**ONLINE THESIS ARCHIVING SYSTEM**

A PROJECT REPORT

Submitted in partial fulfillment of requirements to

Project –I (CS- 362)

III/IV B.TECH CSE (VI Semester)

Submitted by

## Batch10

## G.Sai Pranathi(Y19CS043)

**B.Rajesh Naik(Y19CS008)**

**J.Deepthi(Y19CS058)**



**April 2022**

## R.V.R. & J.C. COLLEGE OF ENGINEERING

(Autonomous)

Approved by AICTE :: Affiliated to Acharya Nagarjuna University Chowdavaram, Guntur - 522019, Andhra Pradesh, India.

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that project work titled **“ONLINE THESIS ARCHIVING SYSTEM“**is the work done and submitted by **G.Sai Pranathi (Y19CS043) ,B.Rajesh Naik(Y19CS008), J.Deepthi (Y19CS058),** in partial fulfilment of the requirements to the **Project-I (CS-362),** during the academic year 2021-2022.

## Smt.M.Vasavi Smt. S J R K Padminivalli V Dr.M.Sreelatha

Project Guide Project In-charge Prof. and HOD, CSE

**ACKNOWLEDGEMENT**

The successful completion of any task would be incomplete without a proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Project “**ONLINE THESIS ARCHIVING SYSTEM**”.

We are very glad to express our special thanks to **Smt. M. Vasavi**, our Project Guide, who has inspired us to select this topic, and also for her valuable advices in preparing this Project work.

We express our sincere thanks to **Dr.M.Sreelatha**, Head of the Department of Computer science and Engineering for her encouragement and support to carry out this project successfully.

We are very thankful to **Smt** **S J R K Padminivalli V**, Project In charge for the Project who extended her encouragement and support to carry out this Project successfully.

We are very much thankful to **Dr. K. Ravindra**, Principal of **R.V.R. & J.C.**

**College of Engineering**, Guntur for providing the supportive Environment.

Finally we submit our reserves thanks to lab staff in Department of Computer Science and Engineering and to all our friends for their cooperation during this project work preparation.

**G.Sai Pranathi(Y19CS043)**

**B.Rajesh Naik(Y19CS008)**

**J.Deepthi(Y19CS058)**

**ABSTRACT**

The Online Thesis Archiving System is an intranet based application that can be accessed throughout the organization or a specified group. This will be useful for storing all the thesis or projects in the website .Admin register alumni students of the organization. Admin will manage the accounts of himself. Admin will verify the identity of the alumni student and can approve or disapprove the students. The Admin can publish or unpublish the thesis or projects submitted by alumni students.

An alumni student should be able to login to the system through the home page, change the password, and submit the thesis, and will logout of the system. As soon as the Alumni student submit the thesis or their projects, the Admin will update the status of the project either publish to unpublish or unpublish to publish. The lower year students can be able to view the projects only after the admin publish the projects. The Lower year students can take references to the projects. Our Community partner is project incharge of CSE department Sri. P. Rama Krishna.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **DESCRIPTION** | **PAGE NO** |
|  | TITLE | i |
|  | CERTIFICATE | ii |
|  | ACKNOWLEDGEMENT | iii |
|  | ABSTRACT | iv |
|  | LIST OF FIGURES | vii |
|  | LIST OF TABLES | viii |
|  | LIST OF ABBREVATIONS | ix |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 Background | 1 |
|  | 1.2 Problem Statement | 1 |
|  | 1.3 Significance of Work | 2 |
|  | 1.4 Objectives | 3 |
| **2** | **SYSTEM ANALYSIS** | **4** |
|  | 2.1 Requirement Specification | 4 |
|  | 2.2 Use-Case View | 8 |
|  | 2.3 Logical View | 30 |
| **3** | **SYSTEM DESIGN** | **46** |
|  | 3.1 Implementation Diagrams | 46 |
|  | 3.2 Architecture of the System | 48 |
|  | 3.3 Work Flow | 49 |
|  | 3.4 Database Design | 51 |

|  |  |  |
| --- | --- | --- |
| **4** | **IMPLEMENTATION** | **52** |
|  | 4.1 Technologies Used | 52 |
|  | 4.2 Implementation of Technical Service Layer | 53 |
| **5** | **TESTING** | **55** |
|  | 5.1 Introduction of Testing | 55 |
|  | 5.2 Test Cases | 57 |
| **6** | **RESULTS** | **58** |
| **7** | **CONCLUSION** | **63** |
|  | **REFERENCES** | **64** |

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **S.NO** | **DESCRIPTION** | **PAGE NO** |
| 2.1 | Basic Use Case Diagram | 18 |
| 2.2 | Use Case diagram for Thesis Archiving System | 19 |
| 2.3 | Use Case diagram with stereotypes | 19 |
| 2.4 | Login Prototype | 25 |
| 2.5 | Registration Prototype | 25 |
| 2.6 | Activity diagram for Thesis Archiving System | 29 |
| 2.7 | Activity diagram for Submitting Thesis | 30 |
| 2.8 | Sequence diagram for Submitting Thesis | 35 |
| 2.9 | Collaboration diagram for Thesis Archiving System | 36 |
| 2.10 | Class Diagram for Thesis Archiving System | 42 |
| 2.11 | State Chart Diagram for Managing Thesis | 45 |
| 3.1 | Component Diagram for Manage Thesis | 46 |
| 3.2 | Deployment Diagram for Thesis Archiving System | 47 |
| 3.3 | 3-Tier Architecture of System | 48 |
| 6.1 | User Interface of Admin Login | 58 |
| 6.2 | Admin Dashboard | 58 |
| 6.3 | Manage Departments | 59 |
| 6.4 | Manage Curriculums | 59 |
| 6.5 | Registration Of Alumni Students | 60 |
| 6.6 | Interfaces of Students’ Login | 60 |
| 6.7  6.8  6.9 | Interfaces of Admin Submitting Thesis  Lower Year Student Interface  Archive List | 61  61  62 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **DESCRIPTION** | **PAGE NO** |
| 2.1. | Requirement Elicitation | 4 |
| 2.2. | Description of Actors | 9 |
| 2.3. | Description of use Cases | 12 |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **FULL FORM** |
| 1. | HTML | Hyper Text Markup Language |
| 2. | CSS | Cascading Style Sheet |
| 3. | PHP | Hypertext Preprocessor |
| 4. | SQL | Structured Query Language |

# 1. INTRODUCTION

**1.1 Background**

An **Online Thesis Archiving System** ,is a platform that enables a organization to easily store and maintain the projects submitted by the Alumni students.

In modern practice there are some websites for exploring the projects but with cost of money. Tied together with the different structures of organizations and the requirements to request and to download the projects can often allocate too much time and resources to manage it.

Today’s thesis archiving system is more often found in the form of a web based application due to its easy accessibility and ability to run on almost any device and operating system that has a web browser. Thesis archiving systems make it easy for an user to view and list the projects from their own desk or even from home if sick.

The alumni student on submitting the thesis the admin will update the status to publish or unpublish the projects they must be entered into the thesis archiving system. This is particularly beneficial to an organization or colleges.

Having a detailed records of all the projects will be helpful to the lower year students in the organization. It helps in storage management because to store all the projects we need a separate database and it costs a lot and we also need manual power to operate it.

A Thesis Archiving System automates the entire process resolved around within an organization, saving time and resources. It focus on important tasks before them and eliminating the traditional need to record and file all the projects.

**1.2 Problem Statement**

How to develop a web application to store and manage the projects submitted by Alumni students. How to manage projects submitted by students ? How to maintain project records of all students ? How to publish and unpublish the thesis?

**1.3 Significance of the Work**

In the existing Thesis Archiving System, every organization follows manual procedure to store all the thesis. At the end of each month, collecting and storing all the thesis which is a time taking process and there are chances of losing data or errors in the records.

The students needs to submit their projects manually to their respective authorities. This increases the paperwork & maintaining the records becomes tedious. In an organization if any lower year student wants to view all the previous projects for taking as reference they should approach the respective project incharge for viewing them.

For a single teacher it is difficult to maintain all thesis of all students in a class and sometimes leads to loss of data or errors.

The purpose of Thesis Archiving System for organization is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Thesis Archiving System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

Thesis Archiving System is a web application for an organization which is used for storing thesis by students and teachers and maintain those all project records easily

## 1.4 Objectives

The main objective of the Project on Thesis Archiving System for organization is to manage all the projects submitted by the alumni students.

* 1. To design a system that can solve the most of student's time constraint and distance by help them to find references using online method.
  2. The project is built in such a way that an admin can add alumni students. An admin will verify the identity of the alumni student.
  3. An alumni student can submit their thesis or projects.
  4. The purpose of the project is to build an application program to reduce the manual work of maintaining the records about projects.
  5. To develop a system that can maintain the safety of the thesis and avoid physical damage.

# 2. SYSTEM ANALYSIS

## 2.1 Requirement Specification

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed.

In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

## Requirements Elicitation

**Table 2.1: Requirement Elicitation**

|  |  |  |  |
| --- | --- | --- | --- |
| **RID** | **REQUIREMENTS** | **REQUIREMENT**  **NATURE** | **Moscow Rule** |
| R1 | Intuitive graphical user interface to be offered | Functional | Should Have |
| R2 | Thesis Archiving should run in real time | Functional | Must Have |
| R3 | Different roles of admin should be allowed | Functional | Must Have |
| R4 | Information of different students should be maintained by admin | Functional | Must Have |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R6 | Every Alumni student must have a unique username and a password. | Functional | | Must Have |
| R7 | Login mechanism is imperative | Functional | | Must Have |
| R8 | Admin shall be able to change password | Functional | | Could Have |
| R9 | Student shall be able to change password | Functional | | Could Have |
| R10 | Admin can be able to view the dashboard | Functional | | Must Have |
| R11 | Admin can be able to change the status of the project | Functional | | Must Have |
| R12 | Admin will add new curriculums | Functional | | Must Have |
| R13 | Admin will add new departments | Functional | | Must Have |
| R14 | Admin will manage the thesis | Functional | | Must Have |
| R15 | Alumni student will submit their thesis | Functional | | Must Have |
| R16 | Lower year students can be able to view departments | Functional | | Could Have |
| R17 | Lower year students can be able to view curriculums | Functional | | Could Have |
| R18 | Lower year students should be able to view the projects | Functional | | Must Have |
| R19 | Admin has to maintain all the projects submitted | Functional | | Must Have |
| R20 | Database interface should be provided | Non-Functional | Want to Have | | |
| R21 | Giving help on different features of the system | Functional | Must Have | | |

## User Requirements

1. User Friendly and easy to understand
2. Fast in Execution(Less execution Time)
3. Maintenance of Records

## System Requirements

### Hardware requirements

Minimum : Pentium IV Processor with 1.2GHZ

Hard Disk : 500GB

Ram : 2GB

Display : LCD Monitor

### Platform Specification

Operating system : Any Operating System after WINDOWS 2000 Front End : HTML, CSS

Database : PHP, MYSQL Database

Hard disk capability can be more than 500 to maintain the records efficiently.

So there is not any constraint on it. PHP is a server side scripting language. It is used to manage dynamic content, databases, session tracking, even build entire e commerce sites.

7

## 2.1.4 Non-Functional Requirements

### Execution qualities:

Efficiency: The state or quality of being efficient, i.e., able to accomplish something

with the least waste of time and effort; competency in performance.

### Evolution qualities:

Testability: The means by which the presence, quality, or genuineness of

anything is determined.

Extensibility: To enlarge the scope of, or make more comprehensive, as

operations, influence etc.

