Fees Management System
(Web Application)

A PROJECT REPORT
for
Major Project (KCA-451)
Session (2024-25)

Submitted By
Arun Kumar
(2200290140039)

Submitted in partial fulfillment of
the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Mr Praveen Kumar Gupta
Assistant Professor

Submitted to
DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(MAY, 2024)

DECLARATION

I hereby declare that the work presented in report entitled “fees management system” was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University of Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Arun Kumar

Roll No.: 2200290140039

(Candidate Signature)

CERTIFICATE

Certified that **Arun Kumar (University Roll No.- 2200290140039)** has carried out the projectwork having "**Fees Management System**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the studenthimself/herself and the contents of the project report do not form the basis for the award of any otherdegree to the candidate or to anybody else from this or any other University/Institution.

Arun Kumar (2200290140039)

This is to certify that the above statement made by the candidate is correct to the best of myknowledge.

Date:

Mr. Praveen Kumar Gupta	**Dr. Arun Kumar Tripathi**
Assistant Professor	**Professor & Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

FEE MANAGEMENT SYSTEM

Arun Kumar

ABSTRACT

The "Fees Management System" is a comprehensive desktop application developed to optimize the management of student fee records in college administration. Utilizing PHP, CSS, and JavaScript, this system replaces manual paperwork with a secure, efficient, and user-friendly digital interface. The primary aim is to provide administrators with complete control over student and branch management, fee transactions, and financial reporting.

Key functionalities of the system include the ability to add, edit, and delete student and branch records, ensuring the database is always up-to-date with accurate information. Administrators can manage fee payments efficiently, including recording paid amounts, updating payment dates, and adding remarks for each transaction. The system automatically updates the fee status, removing names from the dues list once payments are completed.

A robust reporting module generates detailed fee reports, providing insights into fee collection, outstanding dues, and the overall financial status of the institution. This module enhances the transparency and accountability of the fee management process.

The system's architecture is designed for maintainability and scalability, with a three-tier structure separating the user interface, business logic, and database layers. The backend, powered by PHP and MySQL, ensures secure data transactions and storage, while the frontend, developed using HTML, CSS, and JavaScript, offers an intuitive and responsive user experience.

Implementing the Fees Management System leads to significant improvements in administrative efficiency and data accuracy, reducing the workload of administrative staff and enhancing the reliability of the fee management process. This system is a valuable tool for educational institutions seeking to streamline their fee management operations and improve overall efficiency.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Mr. Praveen Kumar Gupta** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Arun Kumar

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION:

This project “**Fee Management System**” is a desktop system enables efficient storage of student records to properly manage the fee records of the students. And it also generates messages for due balances its student’s fee.

The system is designed for fee management of a college administration department. It makes searching records easier and faster.

1.1 ABOUT PROJECT:

Fees Management System project is developed using PHP, CSS, and JavaScript. Talking about the project, it has all the essential features. This project has an administration side from Where he/she can view branch, students, fees, report, manage fees, students, branches, settings. In this project, all the functions are performed from the admin side which means there is no user side.

OBJECTIVE:

- To reduce paperwork
- To make storage of information more efficient and secure.
- To have a friendly interface.
- To operate it easily and with minimum experience.
- To save time and energy of the admin.

1.2 ABOUT SYSTEM:

Admin has full control of the system, he/she can view branch, students, fees and manage branch, students, fees from the system. The project also includes a Fees report of students in Report module, which displays Fees Information as well as respective Student information. He/she can add, edit, delete, view Branch. While adding Branch, he/she has to provide Branch Name, Address, and Detail. Likewise while adding students, he/she has to provide Personal information like Name, Contact, branch, DOJ, Fees Information like Total Fees, Advance Fee, Remarks and Optional Information like About student and Email id. To take Fees for a student, the user has to provide Paid amount, date and Remarks. After paying fees of the student, that particular name will be removed from Fees module.

1.3 SCOPE OF THE SYSTEM:

- Specifically designed for a individual college.
- Inserting new student records are not possible.
- It is based on desktop application.
- Not including fees other than academic like bus fees and etc.
- There are only limited number of modules for fees management

1.4 EXISTING SYSTEM AND DRAWBACK:

- In the existing system, colleges have to maually maintain information regarding to Fees deposited by the students.
- College management system are complex and time consuming to maintain fees of students by that very difficult.
- It is not properly capable to manage the student records with their fee details at a single place.

- Managing collection of student fees, issuing fee receipts and fee register updation is a laborious manual process, leading to data inaccuracy and / or reconciliation.
- Preparing receipts manually everyday needs additional clerical staff.
- To generate due fees report is required a complete manual procedure, which involves a lot of time and clerical staff manpower.
- Re-entry of fees receipts in accounting software separately leads to double manpower cost and time.

1.5 PROPOSED SYSTEM & ADVANTAGES:

- The fees management System is a desktop system aimed to maintaining students records and their fees details.
- It also generate records like I.e. Feed Paid, dues, and etc.
- The system requires small amount of time to generate reports needed to manage the fees of the student.
- Managing collection of school fees, issuing fee receipts and fee register updation is done with the help of software resulting in highly accurate data.
- Software provides facility to print receipts automizing office work.
- This Software provides facility to generate due fees report easily and at any point of time.
- Software directly enter fees receipts to the accounts of the school.

CHAPTER 2

DESCRIPTION

2.1 MODULE DESCRIPTION:

The core functionalities that are to be included in the system are the follows.

- Admin Login
- Student Details
- Fee Details
- Course Scheme
- Payment
- Daily Reports
- Dues

2.2 SOFTWARE DESCRIPTION

2.2.1 PROJECT DESIGN:

Software design is an interactive process through which requirements are translated into a 'Blue Print' for constructing the software. The design is represented at high level of abstraction, a level that can be directly translated to specific data, functional and behavioural requirements. Preliminary design is concerned with the transformation of requirements into data and software architecture. Detained design focuses on refinements to the architectural representation that lead to detailed data structure and algorithmic representation for software.

2.2.2 SYSTEM SPECIFICATION:

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behaviour of a system or software application. It includes a variety of elements (see below) that attempts to define the intended functionality required by the customer to satisfy their different users.

Whenever you purchase software or hardware for your computer, you should first make sure your computer supports the system requirements. These are the necessary specifications your computer must have in order to use the software or hardware.

CHAPTER 3

TECHNOLOGY STACK

3.1 TECHNOLOGY USED

3.1.1 HARDWARE SPECIFICATION

Processor: Dual core

RAM: 1 GB

ROM: 2 GB

3.1.2 SOFTWARE SPECIFICATION

Technologis: HTML, CSS, java script

Database: MySql

Language: PHP version(7.2).

3.2 Brief overview of the technology:

Front end: HTML, CSS, JavaScript

HTML:

HTML is used to create and save web document. E.g. Notepad/Notepad++

CSS : (Cascading Style Sheets) Create attractive Layout

Bootstrap : responsive design mobile freindly site

JavaScript: it is a programming language, commonly use with web browsers.

Back end: PHP, MySQL

3.3 KEY NOTES

PHP:

Hypertext Preprocessor (PHP) is a technology that allows software developers to create dynamically generated web pages, in HTML, XML, or other document types, as per client request. PHP is open source software.

OVERVIEW OF PHP

PHP is an embedded scripting language that is excellent for creating dynamic web sites based on database content or different characteristics of browsers. It is available when you have a department(Web Central) publishing account, a faculty publishing account, a student organization publishing account or if It can Notable PHP compilers include the following the most popular implementation. Several compilers have been developed.

The PHP language was originally implemented as an interpreter, and this is still the most popular implementation. Several compilers have been developed which decouple the PHP language from the interpreter.

PHP stands for Hypertext Preprocessor. It is a server-side scripting language, like ASP. Also, the PHP scripts are executed on the server. It supports many databases (MYSQL, Informix, Oracle, Sybase, Solid, Postures SQL, and Generic ODBC), PHP is open source software and it is free to download and use.

PHP FILE

• PHP file can contain text, HTML tags and scripts

• PHP files are returned to the browser as plain HTML

• PHP files have a file extension of ".php",".html".

PHP combined with MySQLI are cross-platform (You can develop in windows and serve on a UNIX platform) Advantages of compilation include better execution speed, static analysis, and improved interoperability with code written in other languages.

