**Catechism Management System**

*Project Report Submitted by*

**Amal Biju Thomas**

**Reg. No.: AJC18MCA-I014**

*In Partial fulfillment for the Award of the Degree Of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

# 2022-2023

## DEPARTMENT OF COMPUTER APPLICATIONS

### AMAL JYOTHI COLLEGE OF ENGINEERING

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**Catechism Management System”** is the bona-fide work of **Amal Biju Thomas (Regno: AJC18MCA-I014)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2022-23.

**Ms. Sona Maria Sebastian**

**Internal Guide**

**Ms. Gloriya Mathew**

**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

**DECLARATION**

I hereby declare that the project report **“CATECHISM MANAGEMENT SYSTEM”** is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2022-2023.

**Date: 28-08-2022**

**KANJIRAPPALLY**

**AMAL BIJU THOMAS**

**Reg. No: AJC18MCA-I014**

**ACKNOWLEDGEMENT**

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Gloriya** **Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Sona Maria Sebastian** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

Amal Biju Thomas

## ABSTRACT

.

The Catechism School Management System is a web-based system which will be use for the day-to-day management of a Sunday School. The main objective of this project is to computerize the paperwork in the system and automate the work. The computerization is done so that the storage of all the details regarding students and teachers will be stored in the system which makes system centralized and the chance of duplication of any data is minimized. Automation to the system will reduce the time for storing and managing any data in the system for any long period of time.

The Catechism Management System can be accessed by the admins (Vicar or someone else assigned by the Vicar), faculties and students by entering and verifying the credentials already provided in the database. Only the admin can add new faculties into the system. Admin as well as the faculties can add students into the system. A faculty can add students into particular class to which he/she is assigned as in-charge. For making any change to the data faculties have a limited access, but the admin has full rights to change it.

All the activities of a Sunday school, including creating class, adding members, creating events and exams tabs, etc., can be done in this system.

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No** | | **Topic** | **Page No** | |
| **1** | | **INTRODUCTION** |  | |
| **1.1** | | **PROJECT OVERVIEW** |  | |
| **1.2** | | **PROJECT SPECIFICATION** |  | |
| **2** | | **SYSTEM STUDY** |  | |
| **2.1** | | **INTRODUCTION** |  | |
| **2.2** | | **EXISTING SYSTEM** |  | |
| **2.3** | | **DRAWBACKS OF EXISTING SYSTEM** |  | |
| **2.4** | | **PROPOSED SYSTEM** |  | |
| **2.5** | | **ADVANTAGES OF PROPOSED SYSTEM** |  | |
| **3** | | **REQUIREMENT ANALYSIS** |  | |
| **3.1** | | **FEASIBILITY STUDY** |  | |
| **3.1.1** | | **ECONOMICAL FEASIBILITY** |  | |
| **3.1.2** | | **TECHNICAL FEASIBILITY** |  | |
| **3.1.3** | | **BEHAVIORAL FEASIBILITY** |  | |
| **3.2** | | **SYSTEM SPECIFICATION** |  | |
| **3.2.1** | | **HARDWARE SPECIFICATION** |  | |
| **3.2.2** | | **SOFTWARE SPECIFICATION** |  | |
| **3.3** | | **SOFTWARE DESCRIPTION** |  | |
| **3.3.1** | | **PHP** |  | |
| **3.3.2** | | **MYSQL** |  | |
| **4** | | **SYSTEM DESIGN** |  | |
| **4.1** | | **INTRODUCTION** |  | |
| **4.2** | | **UML DIAGRAM** |  | |
| **4.2.1** | | **USE CASE DIAGRAM** |  | |
| **4.2.2** | | **SEQUENCE DIAGRAM** |  | |
| **4.2.3** | | **STATE CHART DIAGRAM** |  | |
| **4.2.4** | | **ACTIVITY DIAGRAM** |  | |
| **4.2.5** | | **CLASS DIAGRAM** |  | |
| **4.2.6** | | **OBJECT DIAGRAM** |  | |
| **4.2.7** | | **COMPONENT DIAGRAM** |  | |
| **4.2.8** | | **DEPLOYMENT DIAGRAM** |  | |
| **4.2.9** | | **COLLABORATION DIAGRAM** |  | |
| **4.3** | | **USER INTERFACE DESIGN USING FIGMA** |  | |
| **4.4** | | **DATA BASE DESIGN** |  | |
| **5** | | **SYSTEM TESTING** |  | |
| **5.1** | | **INTRODUCTION** |  | |
| **5.2** | | **TEST PLAN** |  | |
| **5.2.1** | **UNIT TESTING** | |  |
| **5.2.2** | **INTEGRATION TESTING** | |  |
| **5.2.3** | **VALIDATION TESTING** | |  |
| **5.2.4** | **USER ACCEPTANCE TASTING** | |  |
| **5.2.5** | **AUTOMATION TESTING** | |  |
| **5.2.6** | **SELENIUM TESTING** | |  |
| **6** | **IMPLEMENTATION** | |  |
| **6.1** | **INTRODUCTION** | |  |
| **6.2** | **IMPLEMENTATION PROCEDURE** | |  |
| **6.2.1** | **USER TRAINING** | |  |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** | |  |
| **6.2.3** | **SYSTEM MAINTENANCE** | |  |
| **6.2.4** | **HOSTING** | |  |
| **7** | **CONCLUSION & FUTURE SCOPE** | |  |
| **7.1** | **CONCLUSION** | |  |
| **7.2** | **FUTURE SCOPE** | |  |
| **8** | **BIBLIOGRAPHY** | |  |
| **9** | **APPENDIX** | |  |
| **9.1** | **SAMPLE CODE** | |  |
| **9.2** | **SCREEN SHOTS** | |  |