Scalability: The ability of something, especially a computer system, to adapt to

increased demands.

## 2.2 Use Case View

**2.2.1 Identification of Actors**

Actors represent system users. They are NOT part of the system. They represent anyone or anything that interacts with (input to or receive output from) the system.

An actor is someone or something that:

* + Interacts with or uses the system.
  + Provides input to and receives information from the system.

The needs of the actor are used to develop use cases. This insures that the system will be what the user expected.

### Graphical Depiction:

An actor is a stereotype of a class and is depicted as a “stickman” on a use-case diagram.

For example:

USER

### Naming:

The name of the actor is displayed below the icon.

### Questions to help to identity actors

1. Who is interested in a certain requirements?
2. Where is the system used within the organization?
3. Who will benefit from the use of the system?
4. Who will supply the system with information, use information, and remove it.

From above mentioned information the actors mainly involved in the Thesis Archiving

System for an organization are

* 1. Admin
  2. Alumni Student
  3. Lower Year Student

### Brief Description of Actors

### Table 2.2 Description Of Actors

1. **Lower year Student:**

|  |  |
| --- | --- |
| **Title** | **Description** |
| Lower year student | They can view the departments and curriculums associated with the organization.The Lower Year students can view and list the thesis or projects submitted by the alumni students. |

1. **Alumni Student**

|  |  |
| --- | --- |
| **Title** | **Desciption** |
| Alumni Student | They can login and submit the projects. |

**C. Admin:**

|  |  |
| --- | --- |
| **Actor** | **Brief Description** |
| Admin | He looks over the Thesis Archiving system, verifies the thesis submitted by alumni students. He will add new departments and curriculums. He maintains the database for storing the projects and as well as the details of the alumni students. |

## 2.2.2 Identification of Use-Cases and Sub Use-Case

Use-cases diagrams graphically represent system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may contain all or some of the use cases of a system.

### A use-case diagram can contain:

1. Actors (“things” outside the system)
2. Use cases (system boundaries identifying what the system should do)
3. Interactions or relationship between actors and use cases in the system Including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

In its simple form, a use case can be described as a specific way of using the system from a user’s (actor’s) perspective. A more detailed description might characterize a usecase as:

* + - 1. A pattern of behaviour the system exhibits.
      2. A sequence of related transactions performed by an actor and the system.
      3. Delivering something ofCapture system requirements
      4. Communicate with the end-users and domain experts
      5. Test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

The UML notation for use case is: **ellipse**

Use-case

The basic shape of a use case is an ellipse.

### Naming:

* A usecase may have a name, although it is typically not a simple name. It is often written as an informal text description of the actors and the sequences of events between objects. Usecase names often start with a verb. For example, names of possible use cases for an ATM machine might include Dispense Cash or Transfer Funds.
* The name of the use case is displayed below the icon. The set of questions used to identify the use-cases are:
  1. What are the tasks of each actor?
  2. Will any actor create, store, change, remove or read information in the system?
  3. What use cases will create, store, change, remove, or read this information?

### Purpose of use cases:

* Well-structured use cases denote essential system or subsystem behaviors only, and are neither overly general nor too specific.
* A use case describes a set of sequences, in which each sequence represents the interaction of the things outside the system (its actors) with the system itself (and its key abstraction).
* Use cases specify desired behavior: they do not dedicate how the behavior will be carried out. It helps you to communicate with your developers (who build system that satisfies your requirements) without getting hung up o details.

Use-cases identified for our system are:

* + 1. Login
    2. Change password
    3. Mange the Departments
    4. Manage the Curriculums
    5. Verify the students
    6. Publish/Unpublish the projects
    7. Submit the thesis
    8. Logout

### Table 2.3 : Description of Use cases

1. Use-case name: **Login**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Login | The admin must login into to the thesis archiving system only when they can manage the thesis or projects. |

UML Notation:

Login

**2.**Use-case name: **Change Password**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Change Password | The Alumni student can change their password by verifying the old password |

UML Notation:

Change Password

**3**.Use-case name: **Manage the Departments**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Manage the Departments | The admin can add,delete and update the departments. |

UML Notation:

Mange Departments

**4**.Use-case name: **Manage the Curriculums**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Manage the Curriculums | The admin can add,delete and update the curriculums. |

UML Notation:

Manage Curriculums

**5**.Use-case name: **Verify Students**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Verify Students | The admin will verify the registered alumni students. |

UML Notation:

Verify Students

**6**.Use-case name: **Publish Projects**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Publish Projects | The Admin will publish or unpublish the projects. |

UML Notation:

Publish/Unpublish Projects

**7.**Use-case name: **Submit the Thesis**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Submit the Thesis | Alumni Student will submit the thesis which are done by them. |

UML Notation:

Submit the Thesis

**8.**.Use-case name: **Logout**

|  |  |
| --- | --- |
| Use case name | Brief Description |
| Logout | The admin / student can logout of the thesis archiving system performing their required functionality. |

UML Notation:

Logout

## 2.2.3 Building Requirements Model through Use Case Diagram

**Definition:** Use-case diagrams graphically represent system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may contain all or some of the use cases system.

A use-case diagram can contain:

1. Actors (“things” outside the system)
2. Use cases (system boundaries identifying what the system should do)
3. Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

### Relations:

**Association Relationship:**

An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association .

### Bi-directional association:

If you prefer, you can also customize the toolbox to include the bi- directional tool to the use-case toolbox.

### Graphical Depiction:

An association relationship is an orthogonal or straight solid line with an arrow at one end:

Actor Use-Case



In An Association Relationship, we can provide Stereotype Communicate also as shown below:

<<communicate>>

USER login

### Dependency Relationship:

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that the operations of the client invoke operations of the supplier.

We can provide here

* + 1. Include Relationship.
    2. Extend Relationship.
* Multiple use cases may share pieces of the same functionality. This functionality is placed in a separate use case rather than documenting it in every use case that needs it.
* Include relationships are created between the new use case and any other use case that "uses" its functionality.

An include relationship is a stereotyped relationship that connects a base use case to an inclusion use case. An include relationship specifies how behaviour in the inclusion use case is used by the base use case.

<<include>>

BASE USE-CASE INCLUSION USE-CASE

### Extend Relationship:

An extend relationship is a stereotyped relationship that specifies how the functionality of one use case can be inserted into the functionality of another use case. Extend relationships between use cases are modelled as dependencies by using the Extend stereotype.

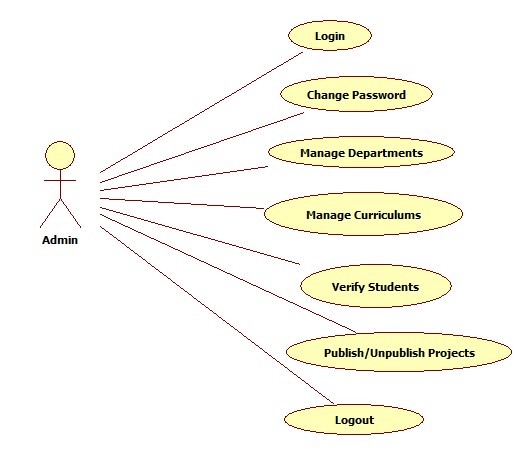
An *extend* relationship is used to show

* 1. Optional behaviour.
  2. Behavior that is run only under certain conditions such as triggering an alarm
  3. Several different flows that may be run based on actor selection.

Finally we can conclude,

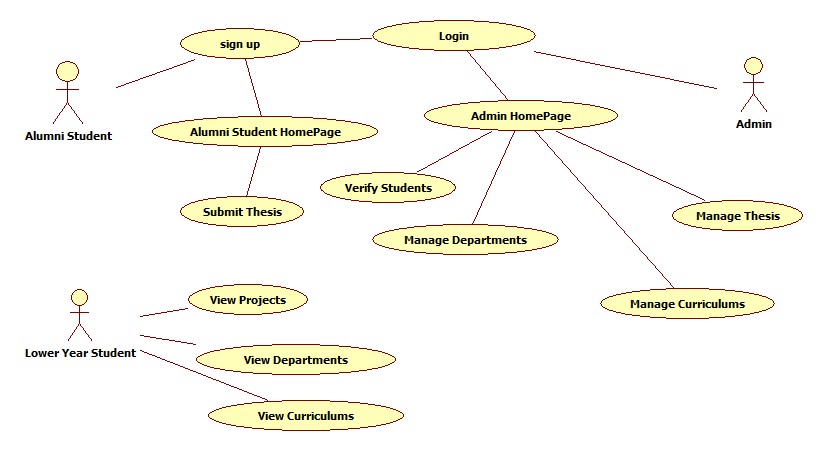
* + - **«extend»** is used when you wish to show that a use case provides additional functionality that may be required in another use case.
    - **«include»** applies when there is a sequence of behavior that is used to model frequently in a number of use cases, and you want to avoid copying the same description of it into each use case in which it is used.

**Basic Use Case Diagram for Thesis Archiving System**



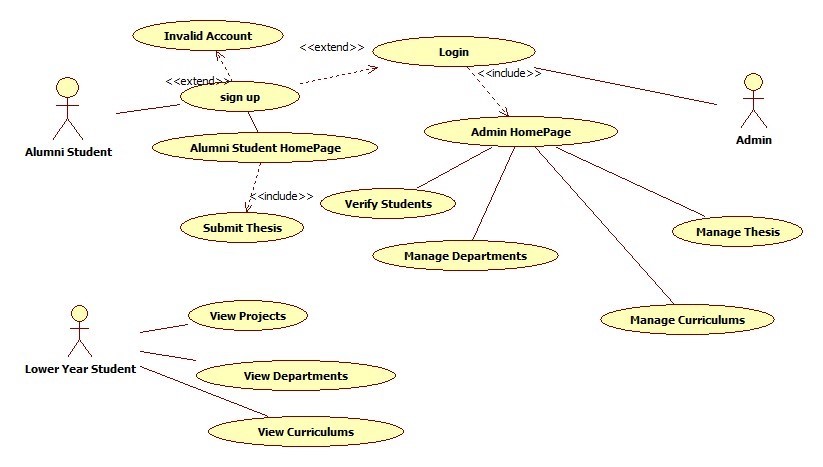
### Figure 2.1: Basic Use Case Diagram

**Use Case Diagram of Thesis Archiving System**

****

### Figure 2.2: Use Case Diagram of Thesis Archiving System

**Use Case Diagram with Stereotypes**

** Figure 2.3: Use Case Diagram with Stereotypes**

## 2.2.4 Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information. Flow of events document is typically created in the elaboration phase.

Each use case is documented with flow of events

1. A description of evens needed to accomplish required behavior
2. Written in terms of what the system should do, NOT how it should do it
3. Written in the domain language, not in terms of the implementation

A flow of events should include:

* 1. When and how the use case stars and ends
  2. What interaction the use case has with the actors
  3. What data is needed by the use case
  4. The normal sequence of events for the use case
  5. The description of any alternate or exceptional flows

The flow of events for a use case is contained in a document called the Use Case Specification. Each project should use a standard template for the creation of the Use Case Specification. Including the following

1. Use Case Name, brief description
2. Flow of Events:
   1. Basic flow
   2. Alternate flow
   3. Pre-Conditions
   4. Extension Points

### Use case specification for Login:

* + - 1. **Use case Number:** 1
      2. **Use case Name:** login
      3. **Brief description:** The admin must login into the Thesis Archiving System in order to manage Thesis.
      4. **Precondition:** Registration of Admin.
      5. **Post condition:** Admin will be able to manage the Thesis.

