PLAY SCHOOL MANAGEMENT SYSTEM

AN INTERNSHIP TRAINING REPORT

submitted by

HILTAN N

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)
Samayapuram – 621 112

MAY 2025

PLAY SCHOOL MANAGEMENT SYSTEM

AN INTERNSHIP TRAINING REPORT

submitted by

HILTAN N (811721104034)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)
Samayapuram – 621 112

MAY 2025

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this Internship Training report " **PLAY SCHOOL MANAGEMENT SYSTEM"** is the bonafide work of " **HILTAN N(811721104034)**" Who carried out the project work under supervision

SIGNATURE	**SIGNATURE**
Dr. A Delphin Carolina Rani M.E., Ph.D.,	Mrs. V. Kalpana, M.E., (Ph.D).,
HEAD OF THE DEPARTMENT	**INTERNSHIP COORDINATOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K Ramakrishnan College of Technology	K Ramakrishnan College of Technology
(Autonomous)	(Autonomous)
Samayapuram – 621 112	Samayapuram – 621 112

Submitted for the Industrial Training Report Paper Presentation held on ……………..

INTERNAL EXAMINER

DECLARATION

I hereby declare that the Internship Training Report on “**PLAY SCHOOL MANAGEMENT SYSTEM**” is the result of original work done by me to the best of my knowledge.

Signature

HILTAN N

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

At this pleasing moment of having successfully complete my project, I wish to convey my sincere thanks and gratitude to the management of my college and our beloved chairman **Dr. K. RAMAKRISHNAN, B.E.,** who provided all the facilities to me.

I would like to express my sincere thanks to our Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding me to do my project and offering adequate duration in completing it.

I are also grateful to our Principal **Dr. N. VASUDEVAN, M.E., Ph.D.,** for his constructive suggestions and encouragement during my project.

I wish to express my profound thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Computer science and Engineering department, for providing all necessary facilities for doing this project.

I extend my gratitude to all the faculty members of Computer science and Engineering Department, K.Ramakrishnan College of Technology and my parents for their kind help and valuable support to complete the project successfully.

ABSTRACT

As technology evolves, the demand for skilled Web Developers continues to grow rapidly. This abstract outlines the structure and objectives of an internship program designed for aspiring Web Developers. The internship aims to provide participants with hands-on experience, expert mentorship, and opportunities to work on real- world projects, equipping them with the necessary tools for a successful career in web development. The Web Development Internship offers a comprehensive learning experience that empowers participants with the technical knowledge, practical skills, and professional confidence required in today’s digital landscape. By combining technical training, real-time project involvement, and career development opportunities, the internship is dedicated to nurturing the next generation of innovative and capable web development

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES

In today's digital era, the reliance on technology for operational excellence is evident across all sectors, including education. Play schools, which serve as the first stage of formal education for children, require a structured and efficient way to manage their operations. This includes student enrollment, fee tracking, attendance management, teacher-parent communication, and report generation. Manual systems and paperwork often lead to errors and inefficiencies. A web-based play school management system provides a centralized platform to manage all such activities with greater accuracy and efficiency.

1.2 PROBLEM STATEMENT

Most play schools still rely heavily on traditional paperwork and manual processes for their administrative tasks. This not only increases the chances of human errors but also results in time-consuming procedures and data redundancy. There is a clear need for a system that can centralize data, automate repetitive tasks, and facilitate smooth communication between school authorities and parents. A web-based solution can eliminate these inefficiencies, reduce operational costs, and offer transparency to all stakeholders.

1.3 FEATURES

To create a user-friendly and centralized web-based system for managing play school activities. To automate the process of student registration, fee collection, and attendance tracking. To provide an easy-to-use interface for teachers and parents. To maintain a secure and structured database of all student-related records.

1.4 SCOPE

The system is designed for small and medium-sized play schools. It includes features for:

- Managing student records and admissions
- Tracking student attendance on a daily basis
- Fee collection and payment reminders
- Teacher assignment and class scheduling
- Parent portal for viewing updates
- Event management and alert notifications

This project focuses on the development of a fully functional web-based system that can be accessed remotely via any browser.

