PROBLEM STATEMENT

Educational institutions handle a large volume of student result data every semester. Traditionally, the process of result preparation, verification, storage, and publication is done manually or through disconnected spreadsheets. This leads to several issues:

- **Time-consuming manual work** for faculty and examination departments.
- **High chances of calculation mistakes** in total marks, grades, and SGPA/CGPA.
- **Difficulty in maintaining historical records** and retrieving past results quickly.
- **Lack of secure access** for students to view results individually.
- **Limited transparency** in result modifications, re-evaluation, and record updates.

There is a clear need for a centralized, secure, and efficient **Student Result Management System** that:

- Automates mark entry and grade calculation.
- Stores all academic records in a structured database.
- Provides role-based access for administrators, faculty, and students.
- Allows students to easily access their semester-wise results anytime, from anywhere.

The proposed system aims to solve these problems by designing and implementing a **web-based result management application** using **PHP and MySQL**.

INTRODUCTION

The **Student Result Management System (SRMS)** is a web-based application that digitizes the entire lifecycle of student result processing in an academic institution. From student registration and subject allocation to marks entry and result publication, the system provides an integrated platform for managing all result-related activities.

In many institutions, examination data is stored in Excel sheets or paper-based records. As the number of students, courses, and semesters grows, maintaining accuracy, consistency, and security of result records becomes a major challenge. In addition, students often rely on notice boards or printed mark sheets to check their results, which is inefficient and slow.

The SRMS addresses these issues by:

- Providing **secure login** for administrators, faculty, and students.
- Enabling **faculty** to enter and update internal and external marks directly through the system.
- Allowing **administrators** to manage students, subjects, semesters, and result approval.
- Allowing **students** to view their results in a simple and user-friendly interface.
- Ensuring that results are **calculated automatically** and stored centrally in a database.

The system is developed using:

- **Frontend:** HTML5, CSS3, JavaScript
- **Backend:** PHP
- **Database:** MySQL

This combination ensures a robust, scalable, and platform-independent solution suitable for deployment in colleges and universities.

OBJECTIVES

7.1 General Objectives

1. To design and develop a **centralized web-based system** for managing student academic results.
2. To **automate the process** of result calculation, storage, and retrieval.
3. To **improve accuracy** by minimizing manual intervention and human errors.
4. To **provide secure role-based access** to administrators, faculty, and students.
5. To **enhance transparency and accessibility** of result-related information.

7.2 Specific Objectives

1. To create a **user authentication system** (Admin / Faculty / Student) with secure login and logout.
2. To design modules for **student registration**, **subject allocation**, and **semester mapping**.
3. To provide interfaces for **faculty to enter internal, external, and practical marks**.
4. To implement **automatic grade and result calculation** based on predefined rules (e.g., percentage, grade points).
5. To allow **students to view their individual results** in a secure and user-friendly manner.
6. To maintain **historical result records** for multiple semesters and academic years.
7. To generate **printable result sheets and summary reports** for departments and examination cells.
8. To provide options for **re-evaluation, result update, and audit trail** (who updated what and when).

SCOPE AND LIMITATIONS

8.1 Scope

- The system is designed for **colleges and universities** offering multiple courses, sections, and semesters.
- It supports **multi-semester result records** for each student.
- It includes **modules for Admin, Faculty, and Student** roles.
- It manages:
    - Student profiles
    - Course and subject details
    - Marks entry and result calculation
    - Result viewing and printing
- The system can be extended to integrate with:
    - Attendance systems
    - Fee management systems
    - ERP or student information systems
    - Mobile applications for students

8.2 Limitations

- The system currently focuses on **result management only**, not full ERP.
- The grading rules (e.g., CGPA calculation) are based on **institution-specific policies** and may require customization.
- It assumes **reliable internet and server availability** for remote access.
- Bulk data migration from old systems must be done manually or through custom scripts.
- Advanced analytics (like performance prediction, AI-based analysis) are not part of the initial version but can be added later.

LITERATURE REVIEW / REFERENCES SUMMARY

A brief summary of related work and existing systems:

1. **Existing Result Management Software (Proprietary Systems)**

   Many commercial result management solutions provide features like grade entry, report generation, and transcript printing. However, they are often expensive, closed-source, and difficult to customize for college-specific workflows.

2. **Spreadsheet-based Systems**

   Traditionally, institutions use Excel or similar tools to calculate totals and grades. While easy to start with, spreadsheets are error-prone, difficult to maintain for large datasets, and lack proper security and access control.

3. **Web-based Academic Portals**

   Several universities have adopted web-based portals allowing students to view results, attendance, and notices. These portals typically use database-driven systems and role-based access management, forming the basis for this project's design.

4. **Research Papers and Journals**

   - Studies on **Education Information Systems** highlight the importance of centralizing academic data and providing timely access to stakeholders.
   - Research on **Database Management Systems** supports the use of relational databases like MySQL for storing structured academic data reliably.
   - Papers on **Human-Computer Interaction (HCI)** emphasize creating user-friendly interfaces to reduce learning curves for faculty and students.

