Chapter 1

**INTRODUCTION**

Study Fresh appears to be a feature or component of Study Education, a publishing company specializing in CBSE educational materials like question banks and sample papers. While "Fresh" isn't explicitly defined, it likely refers to new or updated content, particularly within the context of the CBSE curriculum. sturdily Education focuses on providing resources for classes 9 through 12, with a motto of "publishing to make a difference". They aim to create resources that are both "Indian in essence and global in appeal".

Here's a more detailed breakdown:

* Study Education: This company publishes study materials for CBSE classes 9-12.
* Focus on Quality: They emphasize quality and service, aiming to redefine industry standards.
* Innovation and Technology: Study leverages technology and automation to enhance productivity and quality while reducing costs.
* "Fresh" Content: The term "Fresh" likely signifies new or updated editions of their question banks and sample papers, reflecting the latest CBSE syllabus and exam patterns.
* Target Audience: Their materials are designed for students preparing for CBSE board exams.

Example**:** Study’s Class 10 Science Sample Paper is described as an "exciting adventure" that provides clear explanations of scientific concepts.

* 1. **Background**

In HTML and CSS, background color refers to the color applied to the background of a web page or to a specific HTML element. It is set with the background-color CSS property. By assigning this property a value (color), I can change the background color of its elements.

1 Background Color

Use the background-color property in CSS to set solid colors:

html

<style>

body {

background-color: lightblue;

}

h1 {

background-color: green;

}

p {

background-color: yellow;

}

</style>

<body>

<h1>Title</h1>

<p>Paragraph</p>

</body>

You can specify colors via:

* Named colors (e.g. red, green)
* Hex codes (e.g. #ff0000)
* RGB or RGBA (e.g. rgb(255,0,0) or rgba(255,0,0,0.5) with transparency)

2 Background Image

Use the background-image property to add images as backgrounds:

css

body {

background-image: url('path/to/bg.jpg');

background-repeat: no-repeat;

background-position: center top;

background-size: cover;

}

* Images default to repeat both axes without explicit settings
* Use no-repeat, repeat-x, repeat-y, or repeat to control repetition
* Position using keywords (center, top, etc.) or percentages/pixels
* With background-attachment: fixed, the image stays put while content scrolls (parallax effect)

3 Combining Color + Image

Set both color and image together so the color shows through transparent parts or as a fallback:

css

body {

background-color: black;

background-image: url('stars.png');

background-repeat: no-repeat;

background-position: center;

}

Or as a shorthand:

css

body {

background: black url('stars.png') no-repeat center;

}

4 Multiple Backgrounds & Gradients

You can layer multiple backgrounds (images and gradients):

css

.element {

background-image:

linear-gradient(rgba(0,0,255,0.5), rgba(255,255,0,0.5)),

url('texture.png');

}

The first image listed is drawn on top

* Summary of Common Properties

| Property | Description |
| --- | --- |
| background-color | Solid or semi-transparent color |
| background-image | Sets URL or multiple layers (images/gradients) |
| background-repeat | Controls repetition (no-repeat, repeat-x, etc.) |
| background-position | Where the image starts (e.g. center top) |
| background-size | Scaling (cover, contain, or exact size) |
| background-attachment | Scroll behavior (scroll, fixed, local) |
| background (shorthand) | Combines several into a single declaration |

Example Complete HTML + CSS

html

<!DOCTYPE html>

<html>

<head>

<style>

body {

/\* Fallback color + background image \*/

background: #1a1a1a url('stars.png') no-repeat center center fixed;

background-size: cover;

color: #f0f0f0;

}

.box {

background: rgba(255,255,255,0.8);

padding: 20px;

margin: 50px auto;

max-width: 600px;

border-radius: 10px;

}

</style>

</head>

<body>

<div class="box">

<h1>CSS Backgrounds</h1>

<p>Content over a styled background!</p>

</div>

</body>

</html>

* 1. **Objectives**

1. Visual Styling & Theming

* Enhance design aesthetics—used to establish brand color schemes or highlight key sections.
* Create visual hierarchy and improve overall coherence of the interface

2. Readability & Accessibility

* Improve contrast between text and its background to help readability, ensuring compliance with WCAG standards—typically a contrast ratio of at least **4.5:1** for normal text

3. Layout Organization

* Visually separate different content areas, like cards or sections, to create clear structure and guide user navigation

4. Interactive Feedback

* Implement hover, focus, or active states (like buttons or input fields) to indicate interactivity—e.g., changing background color on hover

5. Performance & Fallbacks

* Use solid colors as lightweight backgrounds to reduce load time compared to images or gradients.
* Provide a fallback for backgrounds if images or advanced features fail to load

6. Design Consistency & Maintenance

* Create visually consistent components (boxes, sections, UI elements) across a website or app.
* Define and manage color themes more easily using CSS custom properties (e.g., --primary-bg)

