Contents

[Chapter 1: Introduction 3](#_Toc202018573)

[1.1. Introduction 3](#_Toc202018574)

[1.2. Problem Statement 4](#_Toc202018575)

[1.3. Objective 5](#_Toc202018576)

[1.4. Scope and Limitation 6](#_Toc202018577)

[1.5. Report Organization 8](#_Toc202018578)

[Chapter 2: Background Study and Literature Review 9](#_Toc202018579)

[2.1. Background Study 9](#_Toc202018580)

[2.2. Literature Review 10](#_Toc202018581)

[Chapter 3: System Analysis and Design 12](#_Toc202018582)

[3.1. System Analysis 12](#_Toc202018583)

[3.1.1. Requirement Analysis 12](#_Toc202018584)

[3.1.2. Feasibility Analysis 14](#_Toc202018585)

[3.1.3. Data Modeling (ER-Diagram Description) 16](#_Toc202018586)

[3.1.4. Process Modeling 17](#_Toc202018587)

[3.2 System Design 18](#_Toc202018588)

[3.2.1. Architectural Design 18](#_Toc202018589)

[3.2.2 Database Schema Design 22](#_Toc202018590)

[3.2.3. Interface Design 22](#_Toc202018591)

[3.2.4. Physical DFD 25](#_Toc202018592)

[Chapter 4. Implementation and Testing 27](#_Toc202018593)

[4.1. Implementation 27](#_Toc202018594)

[4.1.1. Tools Used 27](#_Toc202018595)

[4.1.2. Implementation Details of Modules 29](#_Toc202018596)

[4.2. Testing 32](#_Toc202018597)

[4.2.1. Test Cases for Unit Testing 32](#_Toc202018598)

[4.2.2. Test Cases for System Testing 34](#_Toc202018599)

[Chapter 5: Conclusion and Future Recommendations 36](#_Toc202018600)

[5.1. Lesson Learnt / Outcome 36](#_Toc202018601)

[5.2. Conclusion 38](#_Toc202018602)

[5.3. Future Recommendations 39](#_Toc202018603)

# Chapter 1: Introduction

## 1.1. Introduction

In today's digital world, most schools and colleges are moving towards online systems to manage their day-to-day tasks. However, in many educational institutions, the process of recording marks, preparing results, and informing students is still done manually. This often causes problems like calculation errors, delays, and confusion among students, teachers, and parents.

To solve this issue, I have developed an Online Result Management System. This system helps to manage students’ academic records in a smart and simple way. It allows teachers and admins to add students, enter marks, send notices, and view class results with just a few clicks. Students can also log in to view their results, notices, and updates from their teachers or school admin.

**The system is divided into three main parts:**

* Student Portal: Students can register and log in. After getting approval from the admin, they can view their exam results and personal notices. Teacher Portal: Teachers can log in using thes ID provided by the admin. They can add subjects, insert marks for students, and send notices to selected students or all students.
* Admin Portal: Admin has full control over the system. Admin can approve or delete student applications, create teacher accounts, view the total number of students and teachers, create classes, and send important notices.

The system is built using PHP for backend logic, MySQL for the database, and HTML, CSS, and JavaScript for the user interface. The design is mobile-friendly and clean, so users can access it comfortably from both computers and smartphones.

**With this system:**

* Results are stored safely and can be checked anytime.
* Students do not need to wait for physical report cards.
* Teachers save time and avoid calculation mistakes.
* Admin can manage everything from one place.

This project not only makes result handling easy but also supports the goal of digital education. It improves communication between students, teachers, and administrators and keeps everything well-organized and transparent.

## 1.2. Problem Statement

In many schools and colleges, student results are still prepared and managed manually using paper records or spreadsheets. This traditional method takes a lot of time and effort, especially when dealing with many students and multiple subjects. Teachers often face difficulties in calculating total marks, storing records safely, and updating or correcting results.

Moreover, students usually have to wait for days or weeks to get their results. If any error occurs in the marks or if someone loses the mark sheet, there is no easy way to fix or retrieve the data. Also, communication between teachers and students regarding performance or important notices is often slow and unorganized.

**Some of the main problems in the existing system are:**

* Time-consuming and repetitive manual work for teachers and admins
* Chances of human error in result calculation or entry
* No easy way for students to access their results online
* Poor record-keeping and risk of data loss
* Lack of a proper system for sending updates or notices to students

To overcome these problems, there is a clear need for a smart, secure, and easy-to-use **Online Result Management System** that helps manage academic records digitally and allows fast communication between students, teachers, and admins.

# 

# 1.3. Objective

The main goal of this project is to create a user-friendly and digital result management system that can be used by schools or colleges to manage academic results more easily and efficiently.

**The key objective of this system is:**

* **To build a online result automation system .**

This system will help reduce manual work, avoid result errors, save time, and improve communication between teachers, students, and the admin.

## 1.4. Scope and Limitation

The **Online Result Management System** is designed to digitally streamline the process of recording, managing, and delivering academic results in a secure and organized manner. This system primarily focuses on reducing manual workload, improving accuracy, and ensuring timely availability of results for students, teachers, and administrators.

**The system has three main user roles:**

1. **Admin Panel**:

* Admin has full control over the system.
* They can register, approve, or reject student applications.
* They can also create and manage teacher accounts, assign subjects to classes, and monitor the results being added.