## List of Abbreviation

|  |  |  |
| --- | --- | --- |
| IDE | - | Integrated Development Environment |
| HTML |  | Hyper Text Markup Language. |
| CSS |  | Cascading Style Sheet |
| SQL |  | Structured Query Language |
| UML |  | Unified Modeling Language |
| CMS |  | Catechism Management System |
|  |  |  |
|  |  |  |
|  |  |  |

# CHAPTER 1

# INTRODUCTION

### PROJECT OVERVIEW

The central concept of the application is to allow the vicar and faculties to manage the Catechism of a Parish virtually using the Internet and allow students to manage activities such as leave application, marks viewing and some others. The information pertaining to the students, faculties, vicar, etc. are stored on an RDBMS at the server side. The Server process the data. The application was designed into three modules. First module is for the admin who has the full control of the application. The admin can be the vicar of the church or someone else who is appointed by the admin. Second module is for the faculties who maintains and updates the information pertaining to the students of the class to which they are assigned to, by the admin. The third module is for use of the students of the Sunday School. The student module is the one with least functionalities and least priority. The end user of this product is a Sunday school where the application is hosted on the web and the administrator maintains the database. The application which is deployed at the user database, the details of the items are brought forward from the database for the user view based on the selection through the menu and the database is updated at the end of each transaction. Data entry into the application can be done through various screens designed for various levels of users.

There is a provision to enter the data of faculties and students in bulk quantity using the CSV file. Once the authorized personnel feed the relevant data into the system, several reports could be generated as per the security.

### PROJECT SPECIFICATION

**Admin Module:**

The person in this module has control over the entire system as he/she can add, delete anyone from the system. Admin must have a login. Creating classes, assigning students and teachers to articular class, creating any new events, etc. are the responsibilities of the admin. The admin can view the details of any other person and can edit their details. The users from another module will send the request to admin for any change. From this module, the students can view the details of the assignment given to them, exam date and results.

**Faculty Module:**

This module is for the teachers. Teachers are registered by the admin. They are being assigned to some particular class as its in-charge. They are responsible for updating the marks of the students and their attendance. They can approve the eave reasons of students. A teacher has only the access to the students of the class to which they are being assigned by the admin. They can’t view or edit any data about the other students.

**Students Module:**

From this module, the students can view the details of them, assignment given to them, exam date and results and so on. They can view the details of the faculties and can contact them.

### The various system specification that has been used in developing both the frontend and the back end of the project are being discussed below.

### 1.2.1 Front-End

### HTML, CSS, JAVA SCRIPT are utilized to implement the frontend.

### HTML (Hyper Text Markup Language)

### HTML is a syntax used to format a text document on the web.

### CSS (Cascading Style Sheets)

### CSS is a style sheet language used for describing the look and formatting of a document written in a markup language.

### Java Script

### JS is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side

### 1.2.2 Back-End

### The back end is implemented using Python Flask and Sqlite3 which is used to design the databases.

# 

# CHAPTER 2

# SYSTEM STUDY

### INTRODUCTION

The main objective of Sunday School Management System is to computerize the paperwork in the system and automate the work. This is a web-based system which will be use for the day-to-day management of a Sunday School. The computerization is done so that the storage of all the details regarding students and teachers will be stored in the system which makes system centralized and the chance of duplication of any data is minimized. Automation to the system will reduce the time for storing any data in the system.

### EXISTING SYSTEM

In the present system, it is not easy to store the information related to students, faculty and parents on the paper for a long period of time. As there is too many information when someone tries to access any of stored information it becomes a difficult and time-consuming task. Whereas the storing and retrieving an information is a difficult task, it also requires much amount of unnecessary worker to do the task.

### DRAWBACKS OF EXISTING SYSTEM

* + - Huge storage rooms required to store the complete data of the Sunday school.
    - Difficult to make changes to any data.
    - Difficult to find data of a particular student
    - Difficult to produce reports.

### PROPOSED SYSTEM

The Sunday School Management System will manage all the work in any Sunday School in particular order so that the time requirement and complexity of the system will be reduced, at first it will focus on student related information. As a student gets the admission in the school system will start managing the details regarding the students. It will manage the fee details (like the text book fee), and if the full payment has not done, then it will notify about the fee to a staff of the school. This will then display the date of the exam and when the valuation is completed, teachers will update the marks to the system.

### ADVANTAGES OF PROPOSED SYSTEM

### Almost 90% of the paper works can be stopped after implementing this system.

### Does not require a large office room for handling all the data.

### Easy to update any data.

### Easily find data by search functionality.

### Any sort of reports can be generated easily within very less amount of time.

### The CMS can provide more data security.

### It makes the processes in a Sunday school faster and more efficient.

# CHAPTER 3 REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

## The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

### Economic Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

* + - * The costs conduct a full system investigation.
      * The cost of the hardware and software.
      * The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

The cost of project, DREAMS was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open-source software.

### 3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

* + - * Does the existing technology sufficient for the suggested one?
      * Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

### Behavioral Feasibility

### The following inquiries are part of the suggested system:

### Is there enough assistance for the users?

### Will the suggested system harm anyone?

### The project would be advantageous because, when created and implemented, it would achieve the goals. The project is deemed to be behaviorally feasible after carefully weighing all behavioral factors.

Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)

## SYSTEM SPECIFICATION

### Hardware Specification

Processor - Intel core i5

RAM - 8GB

Hard disk - 1TB

### Software Specification

Front End - HTML, CSS

Backend - MYSQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, J Query, PHP, CSS

## SOFTWARE DESCRIPTION

### PHP

# CHAPTER 4 SYSTEM DESIGN

* 1. **INTRODUCTION**

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

## UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non- software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object

oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

* + - Class diagram
    - Object diagram
    - Use case diagram
    - Sequence diagram
    - Activity diagram
    - Statechart diagram
    - Deployment diagram
    - Component diagram

## USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service- oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

* The boundary, which defines the system of interest in relation to the world around it.
* The actors, usually individuals involved with the system defined according to their roles.
* The use cases, which are the specific roles are played by the actors within and around the system.
* The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

* The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
* G(ive a suitable name for actors.
* Show relationships and dependencies clearly in the diagram.
* Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
* Use notes whenever required to clarify some important points.