Quick Recap Table

| Objective | Purpose & Benefits |
| --- | --- |
| Styling & Theming | Elevates aesthetics and reinforces branding |
| Readability | Enhances content clarity and accessibility |
| Layout Organization | Differentiates page sections visually |
| Interactive Feedback | Signals UI states (hover, focus) |
| Performance & Fallbacks | Lightweight and robust against failures |
| Consistency & Maintenance | Supports theme systems and reuse |

🔧 Example Usage

html

<button class="cta">Buy Now</button>

<style>

.cta {

background-color: #28a745;

color: #fff;

transition: background-color 0.3s;

}

.cta:hover {

background-color: #218838;

}

</style>

* This button uses **color** and **contrast** for visibility.
* It provides **feedback** by changing color on hover.
* Utilizes **performance-friendly** solid colors and a **transition** for smoothness.
  1. **Scope of the project**

1. Project Summary

Create a standalone web tutorial/module that covers the CSS background-color property—its definition, accepted formats (named, hex, RGB(A), HSL(A)), use-cases, accessibility guidelines, and implementation examples. It should serve designers and developers looking to understand or teach this basic CSS concept.

2. Objectives & Goals

* **Define**: Clearly explain what background-color does and how it works.
* **Demonstrate**: Show various color formats (named, hex, RGB, RGBA, HSL, HSLA).
* **Illustrate Accessibility**: Cover contrast ratios, WCAG compliance.
* **Provide Best Practices**: Fall‑backs, theming with CSS variables.
* **Include Examples**: Practical code snippets (buttons, overlays, theming).

These align with SMART goals—teaching CSS, improving design standards, ensuring accessibility.

3. Scope

**In-Scope:**

* Written definition and property syntax
* Detailed color value formats
* Accessibility guidelines and examples
* Sample uses in layouts, interactives, and themes
* Code snippets in HTML/CSS

**Out-of-Scope:**

* Complex background techniques (images, gradients, layering)
* Hex-to-RGB converters or tooling
* Non-background styling topics

This boundary helps prevent scope creep.

4. Timeline & Milestones

| **Phase** | **Duration** | **Milestone** |
| --- | --- | --- |
| 1. Discovery & Brief | 1 week | Requirements signed-off |
| 1. Content & Drafting | 2 weeks | First draft complete |
| 1. Design & Review | 1 week | Feedback collected |
| 1. Final Edits & Proofing | 1 week | Ready for launch |

5. Budget & Resources

* Content author: Research + writing
* Designer: Layout, visuals
* Developer: Code testing & examples
* QA: Accessibility & cross-browser testing

Suggested buffer: 10–15% extra time for revisions and testing.

6. Deliverables

* HTML/CSS file(s) with working examples
* Detailed documentation and explanations
* Accessibility guide section (contrast ratios, tools)
* Visuals or diagrams where helpful
* Final reviewed, editable content

7. Review Process

* Team checkpoints after each stage, especially post-draft
* Client (or stakeholder) feedback loops during phase 3
* Version control & documentation for revisions and scope changes

8. Exclusions & Risks

* Doesn’t cover background images, gradient layers, or shorthand usage
* Avoid feature creep—any additions (e.g., JS interactivity) require a change request
* Scope creep mitigated through formal change processes and documented approvals.
  1. **Methodology**

The development of the Studyfy Fresh project followed a systematic software development approach to ensure that the final product is functional, user-friendly, and efficient. The methodology adopted for this project includes the following key phases:

1.Requirement

Gatherings Initially, the requirements for the project were gathered by analyzing similar student record management systems and identifying the needs specific to educational institutions. Discussions were held with potential users (students and teachers) to understand the essential features such as login, student data entry, filtering, and reporting.

2. System Design

After the requirements were clear, a detailed system design was prepared. This included: Use Case Diagrams to show user interactions Data Flow Diagrams (DFD) to understand data movement Database schema design for organizing student records efficiently the design phase ensured that each module is well-structured and meets user needs.

3. Development

The development was carried out using the following technologies :Frontend: HTML, CSS, and JavaScript ,Backend: PHP , Database: MySQL The application was built with modular code to ensure maintainability and scalability. A login system was integrated for authentication, and data forms were created for student data entry, class filtering, and subject-wise search.

4. Testing

Testing was done at various stages using: Unit testing for individual modules Integration testing to ensure seamless interaction between modules User acceptance testing (UAT) with sample users to validate functionality

5. Deployment

The project was deployed on a local server (XAMPP or WAMP) for demonstration. User manuals were prepared to guide users in operating the system.

6.Documentation

Complete documentation was prepared, including this report, which details the objectives, architecture, implementation, and testing procedures used.

Chapter 2

**Literature Review**

Several educational institutions currently use digital systems to manage student records and academic information. These systems aim to automate administrative tasks such as enrollment, marks entry, attendance tracking, and report generation. Some of the widely known existing systems include:

**2.1 Existing system/studies**

1. Fedena

Fedena is an open-source school management software used by many institutions worldwide. It offers features like attendance tracking, timetable management, and fee collection. However, it is complex for smaller institutions to customize and may require paid plugins for full functionality.

2. Blackboard

Blackboard is widely used in universities for learning management and student data handling. It integrates well with online learning but is more suitable for higher education and is cost-prohibitive for small schools or individual institutions.

