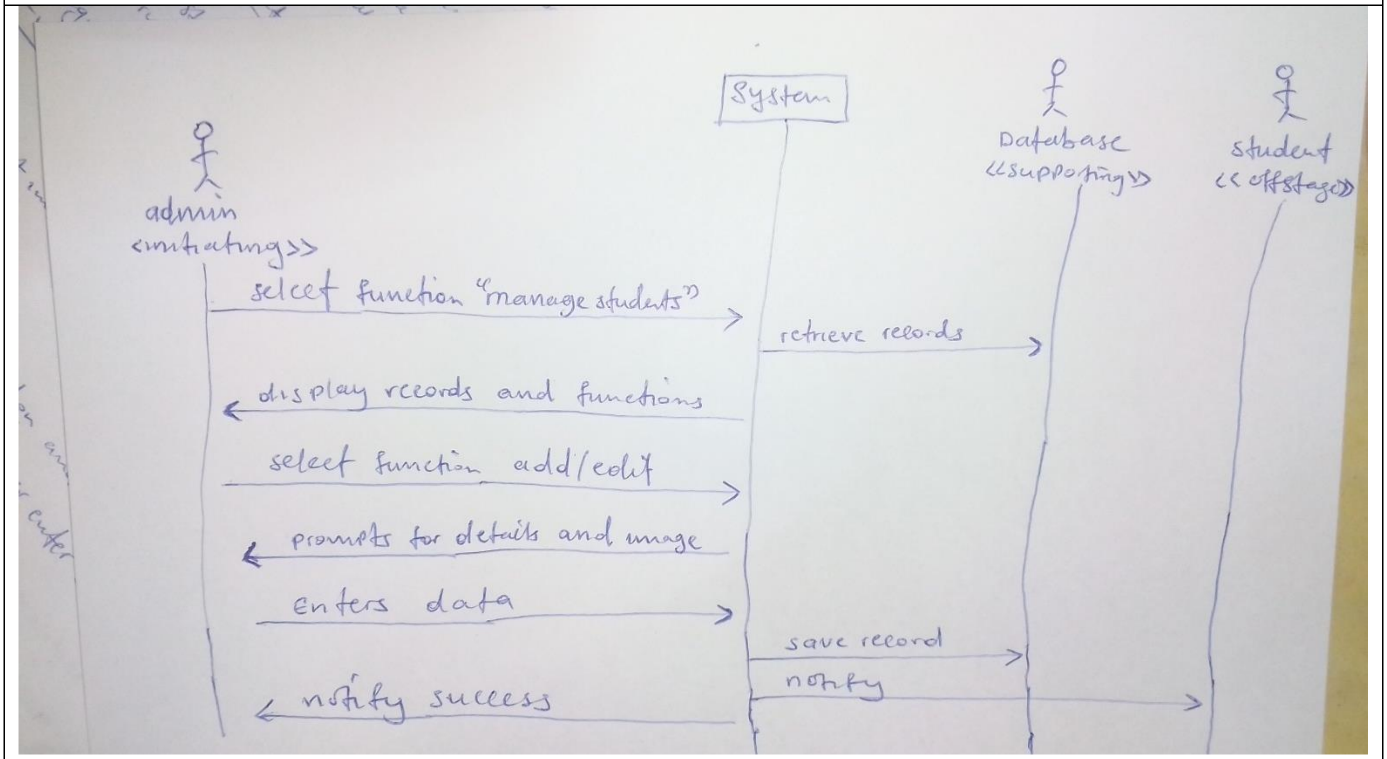


Use case: **UC-5: applyForleave :** main success scenario
 student / staff <<initiating>> System Database <<supporting>> Adminstrator <<participating>> << offstage>> <<include>> UC1: Login select apply for Leave functn Prompts for date and leave reason Enters data save record notify