1. **Teacher Panel**:

* Teachers can log in securely to their own dashboard.
* They can view the list of approved students, send personalized notices, and add subject-wise marks (theory and practical) for selected terms (First, Second, Final).
* Teachers also have access to only the data they are authorized to handle.

1. **Student Panel**:

* After approval from admin, students can log into their dashboard.
* They can view their subject-wise results, term-wise performance, and receive notices from teachers or admin.

**The system also includes other useful features:**

* Dynamic subject loading based on selected class/semester.
* Responsive design that works across mobile, tablet, and desktop.
* Real-time success and error messages for user actions.
* Protection of restricted pages through session management.

In summary, this system helps digitize the result process with accuracy, role-based access, and ease of use for all parties involved.

**Limitations of the Project**

While the system meets the core requirements of an educational institution’s result management, there are certain limitations in the current version:

1. **Limited to Internal Use Only**:

The system is designed for internal use within a single institution. It does not yet support multi-school or large university-level deployments where different faculties or campuses may need isolated modules.

1. **No Bulk Import/Export**:  
   There is no functionality yet to upload or download student marks via Excel/CSV files. Marks must be entered manually for each student.
2. **No Graphical Analysis**:

Although students can view their results, there are no charts or graphs to show academic progress visually over multiple terms.

1. **No Email/SMS Integration**:

Notices are sent within the system only. There is no built-in support for email or SMS alerts to notify students about results or messages.

1. **Single Language Support**:

The interface and system messages are in English only. Users who are not comfortable with English may face difficulty without localization or Nepali translation.

1. **No Real-time Validation of Data Conflicts**:

If a teacher unknowingly tries to add marks for the same student and subject again, a duplicate entry check prevents it, but there’s no alert system or automatic overwrite/prompt option.

1. **Security Measures Are Basic**:

The current version includes basic session validation for authentication. However, advanced features like token-based login, password recovery, or two-factor authentication are not implemented yet.

This system provides a solid foundation for managing academic results in a structured and digital way. With further development and integration of additional features, its usability and coverage can be expanded to handle larger, more complex educational environments.

## 1.5. Report Organization

This project report is systematically organized into several chapters to present a clear and comprehensive understanding of the Online Result Management System and its development process. Each chapter highlights a specific aspect of the project, starting from the conceptual foundation to the final implementation and conclusion.

**Chapter 1: Introduction**

This chapter introduces the project, presents the background of the study, outlines the problem statement, defines the objectives, explains the scope and limitations of the system, and provides an overview of how the report is organized.

**Chapter 2: Background Study and Literature Review**

This chapter discusses the basic theories, technologies, and concepts relevant to the project, including result management, database handling, and web systems. It also includes a literature review of existing similar systems and research that guided the development.

**Chapter 3: System Analysis and Design**

This section focuses on analyzing system requirements and the feasibility of the proposed solution. It includes functional and non-functional requirements, feasibility studies, data modeling through ER diagrams, and system process modeling using DFDs. The system design includes architecture, database schema, user interface designs, and physical data flow.

**Chapter 4: Implementation and Testing**

This chapter provides the technical implementation of the system, including the tools used and the development of key modules. It also covers the testing phase, including unit and system testing with test cases to ensure the application functions as intended.

**Chapter 5: Conclusion and Future Recommendations**

The final chapter summarizes the lessons learned throughout the project, reflects on the outcomes, and provides suggestions for future improvements or expansions of the system.

In addition to the above chapters, the report also includes supporting sections such as appendices (containing screenshots, source code snippets, and supervisor log sheets), references for the materials and technologies used, and a bibliography if applicable.

# Chapter 2: Background Study and Literature Review

## 2.1. Background Study

In recent years, educational institutions have increasingly adopted digital systems to improve academic management and streamline administrative tasks. Among these advancements, the **Online Result Management System (ORMS)** stands out as a crucial innovation for managing and publishing student performance data efficiently. Traditional result processing methods, which relied on handwritten mark sheets, physical registers, and manual data entry, were prone to errors, time-consuming, and lacked transparency. As the volume of students and complexity of academic programs grew, these manual systems became outdated and inefficient.

An Online Result Management System is a web-based platform designed to simplify and automate the process of recording, calculating, and publishing student results. It provides an interface for different users such as administrators, teachers, and students. Teachers can input marks directly into the system, which calculates and stores the results securely. Students can then log in to view their performance instantly, without having to visit the institution in person. Administrators can manage students, faculties, terms, and performance reports from a centralized dashboard.

The growing accessibility of internet-connected devices and the advancement of web technologies like **PHP, MySQL, JavaScript, and HTML/CSS** has made such systems highly feasible and affordable for institutions of all sizes. Moreover, using digital systems helps reduce paperwork, eliminate duplication, enhance security, and support data analytics for better decision-making.

The ORMS developed in this project is designed to meet these modern academic needs. It includes modules for student registration and approval, teacher management, marks entry, and result publication. Each feature is developed considering user roles and data privacy. It also offers notification features, a structured database schema, and a mobile-responsive interface for easy access. Understanding the background of existing systems and the underlying technologies used in this project provides a strong foundation for the design and implementation. The aim is not only to digitize but also to improve the accuracy, reliability, and speed of the overall academic result management process.

## 2.2. Literature Review

The development of an Online Result Management System (ORMS) is not a new concept; several academic institutions and software developers have explored similar systems to streamline the process of recording and publishing student results. This literature review highlights various relevant studies, tools, and technologies that have influenced the design and functionality of this project.