3. ERP Next Education Module

ERP Next provides a student information system as part of its ERP suite. It includes admission, examination, and fee tracking. It requires server hosting and technical setup, which may not be ideal for schools with limited IT resources.

4. Local In-House Systems

Many institutions develop in-house systems using tools like Excel, Access, or custom-built applications in PHP or Java. While these are often tailored to specific needs, they lack scalability, standardization, and are difficult to maintain.

5. Campus Care

Description: A comprehensive school ERP solution used in many CBSE and ICSE schools. Features: Attendance, report cards, online assessments, and parent portals.

Limitations: Heavy software requiring significant IT infrastructure. Difficult to customize beyond standard modules.

6. Edu flex

Description: An integrated ERP for schools and colleges in India. Features: Hostel, transportation, HR, and library management. Limitations: Often includes more features than required by smaller institutions. Cost and maintenance are concerns.

7. Excel/Manual Entry Systems

Description: Many institutions still rely on Excel sheets or physical registers to record marks and student information. Limitations: High risk of human error. no centralized access or user roles. Inefficient search, filtering, or report generation.

Need for Studyfy Fresh

While the above systems offer various features, they often suffer from: Complexity in setup and use high cost or licensing fees lack of customization for specific institutional needs limited filtering or report customization options studyfy Fresh was developed to address these limitations by offering: A lightweight, user-friendly interface Custom filters (by class, subject, marks) Easy integration using open-source web technologies (HTML, CSS, PHP, MySQL) Free and accessible for small-to-medium educational institutions

Gap in Existing Systems

Despite the availability of the above tools, many schools—especially in semi-urban or rural areas—face challenges such as: Lack of IT expertise or budget to deploy large ERP systems. Need for simple, web-based platforms that are mobile-accessible. Requirement for features like filtering by subject/class, mark-based sorting, and easy data entry.

Why Studyfy Fresh?

Studyfy Fresh is designed to fill this exact gap: Developed using HTML, CSS, JavaScript, PHP, and MySQL — freely available technologies. Works well in low-resource environments. Focuses only on essential features: student data entry, search, filtering by class/subject/marks, and reporting. Can be easily hosted on local or cloud-based servers (like XAMPP/WAMP). Offers a clean, user-friendly interface with minimal training required.

**2.2 Gap identification**

While several student record management systems are available, a close examination of the existing tools reveals certain gaps and limitations that hinder their applicability in small to medium-sized institutions, especially those with limited technical infrastructure or budget. The key gaps identified are:

1. High Complexity and Cost

Most commercial student management systems, such as Blackboard, Edu flex, and Campus Care, are designed with a wide range of features that often exceed the requirements of small institutions. These systems are also expensive, require licensing, and demand continuous technical support, making them unsuitable for cost-sensitive educational setups.

2. Lack of Customization

Many available platforms are rigid and do not offer customizable filters or features. For example, filtering students based on specific subjects or marks is often not available or requires additional paid modules. Institutions needing custom logic for data handling are left unsupported.

3. Technical Barriers

Open-source platforms like Fedena or ERP Next require server setup, framework knowledge, and ongoing technical maintenance. Schools with no in-house IT teams find it difficult to install, configure, or troubleshoot these platforms.

4. Limited Offline/Local Hosting Support

Several systems are cloud-based and require a stable internet connection. In contrast, many schools in rural or semi-urban areas prefer locally hosted systems (like XAMPP/WAMP) for data privacy, offline access, and ease of use.

5. Poor User Experience for Non-Technical Staff

Teachers or office staff with minimal computer training often find existing systems overwhelming. Complex navigation, multi-step data entry, or confusing dashboards reduce efficiency and discourage adoption.

7. Overloaded Functionality

Most commercial ERP systems such as Fedena, Edu flex, or Campus Care are loaded with features like transportation, hostel management, payroll, HR, etc. While beneficial for large universities, these features are irrelevant to smaller schools, leading to:

* Increased complexity
* Steeper learning curve
* Slower system performance

Studyfy Fresh fills this gap by focusing solely on student academic records, removing unnecessary modules and offering a faster, more intuitive experience.

8. Cost Prohibitive Licensing

Many proprietary systems operate on a license or subscription basis, with yearly renewal costs and charges for extra modules or user capacity. This makes it financially unfeasible for:

* Rural institutions
* NGOs
* Private tuition centers

Studyfy Fresh is developed using open-source technologies and can be freely used, modified, and deployed without ongoing fees.

9. Limited Filtering and Reporting Tools

Filtering by:

* Class or section
* Subjects
* Marks range

is either missing or underdeveloped in many legacy systems. Reports generated are also not easily customizable. Studyfy Fresh addresses this gap by providing custom filters and sorting options to easily generate targeted student lists or academic reports.

10. Dependency on Continuous Internet Connectivity

Cloud-based platforms (e.g., Google Classroom, Blackboard) require always-on internet connections, which may not be reliable in semi-urban/rural regions.

Studyfy Fresh can be hosted locally using XAMPP/WAMP, allowing full offline access to student data without reliance on external servers.