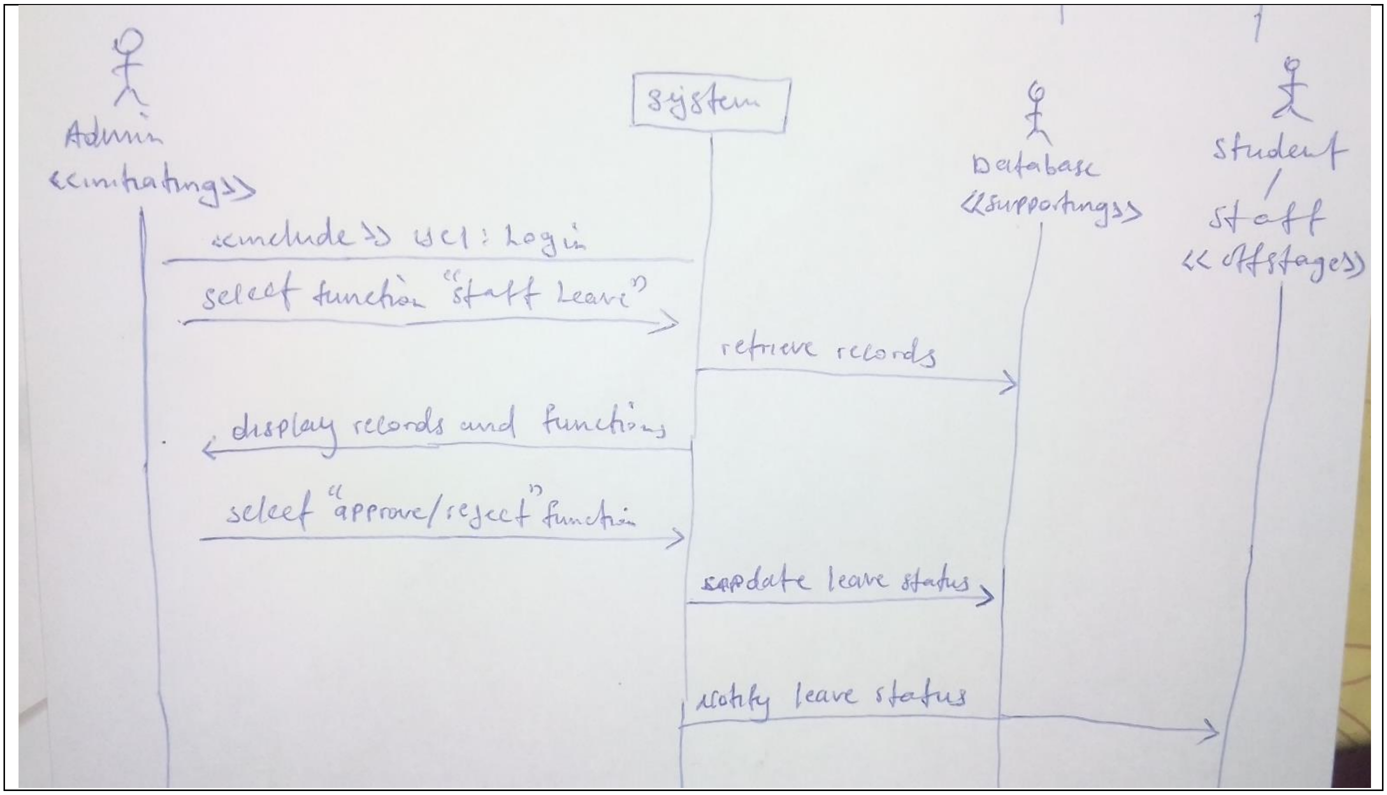
Use case: **UC-3: recordAttendance:** main success scenario