USES OF PHP

PHP is an intuitive, server side scripting language. Like any other scripting language it allows developers to build into the creation of web page content and handle data returned from a web browser. PHP also contains a number of extensions that make it easy to interactive database, extracting data to be displayed on a web page and storing information entered by a web site visitor back into the database.

PHP consists of a scripting language and an interpreter. Like other scripting languages, PHP enables web developers to define the behavior and logic they need in a web page. The scripts are embedded into HTML documents that are served by the web server. The interpreter takes the form of a module that integrates into the web server, converting the scripts into commands the computer then executes to achieve the result defined in the web developer.

OBJECTIVES OF PHP

To develop an understanding of how PHP works it is helpful to first explore what happens when a webpage is served to a user's browser. When a user visits a web site or clicks on a link on a page the browser sends a request to the web server hosting the site asking for a copy of web page.

Now let's consider what kind of web page content a web browser understands. These days a web page is likely to consist of HTML< XHTML and JavaScript. The web browser contains code that tells it what to do with these types of content.

A web browser, however, knows absolutely nothing about any PHP script that may be embedded in an HTML document. The web server receives the request, finds the corresponding web page file on the system and sends it back, over the internet, to the user's browser.

Typically the webpage file system and sends it back, over the internet. If a browser was served a web page containing PHP it would not know how to interpret that code newer. The most common way of installing PHP is compiling it from the source code.
When Php is installed and used in cloud environments.

COMPONENTS OF PHP

In terms of web page content we have two extremes. At one extreme have to HTML which is completely static. There is very little that can be done with HTML to create dynamic content in a web page .At the other extreme we have scripting languages like java script provides a powerful mechanism for creating interactive and dynamic web pages.

When talking about JavaScript it is important to understand that it is, by design, a client side scripting language. By this we mean that the script gets executed inside the user's browser and not on the web server on which the web page originated.

While this is fine for many situations it is often the case that by the time a script reaches the browser it is then either too late, or in efficient, to do what is needed.

Prime example of this involves displaying a web page which contains some data from database table.

Since the database resides on a server (either the same physical server which runs the web server or on the same network as the web server connected by a high speed fiber network connection) it makes sense for any script that needs to extract data from the database to be executed on the server, rather than waiting until it reaches the browser.

It is for this kind of task that PHP is perfectly suited. It also fast and efficient because the script is executed on the server it gets to take advantage of multi-processing, large scale memory and other such enterprise level hardware features.

In addition to the advantages of being a server side scripting language PHP is very easy to learn and use. The fact that PHP works seamlessly with HTML makes it accessible to a broad community of web designers. These scripts can also used for simple text processing tasks. Perhaps one of the most significant advantages of PHP to some is the ease with which it interacts with the MySQLI database and store data. The web server receives the request, finds the corresponding web page file on the system and sends it back, over the internet, to the user's browser.

SCRIPTING OF PHP5

An application programming interface or API defines the classes, methods, functions and variables that your application will need to call in order to carry out its desire task. In the case of PHP applications that need to communicate with databases the necessary APIs are usually exposed via PHP extensions.

APIs can be procedural or object-oriented with a procedural API you call functions to carry out tasks, with the object-oriented API you instantiate classes and then call methods on the resulting objects. Of the two the latter is usually the preferred interface, as it is more modern and leads to better organized code.

When writing PHP applications that need to connect to MySQLI server there are several API options available. This document discussed what is available and how to select the best solution for your application.

SERVER –SIDE SCRIPTING

This is the most traditional and main target fields for PHP. Hey need for three things to make this work.PHP parser a web server and a web browser. They need to run the web server, with a connected PHP installation. They can access the PHP program with a web browser, viewing the PHP page through the server. All these can run on your home machine if your are just experimenting with PHP programming.

COMMAND LINE SCRIPTING

In this project can make a PHP script to run it without any server or browser. They only need the PHP parse to use it this way. This type of usage is ideal for scripts regularly executed using crone or task scheduler.

WRITING DESKTOP APPLICATIONS

PHP is probably not the best language to create a desktop application with graphical user interface, but they know PHP very well, and would like to use some advanced PHP features in your client-side applications.

They also have to write cross platform applications this way. PGP_GTK is an extension to PHP, not available in the admin distribution. They are also having a DBX database abstraction extension allowing you to transparently use any database supported by that extension. Additional php supports ODBC, the open database connection standard, so you can connect to any other database supporting this world standard.

CONNECTOR

In the MySQLI documentation, the term connector refers to a piece of software that allows application to connect to the MySQLI database server. MySQLI provides connectors for a variety of languages, including PHP.

In PHP application needs to communicate with a database server will need to write PHP code to perform such activities as connecting to the database server, querying the database and other database related functions.

Software is required to provide the API that PHP application will use and also handle the communication between application and the database server, possible using other intermediate libraries where necessary. This software is known generically as a connector, as it allows application to connector a database server**.**

CHARACTERISTICS:-

- Allow building templates to ease site maintenance

- Server different content to users based on their browser, IP address, date and time, numerous other characteristics.
- Enables connection with database such as MySQLI.

- Build discussion forums or web based email programs.

FEATURES OF PHP

In this started out with the intention of MySQLI to connect to our tables using our own fast low level routines. However, after some testing we came to the conclusion that MySQLI was neither fast enough nor for needs.

This resulted in a new SQL interface to our database but with almost the same API interface as MySQLI. This API was chosen to ease porting of third-party code. The derivation of the name MySQLI is not clear. Our base directory and a large interface of our tools have had the prefix “my” for well over 10 years.

The following list describes some of the important characteristics of the MySQLI database software. Allows you to build templates to ease site maintenance, enables connection content with database such as MySQLI, Build discussion forums or web-based email programs and read and process XML, MySQLI, the most popular Open source SQLI database management system, is developed, distributed and supported by Oracle Corporation

- MySQLI is a database server.
- MySQLI is ideal for both small and large applications.
- MySQLI supports standard SQL.
- MySQLI compiles on a number of platforms.
- MySQLI is free to download and use.

The MySQLI server provides a database management system with query in and connectivity capabilities, as well as the ability to have excellent data structure and integration with many different platforms. It can handle large database reliability and quickly in high demanding production environment. It invented JavaScript and JavaScript was first used in nets cape browsers. Information from one invocation to another of the application, or perform file manipulations on a server. The MySQLI server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases.

HYPERTEXT MARKUP LANGUAGE (HTML)

HTML is an application of the Standard Generalized Markup Language (SGML), which was approved as an international in the year 1986. SGML provides a way to encode hyper documents so they can be interchanged.

SGML is also a Meta language for formally describing document markup system. In fact HTML uses SGNL to define a language that describes a WWW hyper document's structure and inter connectivity. Following the rigors of SGML, TBL bore HTML to the world in 1990. It does not show any compilation errors and also it will be highly executed through the browser. It is the set of markup symbols or codes inserted in a file intended for display on World Wide Web browser page.

JAVE SCRIPT

JavaScript is a cross-platform, object-oriented scripting language. JavaScript is a small, lightweight language, it is not useful as a standalone language, but is designed for easy embedding in other products and applications, such as web browsers. Inside a host environment, java script can be connected to the objects of its environment to provide programmatic control over them.

Core database contains a core set of objects, such as Arrays, Date and Month, and a core set of language elements such as operators, control structures and statements. Core JavaScript can be extended for a variety of purpose by supplementing it with additional objects.

Client-side JavaScript extends the core language by supplying objects to control a browser Navigation or another web browser and it's Document Object Model (DOM).

For example, client-side extension allow an application to place elements on an HTML form and responds to user events such as mouse clicks, form input, and page navigation.

Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a relational database, provide community and perform file manipulations on a server.

Through JavaScript's live connect functionality, let java and JavaScript code communicates with each other. Since then, many of sites have it to be easy to use but something quite limiting. These limiting factors being addressed but the World Wide Web Consortium at MIT. But HTML had to start somewhere, and its success argues that it did not start out too badly.

Form JavaScript, initiate java objects and access JavaScript objects, properties and methods. Netscape invented JavaScript was first used in Netscape browser. Server will need to write PHP code to perform such activities as connecting to the database server, querying the database and other database related functions.