### Flow of events:

* 1. **Basic flow:**
* The admin will go to admin tab. Then the admin will fill the details like username and password.
* If the username or password is incorrect then the system will execute the alternative flow.

### Alternative flow:

Informs the admin to enter correct username or password.

**7.Extension Point:** nothing

### Use case specification for Login:

1. **Use case Number:** 2
2. **Use case Name:** login
3. **Brief description:** The admin must login into the Thesis Archiving System in order to manage Thesis.
4. **Precondition:** Registration of Admin.
5. **Post condition:** Admin will be able to manage the Thesis.

### Flow of events:

* 1. **Basic flow:**
* The admin will go to admin tab. Then the admin will fill the details like username and password.
* If the username or password is incorrect then the system will execute the alternative flow.

### Alternative flow:

Informs the admin to enter correct username or password.

**7.Extension Point:** nothing

### Use case specification for Change Password:

1. **Use case Number:** 3
2. **Use case Name:** change password
3. **Brief description:** Admin can change their password by entering old password.
4. **Precondition:** Registration of admin.
5. **Post condition:** Admin can login with the new password.

### Flow of events:

* 1. **Basic flow:**
     + The admin will go to change password tab. The admin will fill the details like old password, new password and re- enter of new password.
     + If the old password is incorrect or both the new passwords do not match then the system will execute the alternative flow.

### Alternative flow:

Informs admin to enter the correct old password or to enter the both new passwords as same.

**7.Extension Point:** Nothing

### Use case specification for Verification Of Alumni Students:

1. **Use case Number:** 4
2. **Use case Name:** Verify Alumni Students
3. **Brief description:** The admin can verify the Alumni Students
4. **Precondition:** Registration of Alumni Students
5. **Post condition:** The Alumni Student is verified and status is updated by the admin.

### 6.Flow of events:

* 1. **Basic flow:** The admin will login and go to Students tab. If the student is not verified then it will execute the alternative flow.
  2. **Alternative flow:** Invalid information by the Student.

**7.Extension Point:** nothing.

### Use case specification for Submitting the Thesis:

1. **Use case Number:** 5
2. **Use case Name:** Submitting the Thesis
3. **Brief description:** The Alumni Students should be able to submit the Thesis.
4. **Precondition:** The Alumni Student must be registered.
5. **Post condition:** Project is submitted.

### 6.Flow of events:

### 6.1 Basic flow: The Alumni Student will login and go to Submit Thesis status tab. If the Login is correct then Thesis tab is displayed. Otherwise the system will execute the alternative flow.

* 1. **Alternative flow:** Invalid username or password **:** Indicates the Student to enter correct username or password.

**7.Extension Point:** Alumni Student can delete the leave the thesis before the approval.

### Use case specification for Submitting the Thesis:

1. **Use case Number:** 6
2. **Use case Name:** Submitting the Thesis
3. **Brief description:** The Alumni Students should be able to submit the Thesis.
4. **Precondition:** The Alumni Student must be registered.
5. **Post condition:** Project is submitted.

### Flow of events:

* 1. **Basic flow:** The Alumni Student will login and go to Submit Thesis status tab. If the Login is correct then Thesis tab is displayed. Otherwise the system will execute the alternative flow.
  2. **Alternative flow:** Invalid username or password **:** Indicates the Student to enter correct username or password.

**7. Extension Point:** Alumni Student can delete the thesis before the approval.

**Use case specification for Manage Thesis:**

1. **Use case Number:** 7
2. **Use case Name:** Manage Thesis
3. **Brief description:** The admin can manage the Thesis (if student submitted) and admin can publish or unpublish the projects.
4. **Precondition:** Thesis must be submitted by the student..
5. **Post condition:** Thesis status will be updated.

### Flow of events:

**6.1 Basic flow:** The admin has to login and go to manage archives tab*.* If the login is unsuccessful then the alternative flow is executed otherwise Thesis will be opened. Then the admin will manage thesis to publish or unpublish.

* 1. **Alternative flow:** Invalid username or password:

Indicates admin to enter username or password.

**7. Extension Point:** nothing.

### Use Case specification for Logout:

1. **Use case Number:** 8
2. **Use case Name:** logout
3. **Brief description:** Users (Admin/Alumni Student) can logout of the Thesis Archiving System after performing their required functionality.
4. **Precondition:** The user should login.
5. **Post condition:** The user will move out from the Thesis Archiving system.

### 6. Flow of events:

**6.1 Basic flow:** The user should login. If the login is successful then go to logout tab . Otherwise the system will executes the alternative flow.

**s6.2 Alternative flow:** Invalid username or password.Indicates the user to enter correct

password

**7.Extension Point:** nothing

## 2.2.5 Sample Prototypes for Application

In software development, a prototype is a rudimentary working model of a product or information system, usually built for demonstration purposes or as part of the development process. In the systems development life cycle (SDLC) Prototyping Model, a basic version of the system is built.

### Login Prototype:

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | User Name |  |  | | --- | | Password |  |  | | --- | | Submit |   **Online Thesis Archiving System** |

**Figure 2.4: Login Prototype**

## Registration Form:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Online thesis Archiving system  |  | | --- | | Full Name |  |  | | --- | | Email-id |  |  | | --- | | Phone Number |  |  | | --- | | User Name |  |  | | --- | | Address |  |  | | --- | | Submit | |

## 

## Figure 2.5: Registration Prototype

## 

## 2.2.6. Activity Diagram

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations.

* 1. Activity diagrams also may be created at this stage in the life cycle. These diagrams represent the dynamics of the system.
  2. They are flow charts that are used to show the workflow of a system; that is, they show the
  3. Flow control.
  4. At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
  5. Later in the life cycle, activity diagrams may be created to show the workflow for it.

### Activities:

Activity diagrams contain activities, transitions between the activities, decision points, and synchronization bars. An activity represents the performance of some behavior in the workflow. In the UML, activities are represented as rectangles with rounded edges, transitions are drawn as directed arrows, decision points are shown as diamonds, and synchronization bars are drawn as thick horizontal or vertical bars as shown in the following.



### Transitions:

Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity. Transition connects activities with other model elements and object flows connect activities with objects.

They are typically triggered by the completion of the behavior in the originating activity.

### Decisions:

When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point. The transitions from a decision point contain a guard condition, which is used to determine which path from the decision point is taken. Decisions along with their guard conditions allow you to show alternate paths through a work flow.

**Decision Point**

**End State:**

An end state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want t explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.



### End state

**Start State:**

A start state (also called an “initial state”) explicitly show the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram.



### Start state

**Swim lanes:**

Swim lanes are helpful when modelling a business workflow because they can represent organizational units or roles within a business model. Swim lanes are very similar to an object because they provide a way to tell who is performing a certain role. Swim lanes only appear on activity diagrams.

### Synchronization Bars:

In a workflow there are typically some activities that may be done in parallel. A synchronization bar allows *you* to specify what activities may be done concurrently.

Synchronization bars are also used to show joins in the workflow; that is, what activities must complete before processing may continue.

Means, a synchronization bar may have many incoming transitions and one outgoing transition, or one incoming transition and many outgoing transitions.

Vertical Synchronization

Horizontal synchronization

### The activity diagram for overall system is:

* At first alumni student with his credentials to submit the thesis.
* To submit the thesis he should fill out the fields like Project title.
* After submitting the thesis he logs out.
* Then admin will login to the system and will publish or unpublish the thesis.

### Modelling a workflow in an activity diagram can be done several ways; however, the following steps present just one logical process:

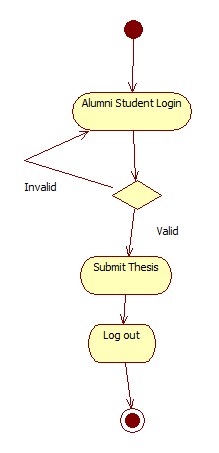
* Decide the pre and post-conditions of the workflow through a start state and an end state. In most cases, activity diagrams have a flowchart structure so start and end states are used to designate the beginning and ending of the workflow. State and end states clarify the perimeter of the workflow.
* Define and recognize all activities and states that must take place to meet your objective. Place and name them on the activity diagram in a logical order.
* Define and diagram any objects that are created or modified within your activity diagram. Connect the objects and activities with object flows.
* Decide who or what is responsible for performing the activities and states through swim lanes. Name each swim lane and place the appropriate activities and states within each swim lane.
* Connect all elements on the diagram with transitions. Begin with the "main" workflow.
* Place decisions on the diagram where the workflow may split into an alternate flow. For example, based on a Boolean expression, the workflow could branch to a different workflow.

### Activity Diagram for Thesis Archiving System:

### 

**Figure 2.6: Activity Diagram**

### Activity Diagram for submitting Thesis

****

**Figure 2.7: Activity Diagram for Submitting Thesis**

## 2.3 Logical View

**2.3.1 Identification of Analysis Classes**

1. For identification to analysis classes we have three approaches, they area:
2. The noun phrase approach
3. The common class patterns approach The use-case driven approach
4. In our application we used Use case driven approach for identifying analysis classes

### Noun Phrase Approach:

In this method, analysts read through the requirements or use cases looking for noun phrases. Nouns in the textual description are considered to be classes and verbs to be methods of the classes all plurals are changed to singular, the nouns are listed, and the list divided into three categories relevant classes , fuzzy classes (the “fuzzy area,” classes we are not sure about), and irrelevant classes.

### Identifying Tentative classes :

* 1. The following are guidelines for selecting classes in an application:
  2. Look for nouns and noun phrases in the use cases.
  3. Some classes are implicit or taken from general knowledge.
  4. Carefully choose and define class names.

### Selecting classes from the Relevant and Fuzzy Categories:

The following guide lines help in selecting candidate classes from the relevant and fuzzy categories of classes in the problem domain.

* + 1. Redundant Classes
    2. Adjectives classes
    3. Attribute classes
    4. Irrelevant classes

### Common Class Pattern Approach:

The second method for identifying classes is using common class patterns, which is based on a knowledge base of the common classes. The following patterns are used for finding the candidate class and object.

* Concept class
* Event class
* Organization class
* People class (also known as person, roles, and roles played class)
* Places class

### Use-Case Driven Approach:

One of the first steps in creating a class diagram is to derive from a use case, via a collaboration, those classes that participate in realizing the use case. Through further analysis, a class diagram are then usually assembled into a larger analysis class be drawn at any scale that is appropriate, from a single use-case instance to a large, complex system.

## 2.3.2 Identification of Responsibility of Classes

### Class Responsibility Collaboration Cards (CRC Cards)

At the starting, for the identification of classes we need to concentrate completely on uses cases. A further examination of the use cases also helps in identifying operations and the messages that classes need to exchange. However, it is easy to think first in terms of the overall responsibilities of a class rather than its individual operations.

A responsibility is a high level description of something a class can do. It reflects the knowledge or information that is available to that class, either stored within its own attributes or requested via collaboration with other classes, and also the services that it can offer to other objects.

Class Responsibility Collaboration (CRC) cards provide an effective technique for exploring the possible ways of allocating responsibilities to classes and the collaborations that are necessary to fulfill the responsibilities.

CRC cards can be used at several different stages of a project for different purposes.

1. They can be used early in a project to help the production of an initial class diagram.
2. To develop a shared understanding of user requirements among the members of the team.
3. CRCs are helpful in modeling object interaction.

The format of a typical CRC card is shown below:

|  |  |
| --- | --- |
| Class Name: | |
| Responsibilities | Collaborations |
| Responsibilities of a class are listed in this section | Collaborations with other classes are listed here, together with a brief description of the purpose  of the collaboration |

CRC cards are an aid to a group role-playing activity. Index cards are used in preference to pieces of paper due to their robustness and to the limitations that their size imposes on the number of responsibilities and collaborations that can be effectively allocated to each class. A class name is entered at the top of each card and responsibilities and collaborations are listed underneath they become apparent.