5. **Open-Source Projects and GitHub Repositories**

   Several open-source result/grade management projects demonstrate practical implementations using PHP and MySQL, offering ideas for database schema design, validation mechanisms, and UI patterns.

In conclusion, existing solutions show the **need and feasibility** of a web-based result management system. However, a **custom, institution-specific implementation** provides more flexibility, control, and integration with local academic policies.

SYSTEM REQUIREMENTS

10.1 Hardware Requirements

For Development Machine (Developer System):

- Processor: Intel Core i5 or AMD equivalent (minimum)
- RAM: 8 GB (minimum)
- Storage: 256 GB SSD (minimum)
- Display: Standard HD monitor
- Peripherals: Keyboard, mouse, internet connection

For Web Server (Deployment Environment):

- Processor: Dual-core or higher
- RAM: 4 GB or higher
- Storage: 100 GB SSD (depending on number of students & years)
- Network: Stable broadband/ LAN connectivity

10.2 Software Requirements

Operating System:

- Windows / Linux / macOS (for development)
- Linux (Ubuntu / CentOS) recommended for production server

Server Stack:

- Web Server: Apache (XAMPP / WAMP / LAMP)
- Backend: PHP 7.4 or above
- Database: MySQL 5.7 or above / MariaDB

Development Tools:

- Code Editor / IDE: Visual Studio Code / Sublime Text / PHPStorm
- Database Tools: phpMyAdmin / MySQL Workbench
- Version Control: Git, GitHub repository

Browser:

- Google Chrome / Mozilla Firefox / Microsoft Edge

Other Tools (Optional):

- Figma / Canva – for UI mockups
- Postman – for API testing (if REST endpoints are used)

SYSTEM DESIGN OVERVIEW (ARCHITECTURE + TECHNOLOGIES)

11.1 System Architecture

The **Student Result Management System** follows a **three-tier architecture**:

1. **Presentation Layer (Client Side):**
    - Technologies: HTML5, CSS3, JavaScript
    - Purpose: User interfaces for admin, faculty, and students; forms for login, marks entry, and result viewing.
2. **Application Layer (Server Side):**
    - Technology: PHP
    - Purpose: Handles business logic, form processing, validation, authentication, session management, and communication with the database.
3. **Data Layer (Database):**
    - Technology: MySQL
    - Purpose: Stores student profiles, subjects, marks, results, user credentials, and logs.

11.2 Key Technologies

- **HTML5 & CSS3:** Structure and styling of web pages.
- **JavaScript:** Client-side validation, dynamic behavior, and improved user experience.
- **PHP:** Backend scripting for form processing, database operations, and role-based access.
- **MySQL:** Relational database storing normalized academic data.

ER DIAGRAM EXPLANATION & TABLE DESCRIPTIONS

The ER model for SRMS consists of the following main entities:

1. **User**
2. **Student**
3. **Course**
4. **Subject**
5. **Semester**
6. **Result**
7. **Exam / Assessment**

12.1 Entity Explanations

1. **User**
    - Represents login-level user accounts (Admin / Faculty / Student).
    - Attributes: user_id, username, password_hash, role, email, created_at.
2. **Student**
    - Stores student profile information and links to a user account.
    - Attributes: student_id, user_id, roll_no, name, course_id, semester_id, dob, contact, address.
3. **Course**
    - Represents academic programs like BCA, B.Tech, etc.
    - Attributes: course_id, course_name, course_code, duration.
4. **Semester**
    - Represents academic semester levels.
    - Attributes: semester_id, course_id, semester_no, session_year.
5. **Subject**
    - Represents subjects taught in each semester.
    - Attributes: subject_id, subject_name, subject_code, course_id, semester_id, max_marks_internal, max_marks_external.
6. **Result**
    - Stores marks and calculated grades for students in each subject.

- Attributes: result_id, student_id, subject_id, internal_marks, external_marks, total_marks, grade, status (Pass / Fail).

7. **Exam / Assessment (optional)**
    - Used if multiple exams per semester are recorded (mid-term, end-term).
    - Attributes: exam_id, exam_type, exam_date, semester_id.

12.2 Sample Table Descriptions

Table: users

- user_id (PK, INT, AUTO_INCREMENT)
- username (VARCHAR, UNIQUE)
- password_hash (VARCHAR)
- role (ENUM: 'admin', 'faculty', 'student')
- email (VARCHAR)
- created_at (TIMESTAMP)

Table: students

- student_id (PK, INT, AUTO_INCREMENT)
- user_id (FK → users.user_id)
- roll_no (VARCHAR, UNIQUE)
- name (VARCHAR)
- course_id (FK → courses.course_id)
- semester_id (FK → semesters.semester_id)
- dob (DATE)
- contact (VARCHAR)
- address (TEXT)