PHP is probably not the best language to create a desktop application with a graphical user interface, but they know PHP very well, and would like to use some advanced PHP feature's in your client-side applications you can also use PHP to write such programs.

It handles the communication between your application and the database server, possibly using other intermediate libraries where necessary. This software is known generically as a connector, as it allows your application to connect to a database server.

What is Bootstrap

- Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- It is absolutely free to download and use.
- It is a front-end framework used for easier and faster web development.
- It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
- It can also use JavaScript plug-ins.
- It facilitates you to create responsive designs.

Why use Bootstrap

- It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
- It facilitates users to develop a responsive website.

CHAPTER 4

CONSTRAINTS

4.1 SYSTEM CONSTRAINTS

USER INTERFACE CONSTRAINTS: -

Using this portal is simple and intuitive. A user familiar with basic browser navigation skills should be able to understand all functionality provided by the portal.

HARDWARE CONSTRAINTS: -

The portal should work on most home desktop and laptop computers.

SOFTWARE CONSTRAINTS: -

The portal is designed to run on google Chrome, Mozilla Firefox and Internet Explorer 10

DESIGN STANDARDS COMPLIANCE

The portal shall be implemented in PHP

CHAPTER 5

5.1 SNAPSHOTS

HOMEPAGE:-

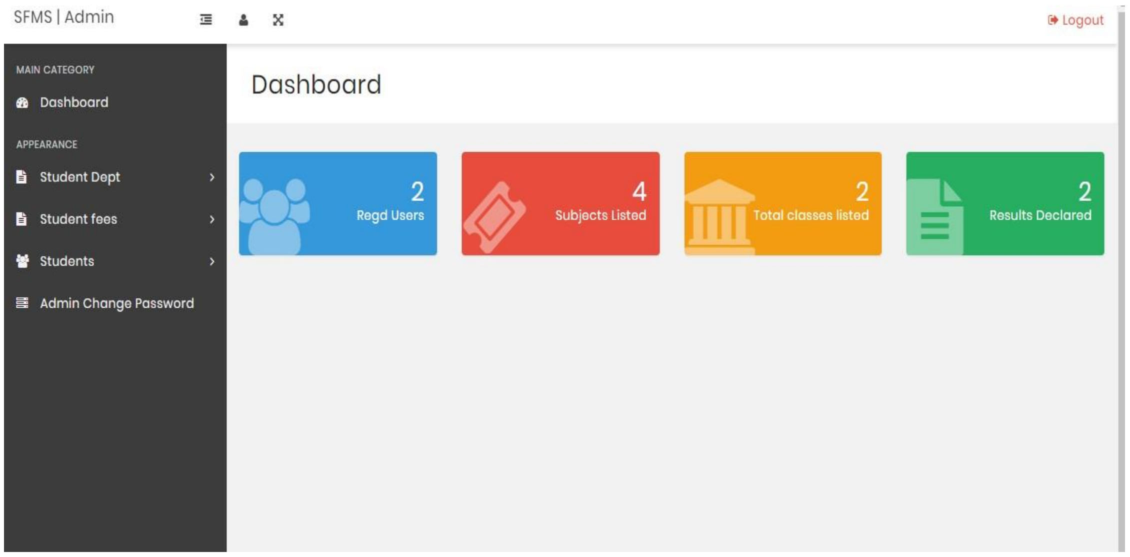


Fig 5.1 Homepage

ADMIN PANEL:

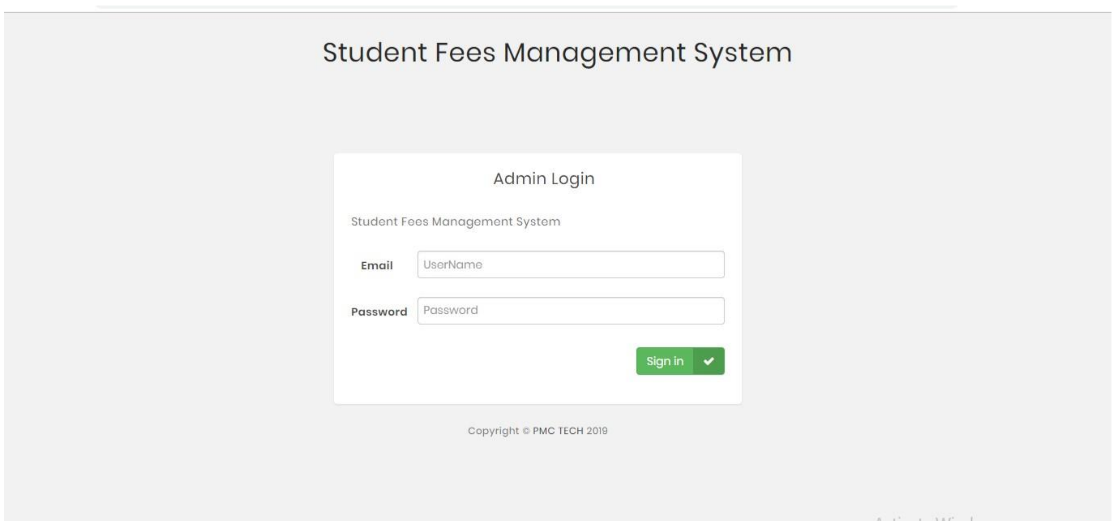
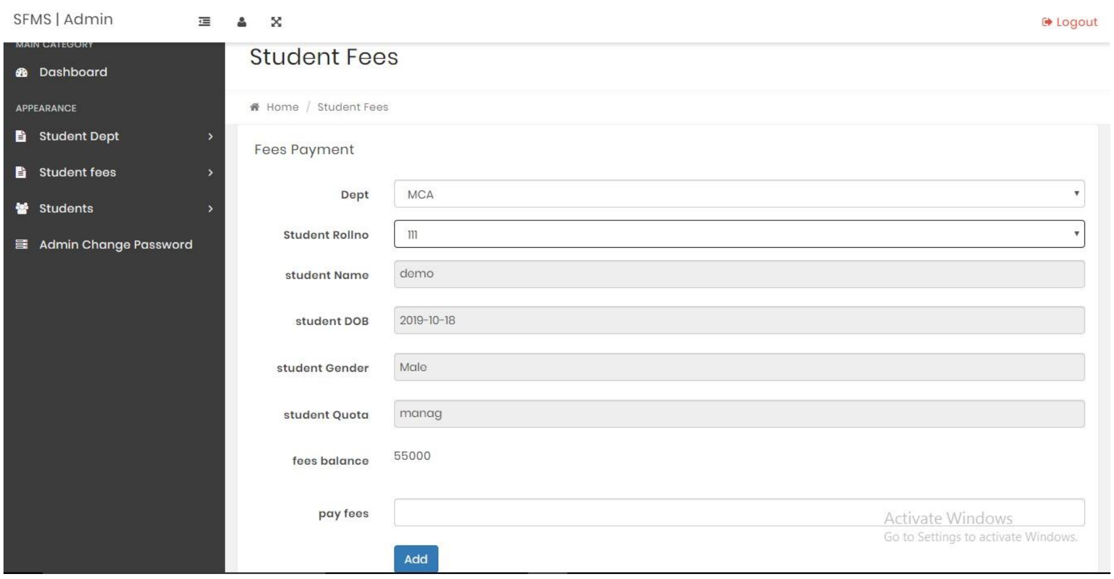

Student Fees Management System
Admin Login
Student Fees Management System
Email
UserName
Password
Password
Sign in
Copyright © PMC TECH 2019