1. Conduct a session to identify which objects are involved in the use case.
2. Allocate each object to a team member who will play the role of that object.
3. Act out the use case: This involves a series of negotiations among the objects to explore how responsibility can be allocated and to identify how the objects can collaborate with each other.
4. Identify and record any missing or redundant objects.

## Use Case Realizations

A use case realization is a graphic sequence of events, also referred as an instance of a use case. These realizations are represented using either a sequence or collaboration diagrams.

Use case Realization Diagrams:



Change password change password



Submit Thesis submit thesis



Manage Departments manage departments



Manage Curriculums manage curriculums

## 2.3.4 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time based sequence-what happens first, what happens next….

Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object’s lifeline. The lifeline represents the object’s existence during the interaction.

### Steps:

1. An object is shown as a box at the top of a dashed vertical line. Object names can be specific (e.g., Algebra 101, Section 1) or they can be general (e.g., a course offering). Often, an anonymous object (class name may be used to represent any object in the class.)
2. Each message is represented by an Arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

The sequence diagram is very simple and has immediate visual appeal—this is its great strength. A sequence diagram is an alternative way to understand the overall flow of control of a program. Instead of looking at the code and trying to find out the overall sequence of behaviour.

The following tools located on the sequence diagram toolbox which enable to model sequence diagrams:

**Object:** An object has state, behavior, and identity. The structure and behavior of similar 1objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

**Message Icons:** A message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrow connecting two lifelines together.

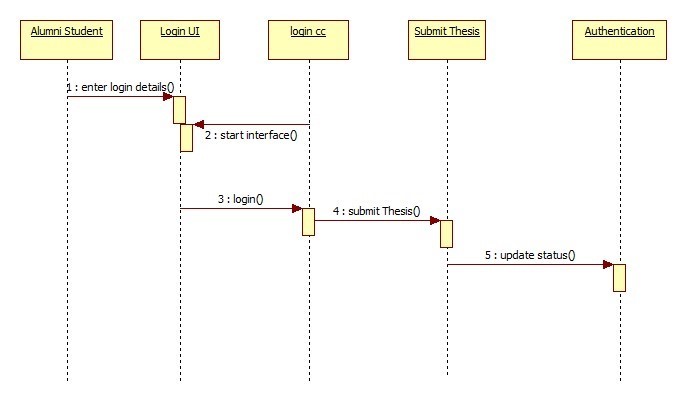
**Focus of Control:** Focus of Control (FOC) is an advanced notational technique that enhances sequence diagrams. It shows the period of time during which an object is performing an action either directly or through an underlying procedure.

**Message to self:** A message to self is a tool that sends a message from one object back to the same object. It does not involve other objects because the message returns to the same object. The sender of a message is the same as the receiver.

**Note:** A note captures the assumptions and decisions applied during analysis and design. Notes may contain any information, including plan text, fragments of code, or references to other documents.

**Note Anchor:** A note anchor connects a note to the element that it affects

**Sequence Diagram for Submitting Thesis**



### Figure 2.8: Sequence Diagram for Submitting Thesis

## Collaboration Diagrams

Collaborations can also be represented in various ways that reveal their internal details.Collaboration diagrams and sequence diagrams are called interaction diagrams. A collaboration diagram shows that the order of messages that implement an operation or a transaction. Collaboration diagrams show objects, their links, and their messages. A collaboration diagram contains

* + - * 1. Objects drawn as rectangles
        2. Links between objects shown as lines connecting the linked objects
        3. Messages shown as text and an arrow that points from the client to the supplier

### Collaboration Diagram for Thesis Archiving System:

### 

### Figure 2.9: Collaboration Diagram for Thesis Archiving System

**Difference between sequence and collaboration diagrams:**

* Sequence diagrams are closely related to collaboration diagrams and both are alternate representations of an interaction.
* Sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

## Identification of Attributes and Methods of Class

### Attributes:

Attributes are part of the essential description of a class. They belong to the class, unlike objects, which instantiate the class. Attributes are the common structure of what a member of the class can 'know'. Each object will have its own, possibly unique, value for each attribute.

Guidelines for identifying attributes of classes are as follows:

* Attributes usually correspond to nouns followed by prepositional phrases. Attributes also may correspond to adjectives or adverbs.
* Keep the class simple; state only enough attributes to define the object state.

Some questions are there which help in identifying the responsibilities of classes and deciding what data elements to keep track of:

* What information about an object should we keep track of?
* What services must a class provide?

Answering the first question helps us to identify the attributes of a class. Answering the second question helps us to identify class methods.

### The attributes identified in our system are

* Attributes for Login: username, password.
* Attributes for Alumni student: Name, Username, Password, Department, Curriculum.
* Attributes for Admin: Username, password, address, phone number.
* Attributes for submit thesis: Project name, year, project members,abstract, documentation.
* Attributes for Manage Departments: Department name.
* Attributes for Manage Curriculums :Curriculum name ,Department name.
* Attributes for Change Password: Old password, New Password

### The responsibilities identified in our system are:

* + Methods for Login: login ().
  + Methods for Thesis: edit(),delete().
  + Methods for Manage Departments: add(),delete(),update()
  + Methods for Mange Curriculums: add(),delete(),update()
  + Methods for Change Password: change ().
  + Methods for Manage Thesis: Publish(),Unpublish()

### Identification of Relationships among Classes

**Need for Relationships among the class**: All systems are made up of many classes and objects. System behavior is achieved through the collaborations of the objects in the system.

Two types of relationships in CLASS diagram are:

* + 1. Associations Relationship
    2. Aggregations Relationship

### Association Relationships:

An Association is a bidirectional semantic connection between classes. It is not a data flow as defined in structured analysis and design data may flow in either direction across the association. An association between classes means that there is a link between objects in the associated classes.

### 2.Aggregation Relationships:

An Aggregation Relationship is a specialized form of association in which a whole is related to its part(s).

Aggregation is known as a "part-of" or containment relationship. The UML notation for an aggregation relationship is an association with a diamond next to the class denoting the aggregate (whole).

### Naming Relationships:

An association may be named. Usually the name is an active verb or verb phrase that communicates the meaning of the relationship. Since the verb phrase typically implies a reading direction, it is desirable to name the association so it reads correctly from left to right or top to bottom. The words may have to be changed to read the association in the other direction. It is important to note that the name of the association is optional.

### Role Names:

The end of an association where it connects to a class is called an association role. Role names can be used instead of association names.

* It is not necessary to have both a role name and an association name.
* Associations are named or role names are used only when the names are needed for clarity.

### Multiplicity Indicator:

Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship. Multiplicity defines the number of objects that are linked to one another. There are two multiplicity indicators for each association or aggregation one at each end of the line.

Some common multiplicity indicators are

1 Exactly one

0.. \* Zero or more

1.. \* One or more

0.. 1 Zero or one

5 .. 8 Specific range (5, 6, 7, or 8)

4.. 7,9 Combination (4, 5, 6, 7, or 9)

### Reflexive Relationship:

Multiple objects belonging to the same class may have to communicate with one another. This is shown on the class diagram as a reflexive association or aggregation. Role names rather than association names typically are used for reflexive relationships.

## UML Class Diagram

Class diagrams contain icons representing classes, interfaces, and their relationships. You can create one or more class diagrams to represent the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model.

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models.

A Class is a description of a group of objects with common properties (attributes) common behavior (operations), common relationships to other objects, and common semantics.

In the UML, classes are represented as compartmentalized rectangles.

* The top compartment contains the name of the class.
* The middle compartment contains the structure of the class (attributes).
* The bottom compartment contains the behavior of the class (operations).

### Stereotypes and Classes:

As like stereotypes for relationships in use case diagrams. Classes can also have stereotypes. Here a stereotype provides the capability to create a new kind of modeling element. Here, we can create new kinds of classes. Some common stereotypes for a class are entity Class, boundary Class, control class, and exception.

### Notations of these stereotypes:

<<Exception>>

control Class Boundary Class Entity Clas

### Entity Classes

1. An **entity class** models information and associated behavior that is generally long lived.
2. This type of class may reflect a real-world entity or it may be needed to perform tasks internal to the system.
3. They are typically independent of their surroundings; that is, they are not sensitive to how the surroundings communicate with the system.

### Control Classes

1. Control classes model sequencing behavior specific to one or more use cases.
2. Control classes coordinate the events needed to realize the behavior specified in the use case.
3. Control classes typically are application-dependent classes.

### Boundary Classes:

Boundary classes handle the communication between the system surroundings and the inside of the system. They can provide the interface to a user or another system (i.e., the interface to an actor). They constitute the surroundings dependent part of the system.

### Boundary classes are used to model the system interfaces

Boundary classes are also added to facilitate communication with other systems. During design phase, these classes are refined to take into consideration the chosen communication protocols.

### Documenting classes:

As classes are created, they should also be documented. The documentation should state the purpose of the class and not the structure of the class.

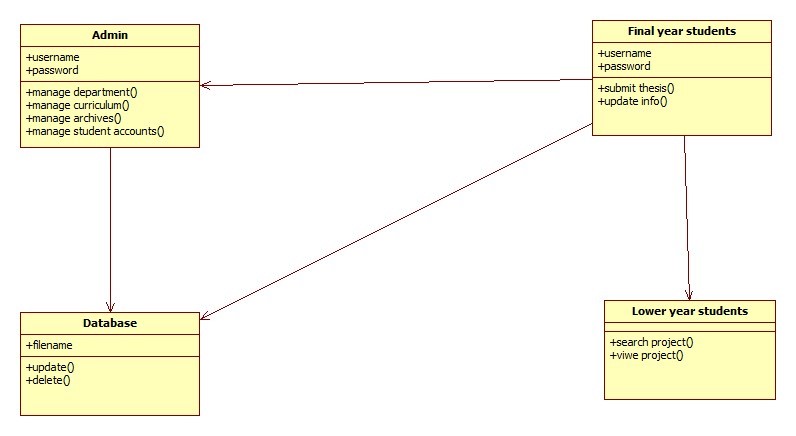
For example, a Student class could be documented as follows: A student is someone currently registered to take classes at the university and this information will be provided for billing purpose.

### Packages:

If a system contained only a few classes, you could manage them easily. Most systems are composed of many classes, and thus you need a mechanism to group them together for ease of use, maintainability, and reusability.

A package in the logical view of the model is a collection of related packages and/or classes. By grouping classes into packages, we can look at the "higher" level view of the model (i.e., the packages).

### Class Diagram for Thesis Archiving System



**Figure 2.10: Class Diagram for Thesis Archiving System**

## 2.3.8 UML State Chart Diagram

A state chart diagram shows the states of a single objects, the events or messages that cause a transition from one state to another and the actions that result from a state change. As I activity diagram, state chart diagram also contains special symbols for start state and stop state.

State chart diagram cannot be created for every class in the system, it is only for those class objects with significant behavior.

### State:

A state represents a condition or situation during the life of an object during which it satisfies some condition, performs some action or waits for some event.

UML notation for STATE is

### Action:

Actions on states can occur at one of four times:

* + on entry
  + on exit
  + do
  + on event.
* **on entry :**What type of action that object has to perform after entering into the state.

syntax: event(args)[condition] : the Action

### State transition:

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

Provide a label for each state transition with the name of at least one event that causes the state transition..