**1. Existing Manual Systems**

Traditional result publication methods involved paper-based evaluation and tabulation processes. While this approach was common in many schools and colleges, it had several limitations. Studies such as "Challenges in Manual Academic Record Management" (Sharma & Adhikari, 2018) point out how manual systems are time-consuming, prone to errors, and often delay result announcements. These issues provided the initial motivation for developing automated result management tools.

**2. Similar Web-Based Systems**

In the study titled "Web-Based Result Processing System for Universities" by Nwachukwu et al. (2016), a system was designed to allow teachers to upload marks and students to view their performance online. This system emphasized the importance of user roles, data validation, and secure authentication, which are also considered in our project.

Another research titled "Design and Implementation of Student Result Management System Using PHP and MySQL" by Rajan B. (2020) inspired the use of open-source web technologies for building affordable and scalable systems. His work showed how basic CRUD operations (Create, Read, Update, Delete) can be performed efficiently in academic systems using structured database tables and a clean user interface.

**3. Technological Frameworks**

Multiple sources support the use of **PHP** as a server-side scripting language due to its ease of use and wide support. “A Review on PHP-based Application Development” (Sapkota & Thapa, 2019) discusses how PHP integrates well with relational databases like **MySQL**, which is ideal for storing student records and result data.

Additionally, JavaScript and AJAX are commonly used for client-side interactivity such as live search, data filtering, and real-time notifications. Responsive design using CSS3 and HTML5 is also a key trend in modern systems to ensure accessibility across desktops, tablets, and smartphones.

**4. Security and Data Integrity**

Security concerns are discussed in various studies including “Securing Web-Based School Result Portals” by Olaitan et al. (2017), where it was emphasized that student data and result records must be stored securely and access must be restricted through authentication systems. This literature reinforced the importance of session management and user role validation, which are applied in our system.

**5. User Experience and Interface**

User interface design is another important aspect. According to Jakob Nielsen’s Usability Heuristics, systems should be easy to navigate, provide feedback, and minimize user errors. These principles were taken into account while designing the dashboards for admin, teacher, and student panels.

# ****Chapter 3: System Analysis and Design****

## ****3.1. System Analysis****

System analysis is the process of studying and understanding the components and requirements of a system. For the Online Result Management System, the goal is to create a platform that allows administrators to manage students and teachers, teachers to assign marks and send notices, and students to view results and receive updates securely and efficiently.

### ****3.1.1. Requirement Analysis****

Requirement analysis is critical in defining what the system is expected to do (functional) and how it should behave (non-functional).

#### ****i. Functional Requirements****

Functional requirements define the core functionalities and features of the system. These include the actions and tasks that each user type (admin, teacher, and student) can perform.

**Admin Functionalities:**

* Register and manage student applications.
* Approve or reject student registrations.
* Register and manage teacher accounts.
* View list of registered students and teachers.
* Maintain records of all results and user activity.
* Secure login/logout access to the admin dashboard.

**Teacher Functionalities:**

* Secure login/logout to the teacher dashboard.
* View list of approved students.
* Assign and submit theory/practical marks to students.
* Select semester, subject, and term while assigning marks.
* View previous result entries.
* Send notices to selected students via dashboard.
* Filter students by name/email/semester during notice dispatch.

**Student Functionalities:**

* Register for an account with personal details (name, photo, phone, email, address, parent’s contact).
* Login securely after approval from admin.
* View their personal dashboard.
* Access their subject-wise marks by semester and term.
* Read notices and updates from teachers.

**Use Case Summary (List Style):**

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **Use Case Description** | **Actor** |
| UC01 | Register as student | Student |
| UC02 | Approve student registration | Admin |
| UC03 | Login to dashboard | Admin/Teacher/Student |
| UC04 | Add teacher to system | Admin |
| UC05 | Add marks for student | Teacher |
| UC06 | View personal result | Student |
| UC07 | Send notice to students | Teacher |
| UC08 | View notices | Student |
| UC09 | View student and teacher lists | Admin |

**Table 1: Use case Summary**

#### ****ii. Non-Functional Requirements****

Non-functional requirements define how the system should behave, covering areas like performance, usability, reliability, and more.

**1. Usability:**

* The user interface must be simple and intuitive, requiring minimal training.
* All dashboards should be responsive and mobile-friendly.

**2. Reliability and Availability:**

* The system should be accessible at all times unless under maintenance.
* All operations (result submission, notices) should be processed without system crash or data corruption.

**3. Security:**

* Login sessions should be securely managed to prevent unauthorized access.
* Data must be protected from SQL injection, XSS, and file upload vulnerabilities.
* Passwords are securely stored (preferably hashed).

**4. Performance:**

* System should load pages and process operations within 2–3 seconds.
* Able to handle concurrent use by students and teachers.

**5. Maintainability:**

* Code should be modular and well-documented for easy maintenance.
* Admins should be able to update records manually via the dashboard.

**6. Scalability:**

* The system should support the growing number of students, teachers, subjects, and classes over time.

**7. Compatibility:**

* The system should be compatible with modern browsers (Chrome, Firefox, Edge).
* Works on both desktop and mobile devices (Android/iOS).