Use case: **UC-10: manageStudents:** main success scenario



Use case: **UC-11: manageLeaveApplication:** main success scenario

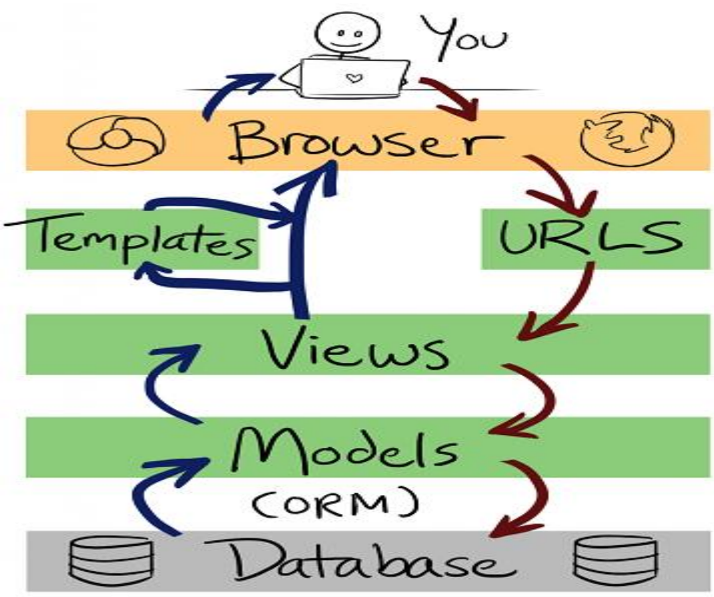


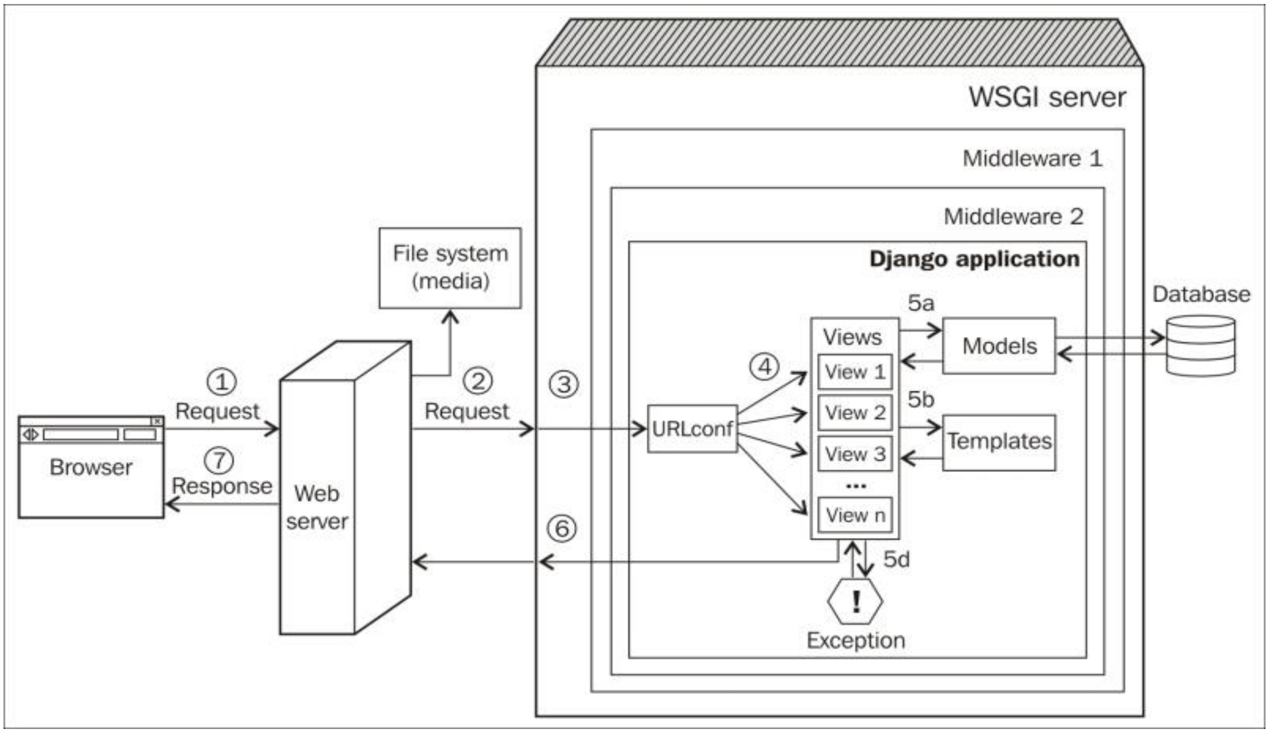
4. User Interface Specification

a. Preliminary Design

Part 3:

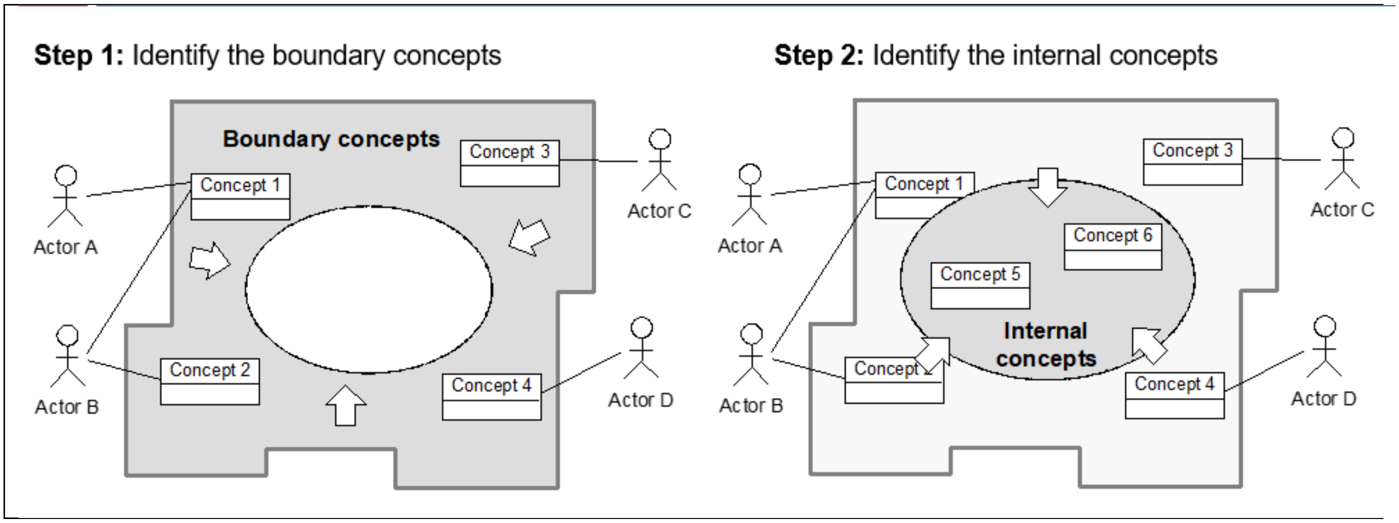
5. Domain Analysis





a. Domain Model

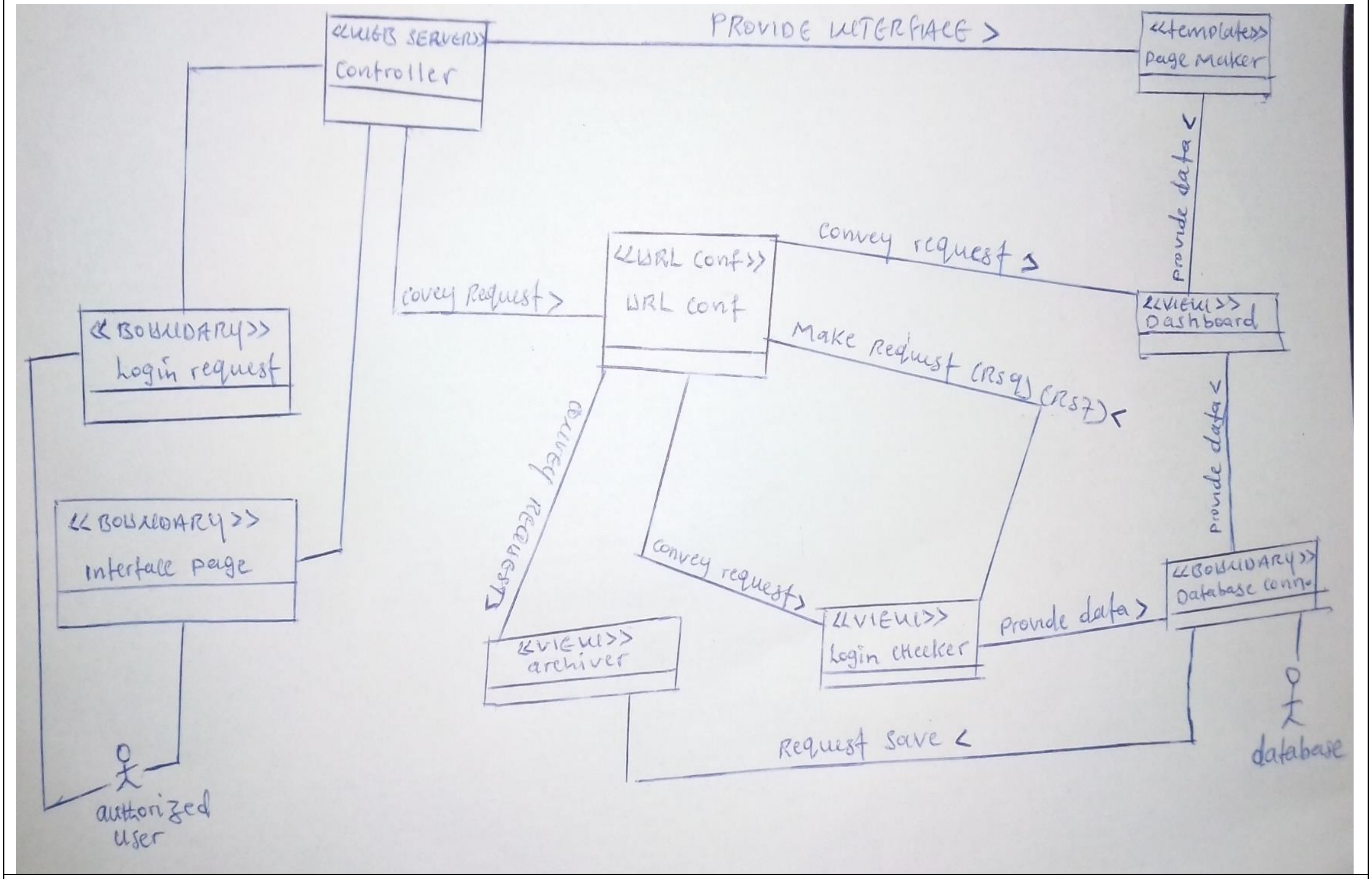
Process of deriving a domain model



DOMAIN MODEL DIAGRAMS		
Use case		
Use Case : (UC-2)	ViewOverallSummary	
Related Requirements:	REQ-1, REQ-6, REQ-11	
Initiating Actor:	Any of: admin/staff/student	
Actor's Goal:	View an overall summary or holistic view of related data	
Participating Actors:	database	
Preconditions:	• set of related data in system database is non-empty • actor is logged in the system	
Postconditions:	none	
Flow of Events for Main Success Scenario:		
--	1.	<<include>> UC1: login
←	2.	The system (a)connect to database and retrieve related data of actor (b) sums the quantitative data and (c) displays it different charts and graphs
Flow of Events for Extensions (Alternate Scenarios):		