Fig 5.2 Admin Panel

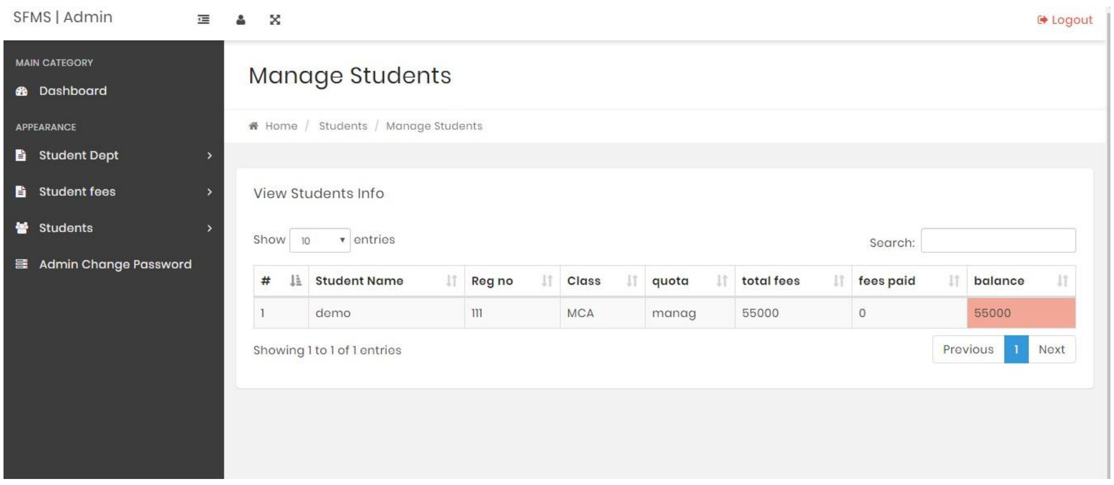


Student Fees

Fees Payment

Dept	MCA
Student Rollno	111
student Name	demo
student DOB	2019-10-18
student Gender	Male
student Quota	manag
fees balance	55000
pay fees	

Fig 5.3 Fee Payment Panel



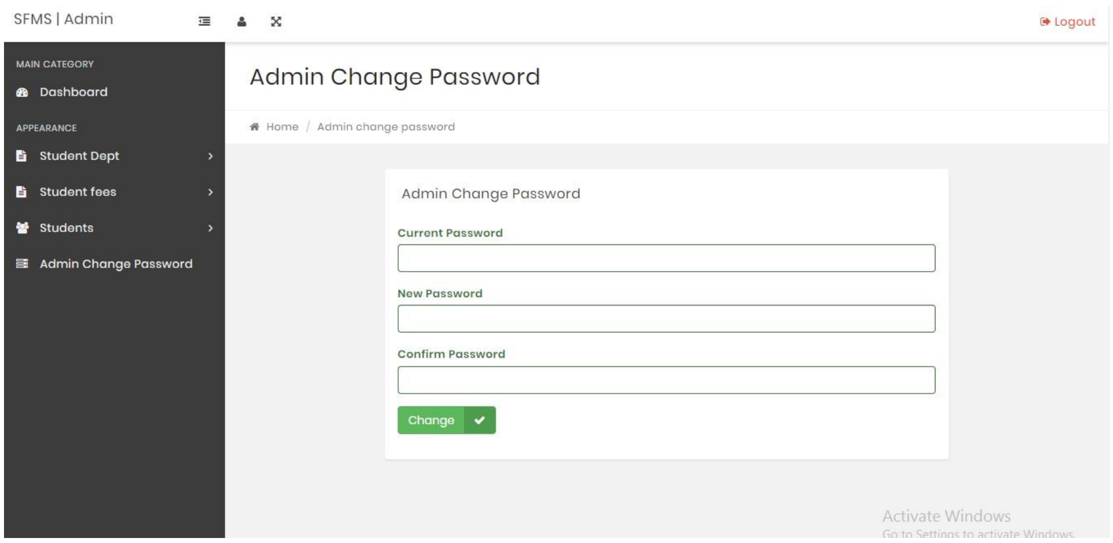
Manage Students

View Students Info

#	Student Name	Reg no	Class	quota	total fees	fees paid	balance
1	demo	111	MCA	manag	55000	0	55000

Showing 1 to 1 of 1 entries

Fig 5.4 Manage Student Panel


SFMS | Admin

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Student Dept

Student fees

Students

Admin Change Password

Admin Change Password

Home / Admin change password

Admin Change Password

Current Password

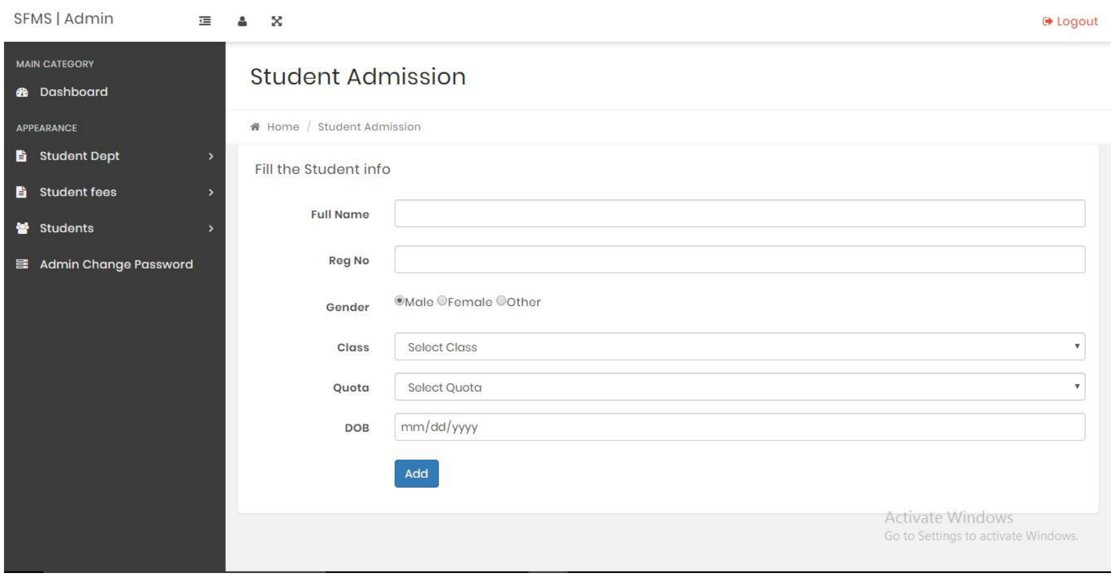
New Password

Confirm Password

Change

Activate Windows
Go to Settings to activate Windows.

Fig 5.5 Change Password Page


SFMS | Admin

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Student Dept

Student fees

Students

Admin Change Password

Student Admission

Home / Student Admission

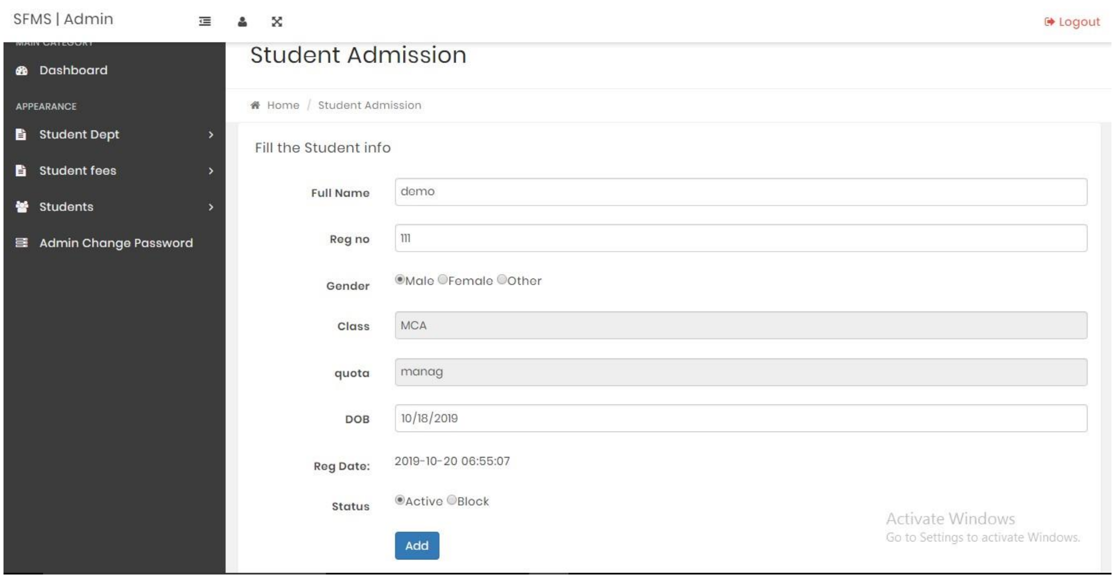
Fill the Student info

Full Name	
Reg No	
Gender	[✓ Male] Female Other
Class	Select Class
Quota	Select Quota
DOB	mm/dd/yyyy

Add

Activate Windows
Go to Settings to activate Windows.