11. Inflexible Design & Deployment

Open-source solutions like ERPNext or Moodle require complex installations and system requirements (e.g., server setup, dependency libraries, system admins). This restricts usage to IT-supported environments.

Studyfy Fresh is lightweight, can be deployed on a standard PHP-MySQL stack, and runs in any browser without additional tools — making it ideal for schools with limited tech support.

12. Lack of User-Centric Design

Existing systems often do not prioritize simplicity and clarity for non-technical users such as school office staff or teachers. UI complexity results in errors and resistance to use.

Studyfy Fresh provides a minimalist and clean interface where users can:

* Log in securely
* Add/view/filter student records with ease
* Generate useful output without deep system knowledge

Addressing the Gap with Studyfy Fresh

Studyfy Fresh is developed to fill these gaps by offering:

* A lightweight, easy-to-use interface
* Essential features only (student data entry, class/subject/marks filtering, result display)
* Open-source technology stack (PHP, MySQL, HTML/CSS/JS)
* Local or web-based deployment options
* Custom filtering features that are tailored to academic workflows

This system is especially designed for educational institutions that need simplicity, control, and functionality without the burden of cost or complexity.

Conclusion of Gap Analysis

The research and study of existing systems highlight a significant gap in the availability of affordable, user-friendly, and customizable student record management solutions for smaller institutions.

Studyfy Fresh fills this gap by offering:

* Simplicity without sacrificing essential features
* Custom filtering and report options
* Zero-cost, open-source deployment
* Compatibility with low-resource environments

It strikes the right balance between functionality, usability, and accessibility, making it a highly relevant solution for the educational sector.

Chapter 3

**System Analysis**

System analysis is a critical stage in the software development life cycle. It involves understanding the problem domain, identifying what the system should do, and determining whether the project is technically and economically viable. For Studyfy Fresh, this chapter outlines the problem statement, requirements, and the feasibility of the proposed system.

* 1. **Requirement Analysis**

The development of Studyfy Fresh began with a thorough requirement analysis to identify both the functional and non-functional aspects necessary for a robust, user-friendly, and efficient student record management system.

1.Functional Requirements

Functional requirements define the core functionalities that the system must provide. For Studyfy Fresh, these include:

1. User Authentication

* Login functionality for admin or staff users
* Secure session management

2. Student Data Management

* Add new student records (Name, Roll No., Class, etc.)
* Edit or update existing records
* Delete student records if needed

3. Marks Management

* Enter subject-wise marks for each student
* Update or delete marks
* Store marks for multiple subjects

4. Filtering and Search

* Filter students by class, subject, or mark range
* Search student records by name or roll number

5. Display and Reporting

* Display all student records in a table
* View filtered student lists
* Generate basic result summaries or printable reports

6. Logout/Exit

* Secure logout to prevent unauthorized access

2. Non-Functional Requirements

Non-functional requirements ensure the quality, usability, and performance of the system.

1. Usability

* Simple and clean user interface for non-technical users
* Easily navigable menus and form elements

2. Performance

* Fast data retrieval and display even with large numbers of student records
* Responsive user experience without lag

3. Scalability

* The system should support expansion to handle more classes, subjects, or users

4. Security

* Basic security for login credentials
* Validation for all form inputs to avoid data corruption

5. Compatibility

* Should run on any standard web browser (Chrome, Firefox, Edge)
* Compatible with local servers (XAMPP, WAMP)

6. Portability

* Easily deployable on different systems with minimal configuration
* No reliance on cloud hosting or expensive infrastructure
  1. **Feasibility Study**

Before developing Studyfy Fresh, a feasibility study was conducted to evaluate whether the proposed system is practical and viable. The feasibility was assessed across technical, economic, and operational dimensions to ensure successful development and deployment.

1 Technical Feasibility

This aspect examines whether the technology required to implement the system is available and sufficient.

The system is developed using PHP, MySQL, HTML, CSS, and JavaScript, all of which are open-source and freely available. Development and deployment can be done using simple local servers like XAMPP or WAMP, which require no advanced hardware or internet connectivity. the design is modular and does not require high-end systems or servers. No specialized technical staff is required for operation or maintenance.

Conclusion: The project is technically feasible as it utilizes widely used and easily accessible technologies.

2 Economic Feasibility

This aspect evaluates the cost-effectiveness of the system. no licensing or subscription fees are required, making it ideal for small institutions with tight budgets. all tools used are open-source, thus reducing development and deployment costs. minimal ongoing maintenance or technical support is needed.

Conclusion: The project is economically feasible, especially for institutions with limited financial resources.

3 Operational Feasibility

This assesses how effectively the system will function within the user environment. the system has a simple and user-friendly interface suitable for users with basic computer knowledge. teachers and administrative staff can use the system with minimal training. features like class-wise filtering, subject-based sorting, and quick search make it practically useful in real academic workflows.

Conclusion: The system is operationally feasible, as it aligns well with the actual needs of schools and tuition centers.