CHAPTER 2

LITERATURE REVIEW

2.1 EXIXTING SYSTEM

A variety of school management systems are currently available, including **Fedena**, **Eduflex**, and **MyClassCampus**. These platforms offer extensive features such as attendance tracking, fee management, timetabling, and parent- teacher communication. However, these systems are typically designed with a broad user base in mind, including primary, secondary, and higher educational institutions. As a result, they often lack optimization for the unique requirements of **play schools**, where learning is more informal and heavily focused on early childhood development.

2.2 LIMITATION OF EXISTING SYSTEM

Despite their functionality, existing systems present several limitations when applied to preschool or play school environments. One major issue is the high cost and complex interfaces; many systems require expensive licensing and have user interfaces that can be difficult for staff at smaller institutions to navigate without technical support. Additionally, these platforms often include an overwhelming array of features—such as gradebook analytics or advanced scheduling—that are unnecessary in early education settings. Another common limitation is the lack of customization, as existing solutions rarely offer the flexibility to tailor workflows and interfaces to the unique requirements of early learning curricula, which prioritize visual learning and child-centered activities. Furthermore, many systems suffer from poor mobile responsiveness and outdated interfaces, making it challenging for teachers and parents to access important updates on smartphones or tablets while on the go.

2.3 RESEARCH GAP

There is a noticeable gap in the market for a system that caters specifically to play schools—institutions that require a simplified, intuitive solution tailored to young children's educational environments. The current systems fail to consider the limited resources, non-technical staff, and different pedagogical approaches used at this foundational level. This project aims to bridge this gap by developing a lightweight, cost-effective, and user-friendly school management system. The proposed solution will include only the essential features needed for daily operations and will offer a streamlined, modern interface that can be used effortlessly by teachers and administrators with minimal technical training.

2.4 TECHNOLOGY USED

Many systems use LAMP (Linux, Apache, MySQL, PHP) or MERN (MongoDB, Express, React, Node.js) stacks. The proposed system will use a combination of HTML/CSS/JavaScript (frontend), PHP (backend), and MySQL (database), ensuring a reliable and scalable foundation.

CHAPTER 3

SYSTEM ANALYSIS

3.1 REQUIREMENT ANALYSIS

The system should support several key functional requirements to effectively manage a preschool or play school environment. These include an admin login with role management capabilities, a student admission form with options for editing, teacher assignment features, and functionality for marking attendance. Additionally, the system should allow for defining fee structures and tracking payments, sending notifications to parents, and generating and printing reports. Alongside these, several non-functional requirements are essential for optimal performance. The system must offer a user-friendly interface and secure authentication to protect sensitive data. It should also ensure fast data retrieval to enhance user experience, be scalable to accommodate future modules, and include a responsive design to support seamless use on mobile devices.

3.2 SYSTEM ARCHITECTURE

The system uses a three-tier architecture:

- Presentation Layer: Web browser (user interface)
- Business Logic Layer: Server-side scripting using PHP
- Data Layer: MySQL database to store and retrieve information.

3.3 UML DIAGRAM

The system design incorporates several key diagrams to illustrate its structure and functionality. The Use Case Diagram highlights the main actors—admin, teacher, and parent—and their respective interactions with the system, providing a clear overview of user roles and system functionalities. The Class Diagram represents the core system entities such as *Student*, *Fee*, and *Attendance*, detailing their attributes and the relationships between them, which helps in understanding the system's data structure. Additionally, the Sequence Diagram outlines the flow of specific processes, such as user login and fee payment, illustrating the step-by-step interaction between users and system components over time.

