

Jaypee University of Information Technology  
Waknaghat, Distt. Solan (H.P.)

## Learning Resource Center

CLASS NUM:

BOOK NUM.:

ACCESSION NO.: SP08024 / SP0812024

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date



# **ACADEMIC RESOURCE MANAGEMENT SYSTEM USING AGILE SOFTWARE DEVELOPMENT METHODOLOGY**

**SUBMITTED BY:**

**Rajat Kumar Jain (081215)**  
**Aditya Singh (081219)**  
**Anil Verma (081308)**

**SUPERVISED BY:**

**Dr. Yashwant Singh**



MAY 2012



Submitted in partial fulfilment of the degree of

**BACHELOR OF TECHNOLOGY  
IN**

**COMPUTER SCIENCE AND ENGINEERING**

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,  
WAKNAGHAT**



# TABLE OF CONTENTS

Chapter No.	Topics	Page No.
	Certificate from the Supervisor	III
	Acknowledgement	IV
	Summary	V
	List of Figures	VI
	List of Acronyms	VIII
<b>Chapter-1</b>	<b>1. Introduction</b>	<b>1</b>
	1.1 Objective	2
	1.2 Project Vision	2
	1.3 Genesis of Problem	3
<b>Chapter-2</b>	<b>2. Literature Survey</b>	<b>4</b>
	2.1 Agile Software Development	4
	2.2 The Agile Manifesto	5
	2.3 Principles	5
	2.4 Agile Methods	6
	2.4.1 Agile Modelling	6
	2.4.2 eXtreme Programming	7
	2.4.3 Dynamic System Development Method	10
	2.4.4 SCRUM	11
	2.4.5 Feature Driven Development	11
	2.4.6 Agile Unified Process	11
	2.4.7 Crystal Clear	12
	2.5 Difference between Agile and Traditional methods	13
<b>Chapter-3</b>	<b>3. Implementation of eXtreme Programming</b>	<b>14</b>
	3.1 User Stories	14
	3.2 Architectural Design	15
	3.3 Planning	15
	3.4 Release Planning	16
	3.4.1 Iteration 1: Administration	16
	3.4.2 Iteration 2: Department	30
	3.4.3 Iteration 3: Account Office	38



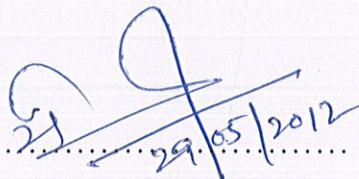
<b>Chapter-4</b>	<b>4. Conclusion and Future Scope</b>	<b>42</b>
<b>Appendix A: Codes</b>		<b>44</b>
<b>Appendix B: Snapshots</b>		<b>48</b>
<b>References</b>		<b>53</b>
<b>Resume of Students</b>		<b>55</b>



## CERTIFICATE

This is to certify that the work titled “**ACADEMIC RESOURCE MANAGEMENT SYSTEM USING AGILE SOFTWARE DEVELOPMENT METHODOLOGY**” submitted by **Rajat Kumar Jain, Aditya Singh and Anil Verma** in partial fulfilment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor: .....

  
29/05/2012

Name of Supervisor: .....

Dr. Vashwant Singh

Designation: .....

Assistant Professor

Date: .....

29<sup>th</sup> May 2012



## ACKNOWLEDGEMENT

Apart from the efforts, the success of any project depends largely on the encouragement and guidelines of many others. Therefore we take the opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would also like to show our appreciation to our project guide Dr. Yashwant Singh. Without his able guidance, tremendous support and continuous motivation the project work would not be carried out satisfactory. His kind behaviour and motivation provided us the required courage to complete our project.

Special thanks to our project panel because it was their regular concern and appreciation that made this project carried out easily and satisfactorily.

*Rajat Kumar Jain*

Rajat Kumar Jain

*Aditya Singh*

Aditya Singh

*Anil Verma*

Anil Verma

Date: 27<sup>th</sup> May . 2012



## SUMMARY

The Academic Resource Management System provides an advanced solution for today's college or university record-keeping challenges. This Academic Resource Management System will give you room to handle your responsibilities whether you're just starting or are an experienced professional.

This Academic Resource Management System provides us the information about student record, college faculty, college department and college examination result. This system provides the flexibility of generating the required documents on screen as well as on printer as and when required.

Academic Resource Management System opens a universe of opportunities to automate the laborious paperwork involved in college or university management. With our proposed record-keeping software the management can more effectively interact with the students as they develop skills and character for success. They will not only have more time to spend with them, but it will be quality time because they will have up-to-date student information to facilitate them.

*Rajat Kumar Jain*

Rajat Kumar Jain

*Aditya Singh*

Aditya Singh

*Anil Verma*

Anil Verma

*Dr. Yashwant Singh*

Dr. Yashwant Singh

(Assistant Professor, CSE Dept.)

Date: 29-05-2012

Date: 29<sup>th</sup> / May / 2012



## List of Figures

Sr. No.	Name	Page No.
1	Fig 2.1: eXtreme Programming Lifecycle	8
2	Fig 2.2: Iterations in XP	9
3	Fig 2.3: Difference between Agile and Traditional method	13
4	Fig 3.1: Architectural Design	15
5	Fig 3.2: Use Case diagram of Administration	16
6	Fig 3.3: Class Diagram	17
7	Fig 3.4: Sequence diagram: Sign in by the Administration	18
8	Fig 3.5: Sequence diagram: Enrol Student by the Administration	18
9	Fig 3.6: Sequence diagram: Add Faculty by the Administration	19
10	Fig 3.7: Sequence diagram: Register Student by the Administration	19
11	Fig 3.8: Sequence diagram: Register Student by the Administration	20
12	Fig 3.9: Sequence diagram: View Student Information by the Administration	20
13	Fig 3.10: Sequence diagram: View Faculty Information by the Administration	21
14	Fig 3.11: Sequence diagram: Edit Student Information by the Administration	21
15	Fig 3.12: Sequence diagram: Edit Faculty Information by the Administration	22
16	Fig 3.13: Sequence diagram: Delete Student Record by the Administration	22
17	Fig 3.14: Sequence diagram: Delete Faculty Record by the Administration	23
18	Fig 3.15: Activity diagram: Sign In by the Administration	23
19	Fig 3.16: Activity diagram: Enrol Student by the Administration	24
20	Fig 3.17: Activity diagram: Add Faculty by the Administration	24
21	Fig 3.18: Activity diagram: Register Student by the Administration	25
22	Fig 3.19: Activity diagram: Add Department by the Administration	25
23	Fig 3.20: Activity diagram: View Student Info by the Administration	26
24	Fig 3.21: Activity diagram: View Faculty Info by the Administration	26



25	Fig 3.22: Activity diagram: Edit Faculty Info by the Administration	27
26	Fig 3.23: Activity diagram: Delete Student Info by the Administration	27
27	Fig 3.24: Activity diagram: Delete Faculty Info by the Administration	28
28	Fig 3.25: Activity diagram: Edit Student Info by the Administration	28
29	Fig 3.26: Entity Relation Diagram	29
30	Fig 3.27: Use Case diagram of Department	30
31	Fig 3.28: Class diagram of Department	31
32	Fig 3.29: Sequence Diagram: Add Course by the Department	31
33	Fig 3.30: Sequence Diagram: Allot Courses by the Department	32
34	Fig 3.31: Sequence Diagram: Refresh Assignment of Courses by the Department	32
35	Fig 3.32: Sequence Diagram: Assign Courses to Faculty in Department	33
36	Fig 3.33: Sequence Diagram: View Staff Info by the Department	33
37	Fig 3.34: Sequence Diagram: View Courses by the Department	34
38	Fig 3.35: Sequence Diagram: View Courses Assigned to Faculty by the Department	34
39	Fig 3.36: Activity Diagram: Add Course by the Department	35
40	Fig 3.37: Activity Diagram: Drop Course by the Department	35
41	Fig 3.38: Activity Diagram: Allot Courses by the Department	36
42	Fig 3.39: Activity Diagram: Refresh Assignment of Courses by the Department	36
43	Fig 3.40: Activity Diagram: Assignment of Courses to Faculty by the Department	37
44	Fig 3.41: View Staff Info in Department	37
45	Fig 3.42: View Courses in Department	38
46	Fig 3.43: Use Case Diagram of Account Office	38
47	Fig 3.44: Class Diagram of Account Office	39
48	Fig 3.45: Sequence Diagram: Submit Fees of Student in Account Office	39
49	Fig 3.46: Sequence Diagram: Pay Salary to Faculty in Account Office	40
50	Fig 3.47: Activity Diagram: Submit Fees of Student in Account Office	40
51	Fig 3.48: Activity Diagram: Pay Salary to Faculty in Account Department	41



## List of Acronyms

Acronym	Description
ARMS	Academic Resource Management System
XP	eXtreme Programming
DSDM	Dynamic Systems Development Method
AUP	Agile Unified Process
EssUP	Essential Unified Process
RUP	Rational Unified Process
ExP	Exia Process
FDD	Feature Driven Development
OpenUP	Open Unified Process
FMI	Functional Model Iteration
DBI	Design & Build Iteration
BAD	Business Area Definition
SAD	System Architecture Definition
OPP	Outline Prototyping Plan
TDD	Test Driven Development
UML	Unified Modelling Language
IDE	Integrated Development Environment
JDBC	Java Database Connectivity



## 1. Introduction

Academic Resource Management System is specifically developed for comprehensive solution to manage academic records of a university. It facilitates university to record all information regarding students and the staff. It handles information about all the departments in the college or university. The main goal of the Academic Resource Management System is to automate the university management, thus reduce paper work and improve the efficiency and enhance the productivity.

Academic Resource Management System is a comprehensive student information management system developed from the ground up to fulfil the needs of independent universities. It connects daily operations in the college or university environment ranging from Admissions and Registration to Finance, Faculty and Department. This reduces data error and ensures that information is always up-to-date throughout the university.

Academic Resource Management System provides a single source of data repository by administration, departments, faculty and the accounts department. It has a simple user interface which ensures that the users spend less time in learning the system and hence, increase their productivity. Efficient security features provide data privacy and maintains data integrity.

Academic Resource Management System is a fast, affordable, low-risk solution with easy implementation and lower maintenance and operational costs. It helps to optimize the use of available resources in a cost effective manner through their proper scheduling and resource allocation. It is an excellent tool to promote and manage enrolment growth and provide accurate enrolment data. Eliminate duplicate data entry and redundant information storage that most often propagates errors.

In short, Academic Resource Management System is a solution to manage all the records related to a university like student records, staff records and handling accounts of a university. It provides a university with an automated system to handle the records that reduces the manual work. It has a user friendly interface which is flexible and easy to maintain. It will be a very good solution to maintain records of a university at one place as every department of the university will keep the records at same place.



ARMS has following entities:

1. **Administration:** Administration is the necessary part of any organization. In Academic Resource Management System, It represents the Registrar office. It will handle the enrolment and registration process of the students, create and distribute grade sheets of them. It also keeps the record of faculty. In short we can say that Administration will be on top of everything in the college or university.
2. **Departments:** In any organization, there are many departments which have their specific tasks to perform. In Academic Resource Management System, Department are the different streams which the university provides for education. Different departments will handle their records like attendance of students, grades of students, subjects registered by students, which course will be taught by which faculty etc.
3. **Faculty:** Faculty has the most important role in academics. He is a person who provides knowledge to his pupils and makes them able to start their career. In Academic Resource Management System, The role of faculty is to mark the attendance and grades of students.
4. **Account office:** The record of student's fees and the salary of staff are also necessary to keep. For this, we have Accounts department. In Academic Resource Management System, It will handle all the data regarding finance like fees submitted by the students, salaries of the staff.

## 1.1 Objectives

- To implement Agile Software Development Methodology (eXtreme Programming).
- To build a secure, flexible, easy to maintain environment for creating and maintaining academic records.
- To develop a user friendly and an eye catching GUI for Academic Resource Management System.
- To learn JAVA swings.
- To learn the JDBC connectivity to the Oracle Database 10g Express Edition.
- To learn Oracle SQL queries.
- To have an experience of working on NetBeans 7.0 IDE.

## 1.2 Project Vision

The vision for the Academic Resource Management System is to provide a software for managing Academic and Financial records. The system will provide the tools to create, maintain



and report on student data in an efficient and convenient manner. Upon completion of the project, the system will improve the quality of the academic experience for students, faculty and staff.

Our vision is to create an environment that:

- Enables a stable, reliable and convenient self-service registration environment.
- Provides more timely, convenient and flexible options to students, faculty and staff.
- Provides the opportunity to improve on current business processes.
- Is conducive to managing and maintaining changes to academic and/or regulatory policies.
- Improves communication by sharing information within functional areas of the University.

### **1.3 Genesis of Problem**

Previously the administration of any university have to make records of students registered and the staff of the university on paper and have to keep them safe in some files. The same thing happens to the accounts department and the different branches as they also need to keep all the records on paper. If anytime, they wanted to get someone's record, first they had to search the file of that record which takes a lot of time. We know that it's very tedious task to keep these much records on paper for a long period of time and in this paperwork, the manual work increases which indeed reduced the utilization of the efficiency of staff. So we require something where we can store as much data as we want and can keep them safe for long period and that must be an automated system by which the utilization of the staff can be increased.

The rest of this report describes our systematic approach towards developing user friendly, flexible and easy to maintain environment for Academic Resource Management System. Chapter 2 describes the Literature survey on the Agile Methodology and different agile methods and their characteristics, after which we are able to decide which method should be used to develop Academic Resource Management System .With Chapter 3 we start with the implementation of eXtreme programming and development of Academic Resource Management System. After that, in Chapter 4, we describe what is our learning from this project and the future scope of the project.



## 2. Literature Survey

### 2.1 Agile Software Development:

Agile Software development is not a single methodology or a set of tools, in fact it is a philosophy which put on paper in 2001 with initial 17 signatories.

Agile is a group of software development methodologies based on iterative and incremental development. Common theme about agile methodologies is that they are all focussed on trying to produce a working solution and be able to respond to changing user/client requirement.

Dictionary meaning of Agile is moving quickly and lightly. It is a method that tries to be responsive to the needs of the software development process. Primary goal of agile methods is to develop working software.

Agile methodologies are used to produce higher quality software in a shorter period of time. Agile methodologies were developed to streamline the development process and remove barriers to accepting business requirement changes during the development process. Agile methodologies do not require that business requirements and design details be locked in for the duration of development.[1]

Agile methods recommend close team cooperation, exchange of knowledge, ideas and code between team members. Instead of specialists for specific parts of development process, it is recommended that a person is included in multiple parts of development process, which can include parts like coding new features, writing coding tests but also gathering user requirements etc. By including team members in various aspects of development process the chance of misinterpretation is reduced. Regardless of their experience each developer can enhance the model of the system.[2]

It focuses on being effective and sufficient.

- Effective in terms of producing working (defect free) software.
- Sufficient in terms of meeting its requirements.



Now, we can say that Agile Methodology is a method that tries to be responsive to the need of the software development process, that is based on practice and that focuses on being effective and sufficient.[5]

## 2.2 The Agile Manifesto

### 1. **Individuals and interactions** over processes and tools.

This refers to the fact that people involvement to the project and their communication with each other is essential. Tools and processes can help but they are still not the overriding influence.

### 2. **Working Software** over comprehensive documentation.

The team must focus on working software and not on the documentation. The main goal of the team should be developing working software as soon as possible and the documentation should be a supportive part of the project.

### 3. **Customer collaboration** over contract negotiation.

There must be a customer representative with the developing team rather than having detailed contract negotiations.

### 4. **Responding to change** over following a plan.

Agile team must be responsive to change rather than saying that it was not in the requirements of the plan, so we can't do it. Note that it does not mean that plan is not important- actually, it is very important but the project adapts itself to its environment.

## 2.3 Principles

- Highest priority is to satisfy the customer.
- Welcome change.
- Deliver working software frequently.
- Business people and developers must work together daily.



- Face to face communication is best.
- Working software is the primary measure of progress.
- Teams should regularly review itself and its processes to try and improve.

## **2.4 Agile Methods**

The following are the core methods of Agile Software Development being worked on:

### **2.4.1 Agile Modelling**

### **2.4.2 eXtreme Programming ( XP )**

### **2.4.3 Dynamic System Development Method (DSDM)**

### **2.4.4 SCRUM**

### **2.4.5 Feature Driven Development (FDD)**

### **2.4.6 Agile Unified Process (AUP)**

### **2.4.7 Crystal Clear**

### **2.4.1 Agile Modelling**

It tries to find an appropriate balance between too little modelling and too much modelling, at the design stage. Modelled enough to explore and document your system effectively, but not so much that it becomes a burden. It provides guidelines on how to create effective models and how to be an efficient modeller. It does not necessarily means that less modelling will be performed while adopting Agile Modelling.[5]

It has three main goals:

1. The definition and promotion of a set of values, principles and practices that help to produce the appropriate models.
2. Guidance on how to apply modelling to an Agile software development.
3. Guidance on how to apply Agile Modelling to other software processes (such as RUP).