←	2a.	System (a) detects database to be empty and (b) signals the actor

Diagram



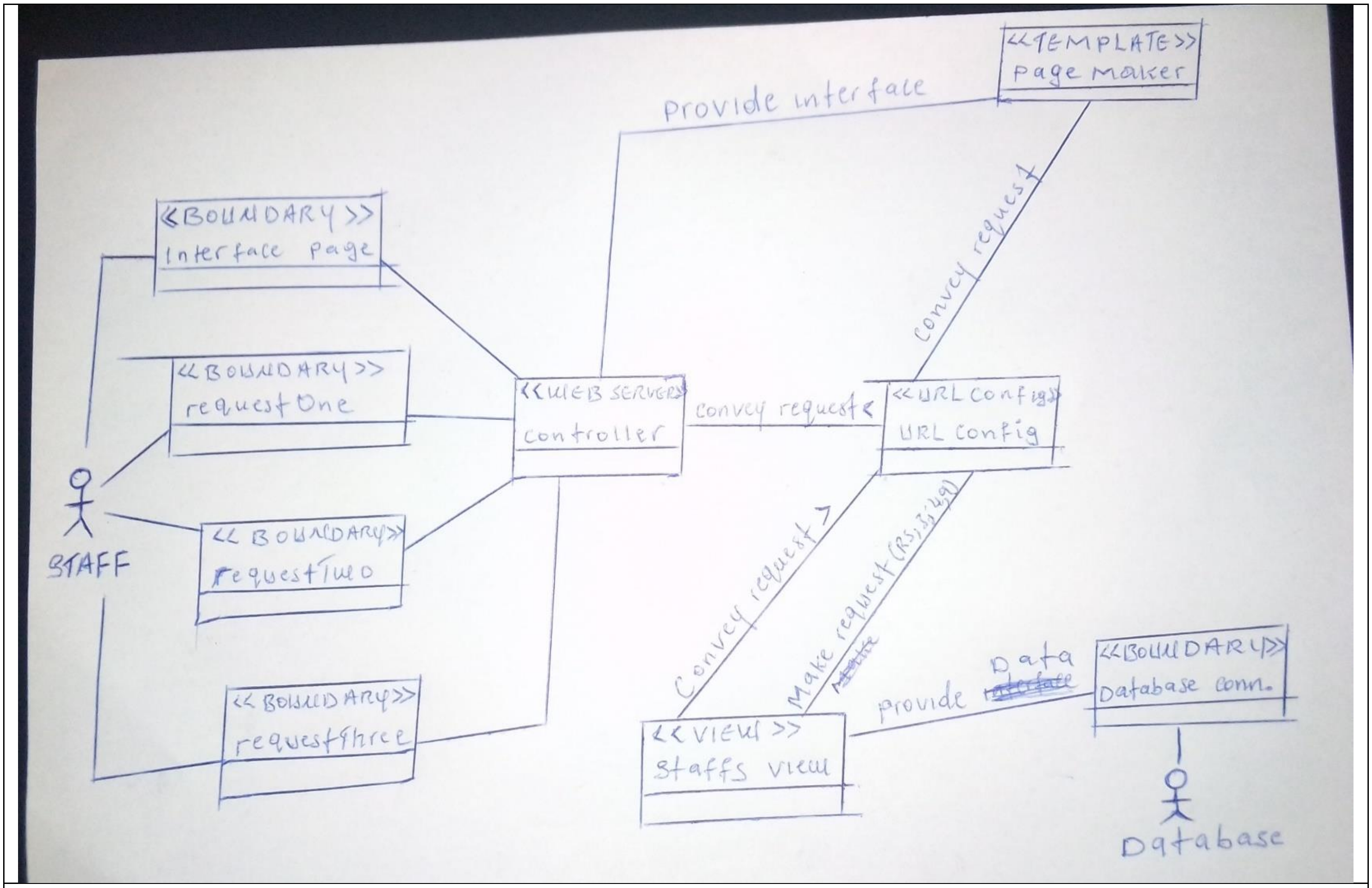
DOMAIN MODEL DIAGRAMS

Use case

Use Case (UC-3) **:**	recordAttendance
Related Requirements:	REQ-2

Initiating Actor:	staff	
Actor’s Goal:	Save a record of students that are present or absent in a subject session	
Participating Actors:	database	
Preconditions:	• a set of students&subjects&session year records are stored in system database • actor is logged in the system	
Postconditions:	none	
Flow of Events for Main Success Scenario:		
→	1.	Actor selects take attendance or update attendance function
←	2.	The system prompts the actor to select a subject and session year
→	3.	The actor selects subject and session year
←	4.	The system prompts for the date and displays all student in the subject with check boxes
→	5.	Actor uncheck students that are absent and select the ‘save ’ function
←	6.	System records data in appropriate relations and notify the actor of a successful record recorded
Flow of Events for Extensions (Alternate Scenarios):		
→	3a.	The actor enters invalid data
←	4a.	System (a) detects invalid credentials and (b) signals the actor