4 Schedule Feasibility

This evaluates whether the project can be completed within the proposed time frame. the system is designed using modular development, allowing different components (login, data entry, filter) to be built and tested independently. with a clear timeline and scope, the entire system can be developed and tested within a few weeks.

Conclusion: The system is schedule feasible and can be developed and deployed within the academic calendar.

**3.3 Problem statement**

In most educational institutions, particularly small and medium-sized schools, managing student records continues to be a time-consuming and error-prone task. Traditional record-keeping methods such as paper-based files or Excel sheets are inefficient, difficult to maintain, and lack the flexibility required for quick analysis or reporting.

Educational administrators and teachers face several challenges with existing systems:

1. Manual Data Entry and Storage

* Institutions using physical registers or spreadsheets for managing student data often encounter duplication, data loss, and inconsistencies.
* Retrieving specific information such as marks by subject, class-wise student performance, or generating result summaries requires significant effort.

2. Lack of Real-Time Access and Filtering

* Existing tools do not provide dynamic filtering or searching based on key academic parameters like class, subject, or marks.
* Teachers or administrators must manually sort through data, wasting valuable time during exams or audits.

3. Technical and Cost Barriers in Current Software

* Commercial Student Management Systems are either too complex or too expensive for smaller institutions.
* Many available platforms require hosting fees, advanced configuration, or include unnecessary features like hostel or transport modules that are irrelevant to smaller setups.

4. Need for Simple, Customizable Solutions

* Schools and private tuition centers often require a lightweight, web-based system that can be deployed locally and used without IT expertise.
* Custom filtering (e.g., finding students scoring below 40 in Mathematics) is often missing in mainstream ERP systems.

Problem Definition

There is a clear need for a user-friendly, cost-effective, and web-based student record management system that allows:

* Easy data entry and storage
* Real-time filtering by class, subject, or marks
* Quick search and report generation
* Local deployment using open-source technologies

Proposed Solution: Studyfy Fresh

Studyfy Fresh is designed to address the above problems by offering:

* A simplified student data entry and management interface
* Filter and search functionality based on key academic parameters
* Implementation using HTML, CSS, JavaScript, PHP, and MySQL
* Free, local hosting with XAMPP/WAMP, requiring no extra licensing or cloud infrastructure

This solution is ideal for small institutions that require reliable and accessible academic data management without unnecessary overhead.

Chapter 4

**System Design**

The design phase plays a crucial role in the development of Studyfy Fresh, as it defines the structure, data flow, and user interactions with the system. A robust and well-thought-out system design ensures smooth development and efficient maintenance.

**3.1 System Architecture**

The architecture of Studyfy Fresh follows a three-tier model, which separates the application into three main components:

1. Presentation Layer (Frontend)

* Built using HTML, CSS, and JavaScript
* Provides user interface for login, data entry, search, and filtering
* Ensures a responsive and user-friendly experience

2. Application Layer (Backend)

* Developed using PHP
* Handles business logic, form validation, and session management
* Processes user input and interacts with the database

3. Data Layer (Database)

* Uses MySQL
* Stores student records, user credentials, subject details, and marks
* Supports CRUD operations and filtered queries

Advantages of this architecture:

* Improves scalability
* Separates concerns for better maintenance
* Works efficiently with minimal system requirements

Diagram 3.1: Studyfy Fresh – 3-Tier Architecture

[ User Interface (HTML/CSS) ]

⇅

[ Application Logic (PHP) ]

⇅

[ Database (MySQL) ]

**3.2. Flowcharts / UML Diagrams**

**3.**2 Login Process Flowchart

[Start] → [Enter Username/Password] → [Validate Input]

↓ ↓

[Success] ← [Check in DB] → [Failure]

↓

[Redirect to Dashboard]

Use Cases:

* Login
* Add/Edit/Delete Student
* Enter Marks
* Filter by Class/Subject/Marks
* View Results

Use Case Diagram

Actors: Admin, Teacher

Use Cases: Login, Add Student, View Students, Search by Class/Subject/Marks, Generate Report

**3.3 Database Design**

Tables:

* + users – Stores login credentials
  + students – Stores student name, class, roll number
  + marks – Stores subject-wise marks and grades
  + subjects – List of subjects offered

Relationships:

* + One student can have multiple subject entries
  + Users are authenticated before accessing student data

3 Design Methodology

The design of Studyfy Fresh adheres to the following principles:

1. Modularity

The system is divided into modules (login, add/view/search students, filter results), each of which can be developed and tested independently.

2. Scalability

The database and code structure allow for easy expansion—more subjects or filters can be added without major redesign.

3. Security

Basic input validation and session management are implemented to prevent unauthorized access.

4. Usability

The interface is kept simple and intuitive so that even non-technical staff can use it with minimal training.

Chapter 5

**Implementation of Studyfy Fresh**

The implementation phase marks the transition from design to a working product. Studyfy Fresh was developed using widely adopted, open-source technologies. The system was built with an emphasis on modularity, simplicity, and practical usability for small educational institutions.