### Transitions are labeled with the following syntax:

event (arguments) [condition] / action ^ target.sendEvent (arguments) Only one event is allowed per transition, and one action per event.

### State Details:

Actions that accompany all state transitions into a state may be placed as an entry action within the state. Likewise that accompany all state transitions out of a state may be placed as exit actions within the state. Behavior that occurs within the state is called an activity.

UML notation for State Details;

StateName

entry/ simple action

entry/ ^class name.eventname do/ simple action

do/ ^class name.event name exit/ ^class name.event name

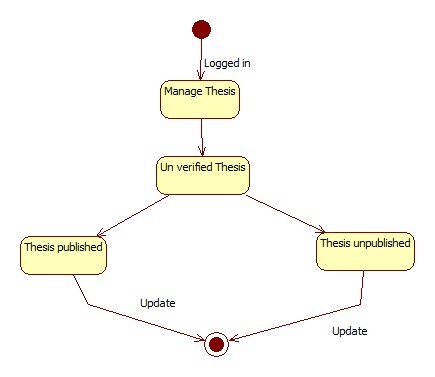
### Purpose of state chart diagrams:

* We use state chart diagram when working on real-time process control applications or systems that involve concurrent processing. It will also be used when showing the behavior of a class over several use cases.
* State chart diagrams are used to model dynamic view of a system.
* State chart diagrams are used to emphasizing the potential states of the objects and the transitions among those states.
* State chart diagrams are used to modeling the lifetime of an object.

### Elements of state chart diagrams:

* + State
  + Event
  + Transition
  + Action state
  + Sequential sub state
  + Concurrent sub state
    - Initial state
    - Final state
    - History state
    - Vertical Synchronization
    - Horizontal Synchronization
    - Guard conditions
    - Forks and joins

**State Chart Diagram for Manage Thesis**

****

### Figure 2.11: State Chart Diagram for Manage Thesis

# 3.SYSTEM DESIGN

## 3.1 Implementation Diagrams

**3.1.1Component Diagrams**

* + Component Diagrams show the dependencies between software components in the system. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time or at runtime.
  + The dependency relationship indicates that one entity in a component diagram uses the services or facilities of another.
  + The dependency relationship may also be used to show calling dependencies among components, using dependency arrows from components to interfaces on other components.

### Types of components:

* Deployment component
* To model source code
* Executable component

### Component Diagram for Manage Thesis

### 

**Figure 3.1: Component Diagram for Manage Thesis**

## 3.1.2 Deployment Diagram

* The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them.
* Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources.
* Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication associations.

### Uses:

* To model embedded system.
* To model client-server system.
* To model distributed systems.

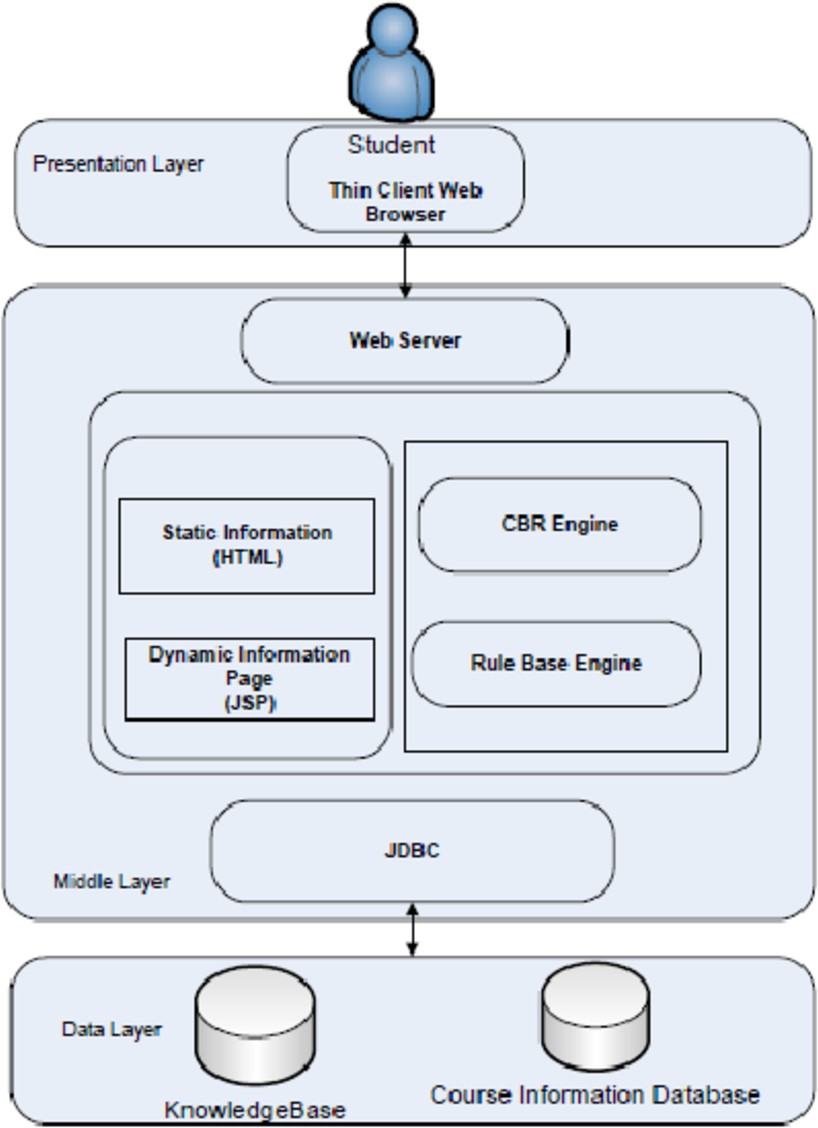
### Deployment Diagram for Thesis Archiving System

### 

**Figure 3.2.: Deployment Diagram for Thesis Archiving System**

## Architecture of the System

Thesis Archiving System for organization is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. The project was developed using 3-tier Architecture.



### Figure 3.3: 3-tier Architecture of System

First tier is **Presentation** tier, in which user interacts with leave management system through any web browser (Google chrome, Internet Explorer etc…).It is interface between user and application.

Second tier is **Business** Tier, in which all system implementation is written. Having dynamic pages written in PHP. This will run whole system in server. It is interface between application and database.

Third tier is **Data Base** tier, in which all the information of users and application is stored. All activities, modifications and whole data is maintained by database. With help of queries user retrieve data from it

## Workflow

A person should be able to login to the system through the login page of the application, change the password, submit the thesis specifying the year, and admin can publish/ unpublish the thesis that are submitted by the alumni students.

The submitted thesis can be viewed by the Lower Year Students.

Project have total 3 modules

* Alumni Student Module
* Lower Year Module
* Admin Module

## 3.3.1Admin Module

First for the initial application page by clicking on Admin, it will redirect to admin login page. For accessing this web application, Admin have to login with his unique username and password. After logged in as Admin he will be redirected to his Admin Module.

Having a detailed records of all the projects will be helpful to the lower year students in the organization. It helps in storage management because to store all the projects we need a separate database and it costs a lot and we also need manual power to operate it.

Admin register alumni students of the organization. Admin will manage the accounts of himself.

will verify the identity of the alumni student and can approve or disapprove the students. The Admin

can publish or unpublish the thesis or projects submitted by alumni students.

Admin module have several functions like:

* **Change Password**: Admin can change his password by giving old password and new password he wants to update.
* **Manage Departments**: In this section of web page admin can add,delete or update the departments.
* **Manage Curriculums**: In this section of web page admin can be able to add,delete or update the curriculums.
* **Manage Archives**: In this section of web page page admin can be able to publish or unpublish the projects submitted by
* **Help**: If a Lower year student doesn’t know how to access this application he can access this page and it gives detailed information of this application and help them to access application.

## 3.3.2Alumni Student Module

Alumni student Module functionalities are:

* **Change Password:** An alumni student can change password by giving old password as a proof of identity and a new password to change.
* **Submit Thesis:** An alumni student will submit the thesis done by them.

## 3.3.3 Lower year student Module

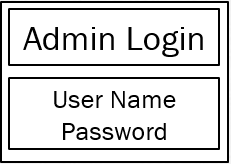
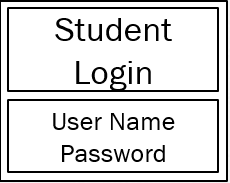
Student Module Functionalities are:

* + - **View Departments** : A student can be able to view and list the departments of the organization.
    - **View Curriculums**: A student can be able to view and list all the curriculums of the organization.
    - **View Thesis:** A student can be able to view and list the projects submitted by their Alumni students.

## 3.4 Data Base Design

To Implement Thesis Archiving system for organizationl a database with 9 tables are created.

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Information Of Alumni student |  |  | | --- | | Id(Primary Key)  Name  Gender  Phone number  Email id  address | |



|  |  |
| --- | --- |
| |  | | --- | | Submit Thesis |   Project Title  Year  Team Members  Abstract  Documentation |

# 4.IMPLEMENTATION

## Technologies Used

* Front End :HTML , CSS
* Back End :PHP
* Data Base:MySQL

**Front End**

### HTML

**CSS**

1. HTML is the standard markup language for creating Web pages
2. HTML describes the structure of a Web page
3. HTML consists of a series of elements
4. HTML elements tell the browser how to display the content
5. HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

### PHP

1. CSS stands for Cascading Style Sheets
2. CSS describes how HTML elements are to be displayed on screen, paper, or in other media
3. CSS saves a lot of work. It can control the layout of multiple web pages all at once
4. External stylesheets are stored in CSS files
5. CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.
6. PHP is an acronym for "PHP: Hypertext Preprocessor"
7. PHP is a widely-used, open source scripting language. PHP scripts are executed on the server
8. It is powerful enough to be at the core of the biggest blogging system on the web. It is deep enough to run the largest social network
9. PHP files can contain text, HTML, CSS, JavaScript, and PHP code

### MySQL

* + MySQL is the most popular Open Source Relational SQL Database Management System.
  + MySQL is one of the best RDBMS being used for developing various web-based software applications.
  + MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

### SQL

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert , delete ,update records of Database.
* SQL can create new databases
* SQL can create new tables in a database

## Implementation of Technical Service Layer

A person should be able to login to the system through the login page of the application, change the password, submit the thesis , and admin can publish/unpublish the projects that are submitted to him / her.

Project have total 3 modules

* + Alumni Student Module
  + Lower Year Student Module
  + Admin Module

The Online Thesis Archiving System is an intranet based application that can be accessed throughout the organization or a specified group. This will be useful for storing all the thesis or projects in the website.

## Admin Module

First for the initial application page by clicking on Admin, it will redirect to admin login page. For accessing this web application, Admin have to login with his unique username and password. After logged in as Admin he will be redirected to his Admin Module.

Admin module have several functions like:

* **Change Password**: Admin can change his password by giving old password and new password he wants to update.
* **Manage Departments**: In this section of web page admin can add,delete or update the departments.
* **Manage Curriculums**: In this section of web page admin can be able to add,delete or update the curriculums.
* **Manage Archives**: In this section of web page page admin can be able to publish or unpublish the projects submitted by
* **Help**: If a Lower year student doesn’t know how to access this application he can access this page and it gives detailed information of this application and help them to access application

**4.2.2 Alumni Student Module**

Alumni student Module functionalities are:

* **Change Password:** An alumni student can change password by giving old password as a proof of identity and a new password to change.
* **Submit Thesis:** An alumni student will submit the thesis done by them.