### 2.4.2 eXtreme Programming (XP)

It was originally designed as a way of supporting small development teams working within uncertain and changing requirements. It was designed as an approach based on software engineering principles, but focussed on the timely delivery of software that meets user's requirements.[4] An important aspect of XP is the empowerment of the actual developers- they should be able to react immediately to changing customer requirements, even late in the development life cycle. It also places great emphasis on the software development team and teamwork. One of its aims is that team should communicate and constantly pay attention to all the details necessary to make sure that the software being developed, matches the user requirements. The main focus of the eXtreme team is on giving the working software to the customer as soon as possible rather than wasting time in doing the documentation.[5]

Four basic principles of XP are as follows:

- **Communication**

Programmer must have communication with his fellow programmer as well as the user representative.

- **Simplicity**

Design should be as simple as possible. Focus on giving the complete working software to the customer as soon as possible rather than wasting time on documentation.

- **Feedback**

Testing must start from day one of the implementation and get feedback from those test cases so that those errors do not repeat in future.

- **Courage**

Programmer must have courage to remove the obsolete code without thinking how much time it took at the time of implementation.

Some key ideas presented in XP are as follows:

- **Code in pairs**

According to XP, Coding should be done in pairs as it assumes that two heads are better than the one.

- **Stay in contact with the customer**

In XP, One user representative always stays with the developer team and whenever developer team needs to ask some information about the problem given by the user, they consult with him.



- **Create tests before coding then test heavily**  
Test scenarios must be prepared at the beginning from the user stories and testing should go hand in hand with coding.
- **Short iterations**  
The whole project is divided into iterations. Iterations must be as short as possible because it will reduce the possibility of errors.
- **Keep it simple**  
Focus on delivering the working software to the customer as soon as possible rather than wasting time on documentation. Design must be as simple as possible.
- **Don't anticipate**  
Don't think about tomorrow. Complete today's work and leave tomorrow work on tomorrow.
- **Collective ownership**  
Everyone within the team has the responsibility for the software. If something goes wrong, nobody will be blamed particularly - everyone will be responsible for that. Thus anybody in the team can fix that problem.

#### ❖ XP Lifecycle

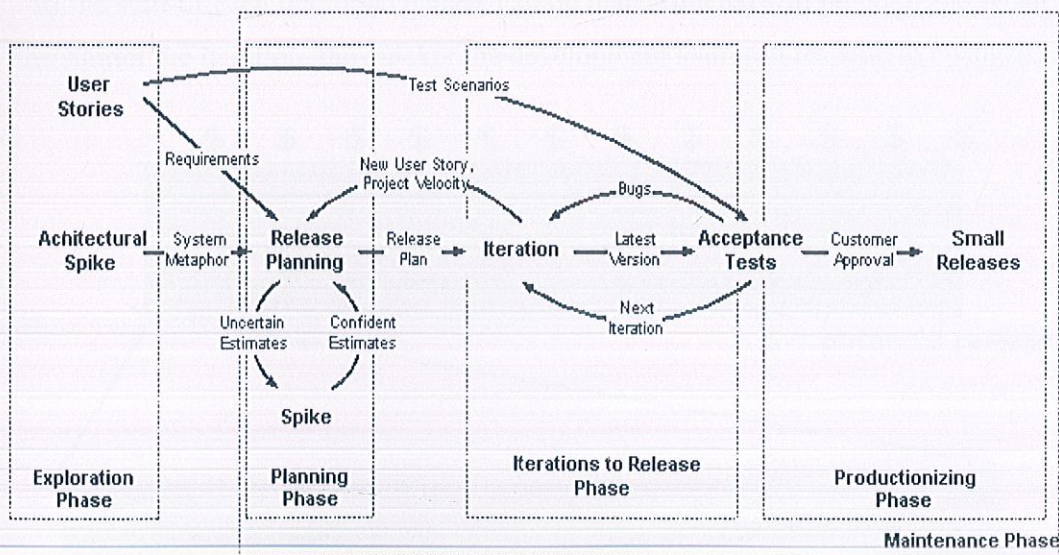


Fig 2.1: eXtreme Programming Lifecycle [4]

This figure 2.1 depicts the lifecycle of eXtreme Programming. It has many keywords which are explained as follows:



### ➤ User Stories

A user story describes problems to be solved by the system being built. These stories must be written by the user and should be about three sentences long. Because user stories are short and somewhat vague, XP will only work if the customer representative is on hand to review and approve user story implementations. This is one of the main objections to the XP methodology, but also one of its greatest strengths.[5]

### ➤ Architectural Spike

A Spike in XP terms is an attempt to reduce the risk associated with an unknown area of the system, technology or application domain. A spike may involve some investigation, research and possibly some software to evaluate the problem.[5]

### ➤ Release Planning

It indicates which user stories will be implemented and in which release this will happen. It also indicates how many iterations are planned and when each iteration will be delivered. Note that individual iterations are planned just before the iteration commences, not in advance.[5]

### ➤ Iterations

At the start of each iteration changes can be made to what will be done and when it will be done. The shorter the iteration, the quicker the development team can respond to changes.[5]

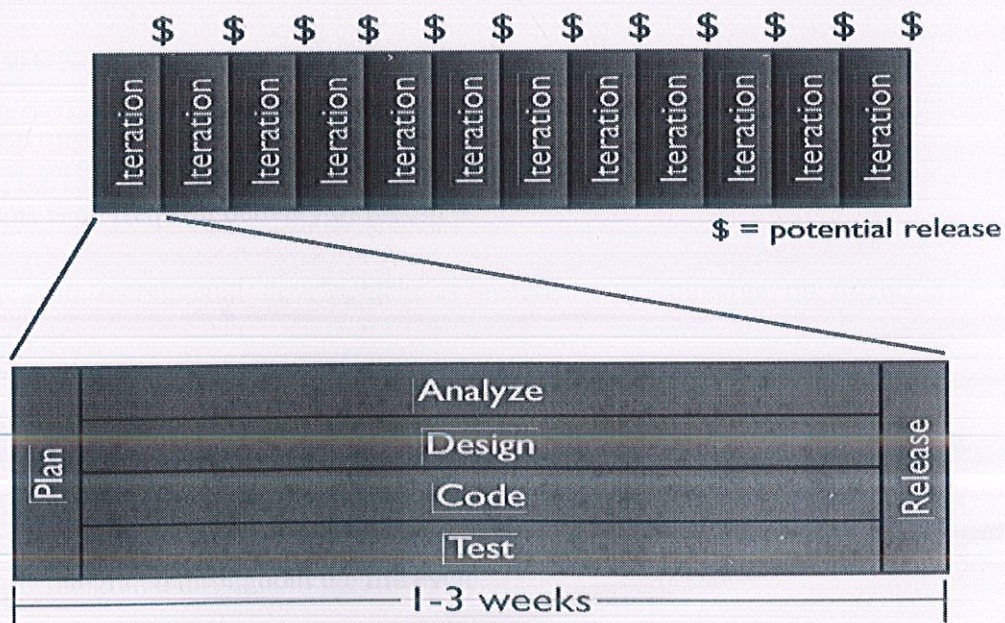


Fig 2.2: Iterations in XP



Fig 2.2 shows how in each iteration, implementation is done. We can depict here that in each iteration, everything from planning to the small release is done. We can see here that each iteration takes 1-3 weeks of time to complete with small release.

#### ➤ **Acceptance Testing**

The acceptance tests are created from the user stories, written at the start of the project. Each iteration implements one or more user stories. A story is not limited to one acceptance test and may be related to many acceptance tests (depending on the complexity of the scenario).[5]

#### ➤ **Release**

XP promotes the concept of “small release” and “release Often”. These releases should be made available to users or the user representative when available. This will allow early and frequent feedback from the user representative rather than relying on the big bang approach and then worrying about the consequences.[5]

### **2.4.3 Dynamic System Development Method (DSDM)**

It is particularly suitable for application development projects that need to develop complex business solutions within tight timeframes. In DSDM, time is fixed for the life of the project, and resources are fixed as far as possible.[5]

It is based on following principles:

1. Active user involvement is imperative.
2. The team must be empowered to make decisions.
3. The focus is on frequent delivery of products.
4. Iterative and incremental development is necessary to converge on an accurate business solution.
5. All changes during development are reversible.
6. Requirements are base lined at a high level.
7. Testing is integrated throughout the life cycle.
8. Collaboration and cooperation between all the stakeholders is essential.



#### **2.4.4 SCRUM**

It aims to manage and control the production of software using iterative, incremental and lightweight processes. After the team completes the project scope and high-level designs, it divides the development process into a series of short iterations called 'sprints'. Each sprint aims to implement a fixed number of backlog items. At the end of a sprint, the team reviews the sprint to check progress. During a sprint, the team has a daily meeting called a scrum. Each team member describes the work to be done that day and progress from the day before. When enough of the backlog has been implemented so that the end users believe the release is worth putting into production, management closes development. The team then performs integration testing, training and documentation as necessary for product release.[5]

#### **2.4.5 Feature Driven Development**

Feature-driven development is another type of Agile Methodologies based on iteration and incremental software development process. It is one of a number of Agile methods for developing softwares. The advantage of using a feature driven approach is the potential for managing an agile project, for handling the uncertainties that an agile approach introduces, for getting to grips with monitoring and reporting on the project.[5]

FDD is a model-driven short-iteration process that consists of five basic activities which are as follows:

- Develop Overall Model
- Build Feature List
- Plan By Feature
- Design By Feature
- Build By Feature

#### **2.4.6 Agile Unified Process**

Agile Unified Process is a simplified version of the IBM Rational Unified Process(RUP). It describes a simple, easy to understand approach to develop softwares using agile techniques. The



Agile Unified Process applies agile techniques including Test Driven Development (TDD), Agile Modelling, agile change management, and database refectory to improve productivity.[5]

It has following disciplines:

- Model.
- Implementation.
- Test.
- Deployment.
- Configuration Management.
- Project Management.
- Environment.

#### **2.4.7 Crystal Clear**

Crystal Clear can be applied to teams of up to 6 or 8 developers working on systems that are not life-critical. The Crystal family of methodologies focus on efficiency and habitability as components of project safety. Crystal Clear focuses on people, not processes.[5]

Crystal Clear requires the following properties:

- Frequent delivery of usable code to users
- Reflective improvement
- Osmotic communication preferably by being co-located

Crystal Clear additionally includes these optional properties:

- Personal safety
- Focus
- Easy access to expert users
- Automated tests, configuration management, and frequent integration



## 2.5 Difference between Agile and Traditional methods

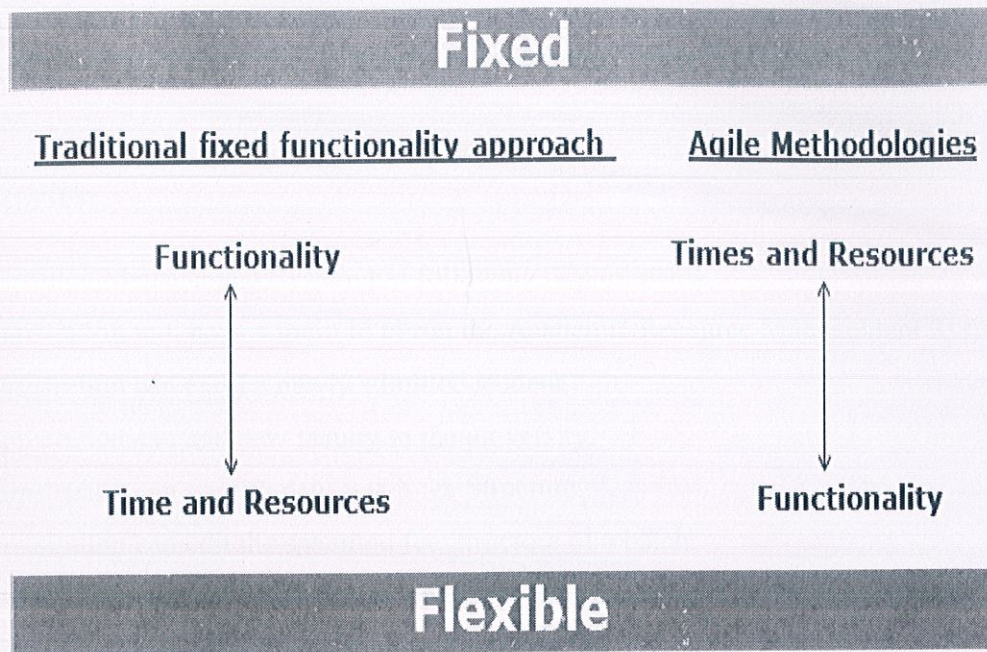


Fig 2.3: Difference between Agile and Traditional methods [5]

This fig 2.3 is showing the difference between the Traditional Methods and the Agile Methods in terms of functionality and the time and resources. In the Traditional approach the functionality is fixed throughout the project and time and resources are flexible i.e. we can delay the development of the project. On the other hand, In Agile Methodologies, it is just opposite of it. In Agile approach, we can vary the functionality but the time and resources must remain same as of the beginning of the project.[5]

In this chapter, we saw many methodologies defined under Agile Methodologies. After understanding all the methods, we decided to implement eXtreme Programming in Academic Resource Management System. In further chapters, we will see the implementation of eXtreme Programming according to its lifecycle.



### 3. Implementation of eXtreme Programming

#### 3.1 User Stories

Academic Resource Management System has following requirements:

- Administration will have a login Id to run the Academic Resource Management System.
- Administration can enrol a newly admitted student.
- Administration can add new faculty to the university.
- Administration can specify new Batch fee Structure.
- Administration can edit the specified fee Structure of a batch.
- Administration can view information of a student or faculty.
- Administration can edit information of a student or faculty.
- Administration can delete record of a student or faculty.
- Administration can register a student in a semester.
- Administration can create Result of a Student.
- Administration can add a department to University.
- Account Office will have a user id to run Academic Resource Management System.
- Account Office can make payments to the Faculty.
- Account Office can receive fees from Students at the time of registration.
- Department will have a user id to run Academic Resource Management System.
- Department can Add Courses.
- Department can drop Courses.
- Department can Assign Courses in the Semester.
- Department can Refresh Courses Assigned to faculty after each semester.
- Department can Assign Courses to the Faculty.
- Department can view Staff Information.
- Department can view courses in the department.
- Department can view courses Assigned to faculty.



### 3.2 Architectural Design

#### Two TIER CLIENT SERVER SYSTEM

In Academic Resource Management System, we need a client and a server. Everything will be handled on the server side. So we can say that it is a Two Tier Thin Client Server System. Design is shown in fig 3.1.

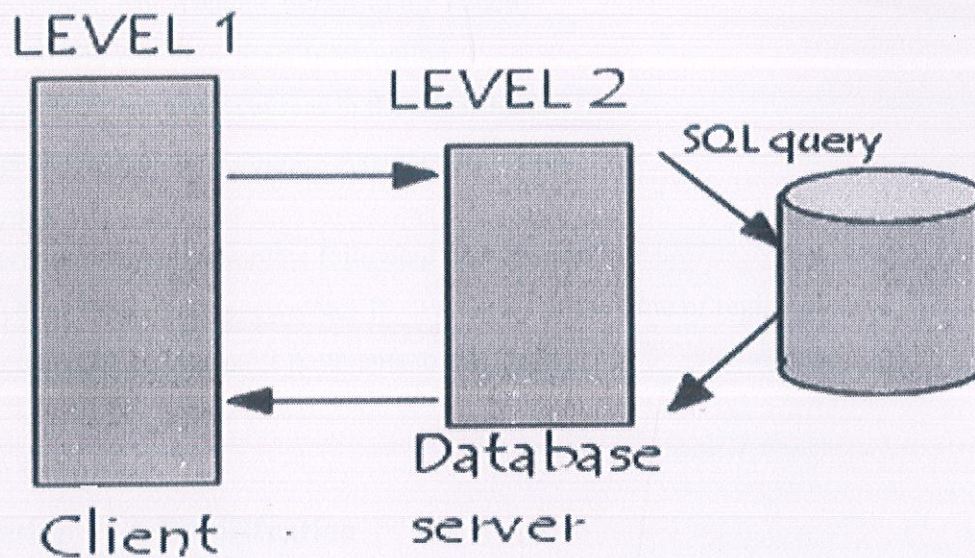


Fig 3.1: Architectural Design

### 3.3 Planning

The Project will be completed in 4 iterations:

#### Iteration 1

In this Iteration, we will complete following user stories:

- Administration can enrol a newly admitted student.
- Administration can add new faculty to the university.
- Administration can specify new Batch fee Structure.
- Administration can edit the specified fee Structure of a batch
- Administration can view information of a student or faculty.
- Administration can edit information of a student or faculty.
- Administration can delete record of a student or faculty.
- Administration can register a student in a semester.
- Administration can create Result of a Student.
- Administration can add a department to University.