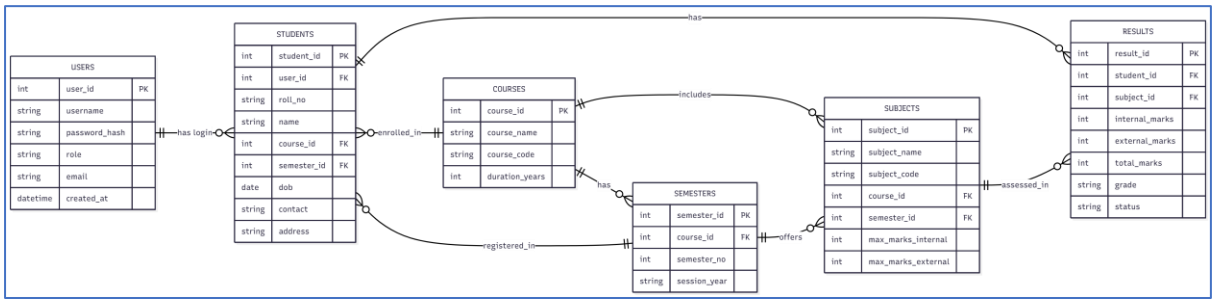
Table: subjects

- subject_id (PK)
- subject_name
- subject_code
- course_id (FK)
- semester_id (FK)
- max_marks_internal
- max_marks_external

Table: results

- result_id (PK)
- student_id (FK)
- subject_id (FK)
- internal_marks
- external_marks
- total_marks
- grade
- status

This ER structure ensures **data normalization**, **consistency**, and **easy retrieval** of result-related information.



DATA FLOW DIAGRAM (DFD) EXPLANATION

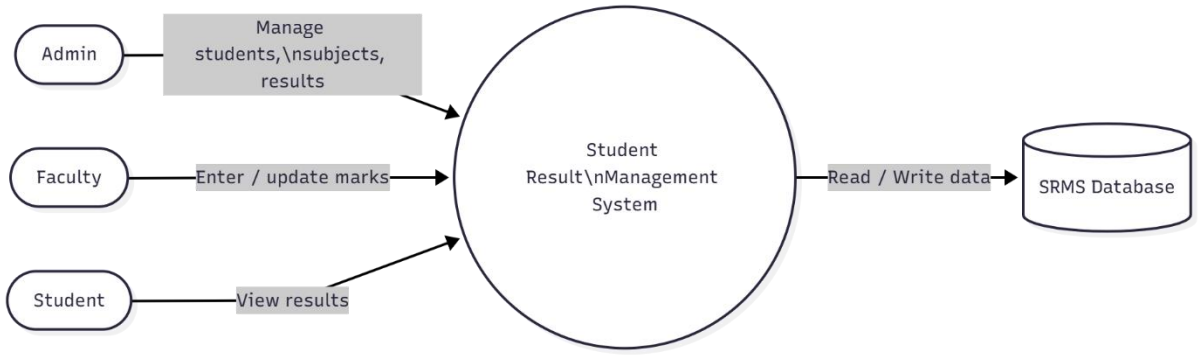
13.1 Context Level Diagram (Level 0)

At the highest level, the **Student Result Management System** interacts with three main external entities:

1. **Admin**
2. **Faculty**
3. **Student**

- Admin manages courses, subjects, students, and users.
- Faculty submits and updates marks.
- Students view results.

All interactions pass through the **SRMS**, which reads/writes data to the database.



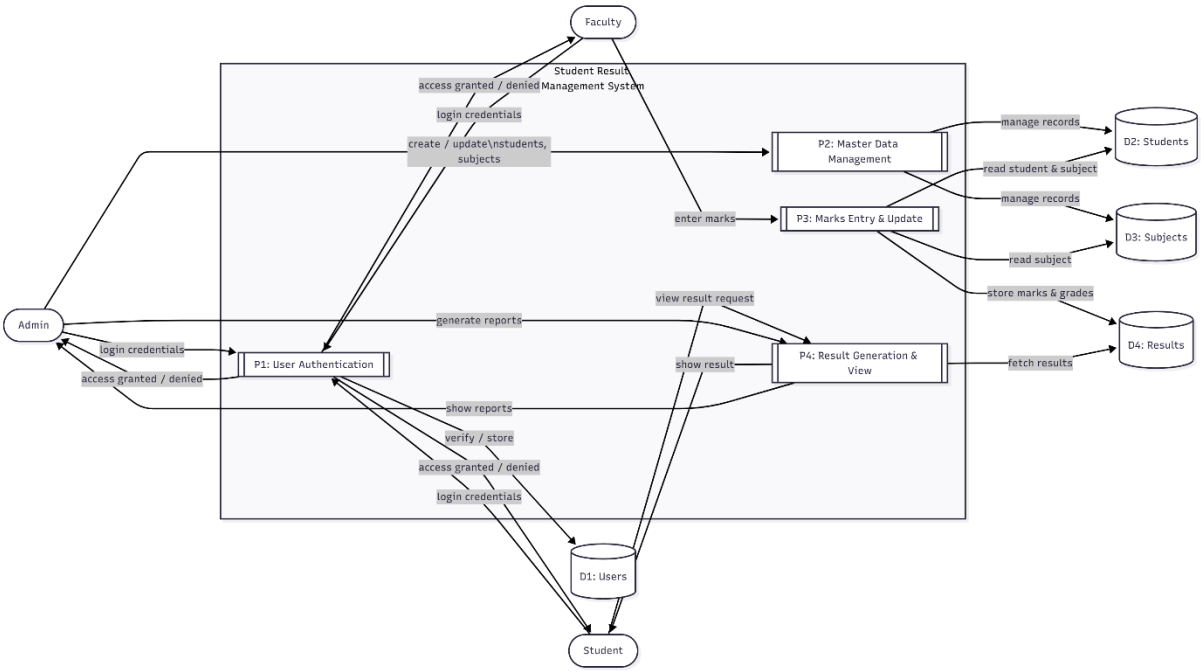
13.2 Level 1 DFD

Processes may include:

1. **User Authentication**
    - Inputs: Login credentials (username, password).
    - Outputs: Access to dashboard or error message.
    - Data Store: users.
2. **Student & Subject Management (Admin)**
    - Inputs: Student data, course data, subject details.
    - Processes: Add, edit, delete operations.
    - Data Stores: students, courses, subjects.
3. **Marks Entry (Faculty)**

- Inputs: Marks per student per subject.
- Processes: Validate and save marks.
- Data Store: results.

4. **Result Generation & Viewing (Student / Admin)**
   - Inputs: Student selection / login.
   - Processes: Fetch results, calculate grades (if needed), and display.
   - Data Store: results, subjects, students.



MODULE-WISE FUNCTIONAL SPECIFICATIONS

14.1 Admin Module

Responsibilities:

- Manage user accounts (create faculty/student logins).
- Manage courses, semesters, and subject masters.
- Approve or lock result entries after verification.
- Generate reports like class-wise and subject-wise results.

Key Functions:

- Admin Login
- Add/Edit/Delete Student
- Add/Edit/Delete Subject
- Assign Subjects to Faculty
- Lock/Unlock Result for a semester

14.2 Faculty Module

Responsibilities:

- Enter and update marks for assigned subjects.
- View student lists for their subjects.
- Generate subject-wise performance lists.

Key Functions:

- Faculty Login
- View Assigned Subjects
- Enter Internal & External Marks
- Edit Marks before result lock
- View Subject Result Summary

14.3 Student Module

Responsibilities:

- View personal profile and semester-wise results.

- Download/print mark sheets.

Key Functions:

- Student Login
- View Profile
- View Subject-wise Results
- View SGPA/Overall Status
- Print Result / Download as PDF (optional feature)

14.4 Optional Support / Examination Cell Module

- Manage academic sessions and exam cycles.
- Handle re-evaluation and result modification requests.
- Generate consolidated result analysis reports.

UI/UX GUIDELINES

To ensure a consistent and user-friendly interface:

1. **Color Scheme:**
    - Use a **clean light background** (white / light grey).
    - Use **institutional colors** like blue/navy for headers and buttons.
2. **Typography:**
    - Use simple fonts like **Roboto**, **Open Sans**, or **Arial**.
    - Maintain consistent font sizes for titles, headings, labels, and table text.
3. **Layout:**
    - Use a **top navigation bar** with logo and menu.
    - Sidebar for admin/faculty dashboards (Students, Subjects, Results, Reports, etc.).
    - Use **cards and tables** to show data clearly.
4. **Forms:**
    - Clearly labelled fields with placeholder text where necessary.
    - Validation messages (e.g., “Please enter valid marks between 0 and 100”).
5. **Accessibility:**
    - Sufficient color contrast.
    - Responsive design for desktops, laptops, and tablets.
6. **Branding:**
    - Place **university logo** on top-left.
    - Footer with university name, copyright year, and contact/email (optional).

CODE TEMPLATES (CLEAN + INTERVIEW READY)

The system is built using **PHP & MySQL** with modular and secure coding standards.

16.1 Database Connection (db_connect.php)

```
<?php
$host = "localhost";
$user = "root";
$pass = "";
$db   = "srms_db";

$conn = mysqli_connect($host, $user, $pass, $db);

if (!$conn) {
   die("Database Connection Failed: " . mysqli_connect_error());
}
?>
```

16.2 Secure Login (login.php)

```
<?php
session_start();
include "db_connect.php";

$username = $_POST['username'];
$password = $_POST['password'];

$query = "SELECT * FROM users WHERE username = ?";
$stmt  = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, "s", $username);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

```
if ($row = mysqli_fetch_assoc($result)) {
   if (password_verify($password, $row['password_hash'])) {
      $_SESSION['role'] = $row['role'];
      $_SESSION['userid'] = $row['user_id'];

      if ($row['role'] == "admin") header("Location: admin_dashboard.php");
      else if ($row['role'] == "faculty") header("Location: faculty_dashboard.php");
      else header("Location: student_dashboard.php");
      exit;
   }
}

echo "Invalid login credentials!";
?>
```

16.3 Marks Entry (marks_entry.php)

```
<?php
include "db_connect.php";

$student_id  = $_POST['student_id'];
$subject_id  = $_POST['subject_id'];
$internal    = $_POST['internal_marks'];
$external    = $_POST['external_marks'];