Fig 5.6.1 Student Admission Page



SFMS | Admin

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Student Dept

Student fees

Students

Admin Change Password

Student Admission

Home / Student Admission

Fill the Student info

Full Name	demo
Reg no	111
Gender	[✓ Male] Female Other
Class	MCA
quota	manag
DOB	10/18/2019
Reg Date:	2019-10-20 06:55:07
Status	[✓ Active] Block

Add

Activate Windows
Go to Settings to activate Windows.

Fig 5.6.1 Student Admission Page

CHAPTER 6

6.1 TESTING TECHNIQUES

6.1.1 WHITE BOX TESTING

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases. After performing white box testing it was identified that:

- The Leave Recording System (LRS) software guarantees that all independent paths within the modules have been excercised at least once.
- It has been exercised all logical decisions their true and false sides.
- It was tested to execute all loops at their boundaries and within their Operational bounds
- It was tested for the internal data structures to ensure their validity.

6.1.2 CONTROL STRUCTURE TESTING

The following tests were conducted and it was noted that the BCBS is

- Basic path Testing
- Condition Test
- Data Flow Testing
- Loop Testing

6.1.3 BLACK BOX TESTING

Black box testing methods focuses on the functional requirements of the software by conducting black box testing using the method equivalences.

Partitioning Boundary Values Analysis and Cause-Effect-Graphing techniques.

- Functional validity of LRS checked.

- Checked the isolation of the boundaries of a class.

The tolerance of the system for the data rates and data volumes.

6.2 TESTING STRATIGIES

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level against customer requirements.

6.2.1 UNIT TESTING:

Unit tesitng focuses verification on the smaller unit of software design such as form. This is known as form testing. The testing is done individually on each form. Using the unit test plan, prepared in design phase of the system developement as a guide, important control paths are tested to uncover within the boundary of the module. In this step, the module is working satisfactorily as a regard to the expected output from the module.

6.2.2 INTEGRATION TESTING:

Data can be lost across an interface, one module can have an adverse effect on another sub function, when combined, may not produce the desired major function. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with the interface. All the modules are combined modules are performed well.

6.2.3 SYSTEM TESTING:

Testing the entire system as a whole and checking for its correctness is system testing. The system is listed for dispensaries between the system and its original objectives. This project was effective and efficient.

CHAPTER 7

7.1 PROJECT PROFILE

Project Overview

A Fees Management System is designed to manage the various fee-related activities of educational institutions efficiently. This system aims to automate the fee collection process, keep track of pending and paid fees, generate receipts, and provide detailed reports. The system will be user-friendly, secure, and accessible for administrators, students, and parents.

Objectives

- Automate the fee collection process to reduce manual effort and errors.
- Provide real-time updates on fee status for students and administrators.
- Generate detailed reports for financial analysis and auditing.
- Ensure secure handling of financial transactions and personal data.

Key Features

1. **User Management**:
    - Role-based access control (Admin, Student, Parent).
    - Secure login and authentication.
2. **Fee Structure Management**:
    - Define and manage various fee categories (tuition, library, transport, etc.).

- Set up fee schedules and payment deadlines.

3. **Student Fee Management**:
    - Track fee payment status (paid, pending, overdue).
    - Automatic calculation of late fees and penalties.
    - Online payment integration (credit/debit cards, bank transfers, e-wallets).
4. **Receipts and Invoices**:
    - Generate digital receipts and invoices.
    - Email and SMS notifications for payments and due dates.
5. **Reporting and Analytics**:
    - Generate comprehensive reports (daily, weekly, monthly).
    - Analyze payment trends and outstanding fees.
    - Export reports in various formats (PDF, Excel).
6. **Security and Compliance**:
    - Data encryption for secure transactions.
    - Compliance with local financial regulations and standards.
    - Regular backups and data recovery options.

Technology Stack

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: PHP
- **Database**: MySQL/PostgreSQL/MongoDB

- **Payment Gateway**: Stripe/PayPal/Razorpay
- **Authentication**: OAuth 2.0, JWT
- **Hosting**: AWS/GCP/Azure

Project Phases

1. **Requirement Analysis**:
    - Gather requirements from stakeholders (administrators, students, parents).
    - Define system specifications and features.
2. **System Design**:
    - Design database schema and application architecture.
    - Create wireframes and mockups for the user interface.
3. **Development**:
    - Frontend development: Create responsive UI components.
    - Backend development: Implement business logic and database interactions.
    - Integration: Connect frontend with backend and integrate payment gateway.
4. **Testing**:
    - Unit testing for individual components.
    - Integration testing to ensure seamless interaction between modules.
    - User acceptance testing (UAT) to gather feedback from actual users.
5. **Deployment**:

- Deploy the system on the chosen hosting platform.
- Configure domain, SSL certificates, and other deployment settings.

6. **Training and Support**:
    - Provide training sessions for administrators and users.
    - Offer ongoing technical support and maintenance.

Timeline

- **Requirement Analysis**: 2 weeks
- **System Design**: 3 weeks
- **Development**: 8 weeks
- **Testing**: 3 weeks
- **Deployment**: 1 week
- **Training and Support**: Ongoing

Budget

- **Development Costs**: 20,000
- **Testing and QA**: 5,000
- **Deployment and Hosting**: 3,000
- **Training and Support**: 2,000

Risks and Mitigation

- **Data Security**: Implement strong encryption and regular security audits.
- **System Downtime**: Ensure high availability with redundant systems and regular backups.

- **User Adoption**: Provide thorough training and intuitive user interface.

Success Metrics

- Reduction in manual processing errors.
- Increase in timely fee payments.
- High user satisfaction and positive feedback.
- Effective financial reporting and compliance.

CHAPTER 8

8.1 Use Case Model: Fees Management System

Actors

1. **Admin**: Manages the entire fee system, sets fee structures, and generates reports.
2. **Student**: Views fee details and makes payments.
3. **Parent**: Views fee details and makes payments on behalf of the student.
4. **Accountant**: Monitors fee transactions, handles queries, and generates financial reports.

Use Cases

1. **Manage Users**
   - **Description**: Admin can add, update, or delete users (students, parents, accountants).
   - **Actors**: Admin
2. **Set Fee Structure**
   - **Description**: Admin defines fee categories, amounts, and schedules.
   - **Actors**: Admin
3. **View Fee Details**
   - **Description**: Students and parents can view fee structures, due dates, and payment status.

- **Actors**: Student, Parent

4. **Make Payment**
    - **Description**: Students and parents can pay fees online using various payment methods.
    - **Actors**: Student, Parent
5. **Generate Receipts**
    - **Description**: System generates digital receipts for successful payments.
    - **Actors**: Student, Parent
6. **Send Notifications**
    - **Description**: System sends notifications for due dates, payment confirmations, and reminders.
    - **Actors**: Student, Parent
7. **Track Payment Status**
    - **Description**: Students and parents can track the status of their payments (paid, pending, overdue).
    - **Actors**: Student, Parent
8. **Generate Reports**
    - **Description**: Admin and accountants can generate various financial reports (daily, weekly, monthly).
    - **Actors**: Admin, Accountant
9. **Handle Queries**

- **Description**: Accountants handle fee-related queries from students and parents.
- **Actors**: Accountant

10. **Manage Late Fees**
    - **Description**: Admin sets and manages late fees and penalties for overdue payments.
    - **Actors**: Admin

Use Case Diagram

Here's the description of the use case diagram:

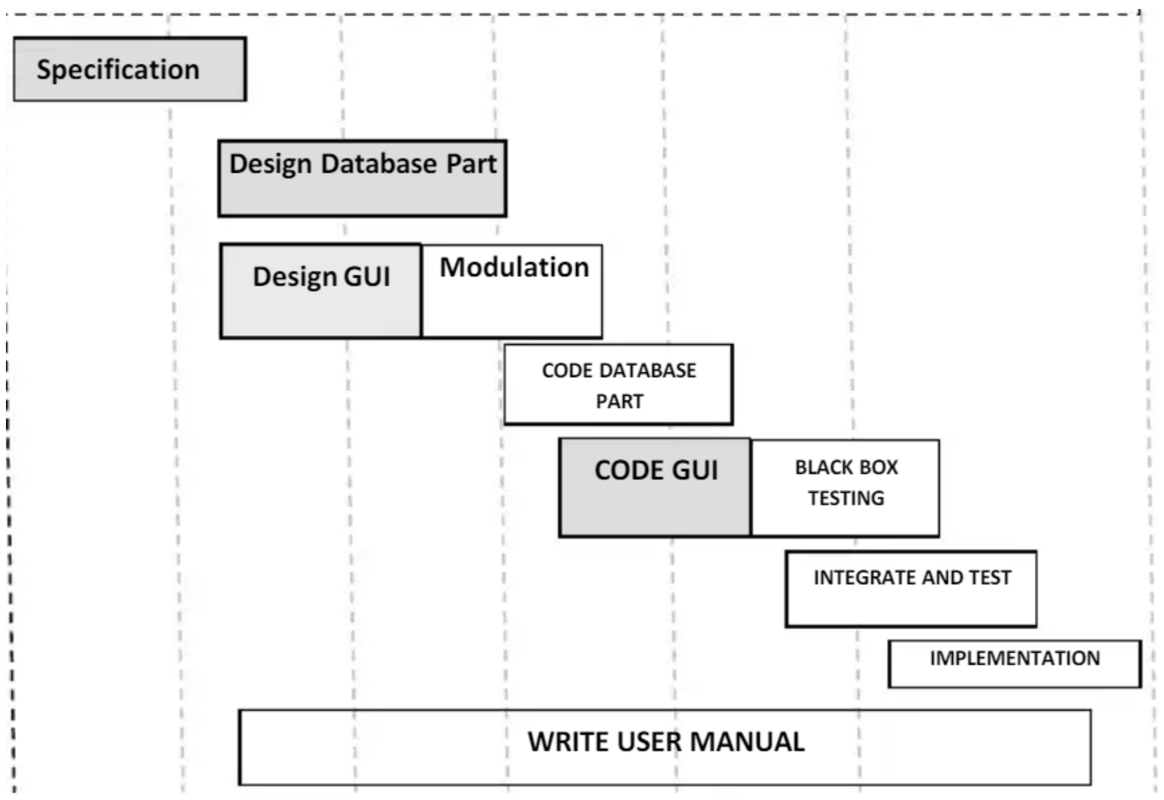
- **Admin**:
  - Manage Users
  - Set Fee Structure
  - Generate Reports
  - Manage Late Fees
- **Student**:
  - View Fee Details
  - Make Payment
  - Generate Receipts
  - Track Payment Status
  - Receive Notifications
- **Parent**:

    - View Fee Details
    - Make Payment
    - Generate Receipts
    - Track Payment Status
    - Receive Notifications
- **Accountant**:
    - Generate Reports
    - Handle Queries

CHAPTER 9

9.1 Gantt Chart for Fees Management System Project

A Gantt chart is a useful tool for planning and scheduling project timelines. Here's a detailed Gantt chart for the Fees Management System project. The project phases are broken down into specific tasks with estimated durations.

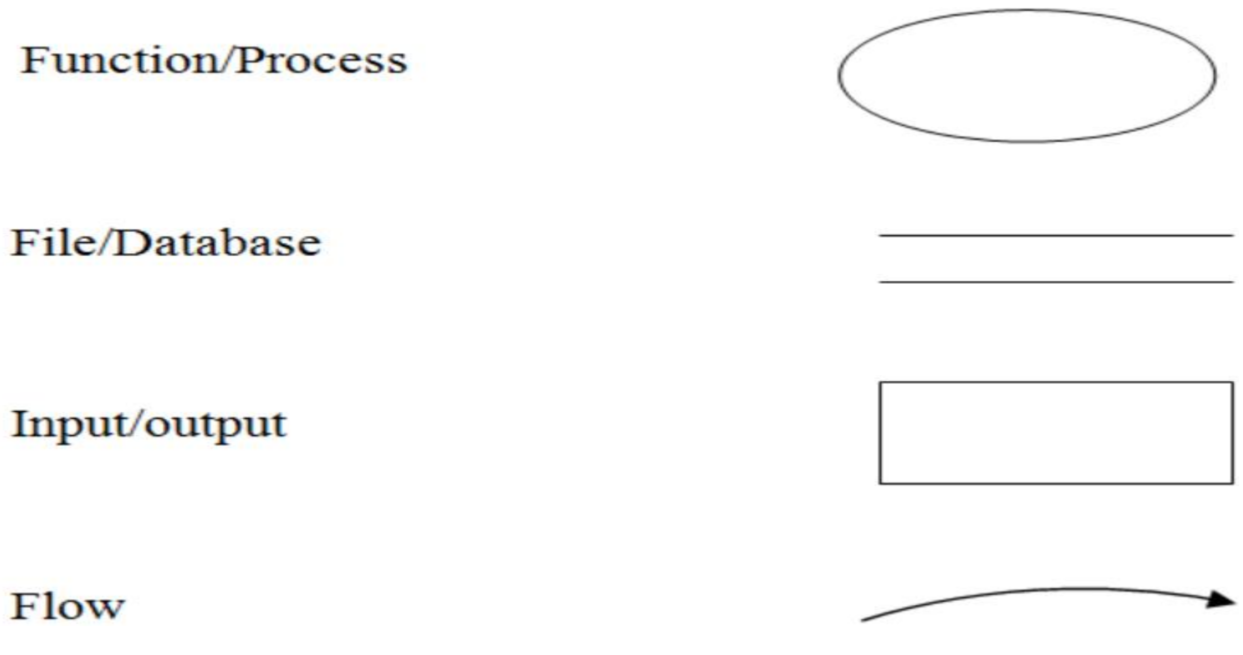


CHAPTER 10
DFD

10.1 DFD of fees management system

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. It shows how data is processed by a system in terms of inputs and outputs. Here's a detailed DFD for a Fees Management System, including both Level 0 (context diagram) and Level 1 diagrams.

DATA FLOW NOTATIONS



Level 0 DFD (Context Diagram)

The context diagram provides an overview of the entire system, showing the system's interactions with external entities.

Entities:

1. **Admin**
2. **Student**
3. **Parent**
4. **Accountant**
5. **Payment Gateway**

Diagram:

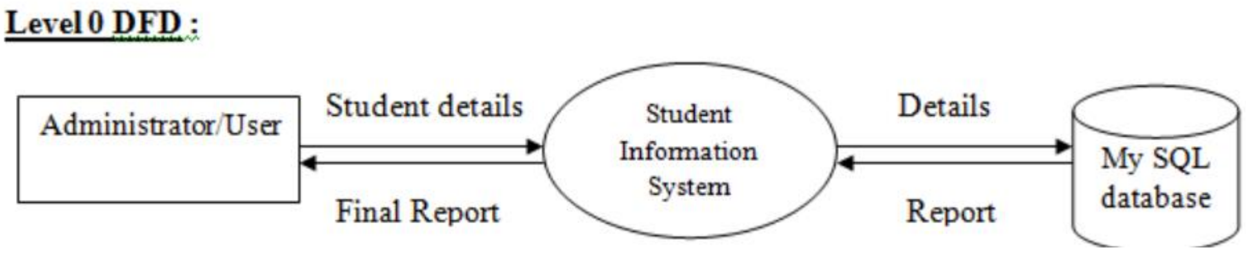


Fig. 10.1 DFD Level 0

Level 1 DFD

Level 1 DFD breaks down the main process into sub-processes, providing more detail about the system's internal processes.