## Iteration 2

In this Iteration, we will complete following user stories:

- Department can Add Courses.
- Department can drop Courses.
- Department can Assign Courses in the Semester.
- Department can Refresh Courses Assigned to faculty after each semester.
- Department can Assign Courses to the Faculty.
- Department can view Staff Information.
- Department can view courses in the department.
- Department can view courses Assigned to faculty.

## Iteration 3

In this Iteration, we will complete following user stories:

- Account Office can receive fees from students at the time of registration.
- Account Office can make payments to the faculty.

### 3.4 Release Planning

#### 3.4.1 Iteration 1: Administration

- Use case diagram

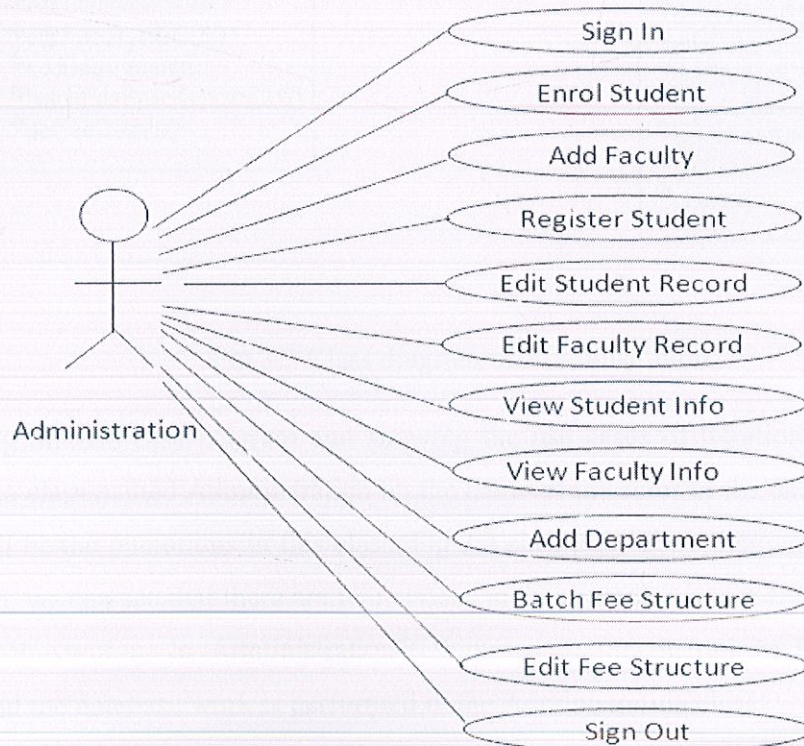


Fig 3.2: Use Case diagram of Administration



**Actor:** Administration

**Use Cases:** Sign In, Enrol Student, Add Faculty, Register Student, Edit Student, Edit Faculty Record, View Student Information, View Faculty Information, Add Department, Batch Fee Structure, Edit Fee Structure, Sign Out.

## • Class Diagram

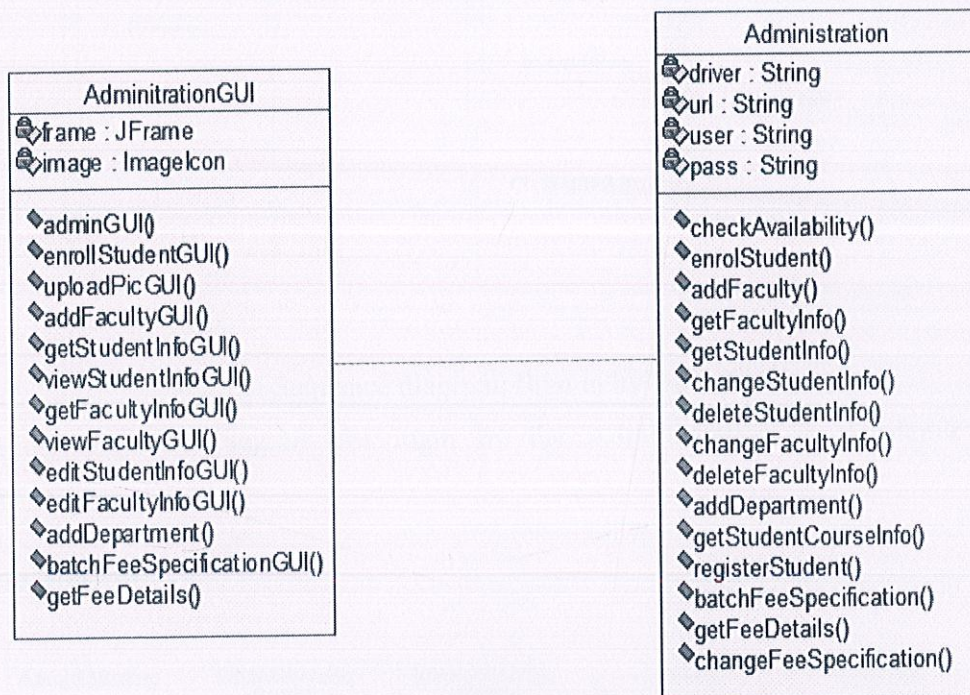


Fig 3.3 Class diagram of Administration

After creating the use case diagram and showing the use cases of iteration 1, we determine that there will be a class named Administration on the name of the actor of the use case diagram and the use cases will be the operations in this class. Fig 3.3 shows the class diagram of iteration 1. In this class diagram, we can see that there are two classes in this iteration, one AdministrationGUI and the other is Administration. In AdministrationGUI class, all the operations related to the GUI is performed and the database work is performed in the Administration class.



- **Sequence Diagrams**

- **Sign in**

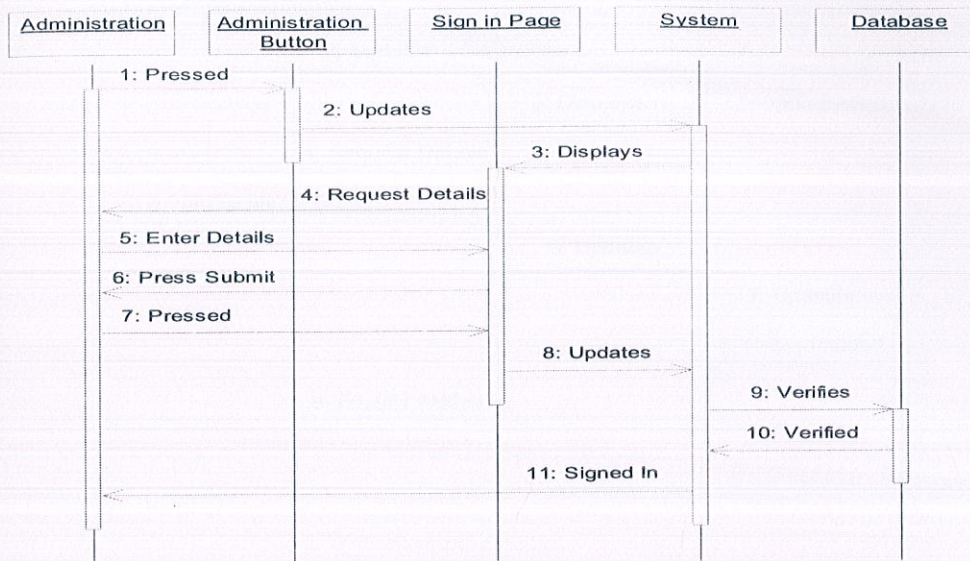


Fig 3.4: Sequence diagram: Sign in by the Administration

Fig 3.4 is the sequence diagram for login by the Administration in Academic Resource Management System.

- **Enrol Student**

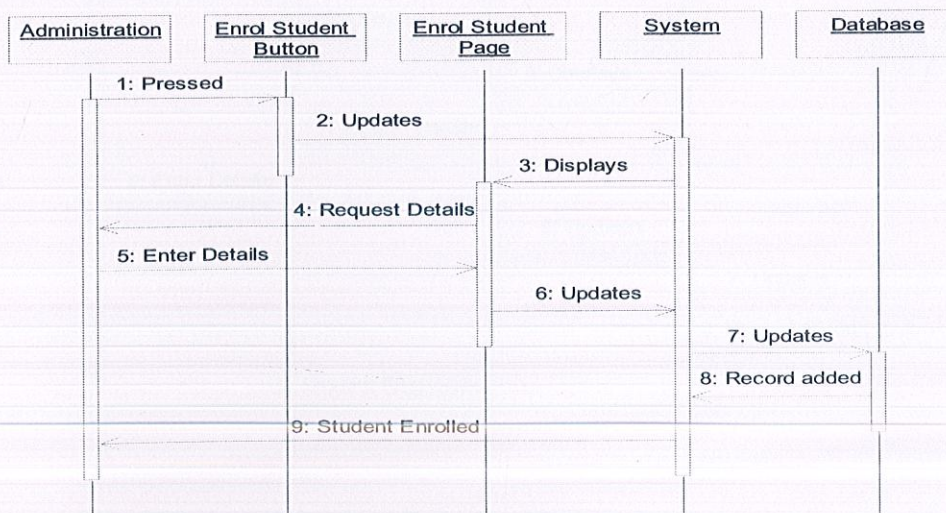


Fig 3.5: Sequence diagram: Enrol Student by the Administration

Fig 3.5 is the sequence diagram of Enrolling a Student in university.



- **Add Faculty**

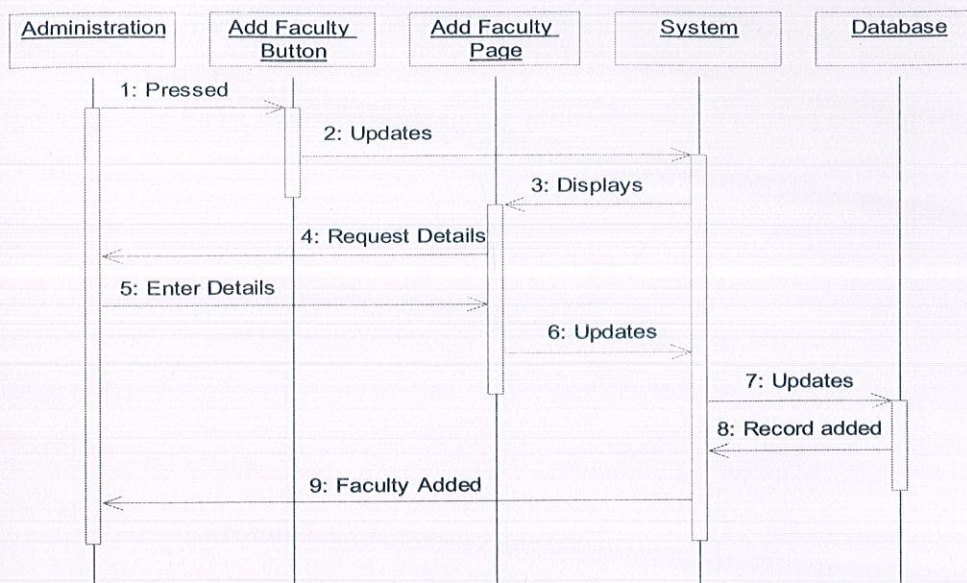


Fig 3.6: Sequence diagram: Add Faculty by the Administration

Fig 3.6 is the sequence diagram of Adding the information of a faculty in the university.

- **Register Student**

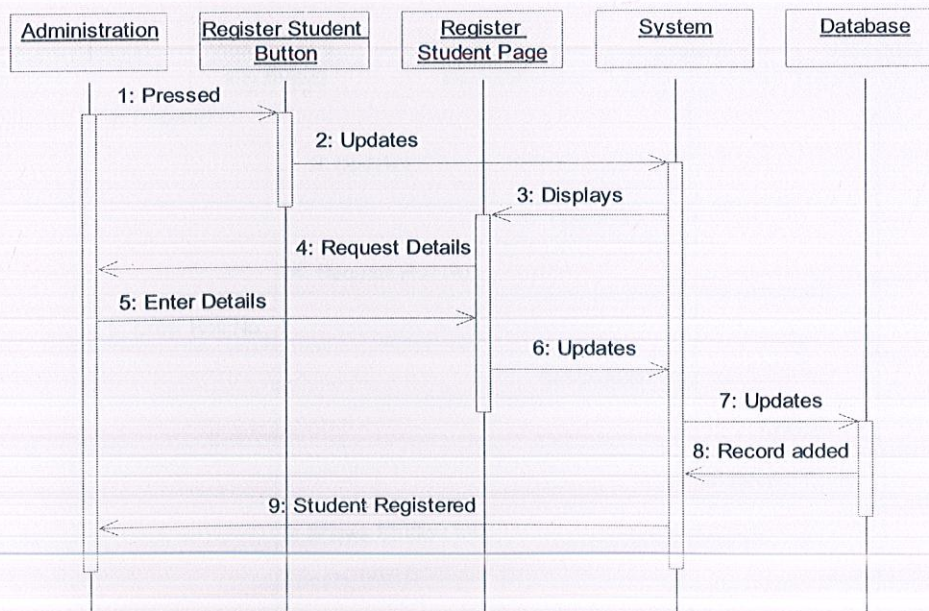


Fig 3.7: Sequence diagram: Register Student by the Administration

Fig 3.7 is the sequence diagram of Registering a student in some semester in the university.



- **Add Department**

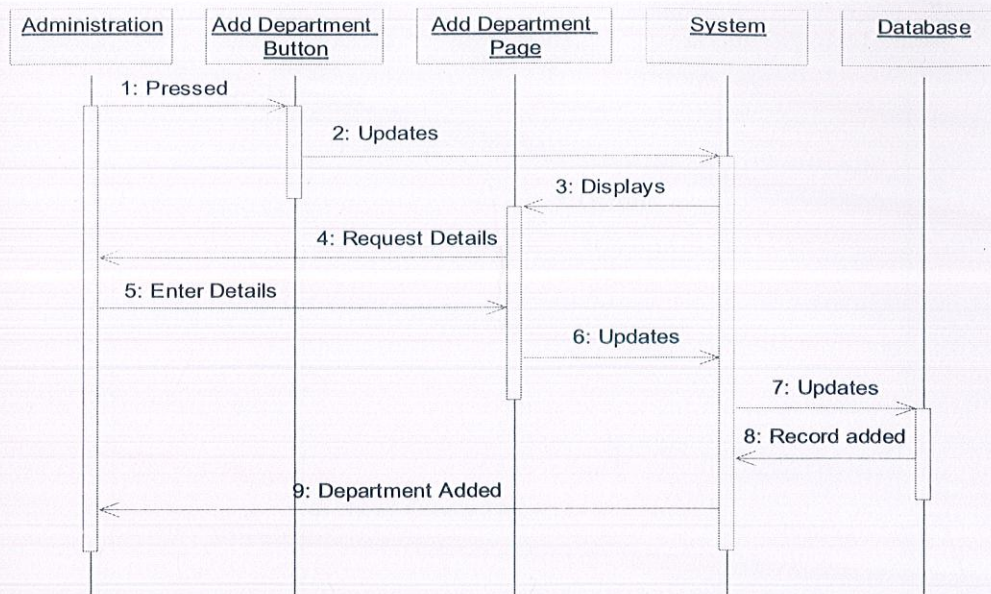


Fig 3.8: Sequence diagram: Register Student by the Administration

Fig 3.8 is the sequence diagram of adding a department (like CSE, IT, ECE etc.) in the university.

- **View Student Info**

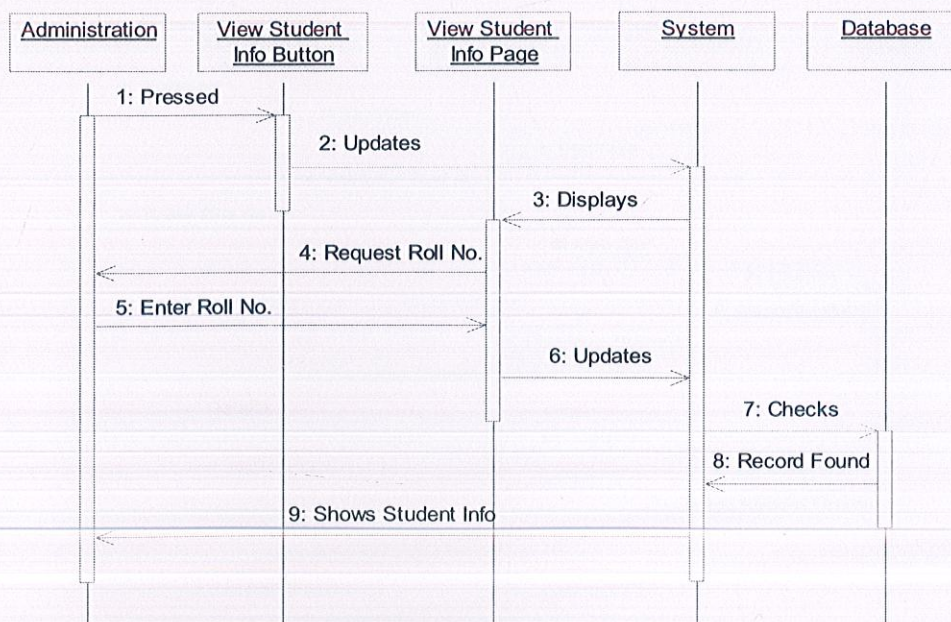


Fig 3.9: Sequence diagram: View Student Information by the Administration

Fig 3.9 is the sequence diagram of Viewing information of a student in the university.