$total = $internal + $external;
$grade = ($total >= 90) ? "A+" :
         ($total >= 80) ? "A" :
         ($total >= 70) ? "B" :
         ($total >= 60) ? "C" :
         ($total >= 50) ? "D" : "F";
```

```
$query = "INSERT INTO results(student_id, subject_id, internal_marks, external_marks, total_marks, grade)
        VALUES (?, ?, ?, ?, ?, ?)";
$stmt = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, "iiiiss", $student_id, $subject_id, $internal, $external, $total, $grade);
mysqli_stmt_execute($stmt);

echo "Marks submitted successfully!";
?>
```

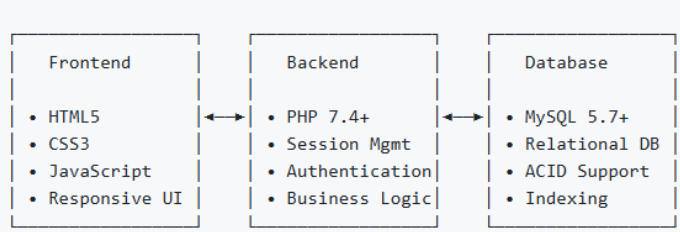
16.4 Student Result View (student_result.php)

```
<?php
session_start();
include "db_connect.php";

$student_id = $_SESSION['student_id'];
$query = "SELECT subjects.subject_name, results.internal_marks, results.external_marks, results.total_marks, results.grade
        FROM results
        JOIN subjects ON results.subject_id = subjects.subject_id
        WHERE results.student_id = $student_id";

$result = mysqli_query($conn, $query);
?>
<table>
<tr><th>Subject</th><th>Internal</th><th>External</th><th>Total</th><th>Grade</th></tr>
<?php while($row = mysqli_fetch_assoc($result)) { ?>
<tr>
<td><?= $row['subject_name']; ?></td>
<td><?= $row['internal_marks']; ?></td>
```

```
<td><?= $row['external_marks']; ?></td>
<td><?= $row['total_marks']; ?></td>
<td><?= $row['grade']; ?></td>
</tr>
<?php } ?>
</table>
```

System Architecture

System Architecture



Code File Structure

Project Structure