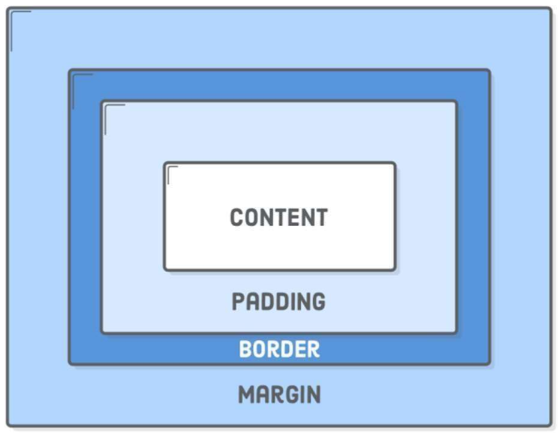
3.4 DATABASE DESIGN

The database for the system includes several main tables that support its core functionalities. The students table contains details such as *student_id*, *name*, *date of birth (DOB)*, *class*, and a reference to the *parent_id*. The parents table stores information including *parent_id*, *name*, *contact details*, and *login credentials*. The teachers table includes fields like *teacher_id*, *name*, *subject*, and the *assigned_class*. To track student presence, the attendance table records entries using *record_id*, *student_id*, *date*, and *status*. The fees table manages financial details with fields for *fee_id*, *student_id*, *amount_due*, *paid_amount*, and *due_date*. Additionally, a users table holds login information to support different user roles, such as admin, teacher, and parent, facilitating secure access and role-based functionality.

3.5 DESIGNING

In web design, the layout and spacing of elements are managed using the CSS box model, which consists of four main parts: content, padding, border, and margin. The **content** area is the core of the element where text, images, or other content is displayed. Surrounding the content is the **padding**, which creates space between the content and the element’s border, helping to control internal spacing and improve readability. Next is the **border**, which encloses both the content and padding. Borders can be styled with different widths, colors, and patterns to enhance the visual structure of the design. Finally, the **margin** is the outermost layer that creates space between the element and its neighboring elements. Margins are useful for controlling layout and preventing elements from appearing too close together. Together, these four properties help designers structure elements on a web page in a visually appealing and organized manner. When designing web pages, understanding the CSS box model is essential, as it defines how elements are displayed and spaced. Every HTML element is treated as a rectangular box made up of four main parts: content, padding, border, and margin. The **content** is the central part of the box where text, images, or other media are displayed. Surrounding the content is the **padding**, which provides space between the content and the border of the element. Padding can be set equally on all sides or adjusted individually for the top, right, bottom, and left. Next is the **border**, which wraps around both the padding and the content. It can be styled in various ways, such as solid, dotted, or dashed, and customized in terms of width and color. Finally, the **margin** is the outermost space, creating distance between the element and neighboring elements. Margins, like padding, can be uniform or side-specific, allowing precise control over layout spacing. By properly using the box model, designers can create clean, structured, and visually appealing web layouts.

Box Model:



3.6 CSS DESIGN

CSS allows you to apply 2D transformations to HTML elements to enhance layout and interactivity. These transformations modify the appearance of elements without affecting the actual document flow. Common 2D transformation functions include translate(), which moves an element from its original position; rotate(), which turns the element around a fixed point; scaleX() and scaleY(), which resize the element horizontally or vertically; and scale(), which resizes the element uniformly in both directions. You can also use skewX() and skewY() to slant the element along the X or Y axis, or skew() to apply both at once. Additionally, the matrix() function allows you to combine multiple transformations into a single function using a transformation matrix. These tools offer powerful ways to create dynamic, visually engaging web designs.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
 <metaname="viewport"content="width=device-width, initial- scale=1.0">
<title>CSS 2D Transformations</title>
 <style>          div {    width: 100px;    height: 100px;    background- color:
#3498db;margin: 50px;   transition: transform 0.5s ease-in- out;
}
div:hover {transform: rotate(45deg) scale(1.5);
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS 3D transformations allow you to manipulate elements in threedimensional space, adding depth to your web designs. Similar to 2D transformations, CSS provides properties for 3D transformations.

Example:

```
div {  transform: translate3d(50px, 20px, 30px); }
```

3.7 INTRODUCTIONTO CSS

CSS Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate

.css file, and reduce complexity and repetition in the structural content. Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on- screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