## 4.2.3 Lower Year Student Module

Student Module Functionalities are:

* + **View Departments** : A student can be able to view and list the departments of the organization.
  + **View Curriculums**: A student can be able to view and list all the curriculums of the organization.
  + **View Thesis:** A student can be able to view and list the projects submitted by their Alumni students

**5.TESTING**

**5.1 Introduction To Testing:**

Testing is a fault detection technique that tries to create failure and erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer.

**5.1.1 Top down approach:** This type of testing starts from upper level modules.

Since the detailed activities usually performed in the lower level routines are

not provided stubs are written.

**5.1.2. Bottom Up Approach:** Testing can be performed starting from smallest and

Lowest level modules and proceeding one at a time. For each module in bottom

Up testing.

**5.1.3 Testing Methodologies**

The following are the Testing Methodologies:

* Unit Testing.
* Integration Testing.
* User Acceptance Testing
* Output Testing

### Unit Testing User Acceptance Testing.

Unit testing focuses verification effort on the smallest unit of Software design that is the

module. Unit testing exercises specific paths in a modules control structure to ensure complete coverage and maximum error detection.

This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification.

### Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted.

The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

### Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

### Bottom-up Integration

The bottom up integration strategy may be implemented with the following steps:

1. The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
2. A driver (i.e.) the control program for testing is written to coordinate test case input and output.
3. The cluster is tested. d. Drivers are removed and clusters are combined moving upward in the program structure. The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

### User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes .

### Output Testing

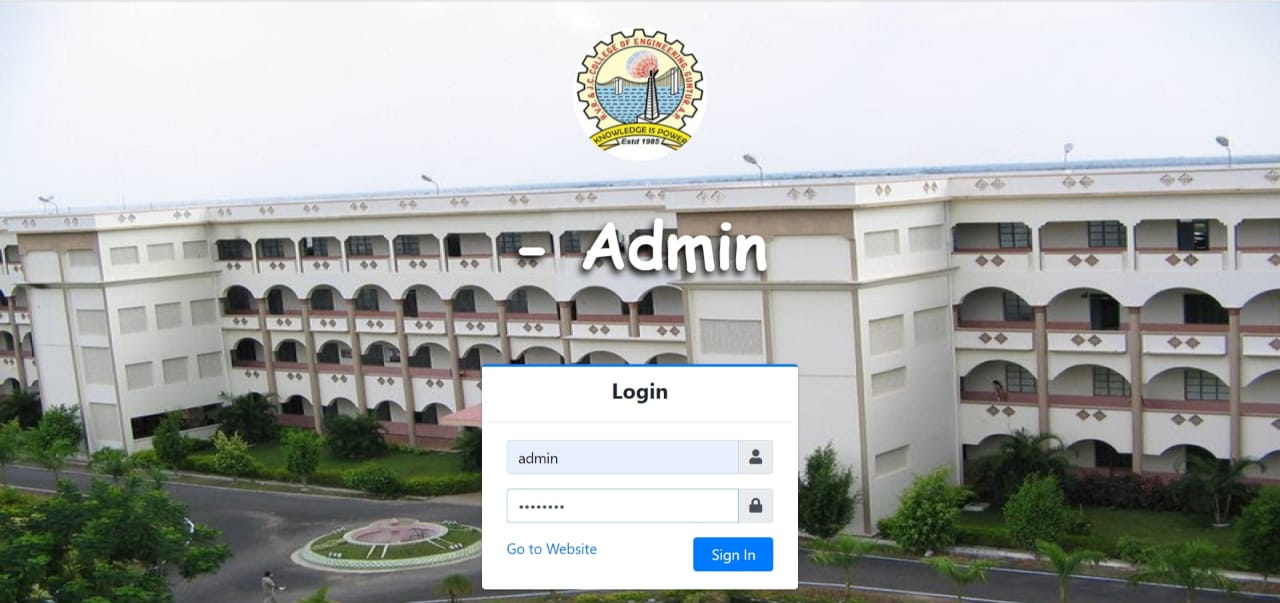
After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output.

**5.2 Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Name** | **Description** | **Steps** | **Expected** | **Actual** | **Test**  **P/F** |
| Login Validation | The username and password of the admin/student should be verified. | 1.Enter username.  2.Enter password.  3.Click the login button. | The login should be Successful. | Successful login. | Pass |
| Add Departments | The admin can add departments that are in the organization. | 1.Click manage departments button.  2.Enter add department.  3.Submit | The department should be added to the list of departments. | The department is added to the list. | Pass |
| Add Curriculums | The admin can add curriculums that are in the organization. | 1.Click manage curriculums button.  2.Enter add curriculum.  3.Submit | The curriculum should be added to the list of curriculums. | The curriculum is added to the list. | Pass |
| Add Thesis | The alumni student can add thesis. | 1.Login to the system.  2.Click Submit Thesis.  3.Add relevant Information.  4.Click submit button. | The lower year student can be able to view the submitted thesis. | The lower year student will view the thesis submitted. | Pass |
| Manage Student | The admin  Can manage students | 1.Login to the system.  2.Click verify student | The student status will be updated and he can be able to login. | Successful registration. | Pass |

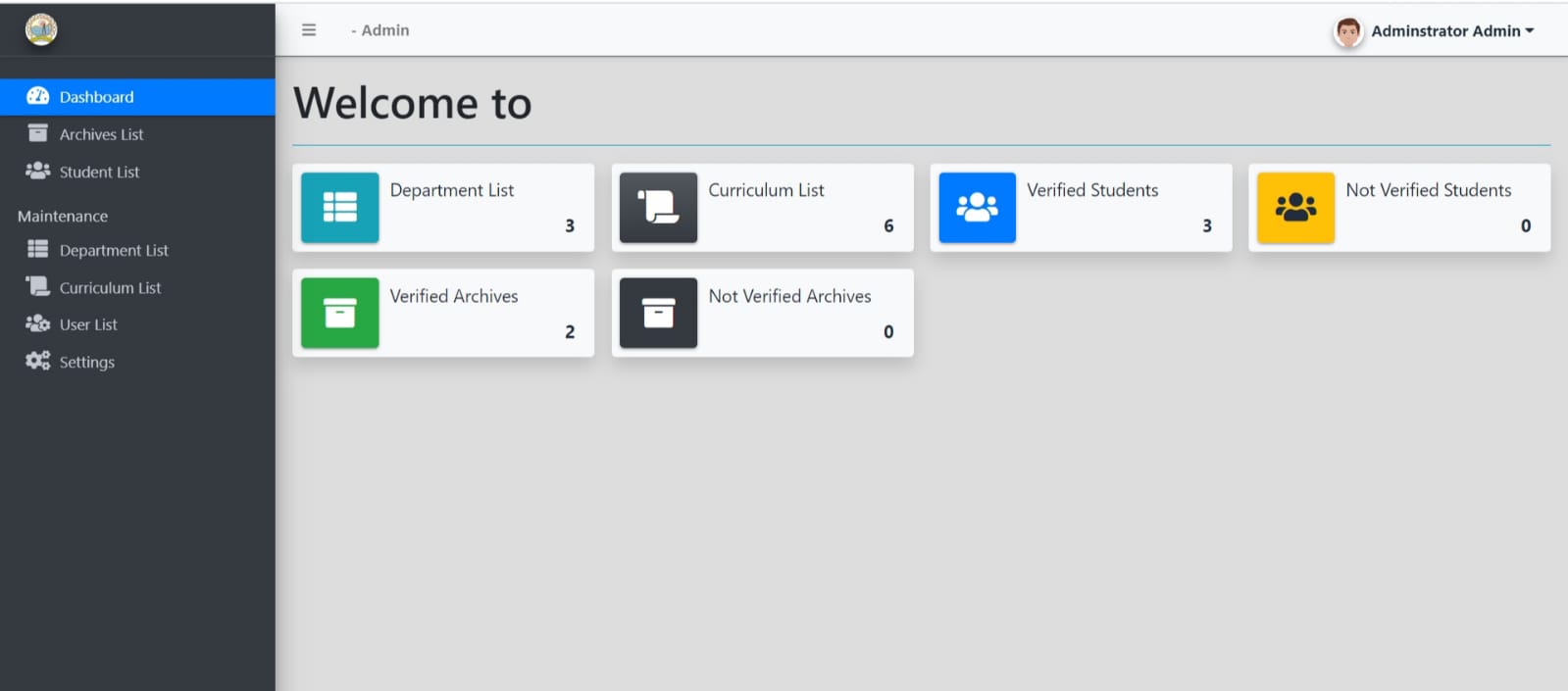
# 6. RESULTS

**The Login Page for admin**

****

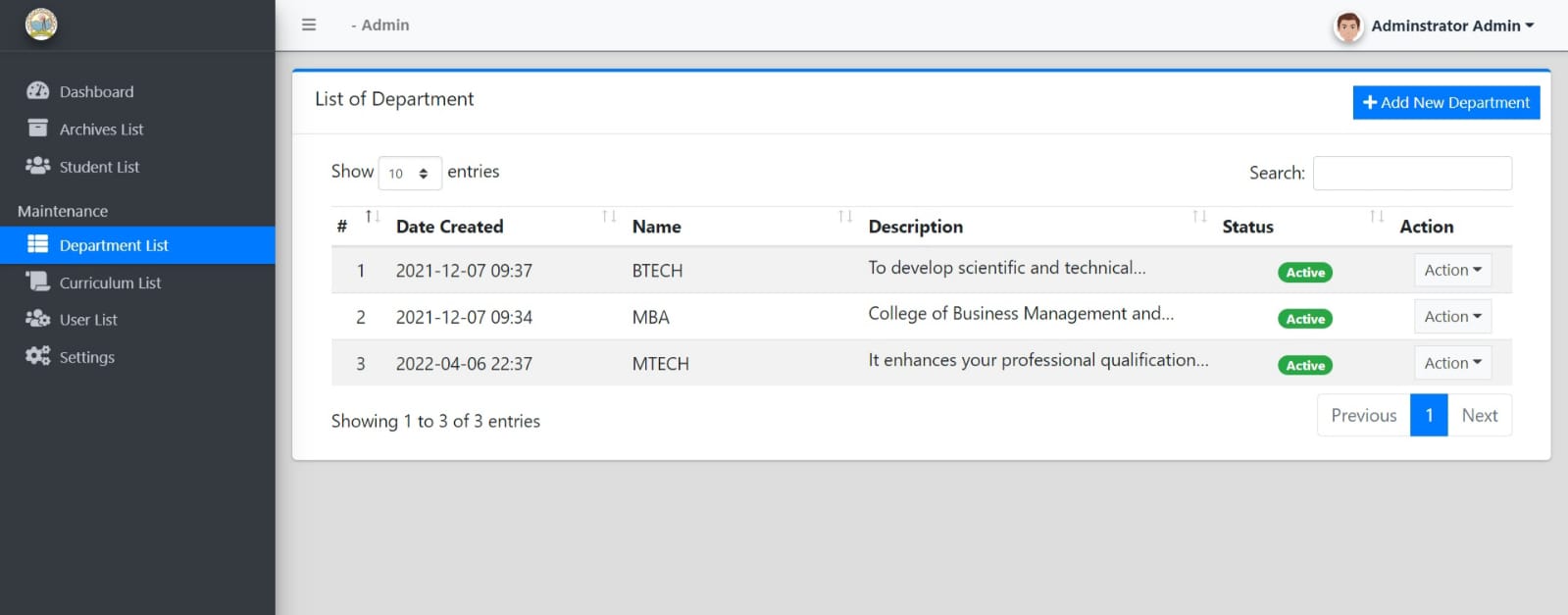
**Figure 6.1: User Interfaces of Admin Login**