```
Student-e-Service-Portal/
├── Backend/                    # PHP Backend Files
│   ├── complaint.php           # Complaint management
│   ├── db_connect.php          # Database connection
│   ├── hostelMaintenance.php   # Hostel maintenance
│   ├── leaveApplication.php    # Leave applications
│   └── login.php               # Authentication logic
├── code/                       # Additional code modules
├── Dynamic Login/              # Login system components
│   └── index.html              # Login interface
├── images/                     # UI assets and screenshots
│   ├── LandingPage_ss.png      # Landing page screenshot
│   └── Rooms_SS.png            # Rooms page screenshot
├── RoleLogin/                  # Role-based login modules
│   ├── admin_dashboard.php     # Admin control panel
│   ├── employee_dashboard.php  # Employee interface
│   ├── student_dashboard.php   # Student interface
│   ├── warden_dashboard.php    # Warden interface
│   ├── index.html              # Role selection page
│   ├── indexupdate.html        # Updated interface
│   ├── login.php               # Role-specific login
│   ├── logout.php              # Logout functionality
│   ├── UpdateLogin.php         # Login updates
│   ├── styles.css              # Dashboard styling
│   └── stylesUpdate.css        # Updated styles
├── index.html                  # Main landing page
├── footercopymmdu.html         # Footer component
├── README.md                   # Project documentation
├── scripts.js                  # Frontend JavaScript
├── styles.css                  # Main stylesheet
└── style_indexpage.css         # Landing page styles
```

IMPLEMENTATION PLAN & MILESTONES

Stage	Duration	Deliverables
Requirement Gathering	Week 1	Feature list, ER diagram, use cases
System Design	Week 2	UI/UX, database design
Database Setup	Week 3	All tables created, sample data inserted