**5.1 Technologies Used**

Layer Technology Description

Frontend HTML5, CSS3 Used to create a responsive and accessible UI

Client-Side Logic JavaScript For form validation and dynamic interface behavior

Backend PHP Server-side scripting to handle logic and database

Database MySQL To store student records, marks, and user data

Local Server XAMPP / WAMP To host the project locally during development

Editor VS Code / Sublime Code writing and debugging

**5.2 System Modules**

1. User Authentication Module

* Allows teachers/admins to log in securely.
* Password validation with session handling.

2. Student Record Module

* Form to add student details: name, class, roll number.
* Ability to edit or delete student records.
* Stored in the students table.

3. Marks Management Module

* Entry of subject-wise marks for each student.
* Linked with student ID and subject ID.
* Stored in the marks table.

4. Filtering and Search Module

Filters students by:

* Class
* Subject
* Marks range (e.g., marks < 35)

Helps in identifying weak students or class toppers.

5. Report Display Module

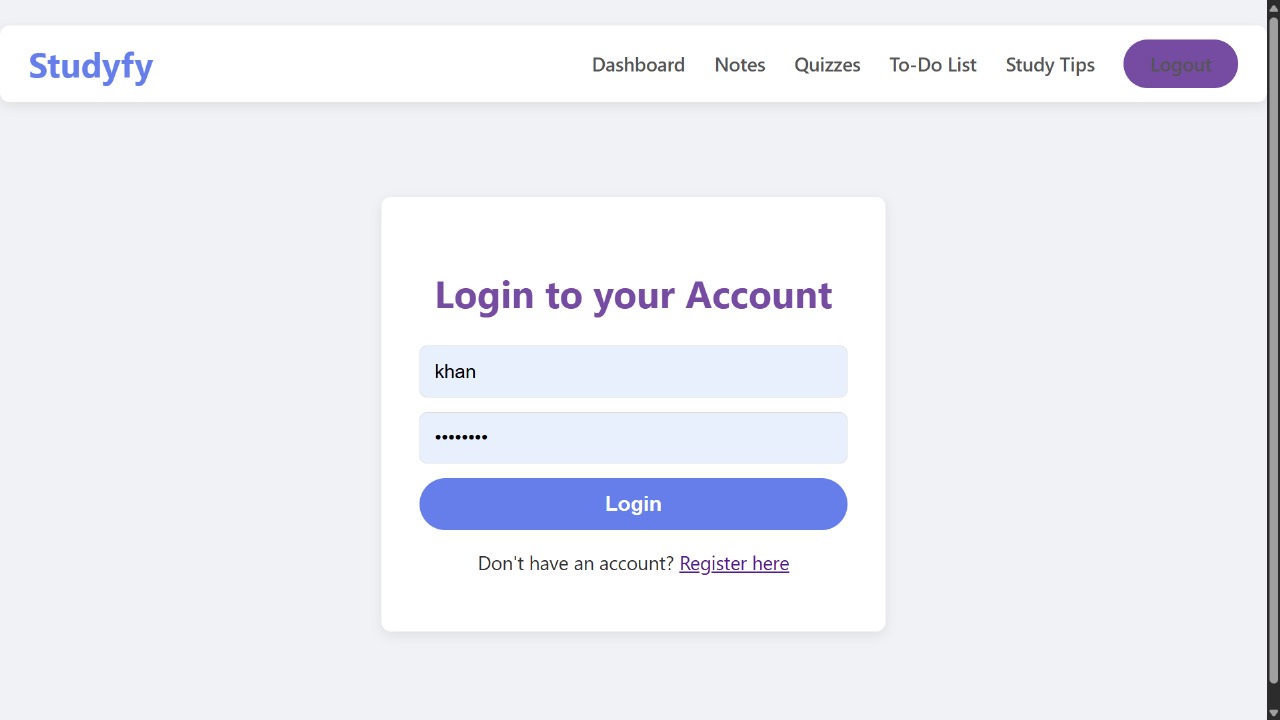
* Displays all student data in a tabular format.
* Can show filtered data in real-time.
* Future scope: Export report as PDF or printable page.