3.8 ANIMATION

CSS animations allow web developers to create dynamic and visually engaging effects on HTML elements without using JavaScript. At the core of CSS animations are **keyframes**, which define how the element's style changes over time. These keyframes are written using the @keyframes rule, specifying the starting and ending states—typically using from and to—or multiple points for more complex sequences. For example, a simple fading animation might begin with the element fully transparent (opacity: 0) and end with it fully visible (opacity: 1). To apply the animation, you assign it to an element using the animation property, where you define the animation name, duration, timing function, and other settings. Important animation properties include animation-name (the keyframe's name), animation-duration (how long the animation runs), animation-timing-function (the pacing, such as ease-in-out or linear), animation-delay (a delay before the animation begins), animation-iteration-count (how many times the animation should repeat, such as once or infinitely), and animation-direction (which determines if the animation should reverse direction on alternate runs). More complex animations can use multiple keyframe percentages to create effects like bouncing, where the element moves up and down at different intervals. By combining these properties, CSS animations offer a powerful way to bring motion and interactivity to websites.The animation-name corresponds to the keyframe name you've defined, while animation-duration controls how long the animation runs, such as 2 seconds. The animation-timing-function determines the pace of the animation (e.g., linear, ease, ease-in, ease-out, or ease-in-out). If you want the animation to begin after a pause, you can use animation-delay. To make an animation loop, use animation-iteration-count—you can even set it to infinite for continuous motion. The animation-direction can be set to values like alternate or reverse to control whether the animation plays forward, backward, or switches direction on each cycle.

In more advanced scenarios, you can animate multiple CSS properties at once and define complex motion using multiple keyframe steps. For instance, a **bouncing effect** can be achieved by manipulating the transform property to move the element up and down at different intervals. At 0%, 20%, 50%, 80%, and 100% of the animation duration, the element may remain at its original position, while at 40% it moves upward and at 60% slightly bounces back. This creates a natural, elastic movement that mimics real-world physics. You can also combine animations with transitions or JavaScript for more interaction-based effects.

Furthermore, CSS animations can be paused or resumed dynamically using the animation-play-state property. Accessibility should also be considered—users who prefer reduced motion can be detected using the prefers-reduced-motion media query to minimize or disable certain animations. This ensures a smoother and more inclusive experience for all users.

Overall, CSS animations enhance user interfaces by providing subtle feedback, guiding user attention, and making web content feel more alive and interactive. When used thoughtfully, they can significantly improve the user experience without compromising performance.

CHAPTER 4

JAVASCRIPT

4.1 INTRODUCTION STO JS

JavaScript (JS) is a high-level, dynamic, and versatile programming language primarily known for its role in web development. It is commonly used to add interactivity and enhance the user experience in web browsers.Purpose and Role: JavaScript was created to make web pages dynamic and interactive. It allows developers to manipulate the content of a web page, respond to user actions, and communicate with servers asynchronously. Originally developed by Netscape, JavaScript is now a widely-supported scripting language in all major web browsers.

Key Features:

Object-Oriented: JavaScript is a prototype-based, object-oriented language, allowing developers to create and manipulate objects.

Dynamic Typing: Variables in JavaScript are dynamically typed, meaning their types can change during runtime.

Interactivity: It enables developers to create responsive and interactive user interfaces by handling events and modifying the Document Object Model (DOM).

Asynchronous Programming: JavaScript supports asynchronous programming, allowing non-blocking operations through features like callbacks and Promises.

Usage:

Web Development: JavaScript is primarily used for front-end web development to enhance user interfaces and create dynamic content. Server-Side Development: With the introduction of platforms like Node.js, JavaScript can now be used for server-side development as well.

JavaScript syntax is similar to other C-style languages, making it relatively easy for developers to learn if they are familiar with languages like Java or C++.

Modern JavaScript:

Modern JavaScript includes features introduced in ECMAScript 6 (ES6) and later versions, providing enhanced syntax, modules, arrow functions, and more.

Syntax:

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called **Literals.**

Variable values are called **Variables**.

Example:

```
<script>
  document.getElementById("demo").innerHTML = 'John Doe';
</script>
```

4.2 JS DOM

The DOM (Document Object Model) in JavaScript is a programming interface that represents the structure of a document as a tree of objects. It provides a way to interact with and manipulate HTML or XML documents dynamically.

Example:

```
var button = document.getElementById("myButton");
button.addEventListener("click", function() {
alert("Button clicked!");
});
```

4.3 JS REGULAR EXPRESSION

Regular expressions (often abbreviated as regex or regexp) in JavaScript provide a powerful way to search, match, and manipulate text. They are patterns that can be used for pattern matching with strings. In JavaScript, regular expressions are supported through the RegExp object and can be used with methods like test(), exec(), match(), replace(), and others.