• View faculty Info

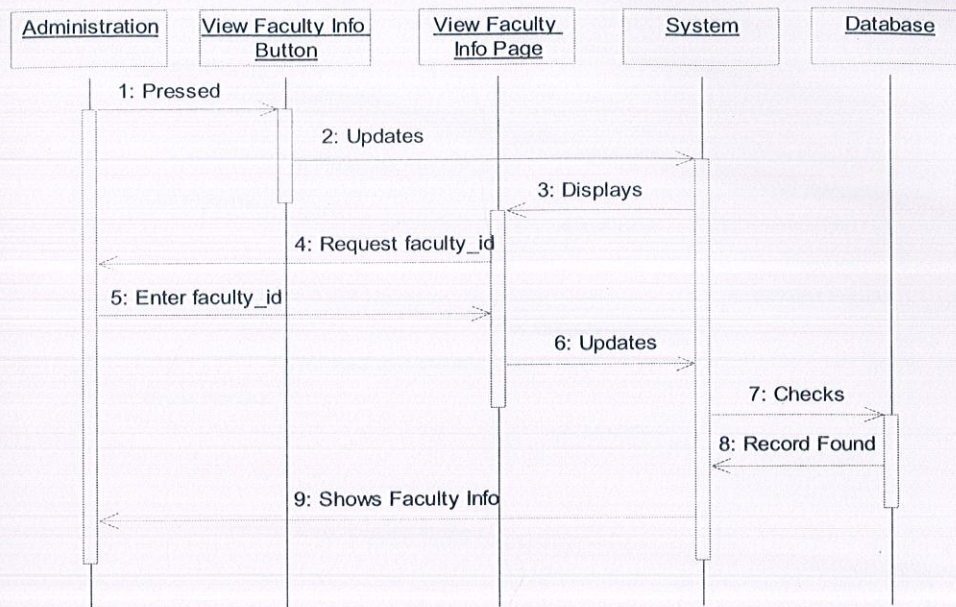


Fig 3.10: Sequence diagram: View Faculty Information by the Administration

Fig 3.10 is the sequence diagram of Viewing information of a faculty member in the university.

• Edit Student Info

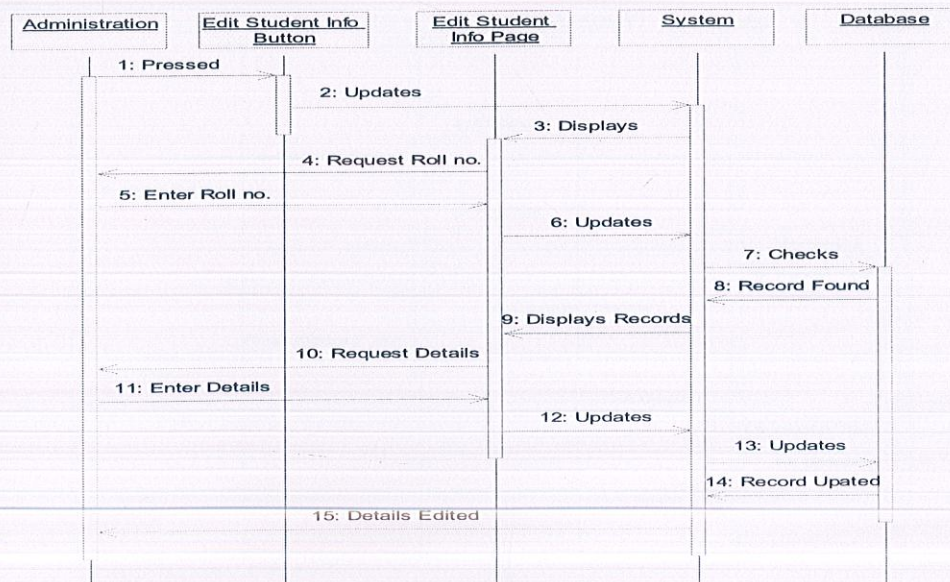


Fig 3.11: Sequence diagram: Edit Student Information by the Administration

Fig 3.11 is the sequence diagram of Editing the information of a student in the university.



- **Edit Faculty Info**

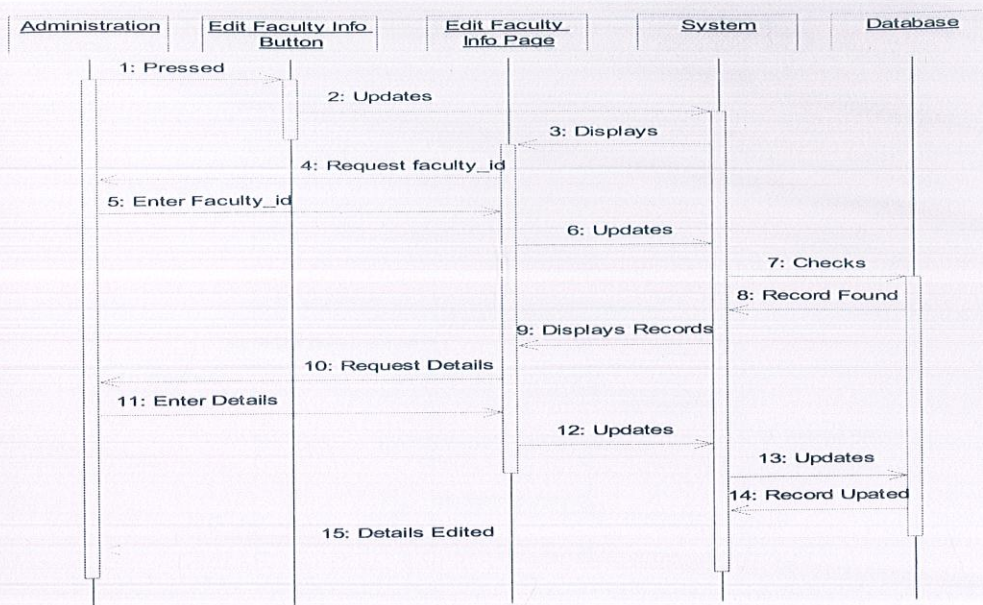


Fig 3.12: Sequence diagram: Edit Faculty Information by the Administration

Fig 3.12 is the sequence diagram of Editing information of a faculty member in the university.

- **Delete Student Record**

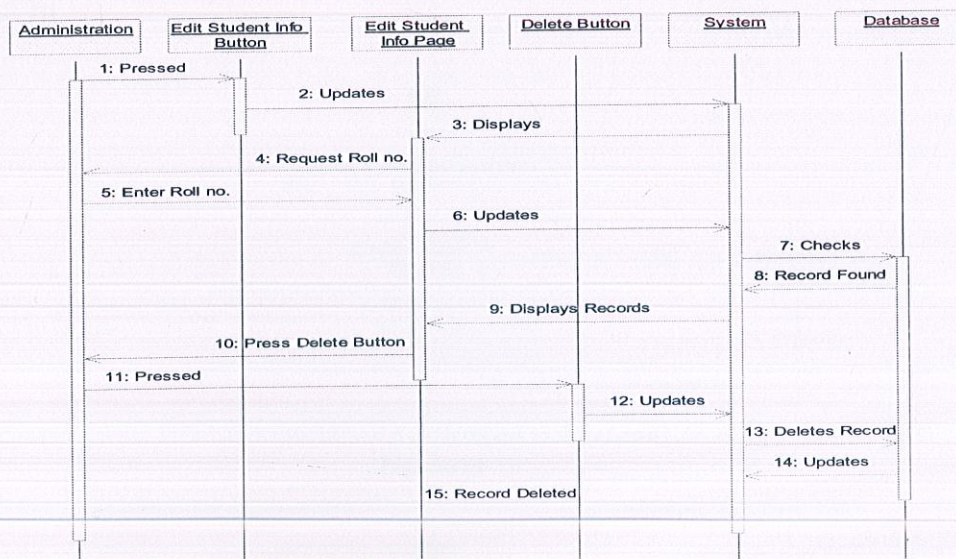


Fig 3.13: Sequence diagram: Delete Student Record by the Administration

Fig 3.13 is the sequence diagram of Deleting Record of a student in the university.



- Delete Faculty info

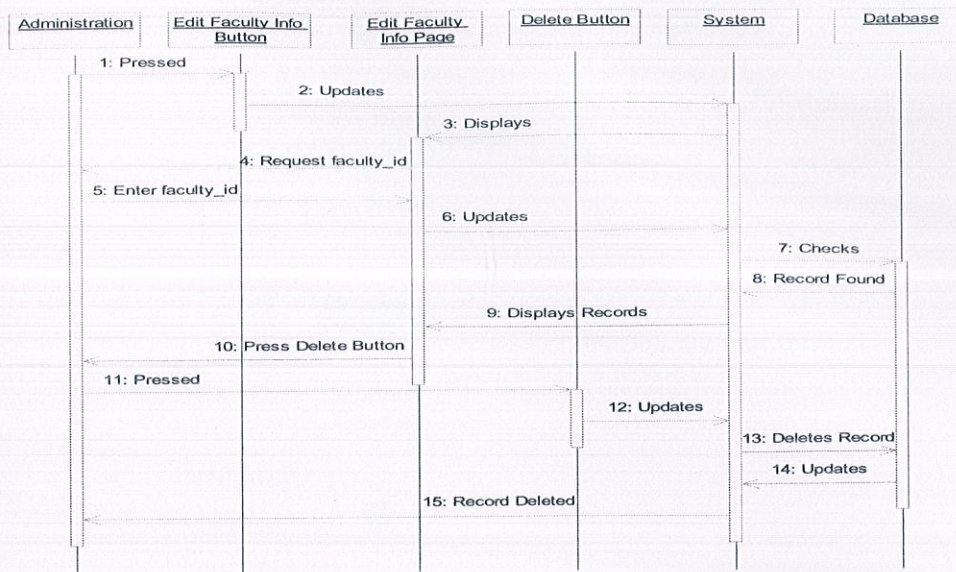


Fig 3.14: Sequence diagram: Delete Faculty Record by the Administration

Fig 3.14 is the sequence diagram of Deleting Record of a faculty in the university.

- Activity Diagrams

- Sign In

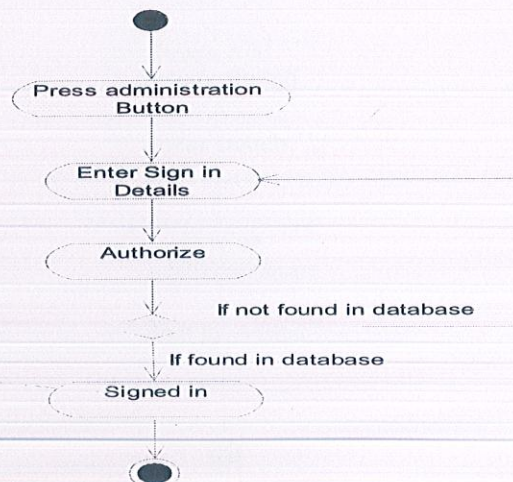


Fig 3.15: Activity diagram: Sign In by the Administration

Fig 3.15 is the activity diagram of login by the Administration in the Academic Resource Management System.



- **Enrol Student**

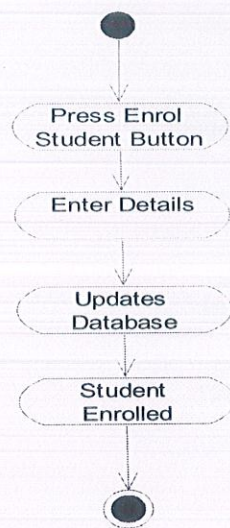


Fig 3.16: Activity diagram: Enrol Student by the Administration

Fig 3.16 is the activity diagram of Enrolling a student by the Administration in the university.

- **Add Faculty**

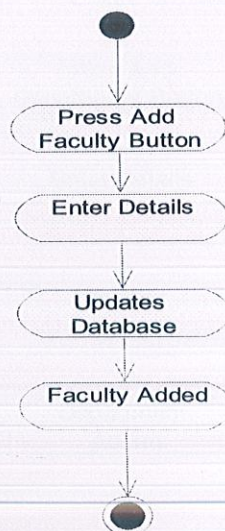


Fig 3.17: Activity diagram: Add Faculty by the Administration

Fig 3.17 is the activity diagram of Adding a faculty member by the Administration in the university.



- **Register student**

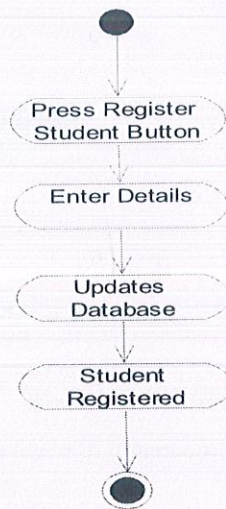


Fig 3.18: Activity diagram: Register Student of Administration

Fig 3.18 is the activity diagram of Registering a student in a semester by the Administration in the university.

- **Add department**

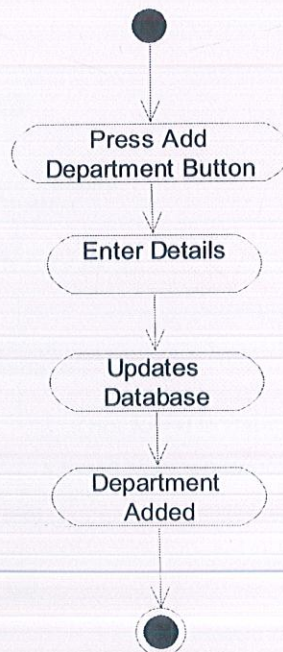


Fig 3.19: Activity diagram: Add Department of Administration

Fig 3.19 is the activity diagram of Adding a department (like CSE, IT, ECE etc.) by the Administration in the university.



- **View Student Info**

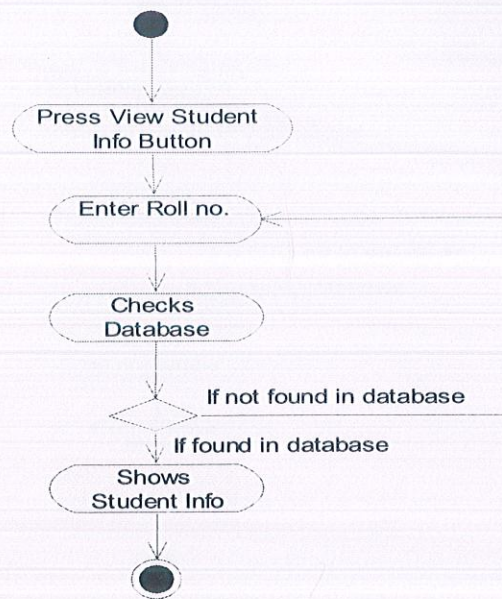


Fig 3.20: Activity diagram: View Student Info of Administration

Fig 3.20 is the activity diagram of Viewing the information of a student by the Administration in the university.

- **View Faculty Info**

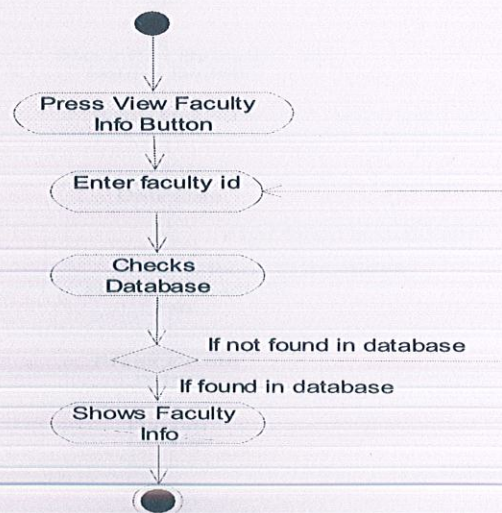


Fig 3.21: Activity diagram: View Student Info of Administration

Fig 3.21 is the activity diagram of Viewing the information of a faculty member by the Administration in the university.