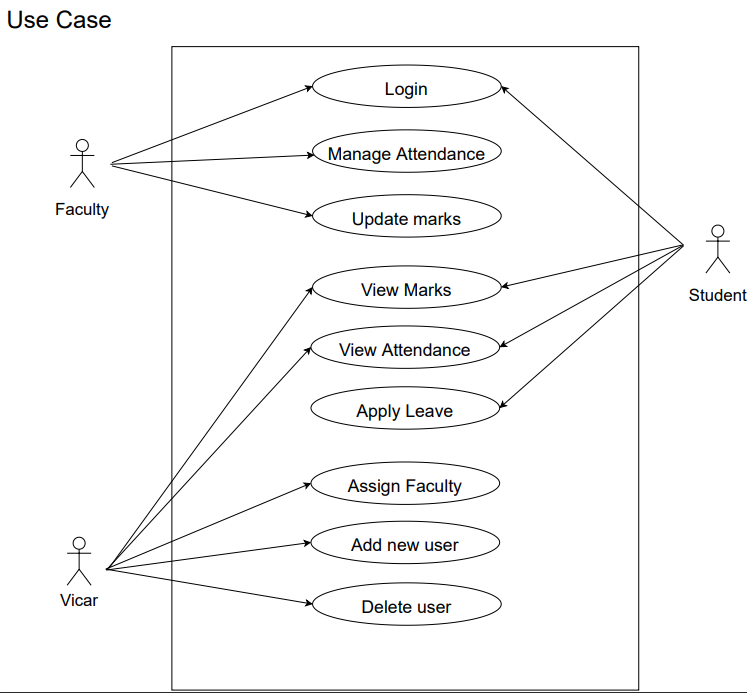


Figure (a) – Use Case diagram

## SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

#### **Sequence Diagram Notations –**

1. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram
3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

* + Synchronous messages
  + Asynchronous Messages
  + Create message
  + Delete Message
  + Self-Message
  + Reply Message
  + Found Message
  + Lost Message

1. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

#### **Uses of sequence diagrams –**

* + Used to model and visualize the logic behind a sophisticated function, operation or procedure.
  + They are also used to show details of UML use case diagrams.
  + Used to understand the detailed functionality of current or future systems.
  + Visualize how messages and tasks move between objects or components in a system.

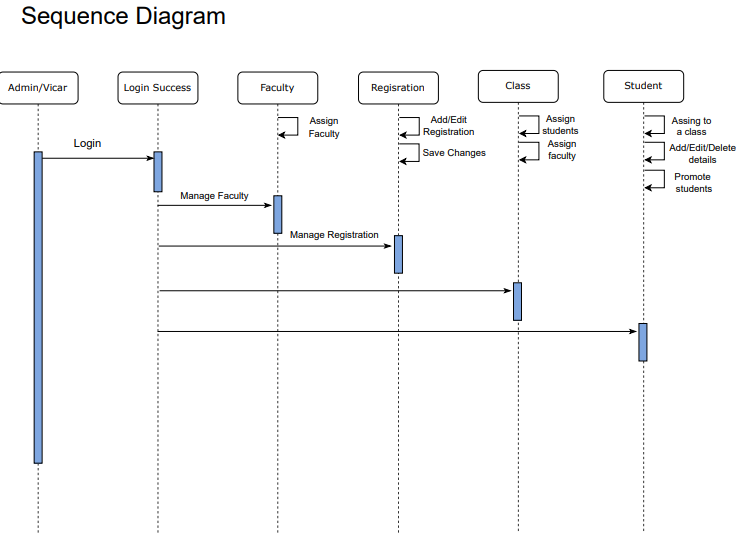


Figure (b) – Sequence Diagram

## State Chart Diagram

The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

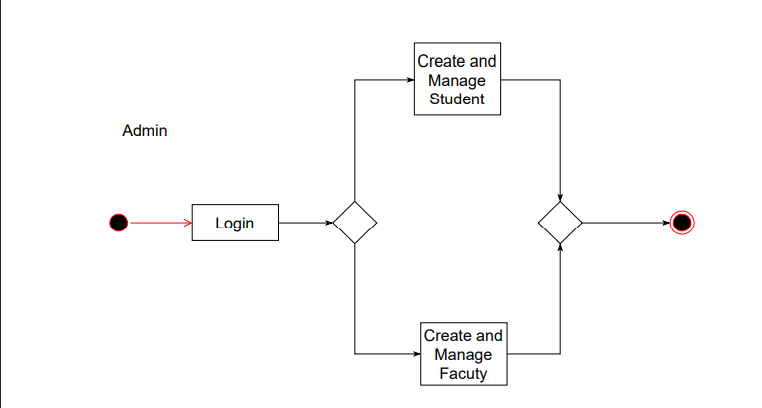
It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system. It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system. Each object/component has a specific state.

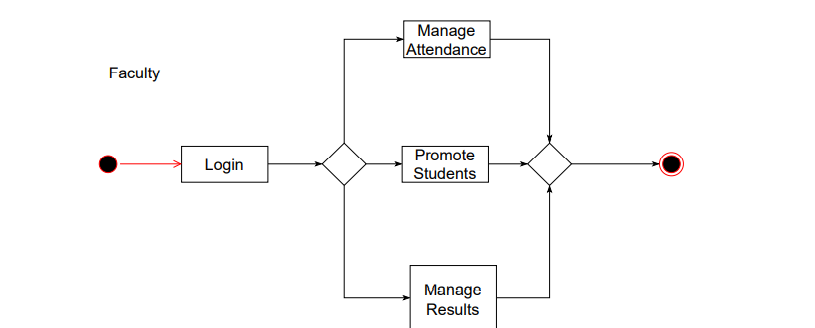
#### **Notation of a State Machine Diagram**

Following are the notations of a state machine diagram enlisted below:

1. **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
2. **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
3. **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
4. **Transition:** A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.