### ****3.1.2. Feasibility Analysis****

Feasibility analysis is performed to assess whether the proposed system is practical and achievable within the existing resources, time, and constraints. The following four types of feasibility are evaluated for the **Online Result Management System**:

#### ****i. Technical Feasibility****

* The system is technically feasible as it uses widely known and supported technologies:
* **Frontend:** HTML, CSS, JavaScript (responsive and accessible design)
* **Backend:** PHP
* **Database:** MySQL
* **Server:** XAMPP/LAMP/WAMP (can run locally or be deployed on hosting)
* No specialized hardware or rare software is required.
* Teachers, admins, and students need only a device with a browser and internet access

**Conclusion:**

Technically feasible with low infrastructure cost.

#### ****ii. Operational Feasibility****

* The system is easy to operate for users with basic digital literacy.
* The interface is clean, user-friendly, and mobile responsive.
* The login-based access system ensures that roles (admin, teacher, student) see only relevant content.
* The workflow (mark entry, student approval, notice sending) mirrors real-world practices.

**Conclusion:**

Operationally feasible and improves efficiency for all stakeholders.

#### ****iii. Economic Feasibility****

* This system is **cost-effective**:
* No expensive licenses or proprietary software are needed.
* Uses open-source tools and frameworks.
* Development cost is minimized by using standard PHP and MySQL.
* Long-term savings include:
* Reduced use of paper and manual result handling.
* Decreased communication delays between students and teachers.

**Conclusion:**

Economically feasible with very low development and maintenance cost.

#### ****iv. Schedule Feasibility****

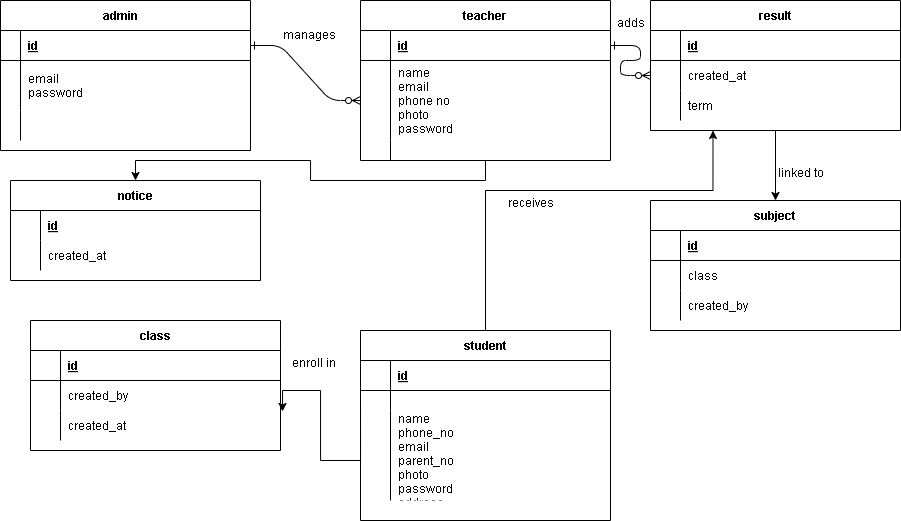
* The system can be developed and implemented within the academic schedule of the BCA 4th Semester Project I.
* The modules are divided logically:
* Admin portal (student and teacher management)
* Teacher portal (result and notice handling)
* Student portal (result viewing)
* Each module is manageable within a standard semester timeline (~3 months).

**Conclusion:**

Schedule is realistic and achievable within the given timeframe.

### ****3.1.3. Data Modeling (ER-Diagram Description)****

Entity Relationship (ER) modeling is used to visually represent the structure and relationships between different entities in the **Online Result Management System**. The ER diagram helps in understanding how data flows and is stored in the system's database.

****

**Fig 1. ER Diagram (Entity-Relationship Diagram)**

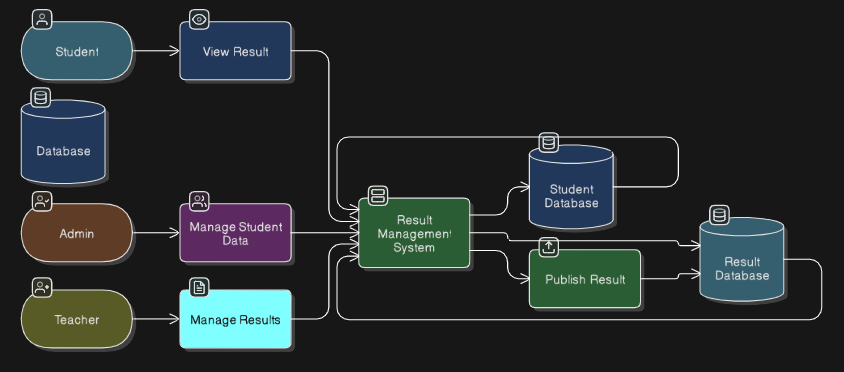
**Relationships**

* A **Student** can be linked to multiple **Results**.
* A **Teacher** can send multiple **Notices** and add multiple **Results**.
* A **Subject** belongs to a **Class** and is related to multiple **Results**.
* A **Result** is a many-to-many relationship among **Student**, **Subject**, and **Teacher**.
* A **Notice** is targeted to either a single student or multiple students (or all).
* **Admin** manages approval of both **Students** and **Teachers**.