Diagram



i. Concept definitions

viewOverallSummary(UC-2)			
Rs-x	**Responsibility Description**	**Type**	**Concept Name**
RS-1	HTML document that shows the actor the current context, what actions can be done, and outcomes of the previous actions.	template	Interface page
RS-2	Form specifying the login parameters for for login checker	request	Login request
RS-3	Conveys HTTP request and returns HTTP response between WSGI server and web browser	Web server	controller

RS-4	Set of patterns/conventions that navigate request to the correct view	URLconfig	URLconfig
RS-5	Receives login request and returns address to dashboard or signals error	view	Login checker
RS-6	Prepares database query and returns records from database	migration	Database connection
RS-7	Processes all fuctions and data needed for charts and graphs	view	dashboard maker
RS-8	Render the retrieved records into an HTML document for sending to actor's Web browser for display.	template	Page maker
RS-9	Update login date	view	archiver

recordAttendance(UC-3)

Rs-x	**Responsibility Description**	**Type**	**Concept Name**
RS-1	HTML document that shows the actor the current context, what actions can be done, and outcomes of the previous actions.	template	Interface page
RS-2	An attendance recording request that request for subjects and sessions years	Request	requestOne
RS-3	Form specifying the specifying subject and session year and requesting all students	request	requestTwo
RS-4	Form specifying attendance date and an array list of students present and requesting to save records	request	requestThree
RS-5	Conveys request and returns response between WSGI server and web browser	Web server	controller
RS-6	Set of patterns/conventions that navigate request to the correct view	URLconfig	URLconfig
RS-7	Receives request and return and appropriate response	view	Staffs view
RS-8	Prepares database query and returns records from database	model	Database connection
RS-9	Render the retrieved records into an HTML document for sending to actor's Web browser for display.	template	Page maker

ii. Association definitions

viewOverallSummary(UC-2)

Concept pair	Association description	Association name
Interface page ↔ controller	Controller receive request from interface page and render response made for displaying	conveys requests
Login request ↔ controller	Controller receives for specifying login request from LoginRequest	Provide data
Controller ↔ URLconfig	URLconfig receive request from controller, send it to the appropriate view, and return response from views	Conveys request
URLconfig ↔ login checker	Login checker receive login request, specify and validate the login request, and returns dashboard maker address, as a response	Conveys request
Login checker ↔ database connection	Database connect receive request to provide valid usernames and passwords	Provide data
URLconfig ↔ dashboard maker	Dashboard maker receive request from URLconfig	Conveys request
Dashboard maker ↔ database connection	Database connection provide related data for specified user to dashboard maker	Provides data
Dashboard maker ↔ page maker	Page maker receive processed related data from dashboard maker	Provides data
Page maker ↔ controller	Controller receive prepared interface page from page maker	Provide interface
urlconfig ↔ archiver	Login checker prompts archive to update database	prompt
archiver ↔ database connection	Requests database to update login date	Request save

recordAttendance(UC-3)

Concept pair	Association description	Association name
Interface page ↔ controller	Controller receive request from interface page and render response made for displaying	conveys requests
Controller ↔ URLconfig	URLconfig receive request from controller, send it to the appropriate view, and return response from views	provides data
requestOne ↔ controller	Request to send a new attendance record	Initiate function
requestTwo ↔ controller	Form specifying subject and session year	Provide data
requestThere ↔ controller	Form containing attendance data	Provide data

URLconfig ↔ staffs veiw	Staffs view receive requests from URLconfig	Conveys request
Staffs veiw ↔ database connection	Database connect receive request to provide requested data	Provide data
Staffs view ↔ URLconfig	Page maker receive processed related data from staffs view and prepares an interface page	Provides data
URLconfig ↔ page maker	Page maker receive request from urlconfig, prepared by staffs view.	
Page maker ↔ controller	Controller receive prepared interface make from page maker	prepares