- **Edit Faculty Info**

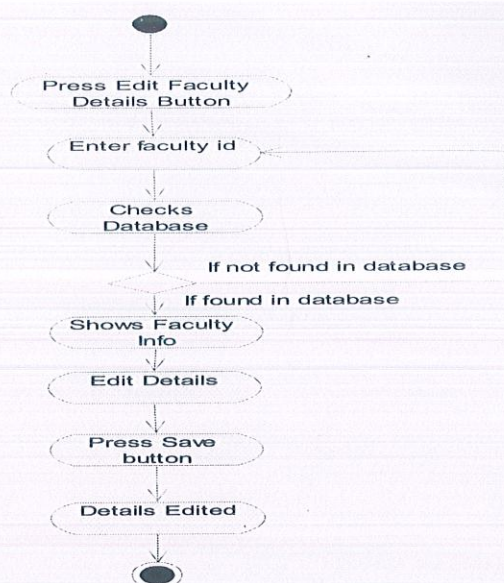


Fig 3.22: Activity diagram: View Student Info of Administration

Fig 3.22 is the activity diagram of Editing the information of a faculty member by the Administration in the university.

- **Delete Student Info**

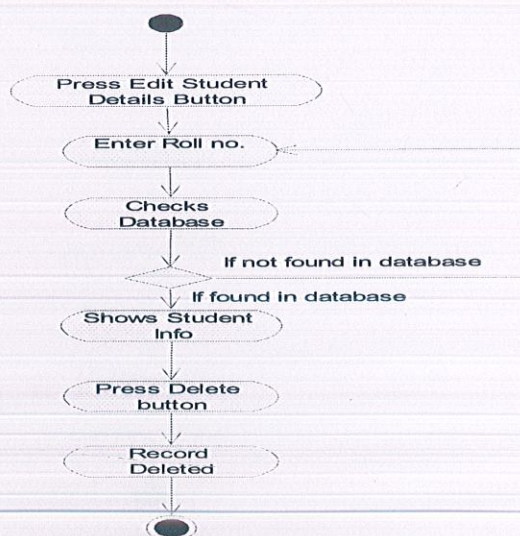


Fig 3.23: Activity diagram: View Student Info of Administration

Fig 3.23 is the activity diagram of Deleting the information of a student by the Administration in the university.



- **Delete Faculty Info**

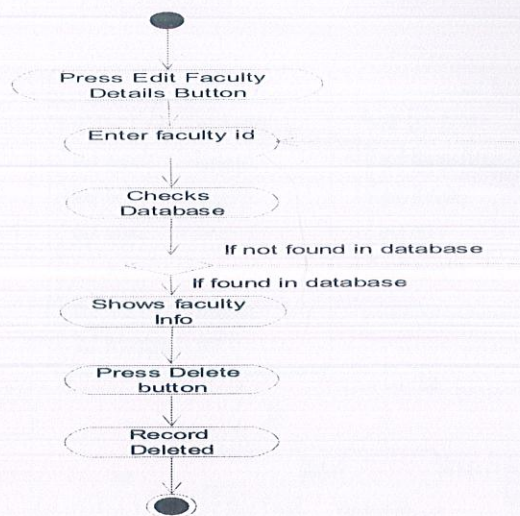


Fig 3.24: Activity diagram: View Student Info of Administration

Fig 3.24 is the activity diagram of Deleting the information of a faculty member by the Administration in the university.

- **Edit Student Info**

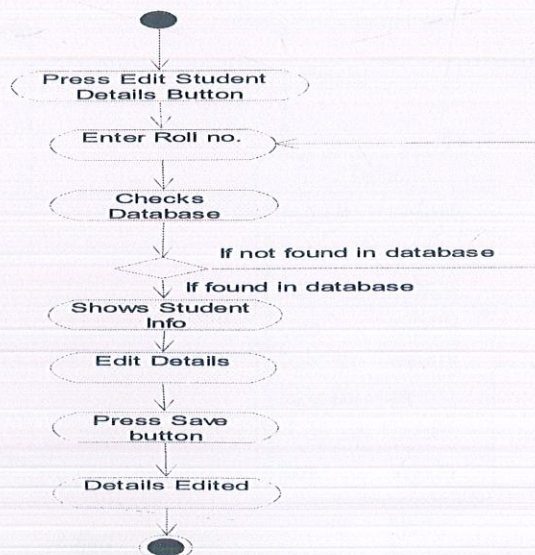


Fig 3.25: Activity diagram: View Student Info of Administration

Fig 3.25 is the activity diagram of Editing the information of a student by the Administration in the university.



## • ER Diagram

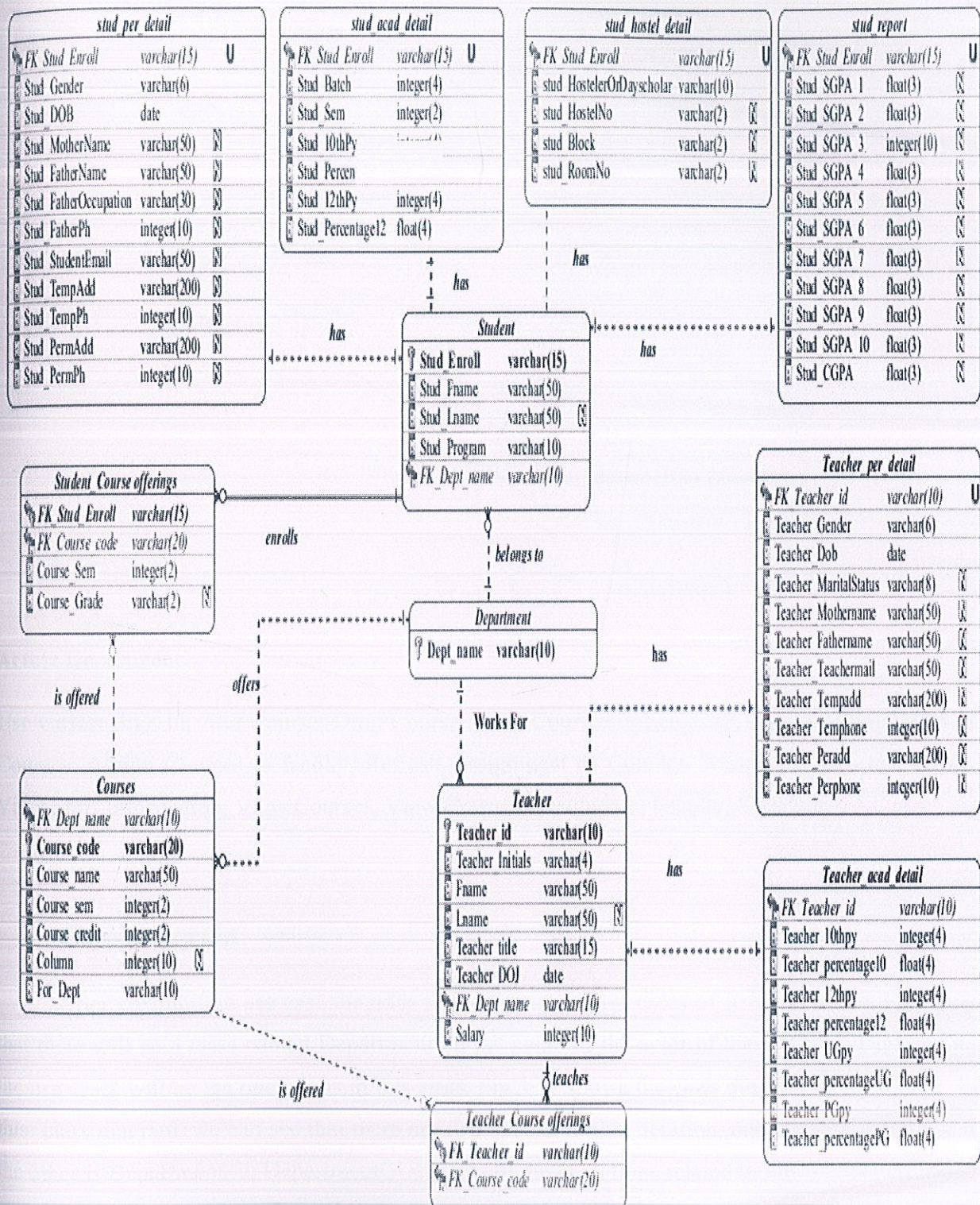


Fig 3.26 Entity Relationship diagram



### 3.4.2 Iteration 2: Department

- Use case diagram

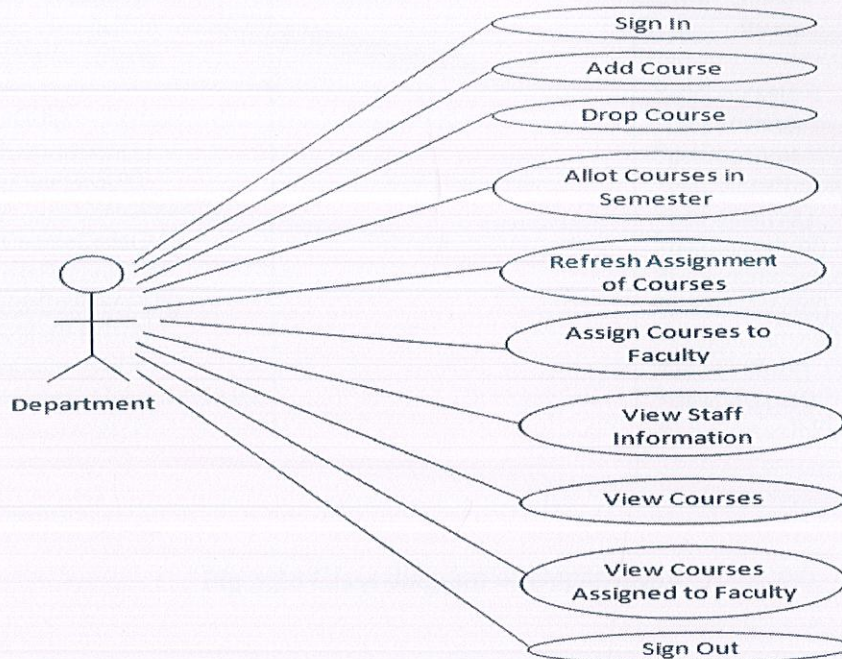


Fig 3.27: Use Case diagram of Department

**Actor:** Department

**Use Cases:** Sign In, Add Course, Drop Course, Allot Courses in Semester, Refresh Assignment of Courses, Assign Courses to faculty, Refresh Assignment of Courses, Assign Courses to Faculty, View Staff Information, View Courses, View Courses Assigned to Faculty, Sign Out.

- Class Diagram

After creating the use case diagram and showing the use cases of iteration 2, we determine that there will be a class named Department on the name of the actor of the use case diagram and the use cases will be the operations in this class. Fig 3.28 shows the class diagram of iteration 2. In this class diagram, we can see that there are two classes in this iteration, one DepartmentGUI and the other is Department. In DepartmentGUI class, all the operations related to the GUI is performed and the database work is performed in the Department class.



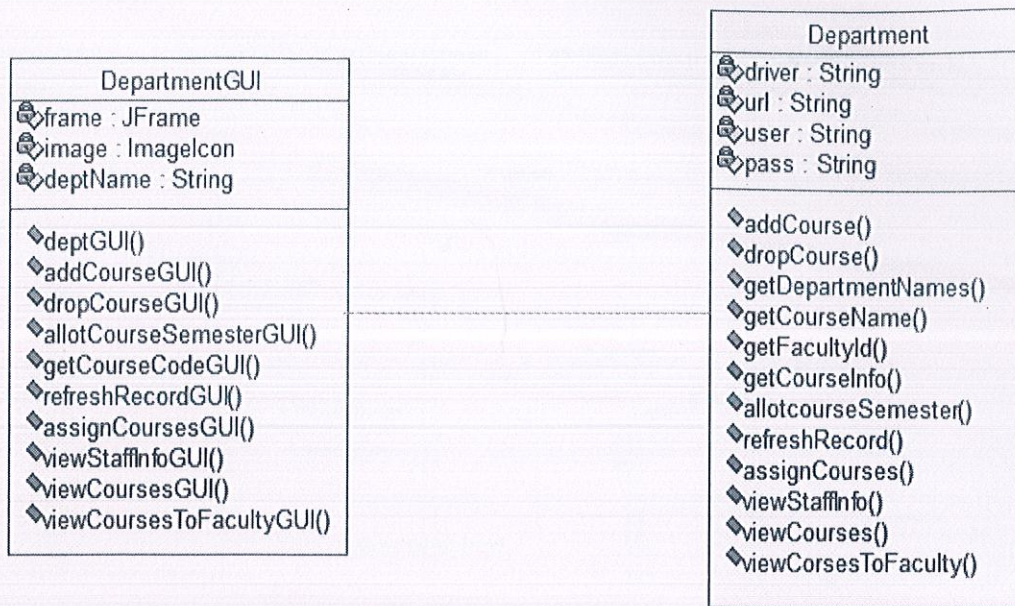


Fig 3.28 Class diagram of Department

## • Sequence Diagrams

### • Add Course

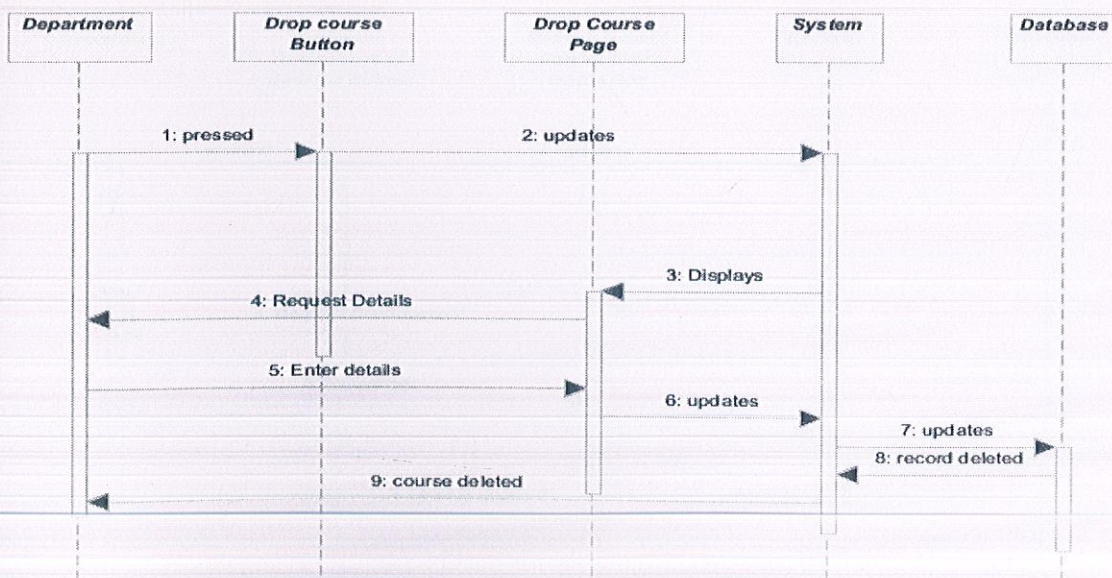


Fig 3.29: Sequence diagram: Add course by the Department

Fig 3.29 is the sequence diagram of Add Course by a Department (like CSE, IT, ECE etc.) in Academic Resource Management System.



- **Allot Courses**

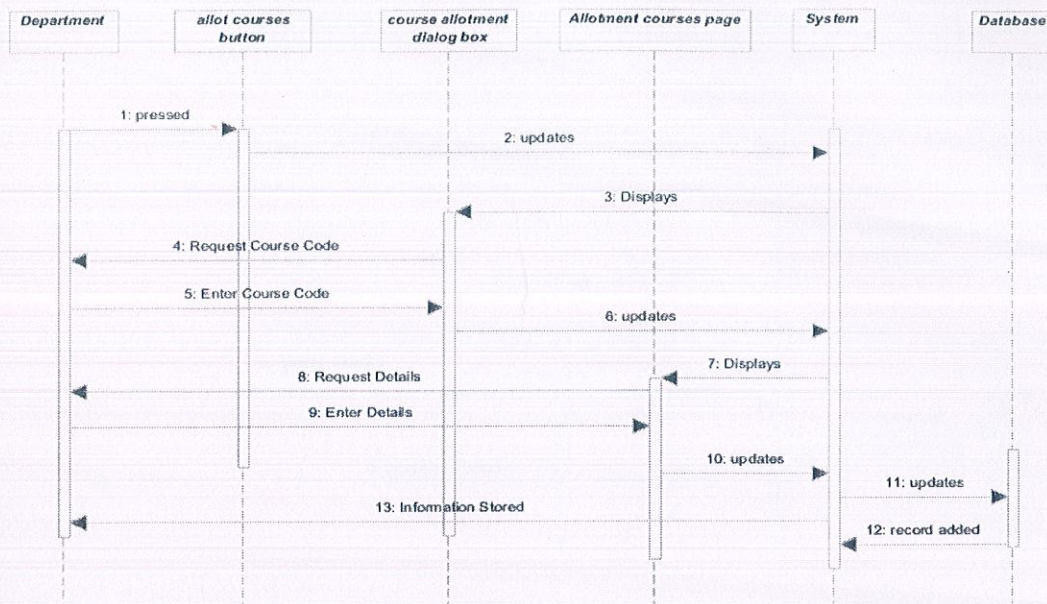


Fig 3.30: Sequence diagram: Allot courses by the Department

Fig 3.30 is the sequence diagram of Allot Courses in semester by a Department in Academic Resource Management System.