**Dashboard for the admin**

****

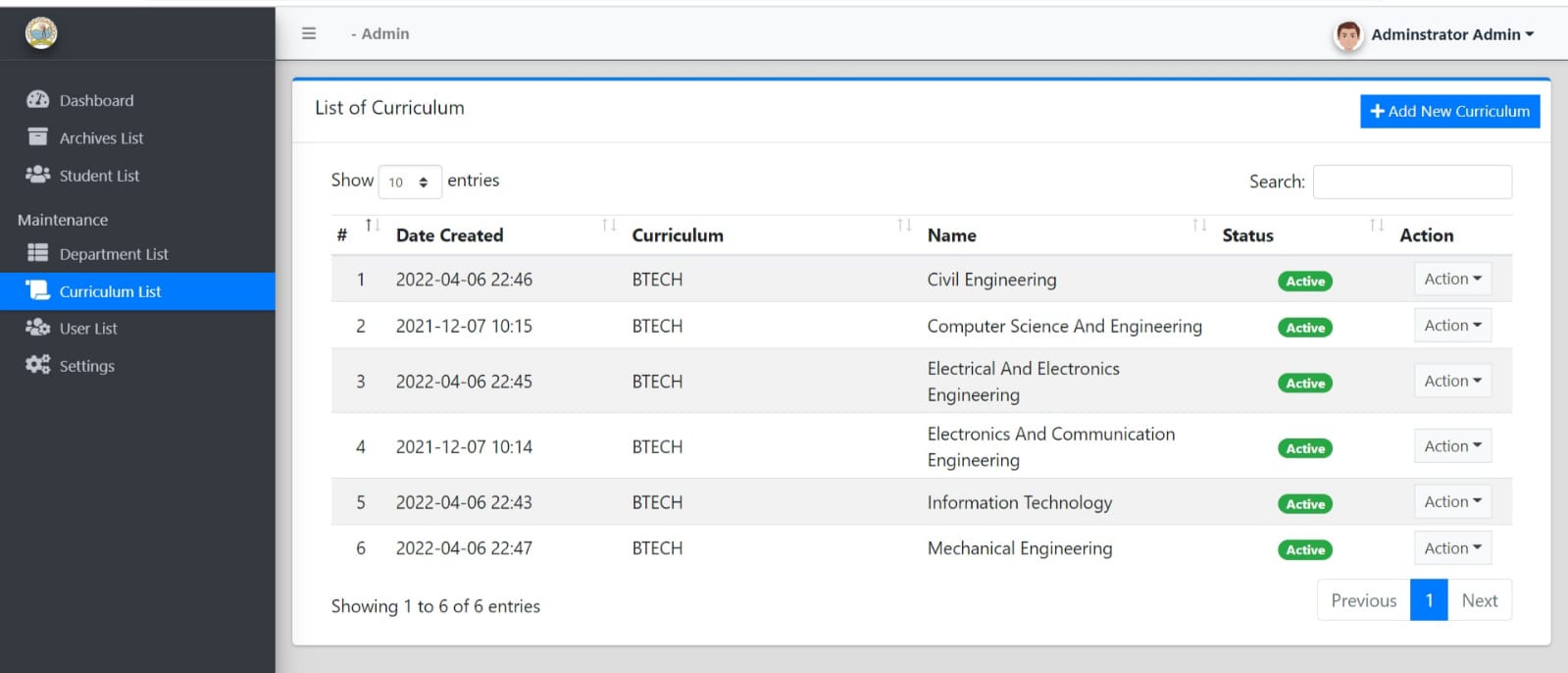
**Figure 6.2:Admin Dashboard**

**Manage Departments**



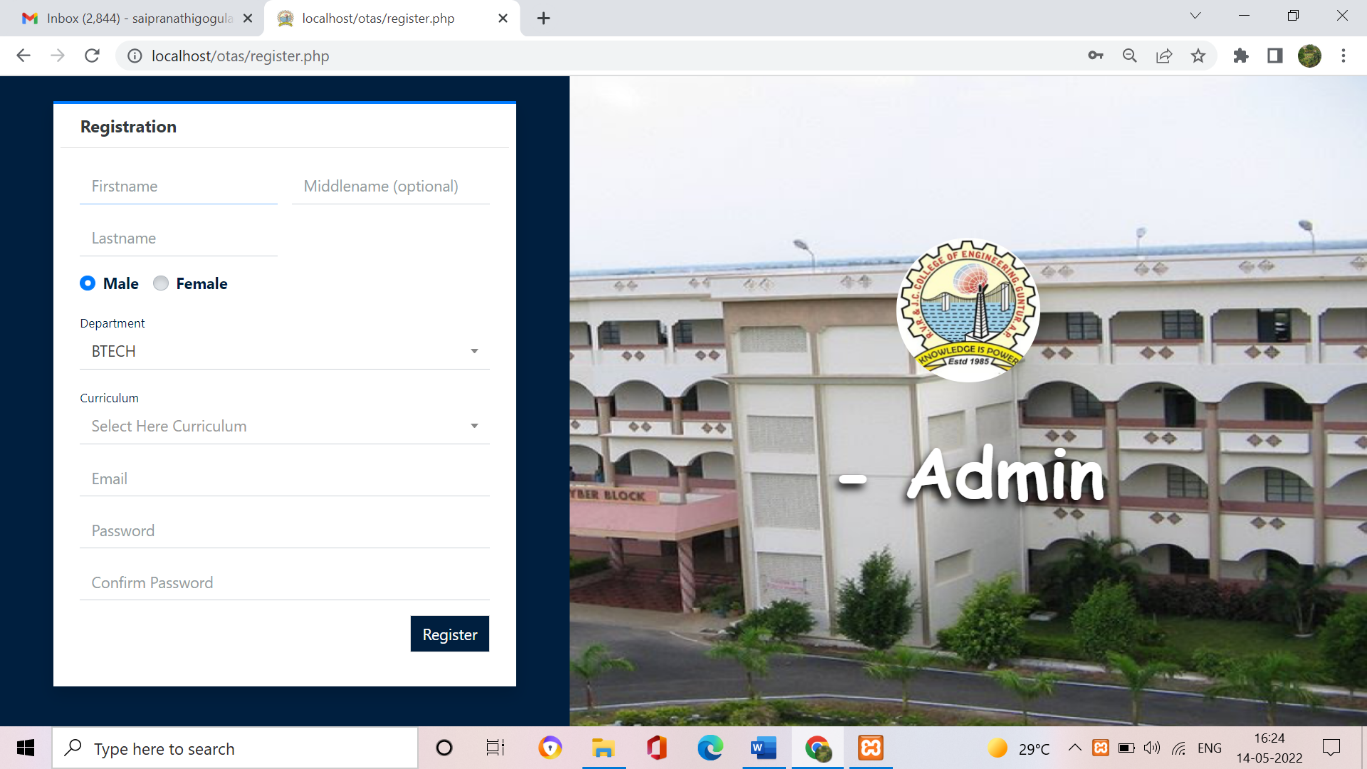
**Figure 6.3:Manage Departments**

**Manage Curriculums**

****

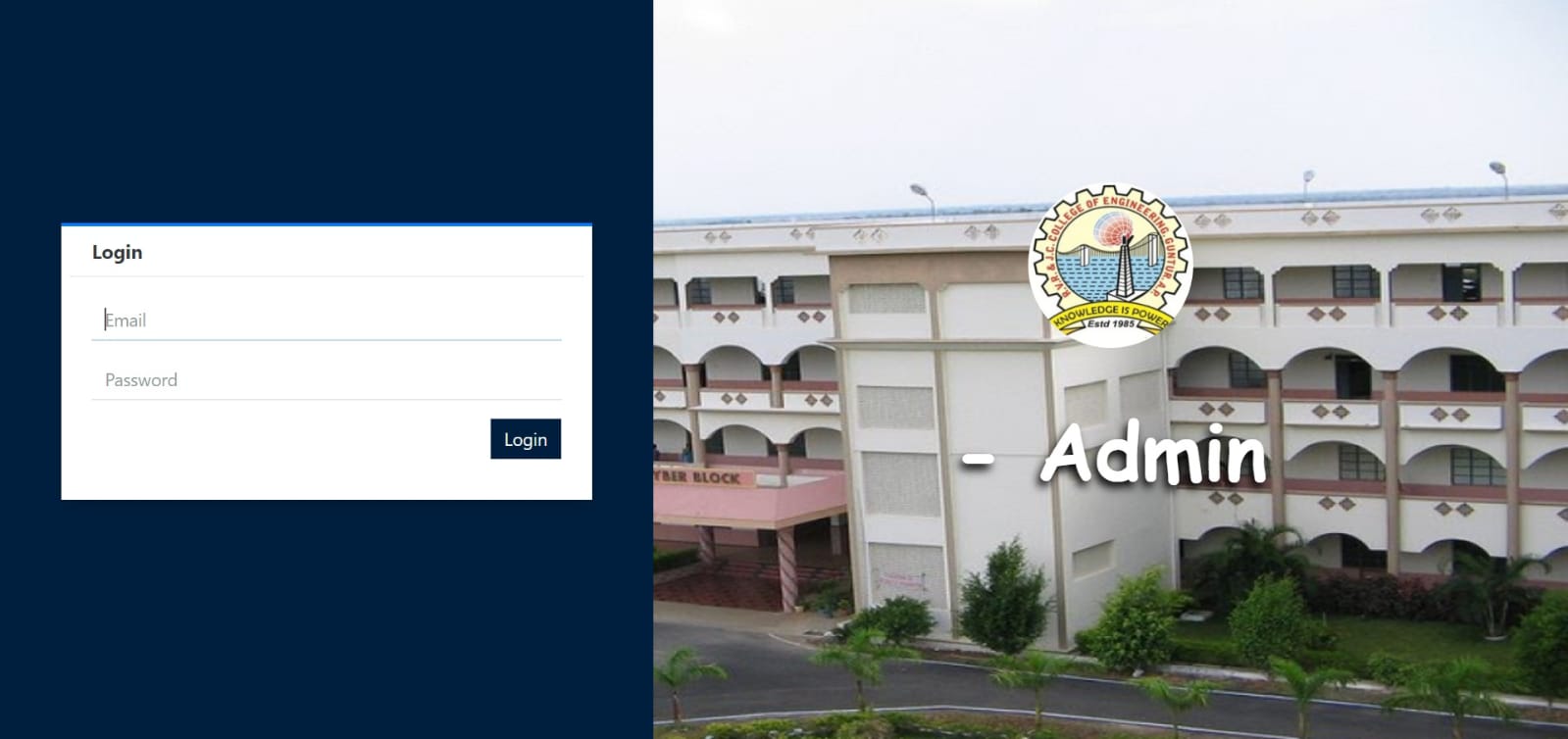
**Figure 6.4 :Manage Curriculums**

**Registration Of Alumni Students**

****

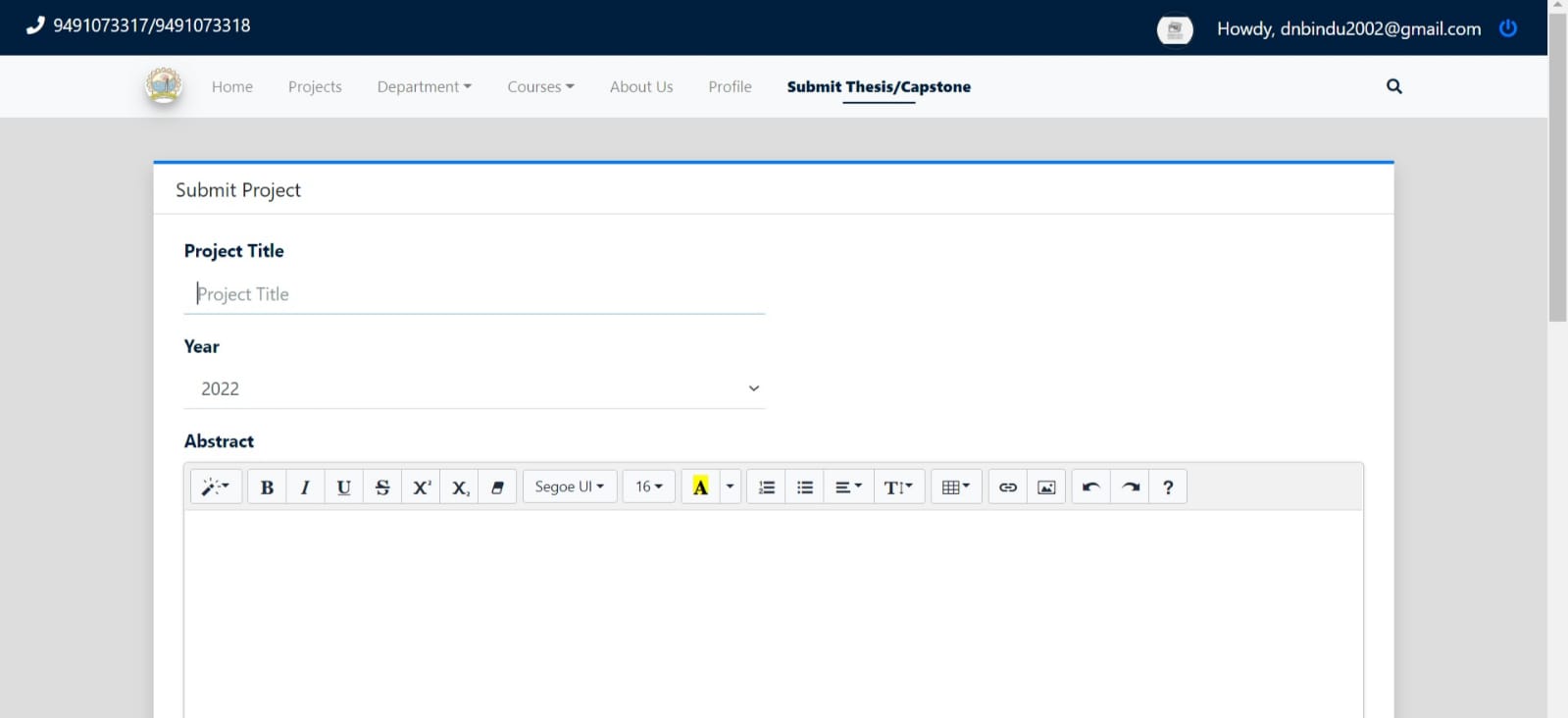
**Figure 6.5:Alumni Students Registration**

**Interface for Alumni Student Login**



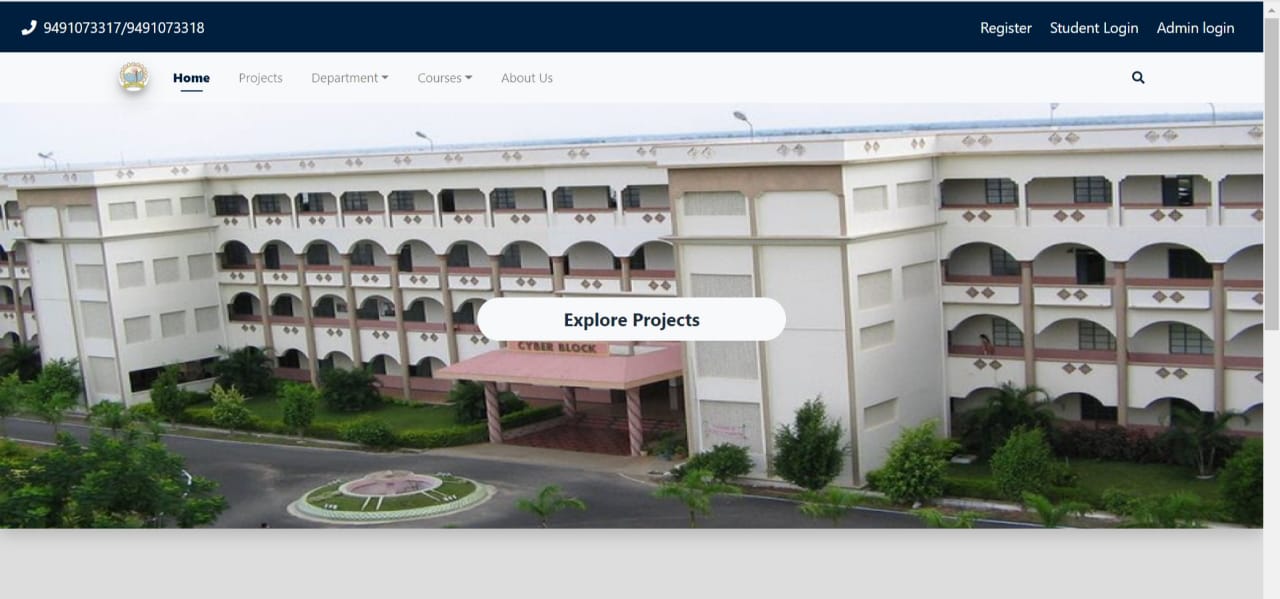
**Fig 6.6 Alumni Student Login**

**Interface for submitting thesis**



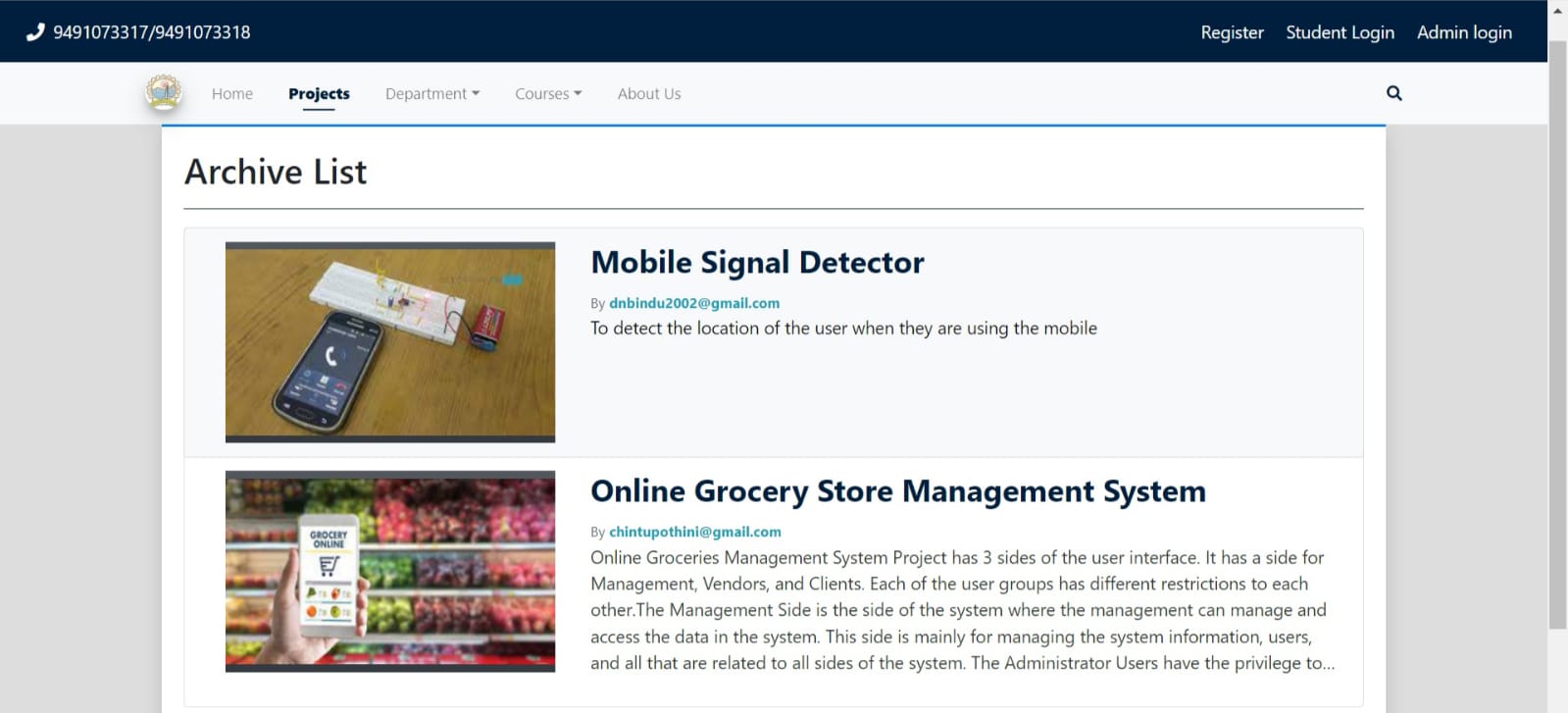
**Figure 6.7: Interface for submitting thesis**

**Lower Year Student Interface**



**Figure 6.8 Lower Year Student Interface**

**Archive List**



**Figure 6.9 Archive List for Lower Year Students**

# 7.CONCLUSION

The new Thesis Archiving System allows the Users to submit the thesis and admin can publish or unpublish the projects,the Lower year students can be able to view the projects and can take the reference to them.

The new system is an improvement on manual Thesis Archiving in the following ways:

1. Alumni students can submit the thesis while being at home.
2. Reduces strain for individuals.
3. More gain in profits.
4. Issue only with security, but with only few websites.
5. It can be accessed at any time and anywhere.
6. Storage is maintained.
7. Cost is reduced.

**REFERENCES**

Object Oriented Analysis and Design 2005 by Mike O’docherty.

Visual Modeling with Rational Rose 2002 by Teny Quatran.

Applying UML and patterns: An Introduction to Object Oriented Analysis and Design, Third Edition 2005 by Craig Larman.