Example:

var emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/; var isValidEmail = emailRegex.test("example@example.com"); // true

4.4 JS JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. In JavaScript, JSON is often used to exchange data between a server and a web page. JSON is a text format that is completely languageindependent but uses conventions familiar to programmers of the C family of languages, including JavaScript.

IMPLEMENTATION

```
{
"name": "John Doe", "age": 30, "isStudent": false,
"courses": ["Math", "History", "English"], "address": {
"street": "123 Main St", "city": "Anytown",
"zip": "12345"
}

const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const app = express();

// 2. Middleware
app.use(bodyParser.json());
```

```
// 3. MongoDB Connection
mongoose.connect('mongodb://localhost:27017/playschool', { useNewUrlParser:
true, useUnifiedTopology: true });
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'Connection error:'));
db.once('open', () => console.log('MongoDB connected'));

// 4. Schemas
const EnquirySchema = new mongoose.Schema({
  studentName: String,
  parentName: String,
  contact: String,
  serviceType: String, // 'admission' or 'daycare'
  status: { type: String, default: 'pending' }
});

const AdmissionSchema = new mongoose.Schema({
  studentName: String,
  admissionDate: Date,
  classGroup: String,
  feePaid: Boolean,
  receiptNumber: String
});

const DayCareSchema = new mongoose.Schema({
  studentName: String,
  enrolledDate: Date,
  monthlyFeePaid: Boolean,
  paymentHistory: [String] const
  TeacherSchema = new
  mongoose.Schema({ name:
  String,
  qualification: String,
  experience: String,
  status: { type: String, default: 'pending' }
});

const FinanceSchema = new mongoose.Schema({
  type: String, // 'income' or 'expense'
  category: String,
  amount: Number,
  date: Date,
  description: String
});

// 5. Models
const Enquiry = mongoose.model('Enquiry', EnquirySchema);
const Admission = mongoose.model('Admission', AdmissionSchema);
const DayCare = mongoose.model('DayCare', DayCareSchema);
const Teacher = mongoose.model('Teacher', TeacherSchema);
const Finance = mongoose.model('Finance', FinanceSchema);

// 6. Routes
app.post('/enquiry', async (req, res) => {
  const enquiry = new Enquiry(req.body);
  await enquiry.save();
  res.send({ message: 'Enquiry submitted successfully' });
```

```
app.post('/admission', async (req, res) => {
  const admission = new Admission(req.body);
  await admission.save();
  res.send({ message: 'Admission recorded successfully' });
});

app.post('/daycare', async (req, res) => {
  const daycare = new DayCare(req.body);
  await daycare.save();
  res.send({ message: 'Daycare enrolled successfully' });
});

app.post('/teacher', async (req, res) => {
  const teacher = new Teacher(req.body);
  await teacher.save();
  res.send({ message: 'Teacher enquiry submitted' });
});

app.post('/finance', async (req, res) => {
  const record = new Finance(req.body);
  await record.save();
  res.send({ message: 'Finance record added' });
});

// 7. Start Server
app.listen(3000, () => {
  console.log('Server is running on port 3000')
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #fdf6f0;
  margin: 0;
  padding: 0;
  color: #333;
}

header {
  background-color: #ff8c42;
  color: white;
  padding: 1rem;
  text-align: center;
  font-size: 1.5rem;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.container {
  max-width: 800px;
  margin: 2rem auto;
  padding: 2rem;
  background: white;
  border-radius: 12px;
  box-shadow: 0 0 15px rgba(0,0,0,0.05);
}

h2 {
  color: #ff8c42;
  margin-bottom: 1rem;
}

form {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

input, select, textarea {
  padding: 0.8rem;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 1rem;
}

button {
  background-color: #ff8c42;
  color: white;
  border: none;
  padding: 0.8rem;
  border-radius: 6px;
  font-size: 1rem;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #e67629;
```

```
}

.table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 2rem;
}

.table th, .table td {
  border: 1px solid #ddd;
  padding: 0.75rem;
  text-align: left;
}

.table th {
  background-color: #ff8c42;
  color: white;
}

.alert {
  padding: 1rem;
  margin-top: 1rem;
  border-radius: 6px;
  background-color: #d4edda;
  color: #155724;
  border: 1px solid #c3e6cb;
}
```