**ER Diagram Summary (Textual Form)**

* **One** Class → **Many** Subjects
* **One** Student → **Many** Results
* **One** Teacher → **Many** Notices and Results
* **One** Subject → **Many** Results

### 3.1.4. Process Modeling



**Fig 2. Process Modelling**

## 3.2 System Design

### 3.2.1. Architectural Design

The architectural design of the Online Result Management System follows a **three-tier architecture** that separates the system into three main layers: the **Presentation Layer**, the **Application Layer**, and the **Data Layer**. This design promotes modularity, scalability, and maintainability of the entire system.

**1. Presentation Layer (Front-End)**

This layer provides the user interface through which users interact with the system. It includes all the **HTML**, **CSS**, and basic **JavaScript** that renders the interface for:

* Student registration and login
* Teacher login and dashboard
* Admin login and administrative panels

Each user role has access to a dedicated portal tailored to their functionalities:

* **Students**: View notices, results, and profile.
* **Teachers**: Add subjects, manage student marks, view submitted notices and result history.
* **Admins**: Approve student registrations, create teacher accounts, assign classes, and send notices.

**2. Application Layer (Logic Layer / Back-End)**

The core business logic of the system is handled in this layer using **PHP**. It processes user input, performs necessary validations, manages sessions, and communicates with the database to:

* Authenticate users
* Register students and teachers
* Assign results and subjects
* Approve or reject students
* Send targeted notices

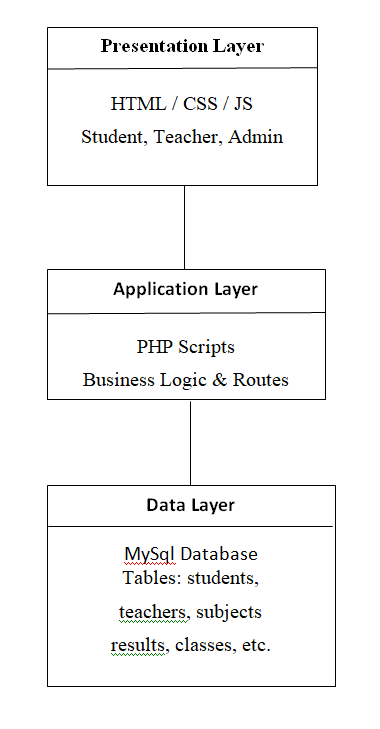
This layer is responsible for ensuring that operations performed by the users are valid and consistent with the system rules.

**3. Data Layer (Database Layer)**

The data layer uses **MySQL** to store and manage the system's data. It maintains separate tables for:

* Students (personal details, approval status)
* Teachers (credentials and contact information)
* Classes or faculties (class names like BCA 1st Sem, Class 10, etc.)
* Subjects (assigned by teachers for specific classes)
* Results (theory and practical marks submitted by teachers)
* Notices (admin communication with students or teachers)

All interactions with the database are done via SQL queries using the PHP mysqli interface. This layer ensures **data persistence, consistency, and integrity**.

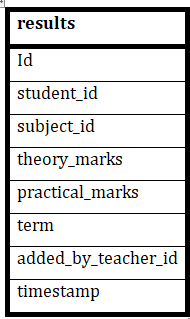
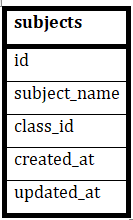
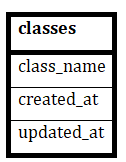
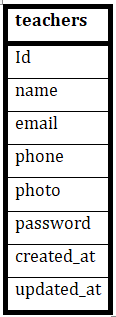
**System Architecture Diagram (Conceptual Overview)**

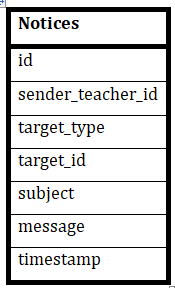
**Fig 3: System Architecture Diagram**

**Justification of Architectural Choice**

* **Modular Development**: Separating presentation, logic, and data layers allows independent development and easier maintenance.
* **Scalability**: The three-tier model supports future enhancements, such as integrating APIs or migrating to frameworks like Laravel.
* **Security**: Keeping logic and database access separate from the user interface helps in minimizing direct exposure of sensitive operations.
* **User Role Separation**: Each role (Admin, Teacher, Student) operates within clearly defined boundaries, reducing complexity.

### 3.2.2 Database Schema Design





**Fig 4. Database Schema**

### 3.2.3. Interface Design

The interface design of the **Online Result Management System** focuses on simplicity, clarity, and role-based navigation for three types of users: **Students**, **Teachers**, and **Admins**. Each interface is designed to be intuitive, ensuring that users can access the required functionalities with ease.

The system follows a **structured and consistent UI layout**, comprising menus, tables, forms, and dashboards. It uses HTML and CSS to create a clean, responsive layout, with specific panels and tabs allocated for each functional module.

**User Interface Structure Overview**

1. **Index Page (Home Interface)**

Entry point of the system.

Contains:

* Student Registration Link
* Login Form (Role selector: Admin / Teacher / Student)

2. **Student Interface**

**Student Dashboard:**

* View personal profile.
* View class results (theory + practical).
* View notices from Admin or Teachers.

**Design Elements:**

* Vertical navigation bar (or top navbar).
* Card-style or tabular result display.
* Notices shown with timestamp and subject.

**Teacher Dashboard:**

* Add new subjects by selecting class.
* Assign marks to students by selecting subject/class.
* View student details.
* View sent results and notices to avoid duplication.

**Design Elements:**

* Class/Subject selection dropdowns.
* Table to list all students in a class.
* Result input form (theory, practical).
* Result history panel.