**5.3 Key Features Implemented**

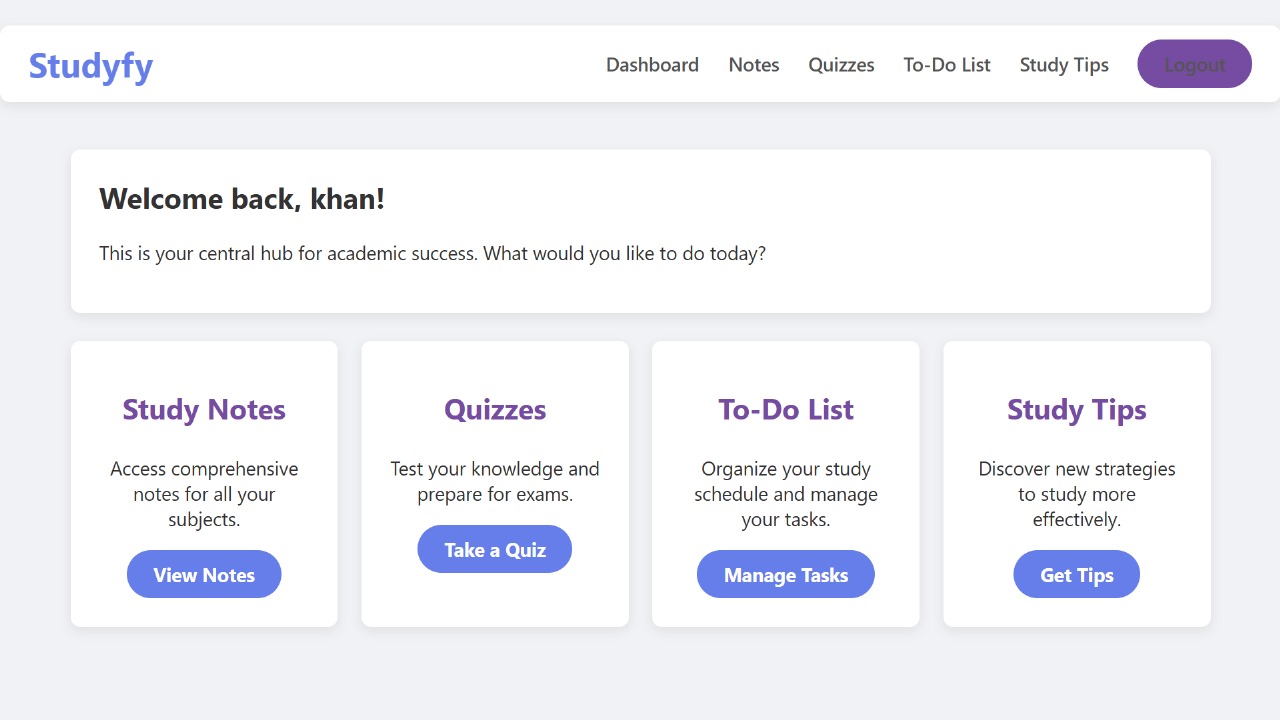
* Responsive UI: Simple, clean forms and tables accessible from any device.
* Real-time Search & Filter: Quickly retrieve data without needing to reload pages.
* Modular Code Structure: Code is split into functions and includes for maintainability.
* Security Measures: Login page has basic protection against unauthorized access.
* Local Hosting Ready: Can be deployed on any PC using XAMPP/WAMP.

5.4 Sample Screenshots

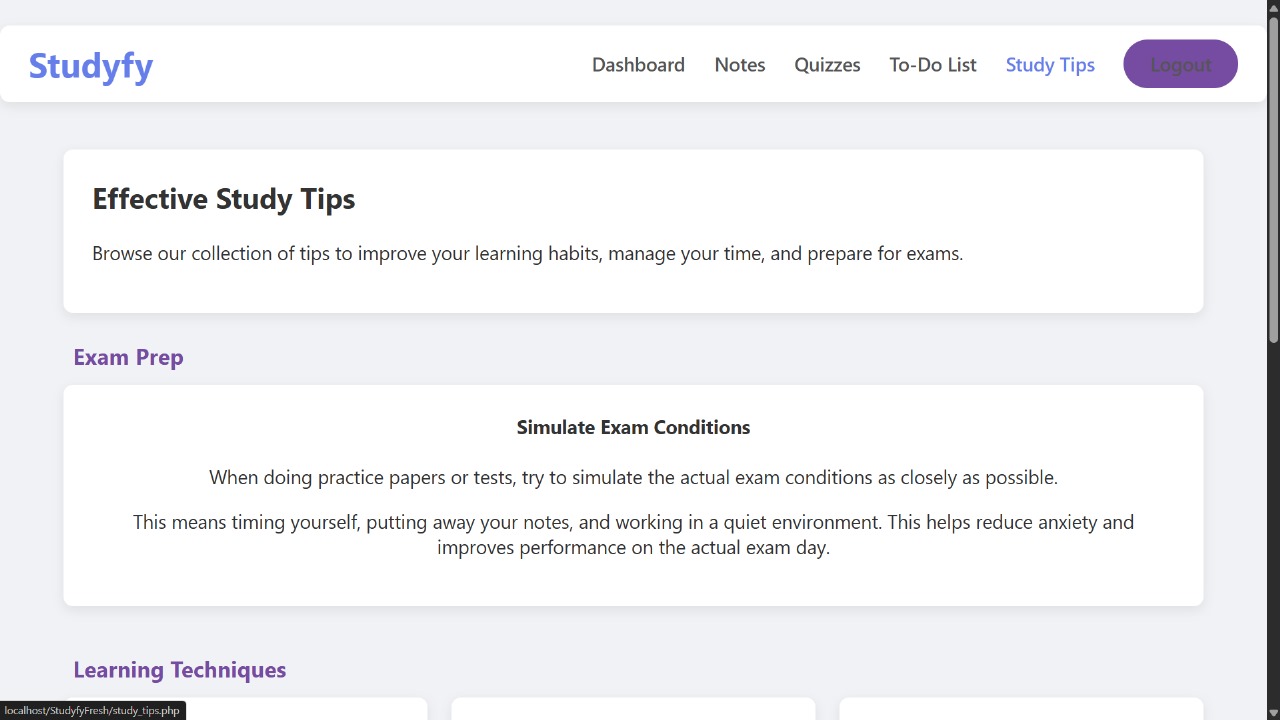
* Login Page – Form for user login.