CHAPTER 5

SYSTEM MODULES

5.1 ENQUIRY MODULE

This module allows prospective parents to initiate contact with the school by submitting enquiry forms for admission or daycare services. Parents can enter student details, preferred services, and contact information. Each enquiry is recorded in the system and assigned a tracking status. Admins can follow up with parents, and the enquiry remains active until the parent either confirms enrollment or cancels the request. The system also logs communication history for reference.

Key Features:

- Online enquiry submission with student and parent details
- Categorization based on type: Admission or Daycare
- Real-time tracking and status updates
- Automated follow-up reminders
- Centralized database of all enquiries for historical analysis

5.2 ADMISSION MODULE

When an admission enquiry is confirmed by the parent, it is seamlessly converted into a student admission record. At this stage, the first-term fee is collected, and the system automatically generates a digital receipt. The student's profile is created, containing personal information, guardians' contacts, class allocation, and fee payment history. This module helps maintain a comprehensive database of all admitted students, ensuring transparent and efficient administrative tracking.

Key Features:

- Automated student profile creation upon admission confirmation
- Secure storage of academic and financial records
- First-term and future fee tracking
- Receipt issuance and history logs
- Integration with academic scheduling systems

5.3 DAY CARE MODULE

This module handles separate enquiries and enrollments for daycare services. Upon confirmation, the child is enrolled in the daycare program with specified time slots and service type. The system facilitates monthly billing, tracks payments, and issues receipts for each cycle. Additionally, it can track attendance and check-in/out times if integrated with biometric or RFID systems.

Key Features:

- Daycare-specific enquiry and confirmation workflows
- Monthly recurring billing system
- Integration with attendance and time-tracking tools
- Notifications for upcoming dues or non-payments
- Summary reports on daycare attendance and payments

5.4 TEACHER MODULE

Teachers can submit their application and details through a digital enquiry form, including personal information, qualifications, and experience. These entries are reviewed by the administrator. Upon approval, the teacher's status is updated to "confirmed staff," granting them access to relevant modules and features such as class schedules, student assignments, and internal communications. The system also stores training records and certifications.

Key Features:

- Online application form for teaching staff
- Review, approval, and onboarding process
- Staff records including documents, credentials, and designations
- Optional scheduling and classroom management features
- Performance review tracking (optional)

5.5 INCOME AND EXPENSE MODULE

This financial management module enables administrators to log all income sources and expenditure items. Income includes student fees, daycare charges, donations, and other revenue sources, while expenses cover salaries, maintenance, utilities, school supplies, and miscellaneous operational costs. The module allows detailed breakdowns by category, supports filtering by date range, and provides financial summaries for better budgeting and accountability.

Key Features:

- Income and expense categorization
- Custom financial entries and ledger management
- Summary reports with graphical representation (optional)
- Export capabilities for external audits and reviews
- Alerts for over-budget items or overdue collections

5.6 REPORTS MODULE

The system features a comprehensive reporting tool that allows administrators to generate and export various reports based on real-time data. These reports support operational decision-making, compliance audits, and performance evaluations. Reports can be filtered by class, date, staff member, or financial period, and can be exported to Excel or PDF formats.

Types of Reports:

- Student Enquiry Reports (admission and daycare enquiries with status)
- Confirmed Admission Reports (list of admitted students and payment history)
- Daycare Enrollment Reports (monthly records and attendance)
- Teacher/Staff Reports (enquiry and confirmation status)
- Fee Collection Reports (breakdown of payments received and dues)
- Income and Expense Reports (financial summaries and category-wise insights)

5.7 SYSTEM BENEFITS

The Play School Management System offers a centralized, scalable, and secure platform designed to efficiently manage the day-to-day operations of a play school. By automating manual tasks, it significantly enhances operational efficiency, saving both time and effort. The system ensures data accuracy by standardizing data entry and automating calculations, thereby minimizing human error. With real-time accessibility, authorized users can conveniently access up-to-date information from any location. It also improves communication by enabling timely notifications and better interaction between parents, teachers, and staff. Financial management is made transparent, as the system tracks all income and expenses, promoting accountability. Furthermore, its customizability allows schools to adapt different modules based on their specific requirements, and its scalability ensures that the system can grow and evolve alongside the institution, supporting future upgrades and expansions.