4. **Admin Interface**

**Admin Dashboard:**

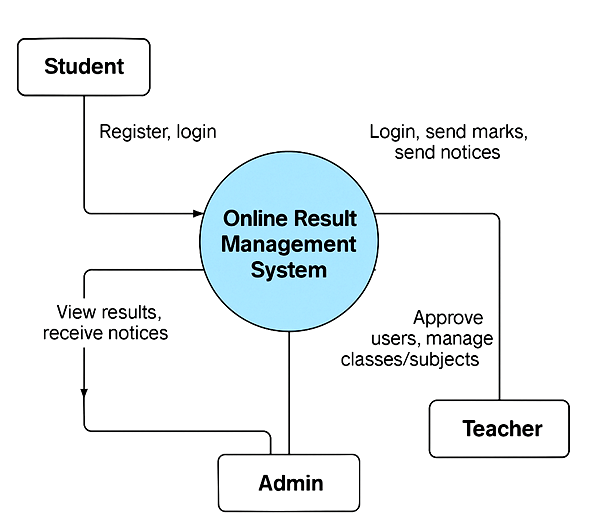
* Approve/Delete student applications.
* Add new teacher details.
* Manage classes/faculty.
* Send notices to teachers/students.
* View summary:
* Total Approved Students
* Total Teachers

**Design Elements:**

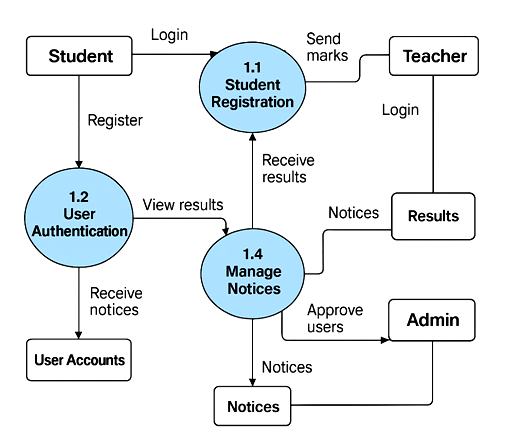
* Table layout for student/teacher lists.
* Simple approval/delete buttons.
* Notice form with dropdown selector.

**Example Screenshot Suggestions**

* Admin Dashboard (with tabs for student/teacher/faculty/notice)
* Teacher marks entry form
* Student result display and notices
* Registration form

3.2.4. Physical DFD

**Fig 5. level 0 DFD**



**Fig 6. level 1 DFD**

# Chapter 4. Implementation and Testing

## 4.1. Implementation

### 4.1.1. Tools Used

This section outlines the various tools, programming languages, platforms, and software used in the development of the **Online Result Management System**. These tools were selected based on ease of use, availability, platform compatibility, and suitability for a web-based application.

**CASE Tools (Computer-Aided Software Engineering Tools)**

|  |  |
| --- | --- |
| **Tool Name** | **Purpose** |
| **Draw.io** | Used to design Data Flow Diagrams (DFD), ER Diagrams, and Architecture Diagrams. |
| **Lucidchart** | For professional-looking system models and UML diagrams. |
| **Microsoft Word** | Documentation and formatting of the final project report. |
| **Google Sheets** | Temporary planning and organizing data structure before DB design. |

**Table 2. Case Tools**

**Programming Languages**

|  |  |
| --- | --- |
| **Language** | **Purpose** |
| **HTML** | For creating the structure and content of the web pages (forms, tables). |
| **CSS** | For styling and designing the web pages to make them visually appealing and responsive. |
| **PHP** | Server-side scripting to handle user sessions, database interactions, and core logic. |
| **JavaScript** (minimal) | Basic interactivity (like dropdown auto-submit, alert confirmations). |

**Table 3. Programming language**

**Database Platform**

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **MySQL** | Used as the relational database to store all system data such as students, teachers, classes, subjects, results, and notices. |
| **phpMyAdmin** | GUI tool for managing MySQL databases, table creation, and data manipulation. |

**Table 4. Database Platform**

**Development Environment**

|  |  |
| --- | --- |
| **Tool/Platform** | **Purpose** |
| **XAMPP** | A local server environment including Apache, PHP, and MySQL. |
| **VS Code** | Source code editor for writing and managing PHP, HTML, and CSS code. |
| **Web Browser** (Chrome / Firefox) | Testing and running the web application during development. |

**Table 5. Development Environment**

**Security Practices Used**

|  |  |
| --- | --- |
| Feature | Description |
| **Session Handling** | PHP session management for role-based login (admin, teacher, student). |
| **Password Storage** | Basic password handling, can be enhanced with password\_hash() for production. |
| **Form Validation** | HTML5 and basic PHP validation to reduce invalid input. |

**Table 6. Security Practices Used**

### 4.1.2. Implementation Details of Modules

This section explains the implementation logic, procedures, and core functions developed for each module of the Online Result Management System. The system is built using **PHP for server-side scripting**, **HTML/CSS for frontend**, and **MySQL** as the backend database.

**The system consists of three main modules:**

* **Admin Module**
* **Teacher Module**
* **Student Module**

Each module has a dedicated dashboard with role-based access and functionalities.