Backend Development	Week 4–5	Admin + Faculty + Student module
Frontend Integration	Week 6	Dashboards, forms, validation
Testing Phase	Week 7	Unit + integration tests
Deployment	Week 8	Hosting on server / localhost
Final Review	Week 9	Report + Presentation + Viva preparation

18. TESTING PLAN & SAMPLE TEST CASES

18.1 Testing Types

Type	Purpose
Unit Testing	Check individual PHP modules
Integration Testing	Check interaction between modules & DB
System Testing	Check complete flow of the system
Security Testing	Verify authentication & session handling
User Acceptance Testing (UAT)	Conducted by faculty/students

18.2 Sample Test Cases

Test Case ID	Input	Expected Output	Status
TC01	Valid login credentials	Redirect to dashboard	Pass
TC02	Invalid login credentials	Error message	Pass
TC03	Marks entry with blanks	Validation warning	Pass
TC04	Total >= 90	Grade = A+	Pass
TC05	Student views result	Display all subjects with marks	Pass

DEPLOYMENT GUIDE

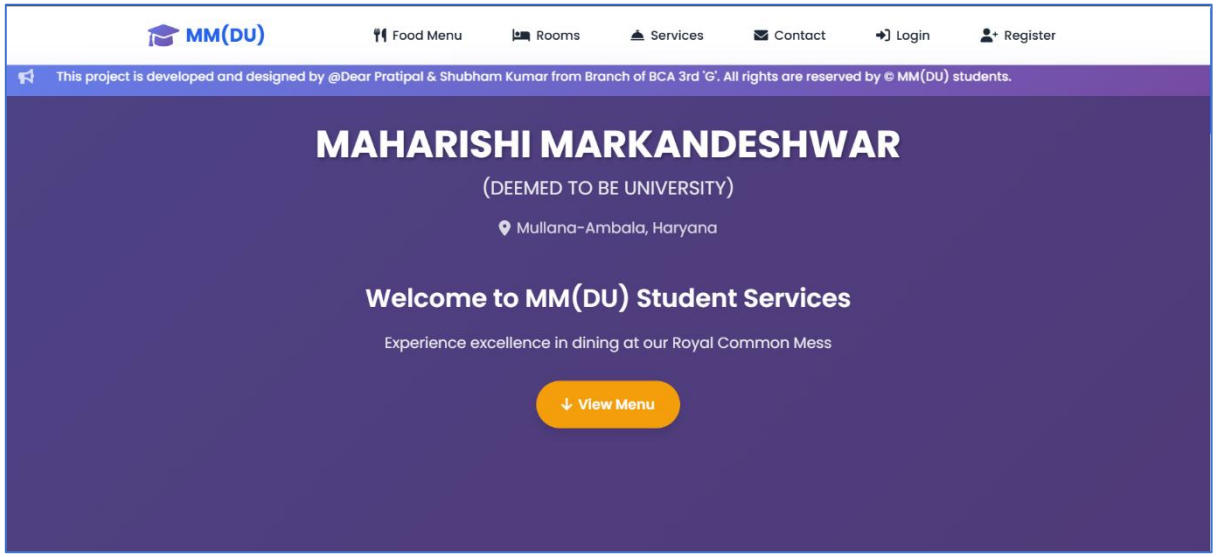
Local Deployment (XAMPP / WAMP)

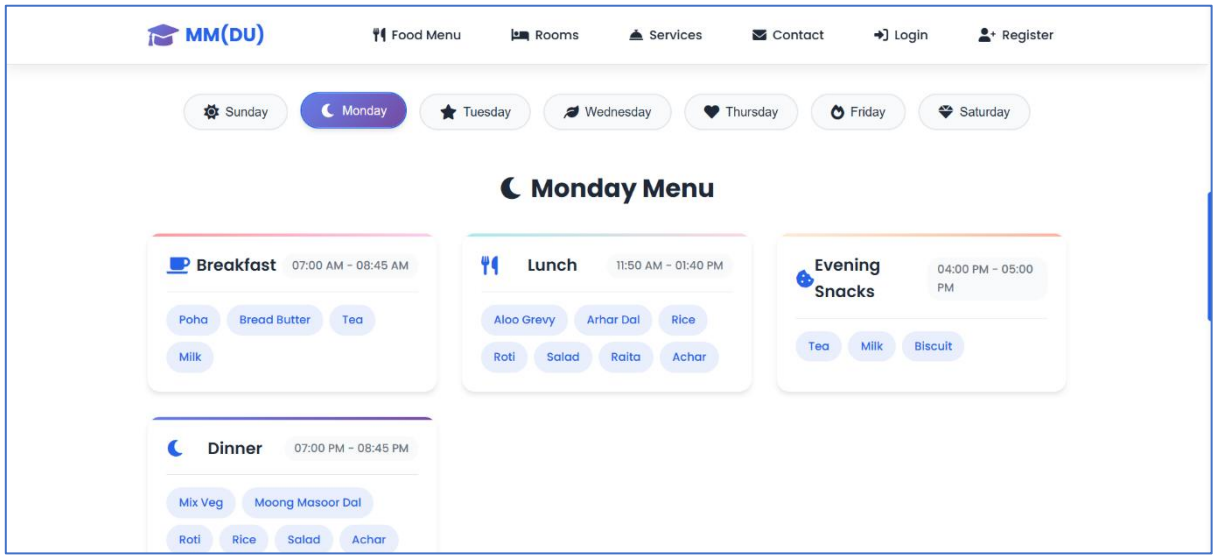
1. Install XAMPP / WAMP.
2. Copy project folder to:
3. C:/xampp/htdocs/srms/
4. Create database srms_db in phpMyAdmin.
5. Import the SQL file.
6. Start Apache + MySQL services.
7. Open in browser:
8. http://localhost/srms/

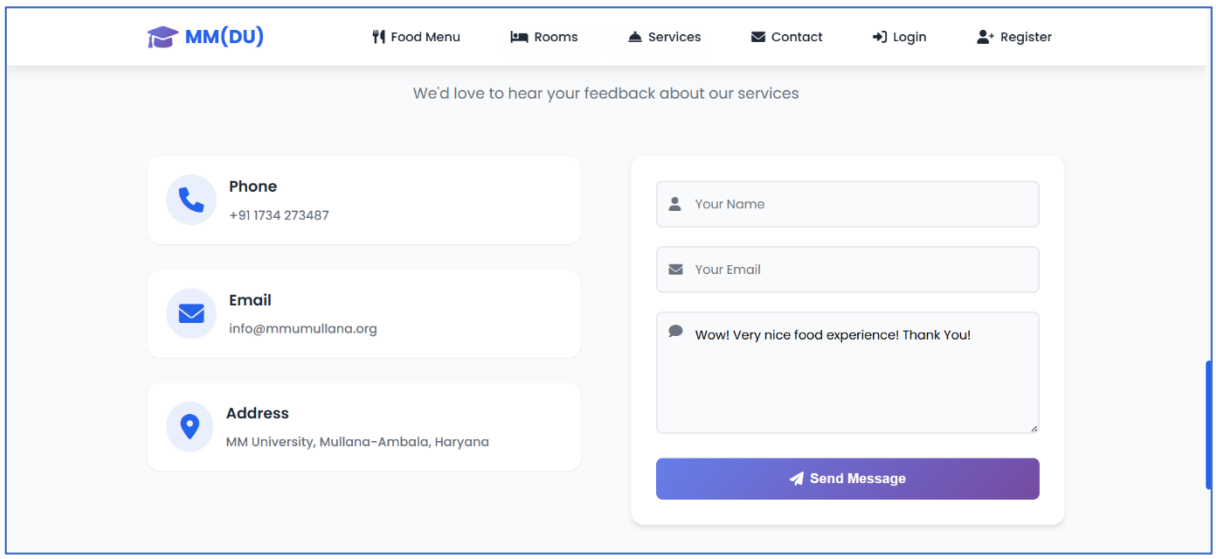
Web Hosting Deployment

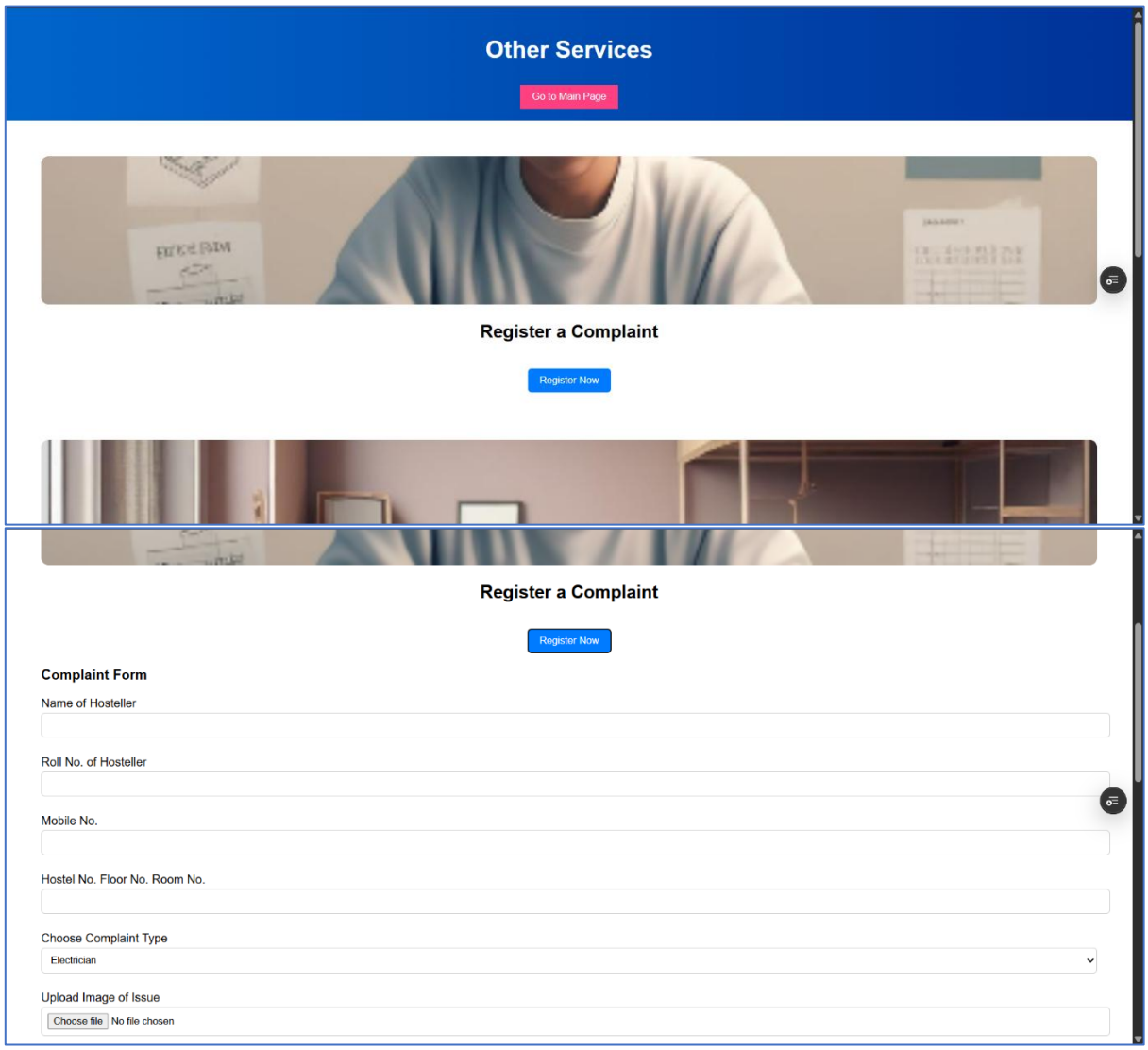
1. Purchase hosting with **cPanel + PHP + MySQL** support.
2. Upload project files to **public_html/**.
3. Create remote database from cPanel.
4. Update database credentials inside db_connect.php.
5. Access with live URL.

OUTPUTS & SCREENS

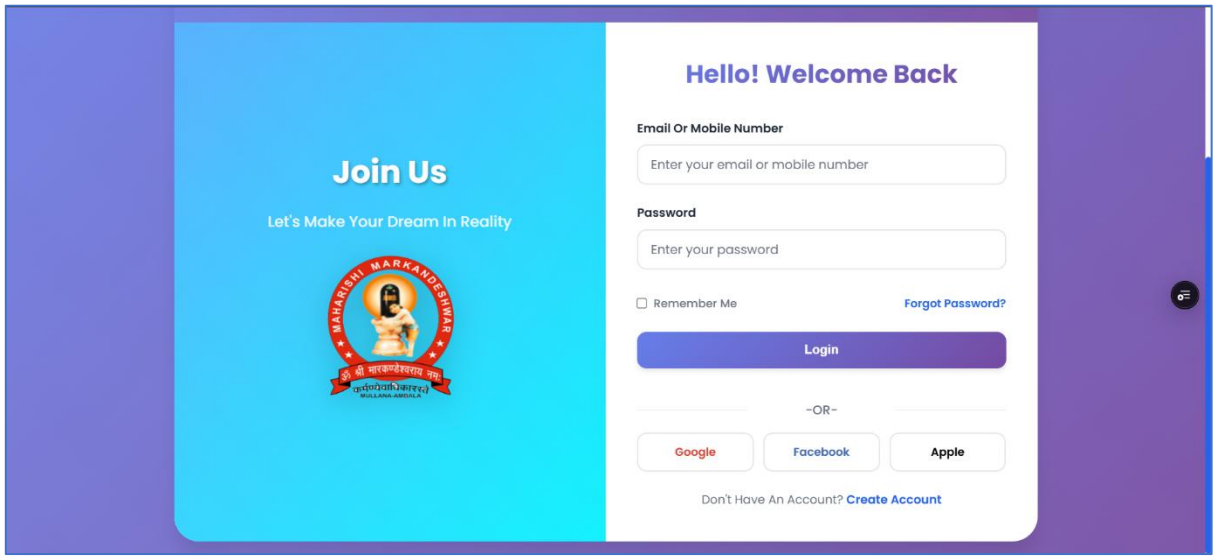








Login/Register Page



MAINTENANCE & FUTURE ENHANCEMENTS

Future Feature	Benefit
SMS / Email alerts	Notify students when results are published