**State box:** It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.





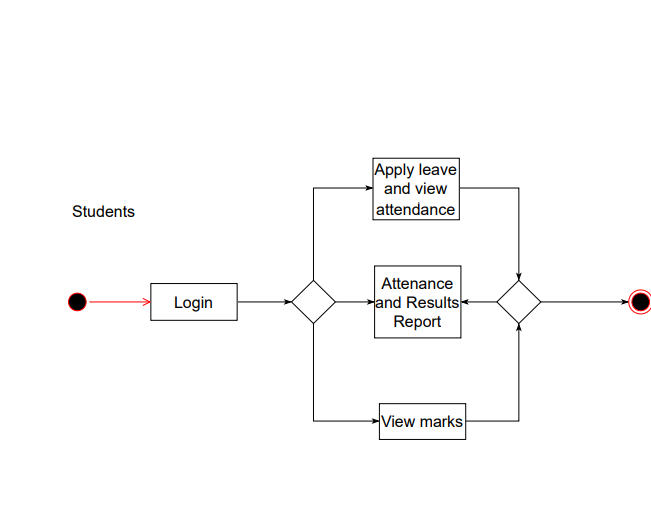


Figure (c) – State Chart Diagram

## Activity Diagram

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

#### **Components of an Activity Diagram**

Following are the component of an activity diagram:

#### **Activities**

The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.

The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity. The activities are initiated at the initial node and are terminated at the final node.

#### **Activity partition /swimlane**

The swimlane is used to cluster all the related activities in one column or one row. It can be either vertical or horizontal. It used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.

#### **Forks**

Forks and join nodes generate the concurrent flow inside the activity. A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.

#### **Join Nodes**

Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.

#### **Notation of an Activity diagram**

Activity diagram constitutes following notations:

**Initial State:** It depicts the initial stage or beginning of the set of actions.

**Final State:** It is the stage where all the control flows and object flows end.

**Decision Box:** It makes sure that the control flow or object flow will follow only one path.

**Action Box:** It represents the set of actions that are to be performed.

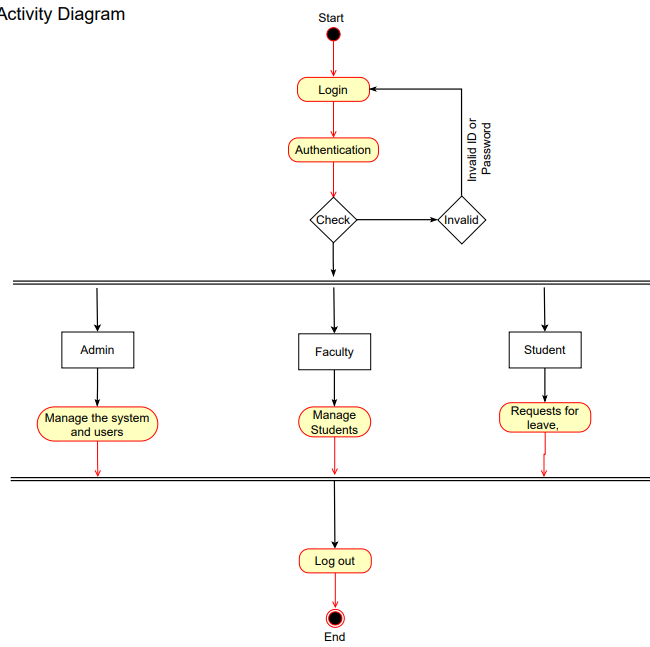


Figure (d) - Activity Diagram

## Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

#### **Components of a Class Diagram**

The class diagram is made up of three sections:

**Upper Section:** The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships,

attributes, operations, and semantics. Some of the following rules that should be taken into account while representing a class are given below:

1. Capitalize the initial letter of the class name.
2. Place the class name in the center of the upper section.
3. A class name must be written in bold format.
4. The name of the abstract class should be written in italics format.

**Middle Section:** The middle section constitutes the attributes, which describe the quality of class.

The attributes have the following characteristics:

* The attributes are written along with its visibility factors, which are public (+), private (-), protected (#), and package (~).
* The accessibility of an attribute class is illustrated by the visibility factors.
* A meaningful name should be assigned to the attribute, which will explain its usage inside the class.

**Lower Section:** The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.

#### **Relationships**

In UML, relationships are of three types:

* **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship.
* **Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class.
* **Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.
* **Multiplicity:** It defines a specific range of allowable instances of attributes. In case if a range is not specified, one is considered as a default multiplicity.
* **Aggregation:** An aggregation is a subset of association, which represents has a relationship. It is more specific then association. It defines a part-whole or part- of relationship. In this kind of relationship, the child class can exist independently of its parent class.
* **Composition:** The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.

#### **Abstract Classes**

In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects.

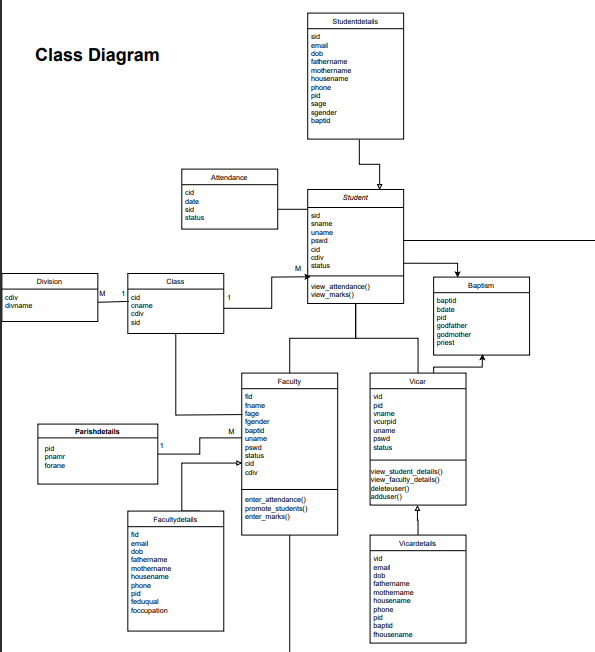


Figure (e) – Class Diagram

## Object Diagram

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant.

Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

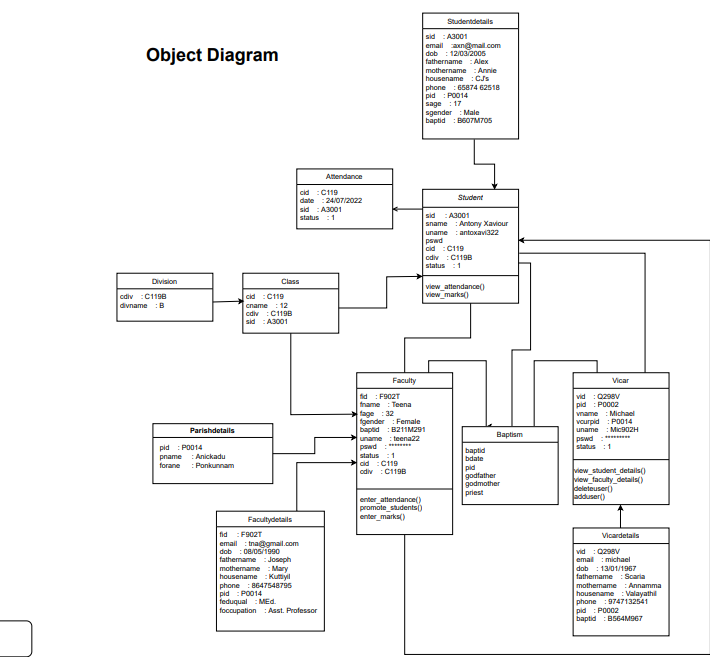


Figure (f) – Object Diagram

## Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

#### **Notation of a Component Diagram**

1. A component
2. A node

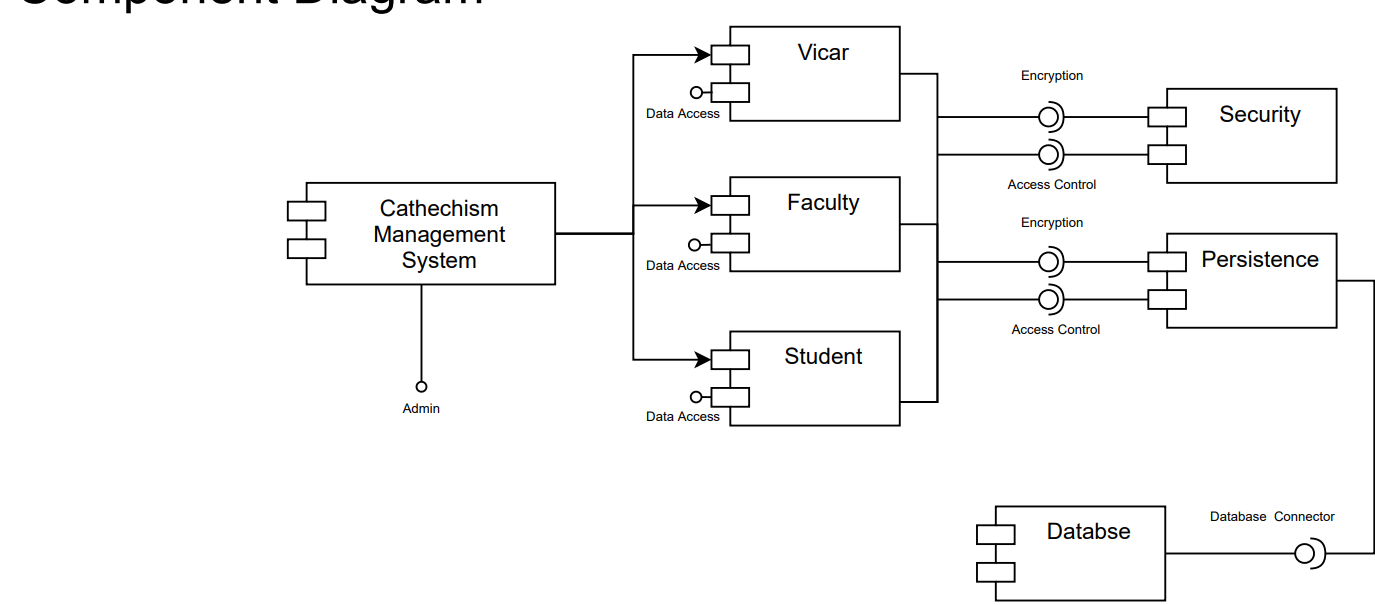


Figure (g) – Component Diagram

**4.2.8 Deployment Diagram**

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

#### **Notation of Deployment diagram**

The deployment diagram consists of the following notations:

1. A component
2. An artifact
3. An interface
4. A node

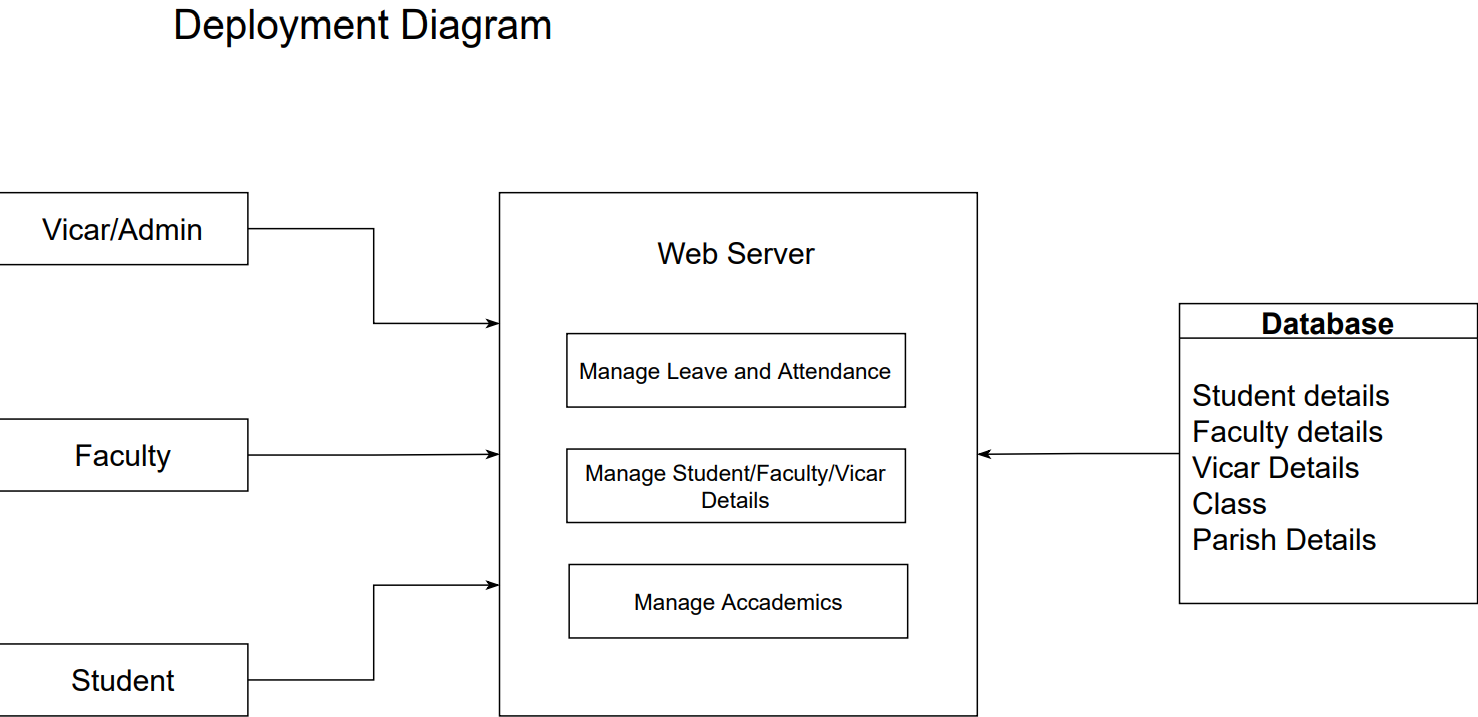
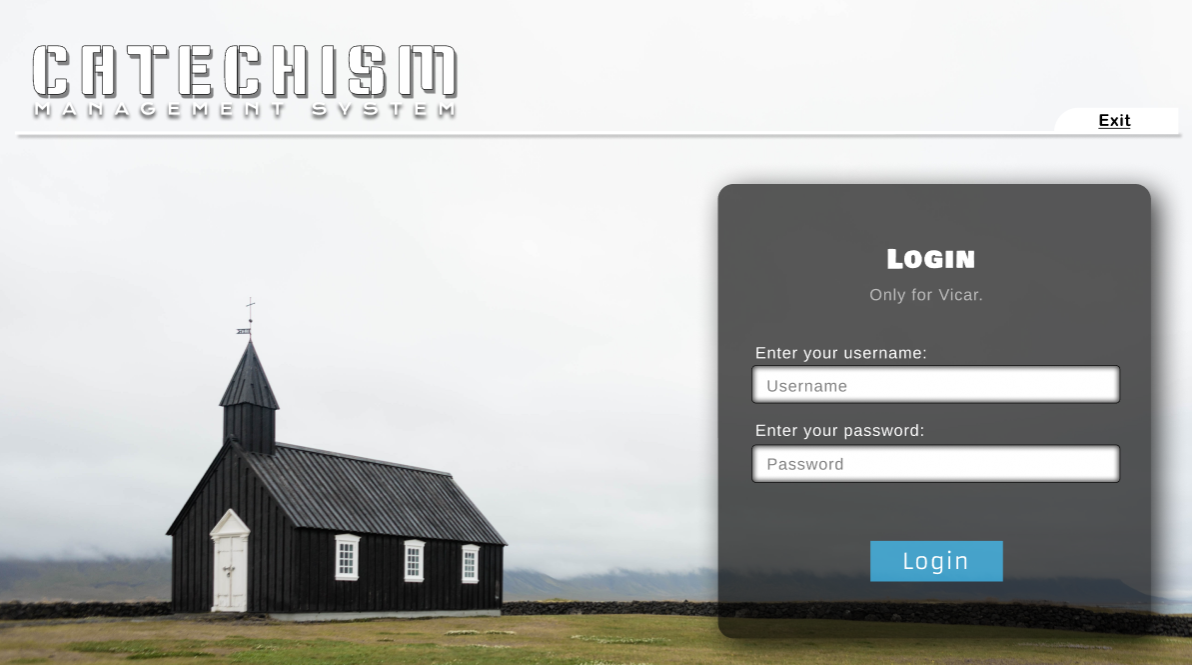