- **Refresh assignment of Courses**

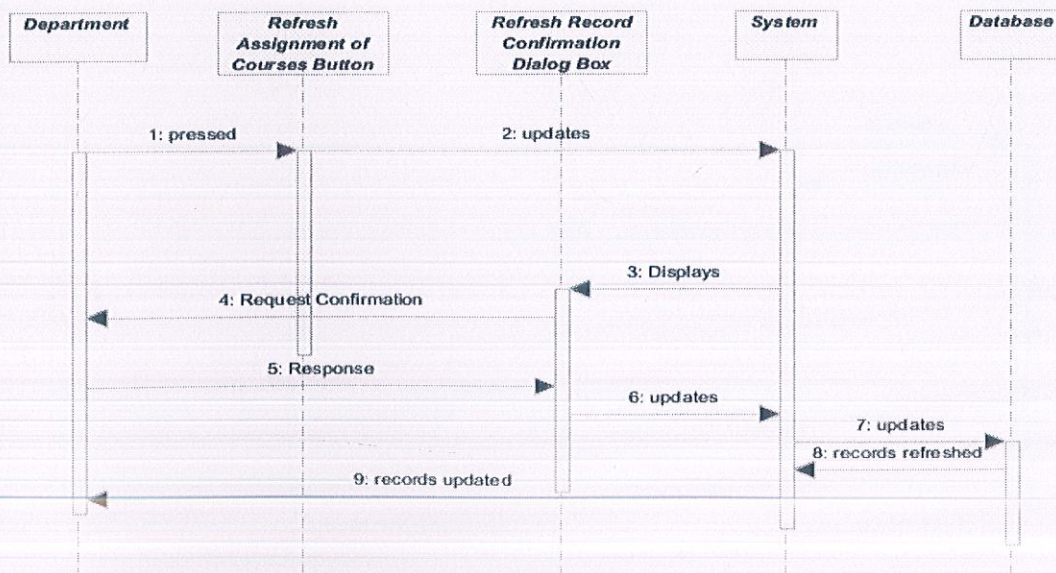


Fig 3.31: Sequence diagram: refresh assignment of courses by the Department

Fig 3.31 is the sequence diagram of Refresh Assignment of Courses in the beginning of every semester by a Department in Academic Resource Management System.



- **Assign Courses to faculty**

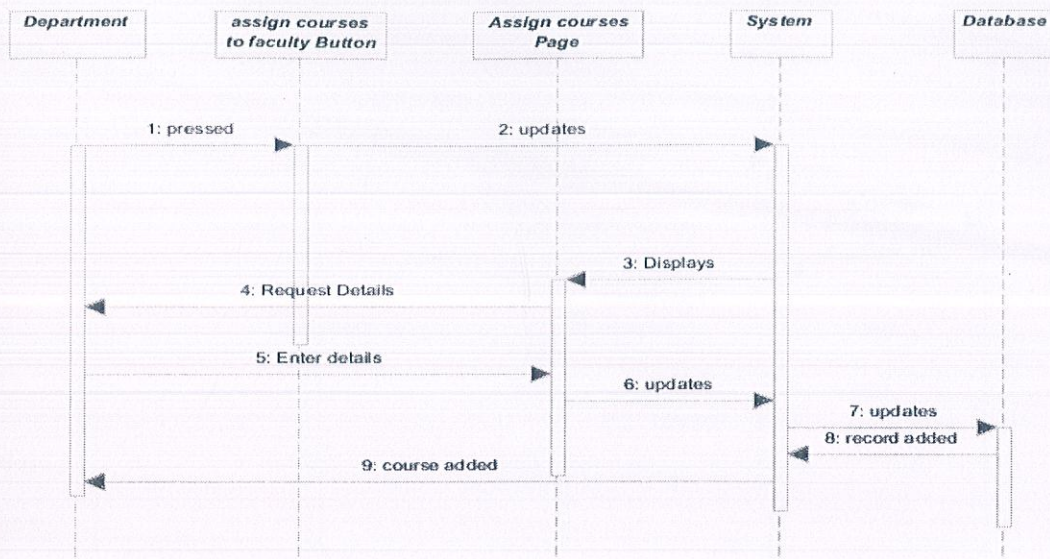


Fig 3.32: Sequence diagram: Assign courses to faculty in department

Fig 3.32 is the sequence diagram of Assign Courses to faculty by a Department in Academic Resource Management System.

- **View Staff info**

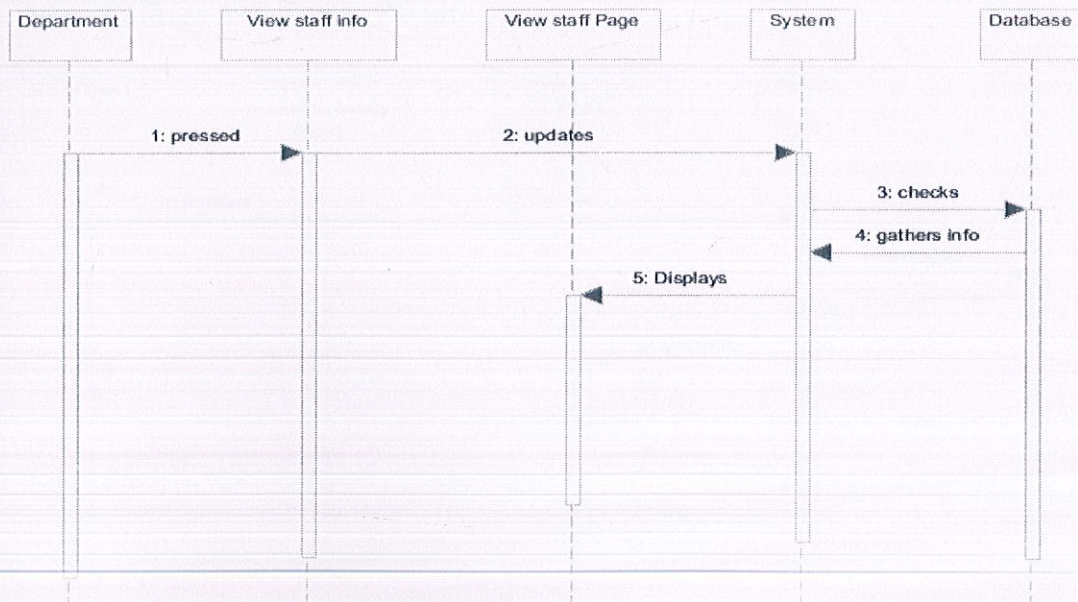


Fig 3.33: Sequence diagram: View Staff info by the Department

Fig 3.33 is the sequence diagram of View Staff information by a Department in Academic Resource Management System.



- **view courses**

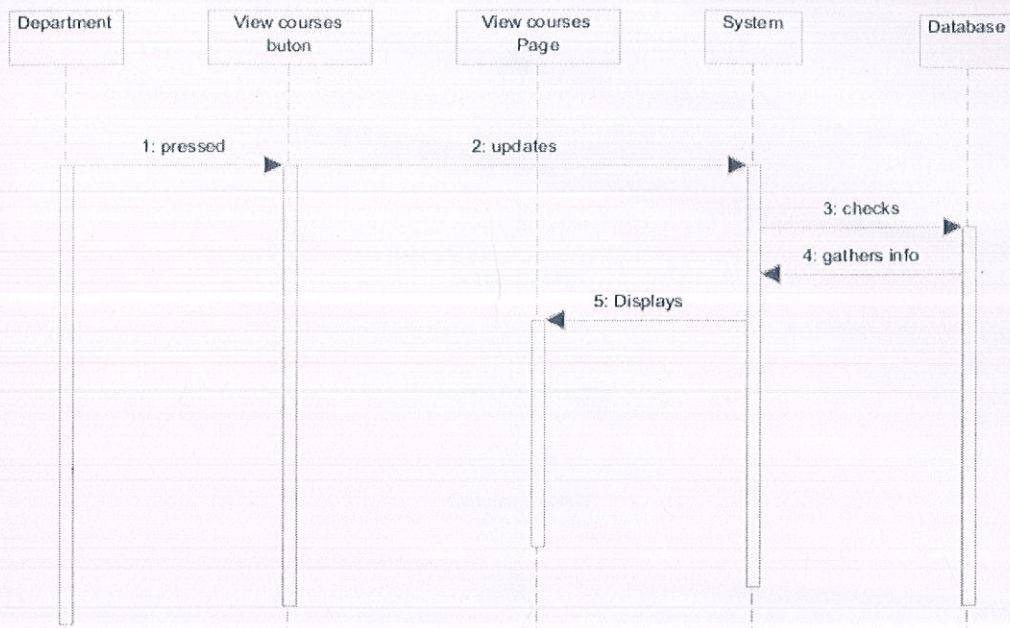


Fig 3.34: Sequence diagram: View courses info by the Department

Fig 3.34 is the sequence diagram of View Courses Information by a Department in Academic Resource Management System.

- **View Courses Assigned to Faculty**

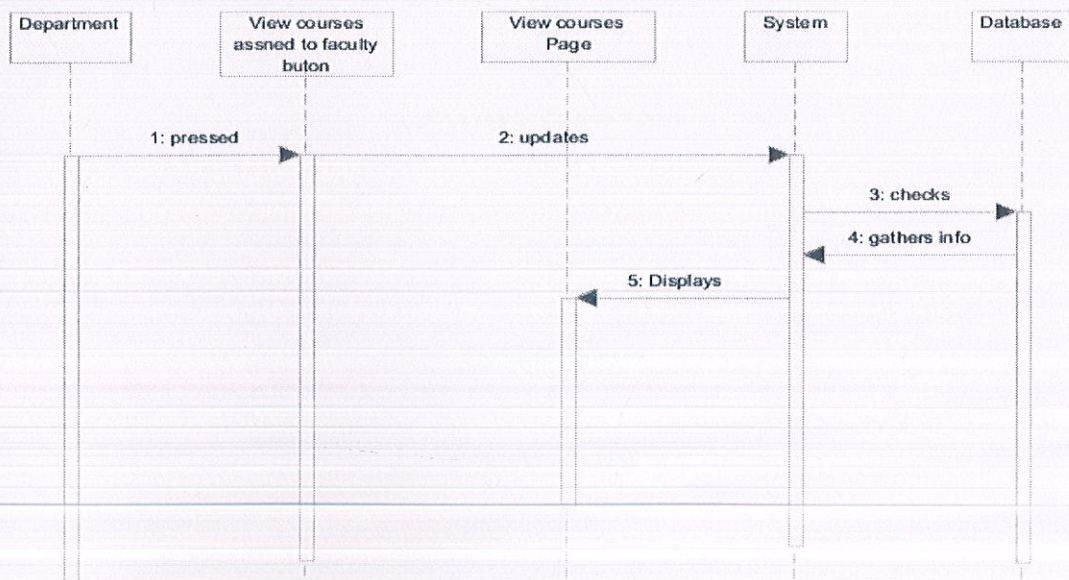


Fig 3.35: Sequence diagram: View Courses Assigned to Faculty by the Department

Fig 3.35 is the sequence diagram of View Courses Assigned to Faculty by a Department in Academic Resource Management System.



- **Activity Diagram**

- **Add Course**

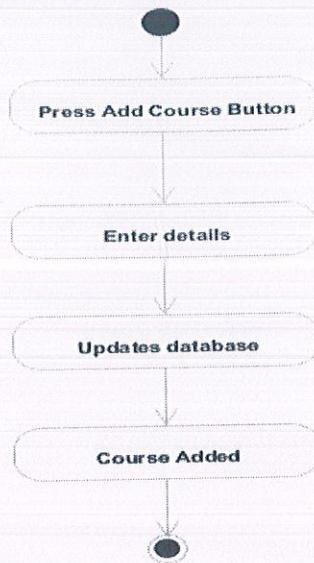


Fig 3.36: Activity diagram: Add Course by the Department

Fig 3.36 is the activity diagram of Add Course by the department in the Academic Resource Management System.

- **Drop Course**

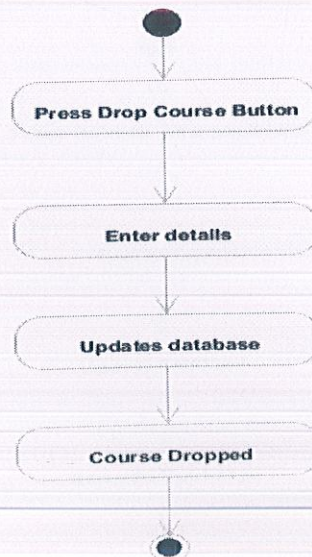


Fig 3.37: Activity diagram: Drop Course by the Department

Fig 3.37 is the activity diagram of Drop Course by the Department in the Academic Resource Management System.



- **Allot Courses**

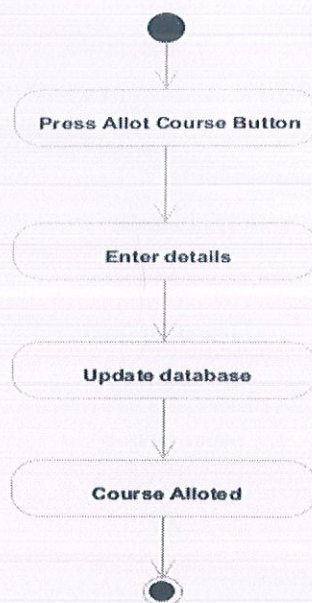


Fig 3.38: Activity diagram: Allot Courses by the Department

Fig 3.38 is the activity diagram of Allotment of Courses by the Department in Academic Resource Management System.

- **Refresh Assignment of Courses**

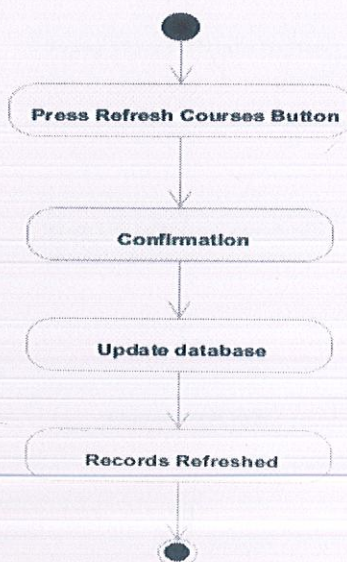


Fig 3.39: Activity diagram: Refresh Assignment of Courses by the Department

Fig 3.39 is the activity diagram of Refresh the assignment of Courses to the faculty in the beginning of every semester by the Department in Academic Resource Management System.



- **Assign Courses to Faculty**

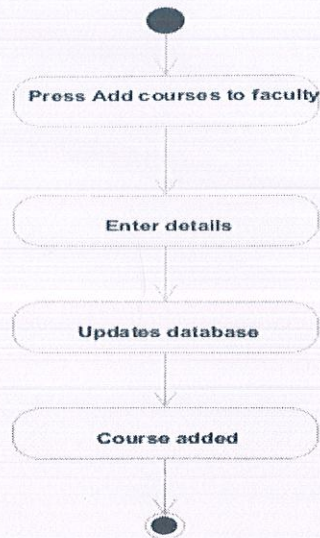


Fig 3.40: Activity diagram: Assignment of Courses to Faculty by the Department

Fig 3.40 is the activity diagram of Assign Course to Faculty by the department in Academic Resource Management System.

- **View Staff info**

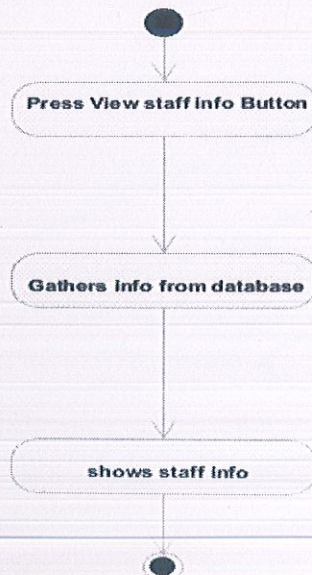


Fig 3.41: Activity diagram:View staff info in department

Fig 3.41 is the activity diagram of Viewing the information of a faculty member by the department in the university.