Mobile App (Flutter)	Instant accessibility
Data Analytics Dashboard	Performance analysis
Parent Login	Parent access to results
Cloud hosting	Scalability and uptime
AI performance predictor	Early warning for slow-performing students

CONCLUSION

The **Student Result Management System** successfully digitalizes and streamlines the result processing workflow in educational institutions. It eliminates manual dependencies and reduces errors related to grade calculations and data handling. With secure authentication and user-friendly dashboards, the system benefits administrators, faculty, and students alike.

The system presents a scalable and future-ready approach, with strong potential for integration with mobile applications, ERP portals, and analytics-based decision support systems. Overall, this project demonstrates a practical and efficient use of modern web technologies to solve real academic challenges.

REFERENCES / BIBLIOGRAPHY

1. Elmasri & Navathe — Database Systems Concepts, Pearson Education.
2. PHP Documentation — https://www.php.net/
3. MySQL Developer Guide — https://dev.mysql.com/
4. Journal of Educational Technology & Society.
5. W3Schools — https://www.w3schools.com/
6. IEEE Papers on Academic Result Management & Automation.

APPENDIX

23.1 Sample SQL Insert

INSERT INTO users(username, password_hash, role, email)

VALUES ('admin', '$2y$10$encryptedpassword12345', 'admin', 'admin@gmail.com');

23.2 Example API Endpoint (if AJAX used)

POST /marks_entry.php

Body: {student_id, subject_id, internal_marks, external_marks}

23.3 Sample README Description for Repo

Student Result Management System

A full-stack web application for digitizing the result workflow across institutions.

Roles: Admin, Faculty, Student

Features: Marks entry, grade calculation, semester result view, print features

Tech Stack: PHP • MySQL • HTML • CSS • JavaScript