**Authentication Module (Login and Registration)**

**File(s):** login.php, register.php  
**Purpose:** Allow users to register (students) and login (all users) securely.

**Functions:**

* **studentRegistration():** Handles student registration and stores the submitted form data in the students table with approval set to false by default.
* **authenticateUser():** Validates login credentials based on selected role (student/teacher/admin) and sets PHP sessions.
* **session\_start():** Used across pages to validate sessions and prevent unauthorized access.

**Admin Module**

**File(s):** admin/dashboard.php, manage\_students.php, create\_teacher.php, manage\_faculty.php, send\_notice.php

**Key Functions and Procedures:**

* **approveStudent($id):** Updates a student's status from unapproved to approved (approved = 1).
* **deleteStudent($id):** Deletes a student record permanently from the database.
* **createTeacher($details):** Accepts form data and inserts new teacher records into the teachers table.
* **addClass($className):** Adds new class/faculty entries into the classes table.
* **sendNotice($recipientType, $recipientId, $subject**): Inserts a notice into the notices table targeted to students or teachers.

**Interface Features:**

* View list of students and approve/delete them.
* View and create teacher accounts.
* Create/manage faculty (like Class 10, BCA 1st Sem, etc.).
* Send individual notices to students or teachers using dropdown lists.

**Teacher Module**

**File(s):** teacher/dashboard.php, add\_result.php, view\_students.php

**Key Functions and Procedures:**

* addSubject($classId, $subjectName, $maxTheory, $maxPractical): Inserts new subject entries into the subjects table linked to the teacher and selected class.
* viewStudentsByClass($classId): Retrieves a list of students enrolled in a specific class.
* assignMarks($studentId, $subjectId, $classId, $marks): Stores marks into the results table with timestamp and teacher ID.
* trackTeacherHistory($teacherId): Shows the notices and results already sent by the teacher to avoid repetition.

**Interface Features:**

* Subject creation form with class and marks input.
* Student list view by class.
* Marks submission form (theory and practical).
* History of sent results and notices.

**Student Module**

**File(s):** student/dashboard.php

**Key Functions and Procedures:**

* viewNotices($studentId): Retrieves all notices addressed to the logged-in student with time of creation.
* viewResults($studentId): Displays marks for all subjects the student is assigned, grouped by class and subject.
* getProfileData($studentId): Loads the student's registered information from the database.

**Interface Features:**

Student dashboard showing:

* Recent notices
* Result history table (subject-wise, class-wise)
* Profile section

**Database Communication Procedures**

* **Connection** is established via db/connection.php using the mysqli object.
* All **queries** use mysqli\_query() or prepared statements to reduce risk of SQL injection.
* **Form handling** and file uploads are processed using $\_POST and $\_FILES arrays.
* **Session-based redirects** ensure security and correct user navigation.

## 4.2. Testing

### 4.2.1. Test Cases for Unit Testing

Unit testing involves testing individual components or functions of the system to ensure that each part works as expected. The following test cases have been designed to validate the critical units (modules and functions) of the Online Result Management System, such as user login, registration, subject creation, and result submission.

**Test Case Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Module** | **Test Description** | **Input** | **Expected Output** | **Result** |
| TC001 | Registration | Register new student with valid data | Name, Email, Phone, Address, Photo | "Registration submitted for approval" message | Pass |
| TC002 | Registration | Register with missing required fields | Missing email | Error or no submission | Pass |
| TC003 | Login | Valid login for approved student | Email: valid, Password: correct, Role: student | Redirect to student dashboard | Pass |
| TC004 | Login | Attempt login before admin approval | Email: valid, Password: correct | "Invalid or unapproved login" message | Pass |
| TC005 | Login | Invalid login credentials | Wrong password | "Invalid login" message | Pass |
| TC006 | Admin Approve | Admin approves pending student | Approve button click | Status updated to approved | Pass |
| TC007 | Admin Delete | Admin deletes student from list | Delete button click | Student record removed | Pass |
| TC008 | Teacher Creation | Admin creates new teacher | Valid name, email, phone, photo | Teacher added successfully | Pass |
| TC009 | Faculty Creation | Admin adds a new class/faculty | Class name = "BCA 1st Sem" | Class inserted into database | Pass |
| TC010 | Notice Sending | Admin sends notice to teacher | Recipient = teacher ID, message = text | Notice saved and shown in teacher dashboard | Pass |
| TC011 | Subject Addition | Teacher adds subject for a class | Subject = Math, Class = BCA, Max Marks input | Subject saved under that teacher’s profile | Pass |
| TC012 | Result Submission | Teacher submits marks for a student | Student, Subject, Marks | Marks stored in results table | Pass |
| TC013 | Result View | Student views their result | Student login | Subject-wise result table is shown | Pass |
| TC014 | Notice View | Student views notices from admin/teachers | Student login | Notice list with timestamps | Pass |
| TC015 | Redundant Result | Teacher tries to send result again for same subject | Already submitted marks | Prevent duplication / overwrite warning | Pass |

**Table 7. Test Case**

**Testing Methodology**

* **Manual Testing** was performed using form inputs, database queries, and result verification on screen.
* **Data Validation** was done to check if inputs were correctly stored in the database.
* **Session Testing** confirmed that pages were protected and role-specific.
* **Boundary Testing** (e.g., empty fields, incorrect types) was also conducted.

### 4.2.2. Test Cases for System Testing

System testing validates the entire **Online Result Management System** as a complete and integrated solution. It ensures that all modules—Admin, Teacher, and Student—work together smoothly and meet the functional requirements of the system. The focus is on the **end-to-end workflows**, including user navigation, data flow between modules, and system behavior under various conditions.