- View Courses

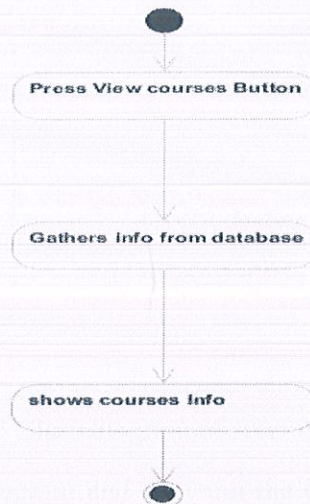


Fig 3.42: Activity diagram:View Courses in Department

Fig 3.42 is the activity diagram of Viewing the Courses in the department by the department in the university.

### 3.4.3 Iteration 3: Account Office

- Use case diagram

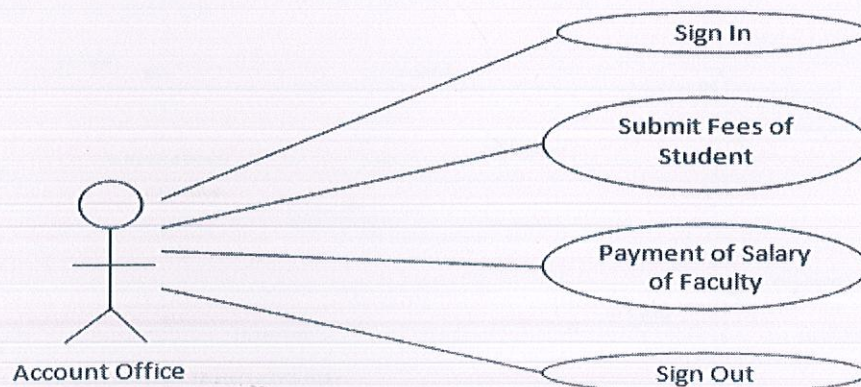


Fig 3.43: Use Case diagram of Account Office

**Actor:** Account Office

**Use Cases:** Sign In, Submit Fees of Student, Payment of Salary of Faculty, Sign Out.



- **Class Diagram**

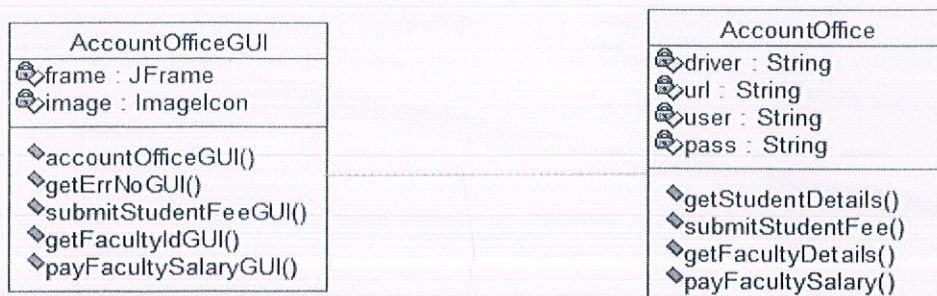


Fig 3.44 Class diagram of Account Office

After creating the use case diagram and showing the use cases of iteration 3, we determine that there will be a class named **AccountOffice** on the name of the actor of the use case diagram and the use cases will be the operations in this class. Fig 3.44 shows the class diagram of iteration 3. In this class diagram, we can see that there are two classes in this iteration, one **AccountOfficeGUI** and the other is **AccountOffice**. In **AccountOfficeGUI** class, all the operations related to the GUI is performed and the database work is performed in the **AccountOffice** class.

- **Sequence Diagrams**

- **Submit Fees of Student**

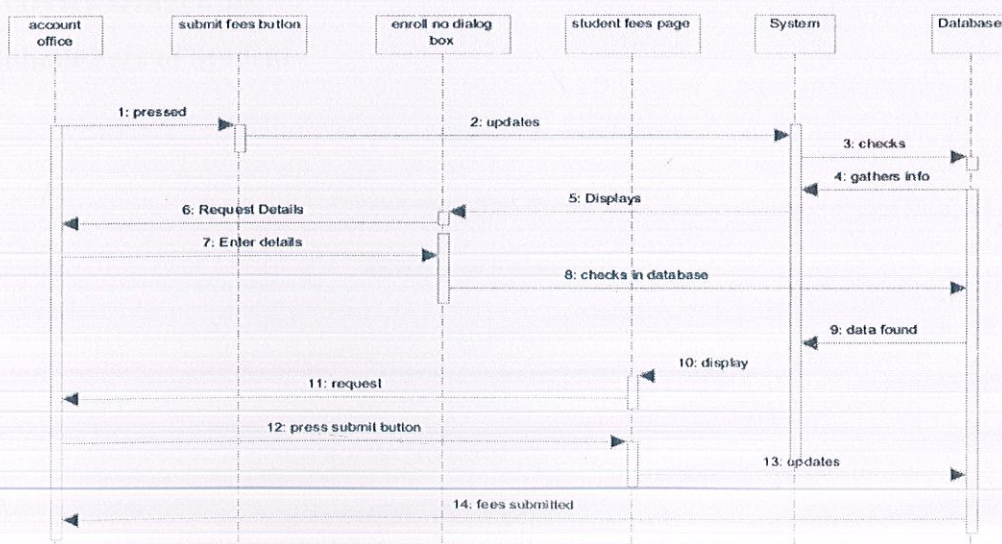


Fig 3.45: Sequence diagram: Submit Fees of Student in Account Office

Fig 3.45 is the sequence diagram of Submit Fees of Student by the Account Office in Academic Resource Management System.



- **Pay Salary to Faculty**

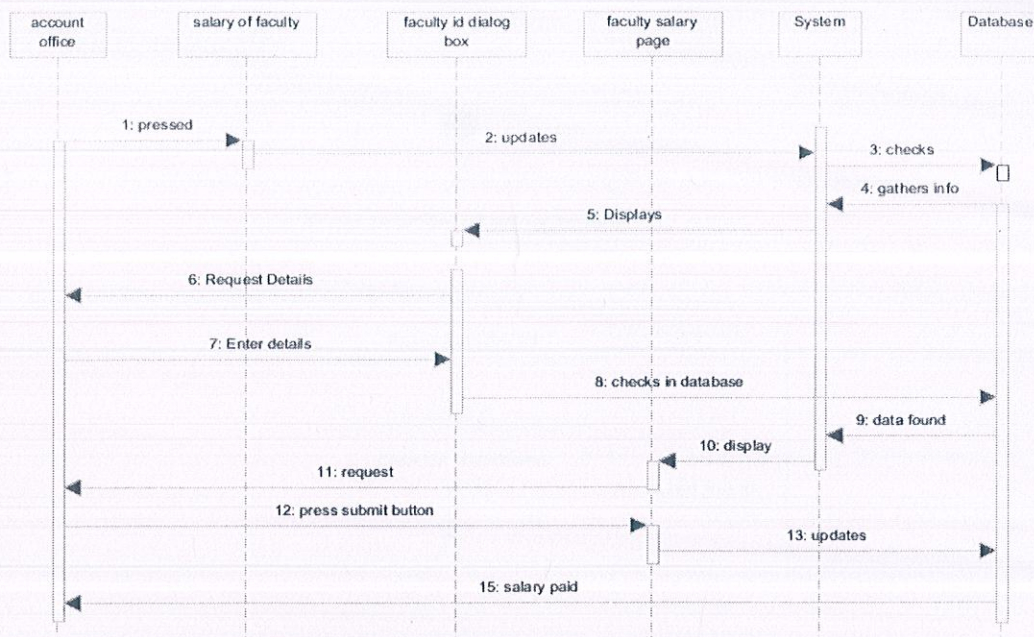


Fig 3.46: Sequence diagram: Pay Salary to Faculty in Account Office

Fig 3.46 is the sequence diagram of Payment of Salary to faculty by the Account Office in Academic Resource Management System.

- **Activity Diagrams**

- **Submit Fees of Student**

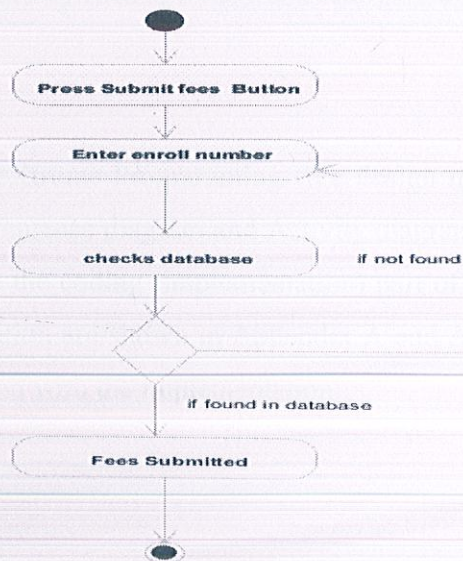


Fig 3.47: Activity diagram: Submit Fees of Student in Account Office



Fig 3.47 is the activity diagram of Submit Fees of Student by the Account Office in Academic Resource management System.

- **Pay Salary to Faculty**

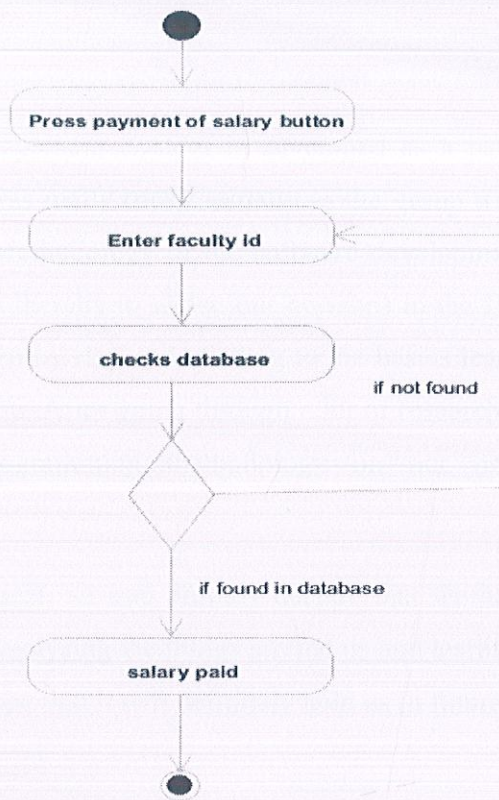


Fig 3.48: Activity diagram : Pay Salary to Faculty in Account Office module

Fig 3.48 is the activity diagram of Payment of Salary to Faculty by the Account Office in Academic Resource management System.

In this chapter, we implement eXtreme Programming. We design many UML diagrams like Use Case diagram, Class diagram, Sequence diagram and Activity diagram. After completing the design of all the iterations, we move to the coding (implementation) part of that iteration. The coding and the snapshots of the implementation are shown in Appendix A and Appendix B. The testing of the implementation goes hand in hand with the implementation.



# 4. Conclusion and Future Scope

## 4.1 Conclusion

Academic Resource Management System is developed as a part of the degree completion process in the fourth year. It was really tricky initially as the group was naïve when it came to the implementation of the Agile Methodology of the software development process. The task in hand was to develop the system and thereby to widen our horizons in the field of Agile Methodologies. The best practice was to go step by step and sticking to the basics learnt in the past 3 years which really worked well for the group. After going through a lot of research, there came a belief that we can implement eXtreme Programming methodology on our project, Academic Resource Management System.

After completing this project, we can proudly declare that we have learned how to go for a project systematically by first analysing the things around us and learning from the mistakes to give the best possible output. We hope that it will definitely help us in future and help us to reach heights in our professional careers

The skills learned in through this project can be summarized as:

- XP which is one of the Agile Methodologies to develop a project.
- How to work on JAVA Swings and how different components in that can be used as and when required.
- JAVA classes.
- Had a very good experience of working on Netbeans IDE 7.0.
- Oracle Database 10g Express Edition, in which database of the project is created and managed.
- JDBC connectivity to the Oracle Database 10g Express Edition.

## 4.2 Future Scope

Academic Resource Management System is a very vast project. In this project, we have completed the administration and department related work and some work of accounts department.



In future, the project can be extended by including some more entities like faculty which is an essential part of academics. When faculty work will be added, one more feature of administration that is creating results can also be added in the project. The accounts department is itself a very big part of academics. That can also be added in the project. Then library management and hostel management are also a part of academics on which we can think. By doing all this, a person or student can get knowledge of programming and many other things that are used in this project.



# Appendix A

## Codes

### 1. ARMS.java

```
public class ARMS
{
    public static void main(String[] args) {}
    public void loginFormGUI() {}
}
```

### 2. Login.java

```
public class Login
{
    public ArrayList<String> matchLoginDetails(){}
}
```

### 3. AdministrationGUI.java

```
public class AdministrationGUI
{
    public void adminGUI() {}
    public void enrolStudentGUI() {}
    public void uploadPicGUI() {}
    public void addFacultyGUI() {}
    public void getStudentInfoGUI() {}
    public void viewStudentInfoGUI() {}
    public void getFacultyInfoGUI() {}
    public void viewFacultyInfoGUI() {}
    public void editStudentInfoGUI() {}
    public void editFacultyinfoGUI() {}
}
```



```

    public void addDepartmentGUI() { }

    public void batchFeeSpecificationGUI() { }

    public void getFeeDetails() { }

}

```

#### 4. Administration.java

```

public class Administration
{
    public int checkAvailability() {}

    public boolean enrolStudent() {}

    public boolean addFaculty() {}

    public int getStudentInfo() {}

    public int getFacultyInfo() {}

    public boolean changeStudentinfo() {}

    public boolean deleteStudentinfo() {}

    public boolean changeFacultyinfo() {}

    public boolean deleteFacultyinfo() {}

    public boolean addDepartment() {}

    public int getStudentCourseInfo() {}

    public int registerStudent() {}

    public int batchFeeSpecification() {}

    public String getFeeDetails() {}

    public int changeFeeSpecification(String str[]) {}

}

```



## 5. DepartmentGUI.java

```
public class DepartmentGUI

{

    public void deptGUI(){}

    public void addCourseGUI(){}

    public void dropCourseGUI(){}

    public void allotCourseSemesterGUI(){}

    public void getCourseCodeGUI(){}

    public void refreshRecordGUI(){}

    public void assignCoursesGUI(){}

    public void viewStaffInfoGUI(){}

    public void viewCoursesGUI() {}

    public void viewCoursesToFacultyGUI(){}

}
```

## 6. Department.java

```
public class Department

{

    public int addCourse(){}

    public int dropCourse(){}

    public String[] getDepartmentNames(){}

    public String getCourseName(){}

    public String[] getFacultyId(){}

    public String[] getCourseInfo(){}

    public boolean allotCourseSemester(){}

}
```



```

    public boolean refreshRecord(){}

    public boolean assignCourses(){}

    public ArrayList<String> viewStaffInfo(){}

    public ArrayList<String> viewCourses(){}

    public ArrayList<String> viewCoursesToFaculty(){}

}

```

## 7. AccountOfficeGUI.java

```

public class AccountOfficeGUI
{

    public void AccOfficeGUI(){}

    public void getErrNoGUI() {}

    public void submitStudFeeGUI(){}

    public void getFacultyIdGUI(){}

    public void payFacultySalaryGUI(){}

}

```

## 8. AccountOffice.java

```

public class AccountOffice
{

    public String getStudDetails(){}

    public int submitStudFee(){}

    public String getFacultyDetails(){}

    public int payFacultySalary(){}

}

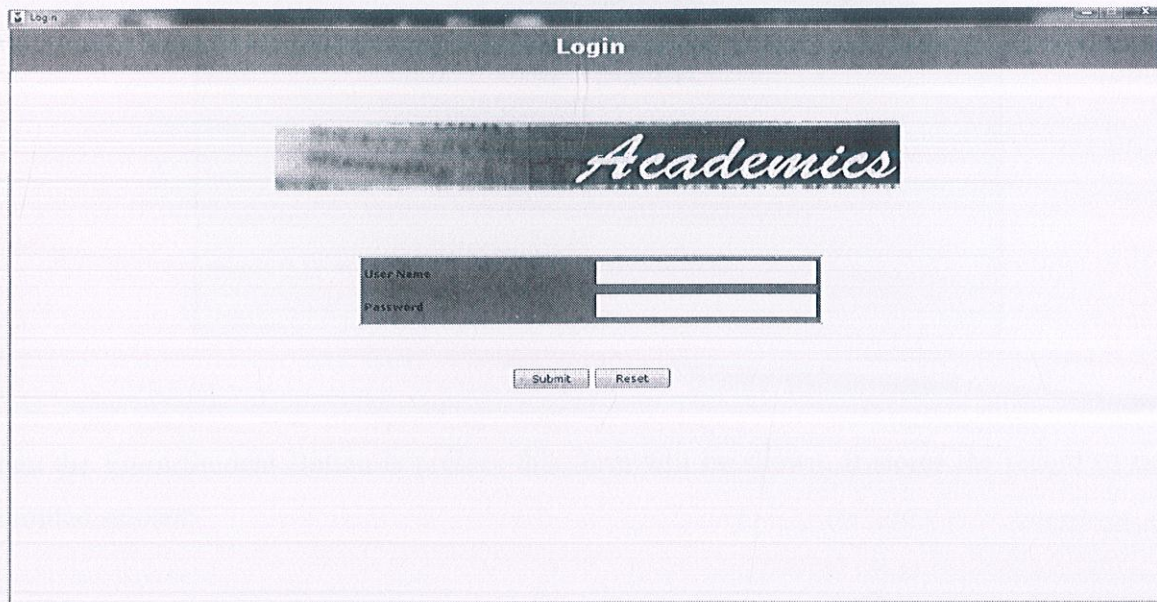
```