Using UML: Software Engineering with objects and components, Second Edition 2006 by Perdita Stevens, R. J. Pooley.

The elements of UML style by Scott W. Ambler.

The Unified Modeling Language User Guide by Grady Booch, James Rumbaugh, Ivar Jacobson.

### Web Links:

[www.w3schools.com/php](http://www.w3schools.com/php)

<https://stackoverflow.com>

**RVR & JC COLLEGE OF ENGINEERING**

**Chowdavaram, Guntur**

**Andhra Pradesh 522019**

Date: April-2022

Every organization need a maintenance of records for storing the thesis submitted by the alumni students. So that it can help the lower year students for taking reference of the projects submitted by the alumni students. When we are looking for a digital platform for our needs, we came across RVR & JCCE students community projects.

I am P. Rama Krishna Project incharge of III/ IV CSE-B of RVR & JCCE . We would like to extend our thanks to the students of RVR & JCCE (G. Sai Pranathi-Y19CS043,B. Rajesh Naik-Y19CS008,J. Deepthi-Y19CS058) of III/IV B.Tech Computer Science and Engineering for fulfilling our

Requirements and providing an application name “**Online Thesis Archiving System**” to maintain our data records as a part of community project.

Now we can easily fulfill our requirements effortlessly within less time by web based Thesis Archiving System. The application runs without any errors in our systems and provides the accurate results. We appreciate all their efforts in our systems and provide the accurate results. We appreciate all their efforts in providing us this application as a part of their semester course.

P. Rama Krishna

(**Project Incharge**)

### 

### 

### 

# 