iii. Attribute definitions

viewOverallSummary(UC-2)		
Concept	**Attributes**	**Attribute Description**
login Request	Username	Used for user specification
	password	Used for user validation
	Device ID	Used for identifying device that sends request
Login checker	isUserLogin	Used to store user login status
	userType	Used to store the type of user
	dbusername	Used to store username received from database
	dbpassword	Used to store password from database
	isValid	Used for validity of user data
	Device ID	Used for identifying device that sends request
Dashboard maker	isUserLogin	Copied from login checker (rs5)
	isValid	Copied from login checker (rs5)
	userType	Copied from login checker (rs5)
	Device ID	Used for identifying device that sends request
	sum	Used for charts and graphs
	available	Used for functions available to user
archiver	currentDate	Used for updating the current date of login

recordAttendance(UC-3)		
Concept	**Attributes**	**Attribute Description**
requestOne	function	Used for storing function name (“staff_take_attendance”)
	Device ID	Used for identifying device that sends request
Staffs view	isUserLogin	Used to store user login status
	subjects	Arrays list that contains all subjects

	sessionYears	Array list that contains all session years
	currentDate	Used to store attendance date
	students	Array list to store all students in the particular session year session year and that offers the subject (student ID and name)
	AttendanceData	An association list that conatins student ID associated with attendanceStatus
	Device ID	Copied from requestOne (rs2)
requestTwo	subject	Used for specifying subject to take attendance for
	session	Used for specifying session to take attendance for
	Device ID	Copied from staffs view (rs7)
requestThree	attendanceData	Copied from staffs view (rs7) with statusvariable modified by user

iv. Traceability matrix

		DOMAIN CONCEPTS											
USE CASE	PW	INTERFACE PAGE	CONTROLLER	URLCONFIG	LOGIN CHECKER	DATABASE CONNECTION	DASHBOARD MAKER	PAGE MAKER	REQUEST-X	STAFFS VIEW	ADMIN VIEW	STUDENTS VIEW	FORM VALIDATOR
UC-1	8	x	x	x	x	x		x	x				x
UC-2	24	x	x	x	x	x	x	x	x				x
UC-3	5	x	x	x	x	x		x	x	X			x
UC-4	5	x	x	x	x	x		x	x	X			x
UC-5	10	x	x	x	x	x		x	x	X		x	x
UC-6	4	x	x	x	x	x		x	x	x		x	x
UC-7	10	x	x	x	x	x		x	x		x	X	x
UC-8	5	x	x	x	x	x		x	x			x	x
UC-9	4	x	x	x	x	x		x	x		x		x
UC-10	4	x	x	x	x	x		x	x		x		x
UC-11	4	x	x	x	x	x		x	x		x		x
UC-12	4	x	x	x	x	x		x	x		x		x
UC-13	4	x	x	x	x	x		x	x		x		x
UC-14	2	x	x	x	x	x		x	x		x		x
UC-15	2	x	x	x	x	x		x	x		x		x

UC-16	16	x	x	x	x	x		x	x	x	x	x	x
MAX PW		24	24	24	24	24	24	24	24	16	16	16	24
TOTAL PW		111	111	111	111	111	24	111	111	40	49	29	111

b. System Operation Contracts

Name:	
Responsibilities:	
Cross References:	
Exceptions:	
Preconditions:	
Post conditions:	

c. Mathematical Model

6. Project size estimation based on use case points

UC-X	NAME	Use case points
UC-1	login	8
UC-2	ViewOverallSummary	24
UC-3	recordAttendance	5
UC-4	addResults	5
UC-5	applyForLeave	10
UC-6	giveFeedBack	4
UC-7	viewAttendance	10
UC-8	ViewResult	5
UC-10	ManageStudents	4
UC-14	ManageLeaveApplications	2
	Project size estimation	**77**

7. Plan of Work

8. References

- https://www.brainkart.com/article/UML-Operation-Contract_9993/ (Friday, 4 November 2021)
- https://static.packt-cdn.com/products/9781783986644/graphics/6644OS_01_01.jpg (Accessed: 1 november 2021)
- https://docs.djangoproject.com/en/3.2/ (1 November 2021)
- https://i.ytimg.com/vi/bgSToGS5J1E/maxresdefault.jpg Thursday, 4 November 2021)
- https://www.lucidchart.com/pages/ (Accessed: October 29 2021)