**System Testing Test Case Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Test Steps** | **Expected Outcome** | **Result** |
| ST001 | End-to-end student registration to login | Student fills registration form → Admin approves → Student logs in | Student accesses dashboard after approval | Pass |
| ST002 | Admin sends notice to student | Admin logs in → Selects student → Sends notice → Student logs in | Notice appears in student dashboard | Pass |
| ST003 | Admin sends notice to teacher | Admin logs in → Selects teacher → Sends notice → Teacher logs in | Notice appears in teacher dashboard | Pass |
| ST004 | Teacher adds subject for a class | Teacher logs in → Selects class → Enters subject and marks range | Subject is saved and linked to class and teacher | Pass |
| ST005 | Teacher assigns marks to student | Teacher selects class → Chooses student and subject → Submits marks | Result saved and appears on student portal | Pass |
| ST006 | Student views result | Student logs in → Navigates to results section | Sees list of subjects with theory and practical marks | Pass |
| ST007 | Teacher views previously submitted results | Teacher logs in → Opens result history | Sees list of submitted results with student details | Pass |
| ST008 | Admin dashboard summary | Admin logs in → Views dashboard summary (total students, teachers, faculties) | Accurate and updated counts displayed | Pass |
| ST009 | Duplicate result prevention | Teacher tries to resubmit the same result again | System prevents duplication or gives a warning | Pass |
| ST010 | Unauthorized access to dashboard | Try accessing teacher/student/admin dashboards without login | Redirects to login page | Pass |

**Table 8. System testing test case**

**System Testing Focus Areas**

* **Data flow consistency** between modules (e.g., student registration → approval → result)
* **Session management** and access control for each role
* **Form submissions and validations**
* **Database updates** reflected across user interfaces
* **Error handling** for invalid or unexpected inputs

**Testing Methodology**

* **Black-box testing** method was used.
* No internal code was assumed; test cases were written solely based on system behavior and requirements.
* **Cross-role integration** was tested: Admin → Teacher → Student.
* **UI and functional verification** were performed simultaneously.

# Chapter 5: Conclusion and Future Recommendations

## 5.1. Lesson Learnt / Outcome

The development of the **Online Result Management System** has been a significant learning experience both technically and analytically. Throughout the project, multiple real-world software development practices were applied, leading to valuable insights and outcomes:

**1. Practical Understanding of Web Development**

* Gained hands-on experience with **front-end technologies** like HTML and CSS for designing user-friendly interfaces.
* Understood the application of **PHP** as a server-side scripting language for managing sessions, form submissions, and database operations.
* Implemented **MySQL** databases to manage complex relational data structures and perform CRUD operations effectively.

**2. Role-Based Access Management**

* Learned to create and manage **multi-user systems** with different privileges for **admins**, **teachers**, and **students**.
* Designed secure login systems using session handling to restrict access and protect sensitive data.

**3. Database Design and Integration**

* Gained experience in **designing normalized relational databases**.
* Created interlinked tables for handling student records, results, teachers, classes, and notices.
* Learned to use **phpMyAdmin** for managing and testing database transactions.

**4. Modular and Scalable Code Structure**

* Implemented the system using a **modular approach**, breaking the project into reusable components (registration, login, dashboard, etc.).
* Achieved **clean and scalable architecture** through separation of concerns (presentation layer, application layer, and data layer).

**5. Testing and Debugging Skills**

* Understood the importance of **unit and system testing**.
* Performed validation of functionalities and fixed issues related to form handling, session control, and data rendering.

**6. Documentation and Reporting**

* Learned how to **document** system features, architecture, implementation details, and testing procedures in a structured format.
* Developed the ability to communicate technical ideas clearly through diagrams and structured reports.

**Overall Outcome**

By completing this project, I gained a complete overview of how real-world result management systems are designed and implemented. It helped strengthen both my **technical skills in programming and database handling**, and **analytical skills in planning, designing, and testing** a working software solution. This project has laid a solid foundation for developing more advanced and professional web applications in the future.

## 5.2. Conclusion

The development of the **Online Result Management System** successfully addresses the need for a streamlined, efficient, and user-friendly platform for managing student results, teacher data, and administrative tasks. This system automates critical processes such as student registration, result entry, approval workflows, and notice dissemination, reducing manual effort and minimizing errors.

Through this project, a robust web-based application was created using PHP, MySQL, HTML, and CSS, incorporating multi-role access control for students, teachers, and administrators. The system ensures data integrity, security, and ease of use for all stakeholders.

Overall, the project has demonstrated the practical application of software engineering principles, database management, and web technologies to solve real-world educational management challenges. It lays a strong foundation for further enhancements and integration with other educational systems.

## 5.3. Future Recommendations

While the current system fulfills its intended functions, several enhancements can be considered to improve its usability, security, and scalability:

* **Enhanced Security**: Implement advanced security measures such as password hashing (e.g., using password\_hash() in PHP), input sanitization to prevent SQL injection, and HTTPS for secure communication.
* **Responsive and Modern UI/UX**: Upgrade the interface using modern frontend frameworks like React or Vue.js for a more dynamic and responsive user experience.
* **Role-based Notifications**: Add automated email or SMS notifications to alert students and teachers about notices, result uploads, or approvals.
* **Mobile Application**: Develop a mobile app version for easier access and notifications on smartphones.