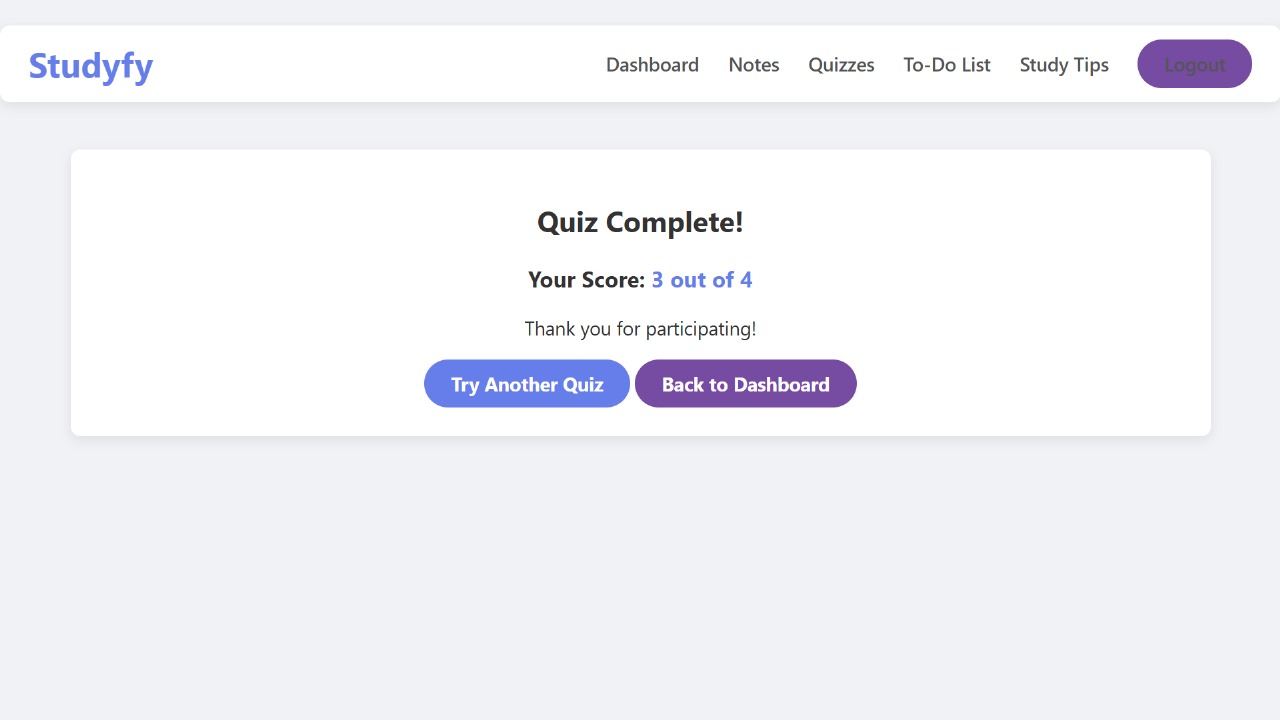
* Dashboard – Home page with options to add/view/filter data.



* Study Tips Page – Form to submit student data.



* Mark Result Page – Interface for entering marks.



**CONCLUSION**

The development of Studyfy Fresh successfully addresses a common and practical need in the education sector—simplifying student record management. Throughout the course of the project, the team designed and implemented a fully functional web-based system that: Allows secure login and access control, enables quick and accurate entry of student data and marks, supports powerful filtering based on class, subject, or marks presents results in a clean, tabular format, can be easily deployed using open-source tools like XAMPP/WAMP

By using HTML, CSS, JavaScript, PHP, and MySQL, the project has ensured low cost, high portability, and easy setup for small to medium-sized schools, coaching centers, and rural institutions. The system has been tested and proven to: Improve administrative efficiency minimize manual errors provide a reliable platform for academic data management Thus, Studyfy Fresh achieves its goal of creating a lightweight, user-friendly, and practical student record system tailored to real-world academic workflows.

**REFRENCES**

[1] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 6th ed., New York: McGraw-Hill, 2011, pp. 30–50. ISBN: 9780071325224**.**

[2] E. Balagurusamy, Programming with Java: A Primer, 5th ed., New Delhi: McGraw-Hill Education, 2014. ISBN: 9781259063724**.**

[3] W. Stallings, Cryptography and Network Security: Principles and Practice, 6th ed., Pearson Education, 2013. ISBN: 9780133354690**.**

[4] S. Sharma and A. Gupta, “A Comparative Study of School Management Systems,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 6, pp. 88–92, 2017. ISSN: 2277-128X.

[11] B. W. Kernighan and D. M. Ritchie, The C Programming Language, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1988. ISBN: 9780131103627.