## Appendix B

### Snapshots

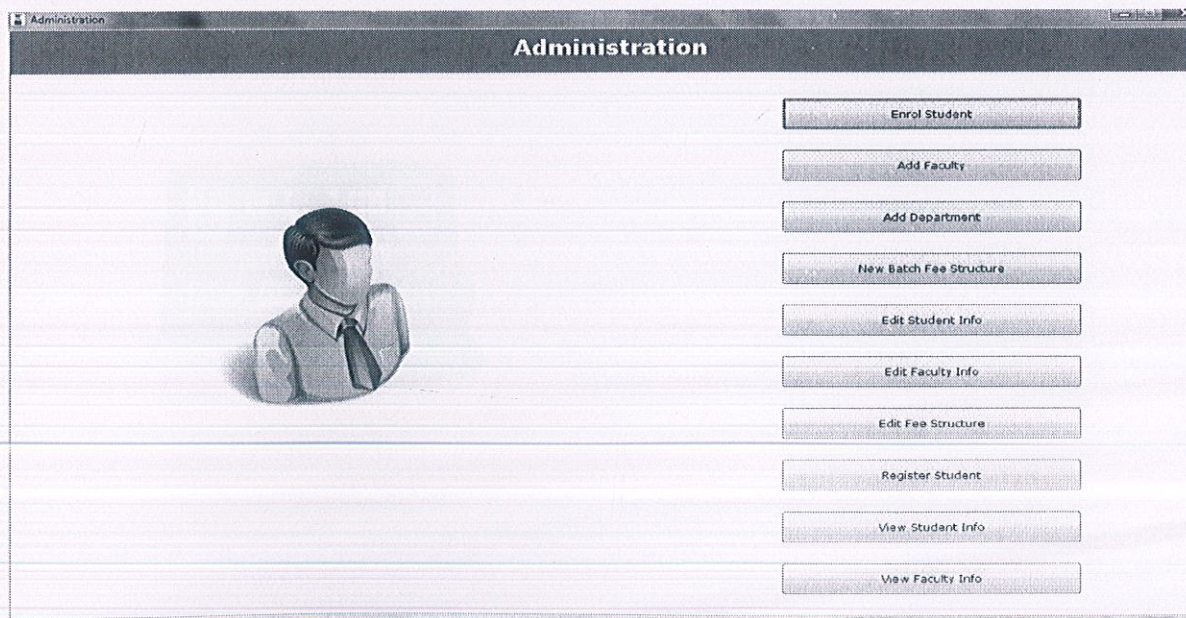
#### Login Form



The screenshot shows a web browser window titled "Login". The main heading is "Academics" in a stylized font. Below the heading, there are two input fields: "User Name" and "Password". At the bottom of the form, there are two buttons: "Submit" and "Reset".

Every entity in the Academic Resource Management System will first login into the system.

#### Administration Form



The screenshot shows a web browser window titled "Administration". On the left side, there is a small illustration of a man in a suit. On the right side, there is a vertical list of buttons: "Enrol Student", "Add Faculty", "Add Department", "New Batch Fee Structure", "Edit Student Info", "Edit Faculty Info", "Edit Fee Structure", "Register Student", "View Student Info", and "View Faculty Info".

When administration login into the system, he will be shown administration form.



## Student Enrolment Form

The screenshot shows a web application window titled "Student Enrolment Form". At the top, there is a header bar with the title. Below the header, on the left, is a graduation cap icon. To its right are four input fields: "Enrolment Number", "Program Name", "Semester", and "Branch", each with a "Select" dropdown menu. Further right is a profile picture placeholder with an "Upload Pic" button. Below these fields is a tabbed interface with three tabs: "Personal Details", "Academic Information", and "Hostel Detail". The "Personal Details" tab is active, showing a form with the following fields: "First Name", "Last Name", "Gender" (with radio buttons for Male and Female), "Date of Birth" (with DD, MM, and YY dropdowns), "Father's Name", "Mother's Name", "Father's Occupation", "Father's Ph. Number", "Student Email ID", "Temporary Address", "Temporary Ph. Number", "Permanent Address", and "Permanent Ph. Number". At the bottom of the form are three buttons: "Save", "Reset", and "Close".

When the Enrol Student Button is pressed this form will be shown. It stores the record of newly admitted student.

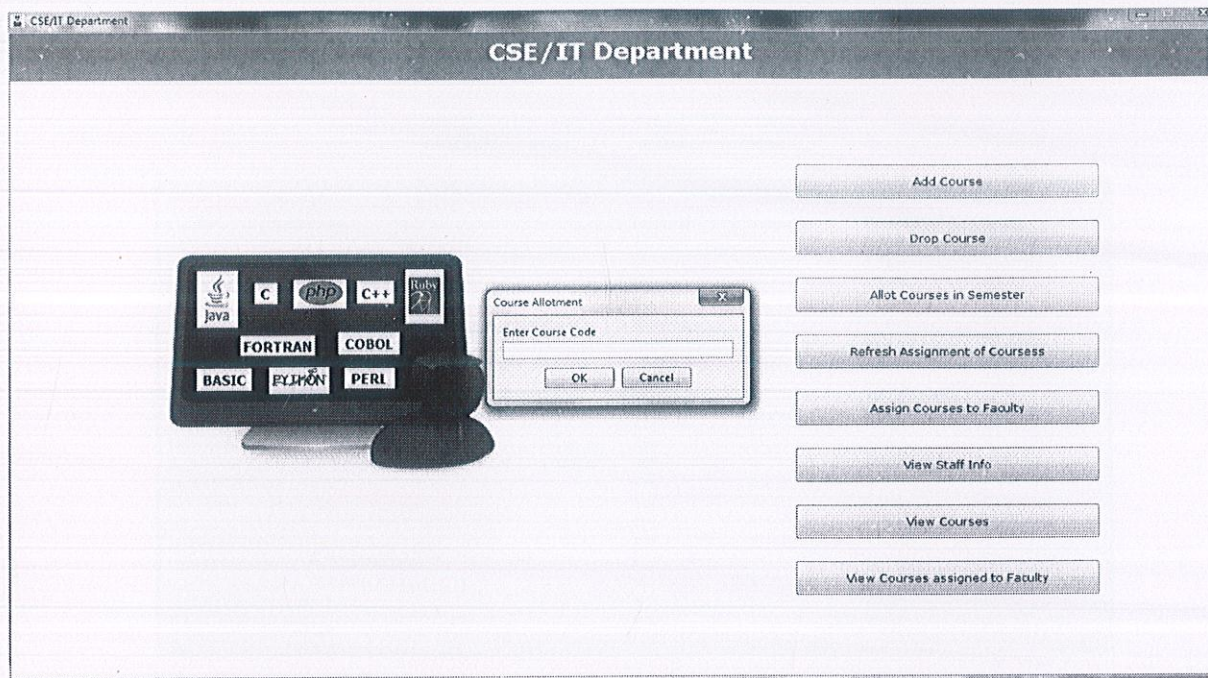
## Department Form

The screenshot shows a web application window titled "CSE/IT Department". On the left side, there is a graphic of a computer monitor displaying various programming language logos: Java, C, PHP, C++, Fortran, COBOL, BASIC, Python, and Perl. To the right of the monitor is a vertical list of eight buttons: "Add Course", "Drop Course", "Allot Courses in Semester", "Refresh Assignment of Courses", "Assign Courses to Faculty", "View Staff Info", "View Courses", and "View Courses assigned to Faculty".

When a department like CSE/IT login into the System, this form with department name will be shown.



## Course Allotment Dialog Box



When any department presses the Allot Courses in Semester Button this dialog box will be shown.

## Allotment of Courses in Semester Form

The screenshot shows a form titled "Allotment of Courses in Semesters". At the top, it displays "Course Code" as 07B21CI702 and "Course Name" as DATA STRUCTURES AND COMPUTER PROGRAMMING LAB. Below this is a section titled "Select Branch and Semester for this Course" containing a table with three columns: "Program Name", "Branch", and "Semester".

Program Name	Branch	Semester
B.Tech.	<input type="checkbox"/> BI	-----Select-----
B.Tech.	<input type="checkbox"/> BT	-----Select-----
B.Tech.	<input type="checkbox"/> CIVIL	-----Select-----
B.Tech.	<input type="checkbox"/> CSE	-----Select-----
B.Tech.	<input type="checkbox"/> ECE	-----Select-----
B.Tech.	<input type="checkbox"/> HSS	-----Select-----
B.Tech.	<input type="checkbox"/> IT	-----Select-----
B.Tech.	<input type="checkbox"/> Maths	-----Select-----
B.Tech.	<input type="checkbox"/> Physics	-----Select-----
M.Tech.	<input type="checkbox"/> CSE	-----Select-----
M.Tech.	<input type="checkbox"/> IT	-----Select-----

At the bottom of the form are three buttons: "Save", "Reset", and "Close".

When the course code will be entered in the previous dialog box this form will be shown.



## Assignment of Courses Form

Assignment of Courses

Faculty ID:  Select.....

Semester: ☒ EVEN ☐ ODD


Course Name (Course Code)	For Dept.	Semester	Program	Select
DATA STRUCTURES AND COMPUTER PROGRAMMING LAB (07B21CI702)	IT	2	B.Tech.	<input type="checkbox"/>
DATA STRUCTURES (07B21CI102)	IT	2	B.Tech.	<input type="checkbox"/>
NETWORK MANAGEMENT (11B1WCI836)	IT	2	M.Tech.	<input type="checkbox"/>
INTRODUCTION TO COMPUTERS AND PROGRAMMING (07B11CI101)	BI	2	B.Tech.	<input type="checkbox"/>
INTRODUCTION TO COMPUTERS AND PROGRAMMING (07B11CI101)	BT	2	B.Tech.	<input type="checkbox"/>
NETWORK MANAGEMENT (11B1WCI836)	CSE	2	M.Tech.	<input type="checkbox"/>
INTRODUCTION TO COMPUTERS AND PROGRAMMING (07B11CI101)	CIVIL	2	B.Tech.	<input type="checkbox"/>
COMPUTER PROGRAMMING LAB (07B11CI701)	BI	2	B.Tech.	<input type="checkbox"/>
COMPUTER PROGRAMMING LAB (07B11CI701)	BT	2	B.Tech.	<input type="checkbox"/>
COMPUTER PROGRAMMING LAB (07B11CI701)	CIVIL	2	B.Tech.	<input type="checkbox"/>
DATA STRUCTURES AND COMPUTER PROGRAMMING LAB (07B21CI702)	CSE	2	B.Tech.	<input type="checkbox"/>
DATA STRUCTURES AND COMPUTER PROGRAMMING LAB (07B21CI702)	ECE	2	B.Tech.	<input type="checkbox"/>
DATA STRUCTURES (07B21CI102)	CSE	2	B.Tech.	<input type="checkbox"/>
DATA STRUCTURES (07B21CI102)	ECE	2	B.Tech.	<input type="checkbox"/>
ADVANCED SOFTWARE ENGINEERING (10M11CI213)	IT	2	M.Tech.	<input type="checkbox"/>
ADVANCED SOFTWARE ENGINEERING (10M11CI213)	CSE	2	M.Tech.	<input type="checkbox"/>
UNIX PROGRAMMING LAB (07B31CI705)	ECE	4	B.Tech.	<input type="checkbox"/>
FUNDAMENTALS OF ALGORITHMS (10B11CI411)	IT	4	B.Tech.	<input type="checkbox"/>
ALGORITHMS LAB (10B17CI471)	IT	4	B.Tech.	<input type="checkbox"/>
ALGORITHMS LAB (10B17CI471)	CSE	4	B.Tech.	<input type="checkbox"/>

Save Reset Close

When department presses the assign courses to faculty button this form will be shown. In this form if we click on the even radio button then subjects only in even semester will be shown.

## Accounts Department Form

Accounts Department



Submit Fees of Student

Payment of salary to Faculty

When Account Office login into the system this form is shown.



## Student Fee Form

Enrollment No.	091215	Name	Rajat Kumar Jain
Current Sem.	8	Branch	CSE
Program	B.Tech.	Batch	2012

Tuition Fee	30000
Hostel Fee	27500
Total	57500

Authority Signature

When Submit Student Fee Button is pressed, this form is shown.

## Faculty Salary form

Faculty Id	abc
Name	Mr. abc abc
Department	ECE
Salary	70000
Payment Date	<input type="text" value="MMM"/> <input type="text" value="YYYY"/>

When Payment of Faculty Salary Button is pressed this form is shown.



## References

- [1]. Jeffrey A. Livermore, "Factors that Significantly Impact the Implementation of an Agile Software Development Methodology", *Journal of Software*, Vol. 3(4), pp. 31-36, April 2008
- [2]. Boris Matijašević, Hrvoje Ronevi, Ognjen Orel, "Agile Software Development Supporting Higher Education Reform", *International Conference on Information Technology Interfaces*, pp.375-380, June 2007
- [3]. Scott W. Ambler, "Agile Database Techniques", Wiley Publishing Inc, 2003.
- [4]. D. Wells, "Extreme Programming: A gentle introduction", <http://www.extremeprogramming.org>, March 2007
- [5]. J. Hunt, *Agile Software Construction*, London: Springer-Verlag press, 2006
- [6]. D. Wells, "The Rules of Extreme Programming", <http://www.extremeprogramming.org>, September 28, 2009
- [7]. Sheetal Sharma, Darothi Sarkar, Divya Gupta, "Agile Processes and Methodologies: A Conceptual Study", *International Journal on Computer Science and Engineering*, Vol. 4(5), pp. 892-898, May 2012
- [8]. D. Lurance, D. Mancl, "Extreme Programming and Agile Processes", [http://princetonaacm.acm.org/downloads/extreme\\_agile.html](http://princetonaacm.acm.org/downloads/extreme_agile.html), December 19, 2002
- [9]. Indika, "Difference Between Agile and Traditional Software Development Methodology", <http://www.differencebetween.com/difference-between-agile-and-vs-traditional-software-development-methodology>, Jul 10th, 2011
- [10]. Beck, K., *Extreme Programming Explained: Embrace Change*: Addison-Wesley, 2000, ISBN 201-61641-6.
- [11]. Erickson, J., Lyytinen, K., and Siau, K., "Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research," *Journal of Database Management*, no. 4, vol. 16, pp. 88 - 100, 2005.
- [12]. Kalra, B., Thomson, C., and Holcombe, M., "The Software Hut - a Student Experience of Extreme Programming with Real Commercial Clients," XP, 2005.
- [13]. Schneider, J. G. and Johnston, L., "Extreme Programming – Helpful or Harmful in Educating



Undergraduates,” Journal of Systems and Software, vol. 74, pp. 121-132, 2005.

[14]. Shukla, A. and Williams, L., “Adapting Extreme Programming for a Core Software Engineering Course,” in Proceedings. 15th Conference on Software Engineering Education and Training (Csec&T), 2002, pp. 184-191.

[15]. Stephens, M. and Rosenberg, D., “Extreme Programming Refactored: The Case against XP”, Berkeley, CA: Apress, 2003, ISBN 1-59059-096-1.

[16]. T. Gaddis, Starting out with JAVA, New Delhi: Wiley Dreamtech India Pvt. Ltd., 2005

[17]. H. Schildt, The Complete Reference JAVA, 7th edition. New Delhi: Tata McGraw-Hill Publishing Company, 2007

[18]. “Creating a Frame”, [http://www.roseindia.net/java/example/java/swing/Swing\\_index.shtml](http://www.roseindia.net/java/example/java/swing/Swing_index.shtml), April 17, 2011.

[19]. “Java Swing Tutorials”, <http://www.roseindia.net/java/example/java/swing>, March 20, 2008.

[20]. “Create a JRadioButton Component in Java”  
<http://www.roseindia.net/java/example/java/swing/CreateRadioButton.shtml>, April 14, 2007.