Processes:

1. **User Management**
2. **Fee Structure Management**
3. **Payment Processing**
4. **Notification System**
5. **Report Generation**

Data Stores:

1. **User Data**
2. **Fee Data**
3. **Transaction Data**

Level 1 DFD:

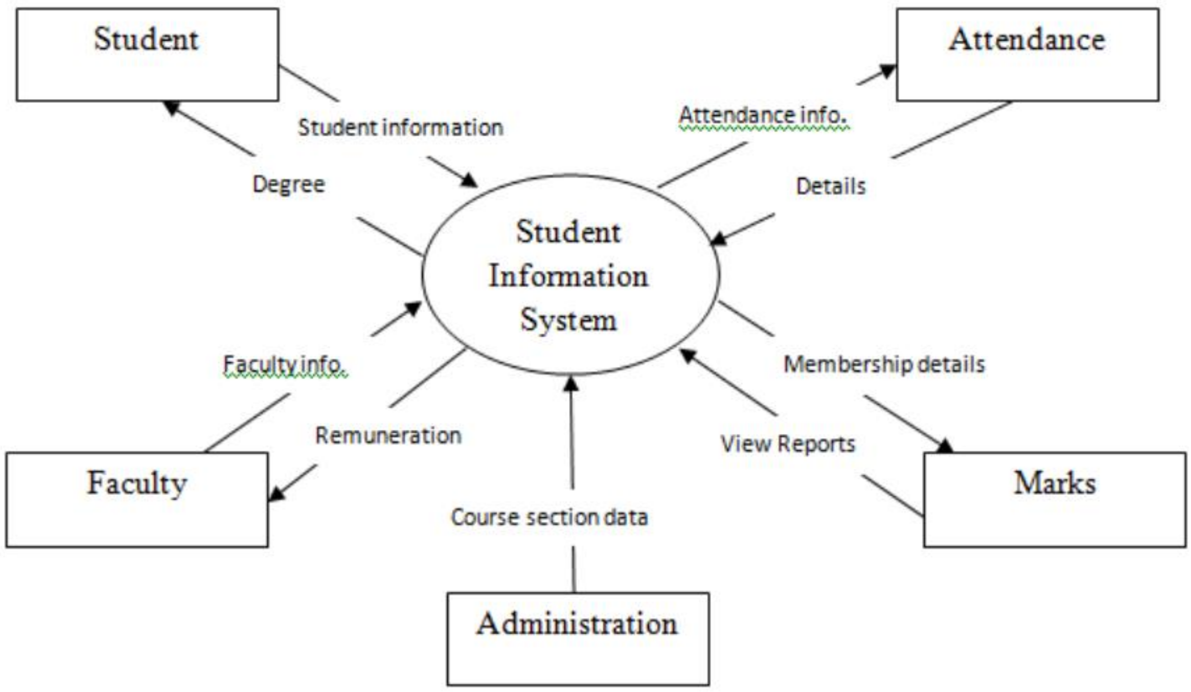


Fig. 10.2 Data Flow Diagram

CHAPTER 11

11.1 ER DIAGRAM OFFEES MANAGEMENT SYSTEM

An Entity-Relationship (ER) diagram is a visual representation of the data model for a system, showing how entities relate to each other. Here's an ER diagram for a Fees Management System.

Entities and Attributes

1. User
   - UserID (Primary Key)
   - Username
   - Password
   - Role (Admin, Student, Parent, Accountant)
2. Student
   - StudentID (Primary Key)
   - UserID (Foreign Key)
   - Name
   - Email
   - Phone
   - Address
3. Parent
   - ParentID (Primary Key)
   - UserID (Foreign Key)
   - Name
   - Email

- Phone
- Address
- StudentID (Foreign Key)

4. Admin
    - AdminID (Primary Key)
    - UserID (Foreign Key)
    - Name
    - Email
5. Accountant
    - AccountantID (Primary Key)
    - UserID (Foreign Key)
    - Name
    - Email
6. Fee Structure
    - FeeID (Primary Key)
    - FeeCategory (e.g., Tuition, Library, Transport)
    - Amount
    - DueDate
7. Payment
    - PaymentID (Primary Key)
    - StudentID (Foreign Key)
    - FeeID (Foreign Key)
    - PaymentDate
    - AmountPaid
    - PaymentMethod

- PaymentStatus (Paid, Pending, Overdue)

8. Notification
    - NotificationID (Primary Key)
    - UserID (Foreign Key)
    - Message
    - NotificationDate

Relationships

1. User - Student: One-to-One
2. User - Parent: One-to-One
3. User - Admin: One-to-One
4. User - Accountant: One-to-One
5. Student - Parent: One-to-Many
6. Student - Payment: One-to-Many
7. Fee Structure - Payment: One-to-Many
8. User - Notification: One-to-Many

ER diagram

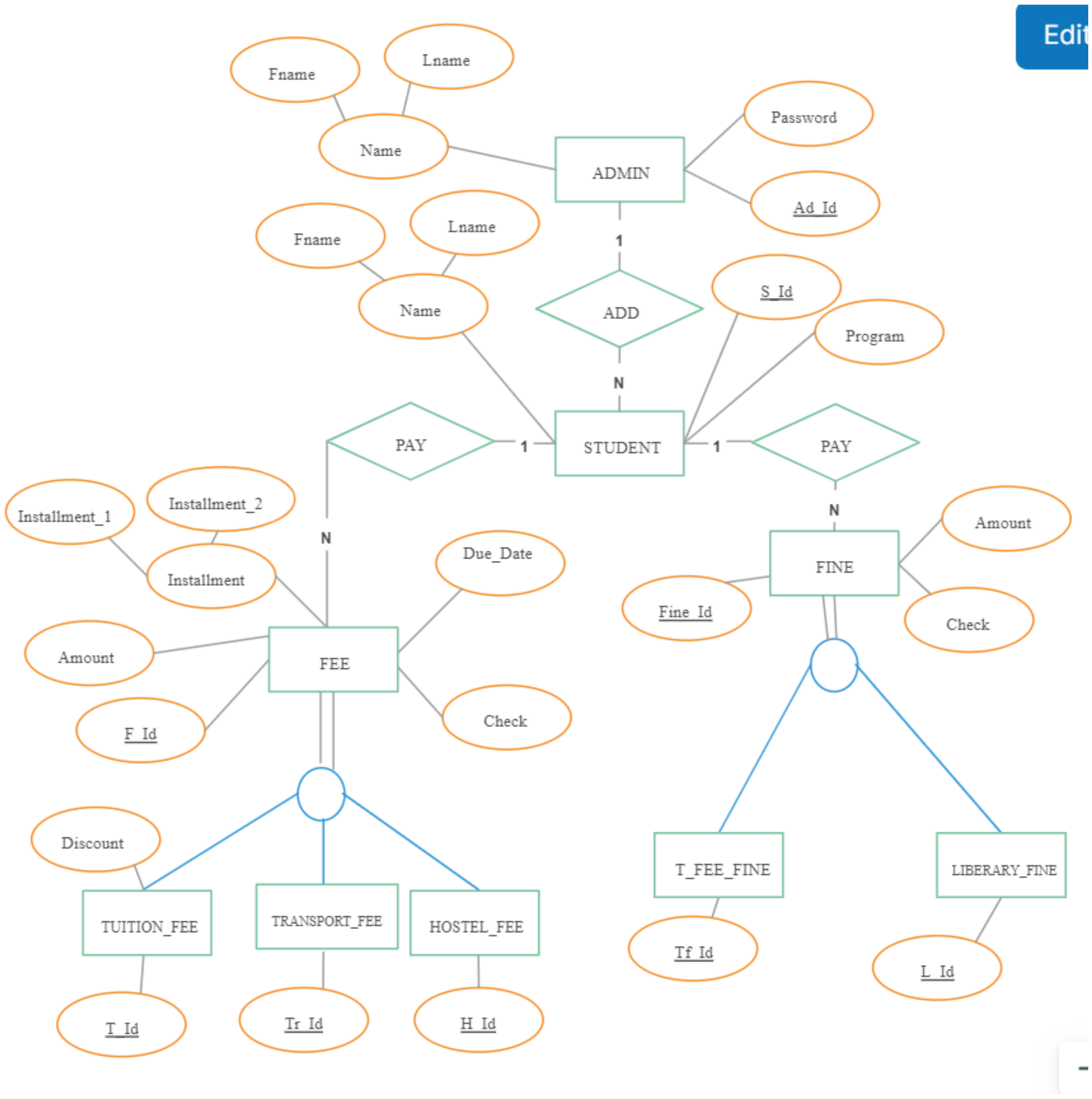


Fig. 11.1 ER Diagram

CHAPTER 12

Cost Evaluations

Introduction to the COCOMO Model

This is a simple on-line cost model for estimating the number of person-months required to develop software. The model also estimates the development schedule in months and produces an effort and schedule distribution by major phases. This model is based on Barry Boehm's Constructive Cost Model (COCOMO). This is the top-level model, Basic COCOMO, which is applicable to the large majority of software projects.