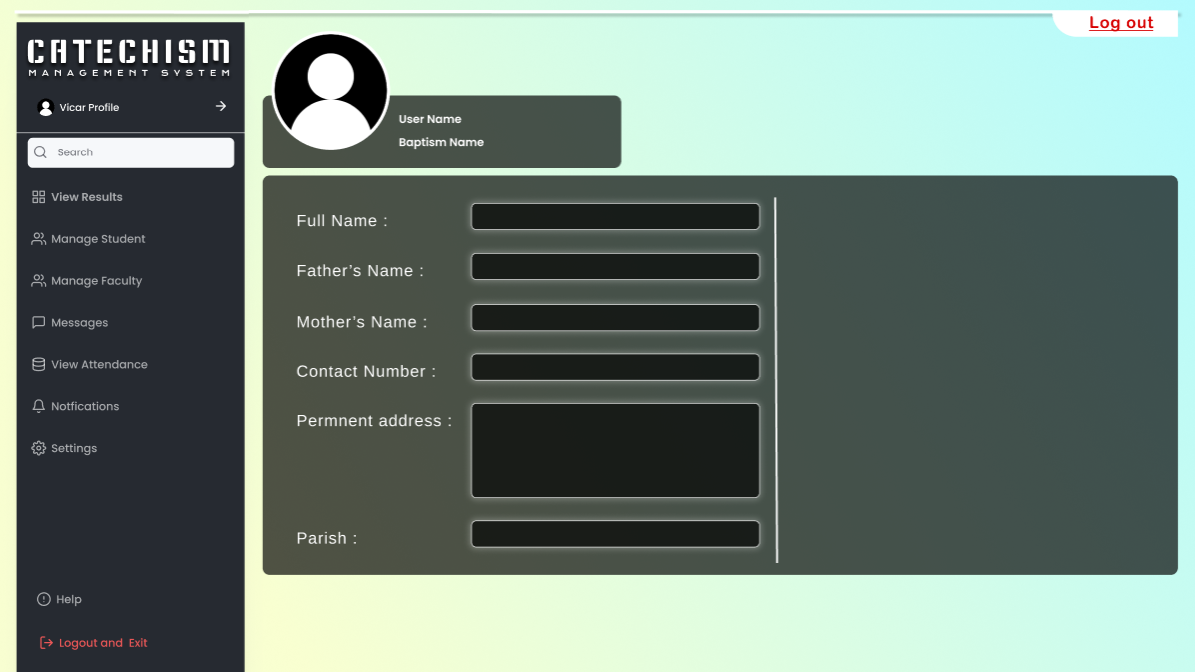
Figure (h) – Deployment Diagram

### 4.3 USER INTERFACE DESIGN USING FIGMA

**Form Name : Login**



**Form Name : Admin Dashboard**



## 4.4DATABASE DESIGN

### 4.4.1 Relational Database Management System (RDBMS)

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

* + - Data Integrity
    - Data independence

#### **Relational Database Management System (RDBMS)**

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

#### **Relations, Domains & Attributes**

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

#### **Relationships**

* Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
* Entity Integrity enforces that no Primary Key can have null values.
* Referential Integrity enforces that no Primary Key can have null values.
* Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.

### 4.4.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

* + - * Normalize the data.
      * Choose proper names for the tables and columns.
      * Choose the proper name for the data.

#### **First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be donor by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

#### **Second Normal Form**

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

#### **Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

### 4.4.3 Sanitization

**4.4.4 Indexing**

### 4.5 TABLE DESIGN

**1.Tbl\_users\_login**

Primary key: **loginid**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype | Key Constraints | Description |
| 1 | userid | varchar | foreign key | Primary key of Registration table |
| 2 | useremail | varchar |  |  |
|  | password | varchar |  |  |
| 3 | role | varchar |  |  |

**CHAPTER 5**

**SYSTEM TESTING**

* 1. **INTRODUCTION**

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

* A good test case is one that has high possibility of finding an undiscovered error.
* A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

* For correctness
* For implementation efficiency
* For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

## TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing

### Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code where removed and ensured that all modules are working, and gives the expected result.

### Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The programme as a whole is tested. Correction is challenging since the size of the overall programme makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All of the modules were integrated after unit testing was completed in the system to check for any interface inconsistencies. A distinctive programme structure also developed when discrepancies in programme structures were eliminated.

### Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions thatwill fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

### Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

* Input Screen Designs,
* Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

* + 1. **Automation Testing**

Automation Testing, often known as Test Automation, is a software testing approach that involves the execution of a test case collection using particular automated testing software tools. On the other hand, manual testing is carried out by a person sitting in front of a computer, methodically carrying out the test processes.

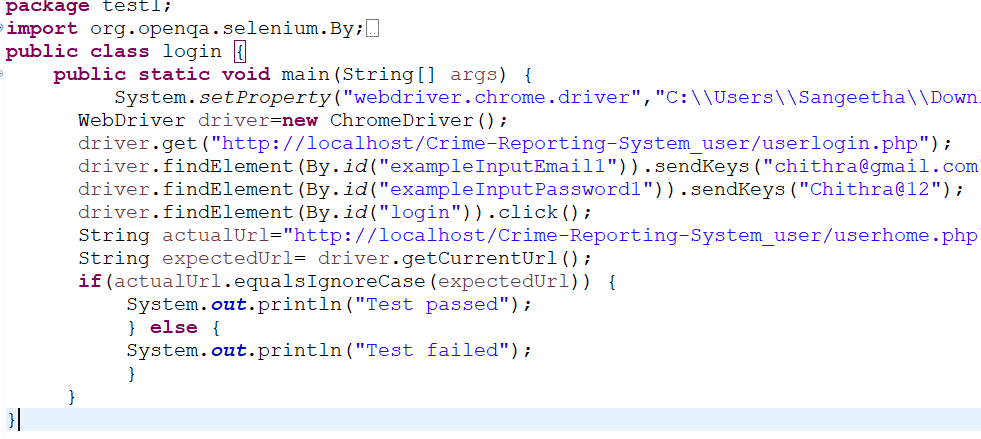
In addition to entering test data into the System Under Test, the automated testing software may analyze predicted and actual outcomes and provide complete test reports. Software Test Automation necessitates significant financial and human resources. Continuous implementation of the same test suite will be required in subsequent development cycles. This test suite may be recorded and replayed as needed using a test automation tool. There is no need for human interaction after the test suite has been automated.