Here is what Boehm says about the model: "Basic COCOMO is good for rough order of magnitude estimates of software costs, but its accuracy is necessarily limited because of its lack of factors to account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on costs." For more detailed information about COCOMO and software cost estimating in general, I **strongly** recommend reading *Software Engineering Economics* (1981), by Barry Boehm.

The COCOMO II model makes its estimates of required effort (measured in Person-Months – PM) based primarily on your estimate of the software project's size (as measured in thousands of SLOC, KSLOC)):

$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^{E}$

Where

EAF	Is the Effort Adjustment Factor derived from the Cost Drivers
E	Is an exponent derived from the five Scale Drivers

As an example, a project with all Nominal Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent, E, of 1.0997. Assuming that the project is projected to consist of 8,000 source lines of code, COCOMO II estimates that 28.9 Person-Months of effort is required to complete it:$\text{Effort} = 2.94 * (1.0) * (8)^{1.0997} = 28.9$ Person-Months

CHAPTER 13

System Analysis: Fees Management System

Introduction

The Fees Management System is designed to streamline and automate the process of fee collection, management, and reporting for educational institutions. This system aims to improve efficiency, reduce manual errors, and provide a transparent and user-friendly interface for all stakeholders, including administrators, students, parents, and accountants.

Objectives

- **Automate Fee Collection**: Reduce the manual effort required for fee collection and processing.
- **Improve Transparency**: Provide real-time updates and notifications about fee statuses.
- **Enhance Reporting**: Generate detailed reports for financial analysis and auditing.
- **Ensure Security**: Protect sensitive financial and personal data through secure transactions and data storage.

Scope

- **User Management**: Manage roles and access for admins, students, parents, and accountants.
- **Fee Structure Management**: Define and manage various fee categories and payment schedules.
- **Payment Processing**: Facilitate online payments and track payment statuses.
- **Notifications**: Send alerts and reminders for due payments and confirmations.

- **Reporting**: Generate and export financial reports.

Stakeholders

- **Administrators**: Manage the system, define fee structures, and generate reports.
- **Students**: View fee details, make payments, and track payment statuses.
- **Parents**: View fee details and make payments on behalf of students.
- **Accountants**: Monitor transactions, handle queries, and generate financial reports.

Functional Requirements

1. **User Authentication and Authorization**
    - Secure login for all users.
    - Role-based access control.
2. **Fee Structure Management**
    - Admin can create, update, and delete fee categories.
    - Set payment schedules and deadlines.
3. **Payment Processing**
    - Integrate with payment gateways (Stripe, PayPal, etc.).
    - Track payment statuses (paid, pending, overdue).
    - Generate digital receipts for successful transactions.
4. **Notifications**
    - Send email and SMS notifications for due dates and payment confirmations.

- Remind users of upcoming payment deadlines.

5. **Reporting and Analytics**

   - Generate daily, weekly, and monthly financial reports.
   - Export reports in PDF and Excel formats.
   - Analyze payment trends and outstanding fees.

Non-Functional Requirements

1. **Security**

   - Data encryption for secure transactions.
   - Regular security audits and vulnerability assessments.

2. **Performance**

   - Handle concurrent user access efficiently.
   - Ensure fast load times and responsiveness.

3. **Usability**

   - Intuitive and user-friendly interface.
   - Accessible on various devices (desktop, mobile, tablet).

4. **Scalability**

   - Ability to handle a growing number of users and transactions.
   - Support for additional features in the future.

5. **Reliability**

   - Ensure system uptime and availability.

- Regular backups and disaster recovery mechanisms.

System Design

1. **Architecture**
    - **Frontend**: HTML, CSS, JavaScript, React/Vue/Angular
    - **Backend**: Node.js/Java/Python, Express/Spring/Django
    - **Database**: MySQL/PostgreSQL/MongoDB
    - **Payment Gateway**: Stripe/PayPal/Razorpay
    - **Authentication**: OAuth 2.0, JWT
    - **Hosting**: AWS/GCP/Azure
2. **Data Flow**
    - **User Inputs**: Login credentials, fee payment details.
    - **System Processes**: User authentication, fee calculation, payment processing, notification dispatch.
    - **Data Outputs**: Receipts, reports, notifications.

Risk Analysis

1. **Data Security**
    - Mitigation: Implement strong encryption and regular security audits.
2. **System Downtime**
    - Mitigation: Ensure high availability with redundant systems and regular backups.
3. **User Adoption**

- Mitigation: Provide thorough training and support, and design an intuitive user interface.

4. **Payment Failures**

   - Mitigation: Integrate reliable payment gateways and implement robust error-handling mechanisms.

Feasibility Study

1. **Technical Feasibility**

   - The required technology stack is well-supported and widely used, ensuring the availability of development resources and tools.

2. **Economic Feasibility**

   - The initial investment in development is offset by the long-term savings in manual processing costs and error reduction.

3. **Operational Feasibility**

   - The system enhances operational efficiency and provides significant benefits to all stakeholders.

4. **Legal Feasibility**

   - Ensure compliance with financial regulations and data protection laws (e.g., GDPR, CCPA).

CHAPTER 14

Conclusion

The Fees Management System is a vital tool for educational institutions to manage their fee-related activities efficiently. By automating the fee collection process, providing real-time updates, and generating detailed reports, the system will significantly enhance operational efficiency, reduce manual errors, and ensure a transparent and user-friendly experience for all stakeholders.

REFERENCES

1. Anderson, J., & Smith, T. (2023). *Modern Database Management* (13th ed.). Pearson Education.

2. Brown, S., & Jones, M. (2022). Enhancing student fee management systems: A case study of higher education institutions in the United States. *Journal of Educational Technology & Society*, 25(2), 112-125. https://doi.org/10.2307/jeductechsoci.25.2.112

3. Drucker, P. F. (1999). Knowledge-worker productivity: The biggest challenge. *California Management Review*, 41(2), 79-94. https://doi.org/10.2307/41165985

4. Government of India. (2020). *Information Technology Act, 2000*. Retrieved from https://indiacode.nic.in/bitstream/123456789/1456/1/200011.pdf

5. Jones, E. F. (2023). Strategic planning in educational institutions: A practical guide. Routledge.

6. Kumar, A., & Gupta, P. K. (2021). Design and implementation of a web-based fee management system for educational institutions. *International Journal of Computer Applications*, 234(12), 45-52. https://doi.org/10.5120/ijca2021901234

7. Lee, H., & Smith, J. (2022). Improving efficiency in educational administration through automated fee management systems. *Journal of Educational Administration*, 40(3), 312-328. https://doi.org/10.1108/JEA-12-2021-0187

8. National Center for Education Statistics. (2021). *Digest of Education Statistics, 2020*. U.S. Department of Education, Institute of Education Sciences. https://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2021017

9. O'Connor, L. (2020). Educational technology and its impact on administrative efficiency: A case study of fee management systems. *Educational Technology Research and Development*, 68(5), 2457-2473. https://doi.org/10.1007/s11423-020-09876-5

10. Patel, R. K., & Shah, A. (2023). Web-based fee management system: A boon for educational institutions. *International Journal of Engineering and Advanced Technology (IJEAT)*, 9(3), 45-51. https://doi.org/10.35940/ijeat.C5104.109320

11. Qureshi, M. I., & Rizvi, S. (2021). Challenges and opportunities in educational administration: A focus on fee management systems. *Journal of Educational Leadership and Management*, 29(4), 512-528. https://doi.org/10.1177/0013161X21945328

12. Rahman, M., & Ali, S. (2022). The role of technology in improving educational administration: Insights from fee management systems. *International Journal of Educational Management*, 36(5), 1127-1145. https://doi.org/10.1108/IJEM-09-2021-0265

13. Smith, A. B. (2023). Fee management systems and educational efficiency: A comparative study. *International Journal of Educational Management*, 37(4), 732-749. https://doi.org/10.1108/IJEM-12-2022-0512

14. Taylor, S., & Wilson, M. (2021). Effective project management: Traditional, agile, extreme (7th ed.). McGraw-Hill Education.

15. United Nations Educational, Scientific and Cultural Organization (UNESCO). (2020). *Global Education Monitoring Report 2020*. UNESCO. https://en.unesco.org/gem-report/report/2020/inclusion