* + 1. **Selenium Testing**

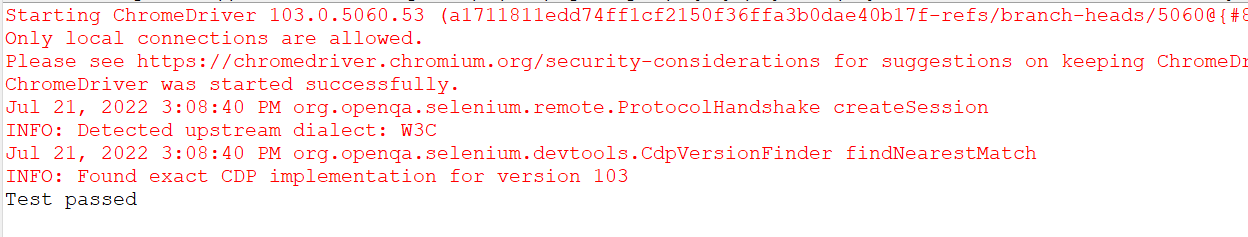
A curated set of various software tools, each striving with a distinct approach of assisting test automation, is known as Selenium Testing. This comprehensive suite of tools showcases the capabilities to satisfy the needs of testing of all types of web applications. Selenium Testing perfectly resembles flexibility. It plays a pivotal role in comparing expected test results against the actual behavior of an application. It allows various options in order to locate UI elements. In short, Selenium Testing is an important support system for the execution of tests on multiple browser platforms.

**Example: Test Case 1**

**code**



**Eg.screenshot**



**Eg.Test report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name:** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID: Test\_1** | | | **Test Designed By:** | | |
| **Test Priority(Low/Medium/High):** | | | **Test Designed Date:** | | |
| **Module Name**: | | | **Test Executed By :** | | |
| **Test Title :** | | | **Test Execution Date:** | | |
| **Description:** | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |  |  |  |
|  |  |  |  |  |  |
| 6 |  |  |
| 7 |  |  |  |  |  |
|  |  |  |  |  |
| **Post-Condition:** | | | | | |

**Test Case 2: minimum 4 testcases**

**Code**

**Screenshot**

**Test report**

# CHAPTER 6

# IMPLEMENTATION

## INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

* + - * Careful planning.
      * Investigation of system and constraints.

Design of methods to achieve the changeover

## IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system.

In many organizations someone who will not be operating it, will commission the

software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

* The active user must be aware of the benefits of using the new system.
* Their confidence in the software is built up.
* Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won’t take place.

### User Training

The purpose of user training is to get the user ready to test and modify the system. The people who will be involved must have faith in their ability to contribute to the goal and benefits anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle error warnings, query the database, call up routines that will generate reports, and execute other important tasks through user training.

### System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the

system environment. Software maintenance is of course, far more than "Finding Mistakes".

### Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

# CHAPTER 7 CONCLUSION AND FUTURE SCOPE

## CONCLUSION

## 

The Catechism Management System is going to decrease the much-complicated paper works of the Sunday schools. It will surely make the process of the storage of data of the students and faculties as compared to the traditional way of management. The proposed system will help all its users to carry out all the required activities in a much better manner, in very less time.

* 1. **FUTURE SCOPE**
* The proposed system can manage the data of the whole Sunday school efficiently.
* It can add and manage members in a very effective way.
* The CMS can handle the results publishing.
* Can conduct quizzes
* This will be able to manage the entire data regarding the various programs conducted in the Sunday school.

# CHAPTER 8 BIBLIOGRAPHY

### REFERENCES:

* + - ..
    - ..
    - ..
    - ..
    - ...

### WEBSITES:

* + - [..](http://www.w3schools.com/)
    - [..](http://www.jquery.com/)
    - [..](http://homepages.dcc.ufmg.br/%7Erodolfo/es-1-03/IEEE-Std-830-1998.pdf)
    - [..](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

# CHAPTER 9 APPENDIX

## Sample Code

30

Project name

* **Config.php ----Database Configuration**

<?php

define('DB\_SERVER','localhost');

define('DB\_USER','root');

define('DB\_PASS' ,'');

define('DB\_NAME', 'catechism');

$con = mysqli\_connect(DB\_SERVER,DB\_USER,DB\_PASS,DB\_NAME);

// Check connection

if (mysqli\_connect\_errno())

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

?>

* **Login.php ---- Code of the login page**

<script type = "text/javascript" >

function preventBack() { window.history.forward(); }

setTimeout("preventBack()", 0);

window.onunload = function () { null };

</script>

<?php

include('config.php');

if(isset($\_POST['submit']))

{

$us=mysqli\_real\_escape\_string($con, $\_POST['adminemail']);

$password=mysqli\_real\_escape\_string($con, $\_POST['adminpassword']);

$query="SELECT \* FROM login\_table WHERE useremail ='$us'and password='$password'";

$res=mysqli\_query($con,$query);

if(mysqli\_num\_rows($res)>0)

{

$row = mysqli\_fetch\_array($res);

$\_SESSION['authentication'] = true;

$\_SESSION['auth\_user'] = [

'user\_id'=>$row['userid'],

'email'=>$row['useremail'],

'role'=>$row['role'],

];

if($row['role'] == 1)

{

session\_start();

$\_SESSION['email'] = $row['useremail'];

header("Location:admindashboard.php");

$\_SESSION['message']="You are Logged In Successfully";

}

elseif($row['role'] == 2 and $row['status'] == 1)

{

session\_start();

$\_SESSION['email'] = $row['useremail'];

header("Location:facultydashboard.php");

exit(0);

}

/\*

elseif($row['role'] == 3)

{

header("Location:studentdashboard.php");

exit(0);

}\*/

else

{

header("Location:login.php");

}

}

/\*

if($count == 1){

header("Location:admindashboard.php");

}

else{

echo "<h1> Login failed. Invalid username or password.</h1>";

}

\*/

else

{

$\_SESSION['message']="Invalid Email or Password";

echo '<script type="text/javascript"> alert("invalid user name or password!!")</script>';

header("Location: login.php");

exit(0);

}

}

?>

Main functionalities

## Screen Shots