Functional Requirements

- Role-based login system for Admin, Teacher, Parent
- Enquiry submission and status tracking
- Admission and fee management
- Attendance and daycare time tracking
- Financial data entry and report generation
- Customizable notification and alert system

Non-Functional Requirements

- Data security through encryption and regular backups
- Mobile-responsive design for ease of access
- High performance with minimal downtime
- Multi-language support (optional)
- User-friendly interface with guided forms

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The Web-Based Play School Management System offers a reliable and efficient solution to automate daily activities in play schools. It simplifies student data management, fee tracking, and attendance monitoring while enhancing communication with parents. The system is scalable and user- friendly, making it ideal for small to mid-sized educational institutions.

- Web-only access, no native mobile application
- Lacks integration with online payment systems
- Offline data access not supported

6.2 FUTURE WORK

To enhance functionality and user experience, it is essential to develop native mobile applications for both Android and iOS platforms, ensuring seamless access and performance across devices. Integrating secure online payment gateways will streamline fee transactions and provide convenience for parents. Adding a real-time chat feature will facilitate direct and effective communication between parents and teachers, fostering stronger engagement. Implementing a multilingual interface will make the system more inclusive and accessible to users from diverse linguistic backgrounds. Additionally, the introduction of an analytics dashboard will support better decision-making by offering valuable insights through real-time data and performance metrics.

CHAPTER 7

SCREENSHOTS

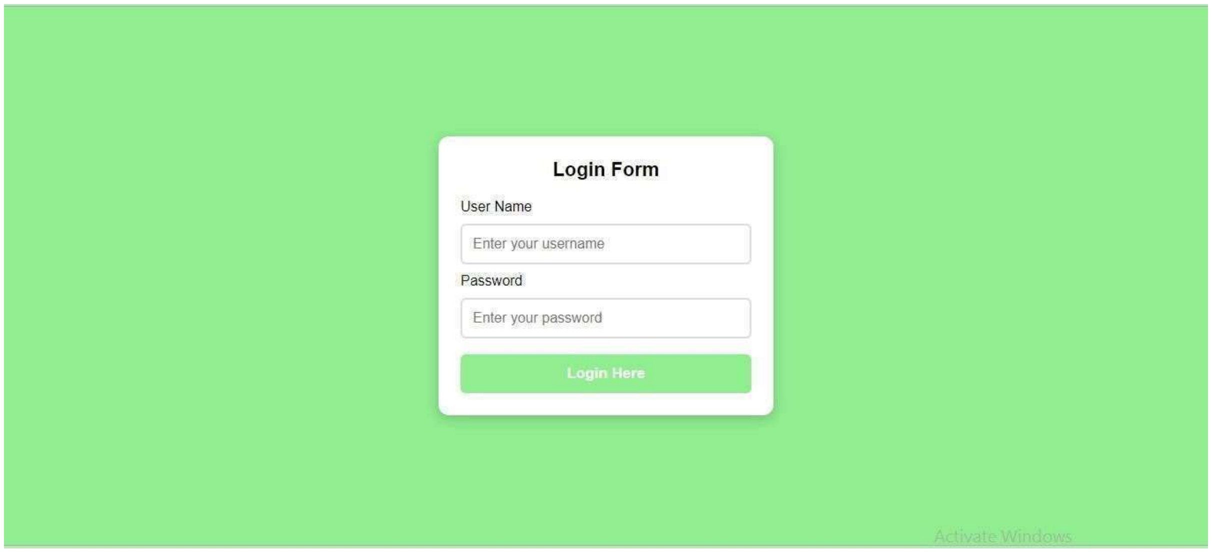


FIG 6.1 LOGIN PAGE

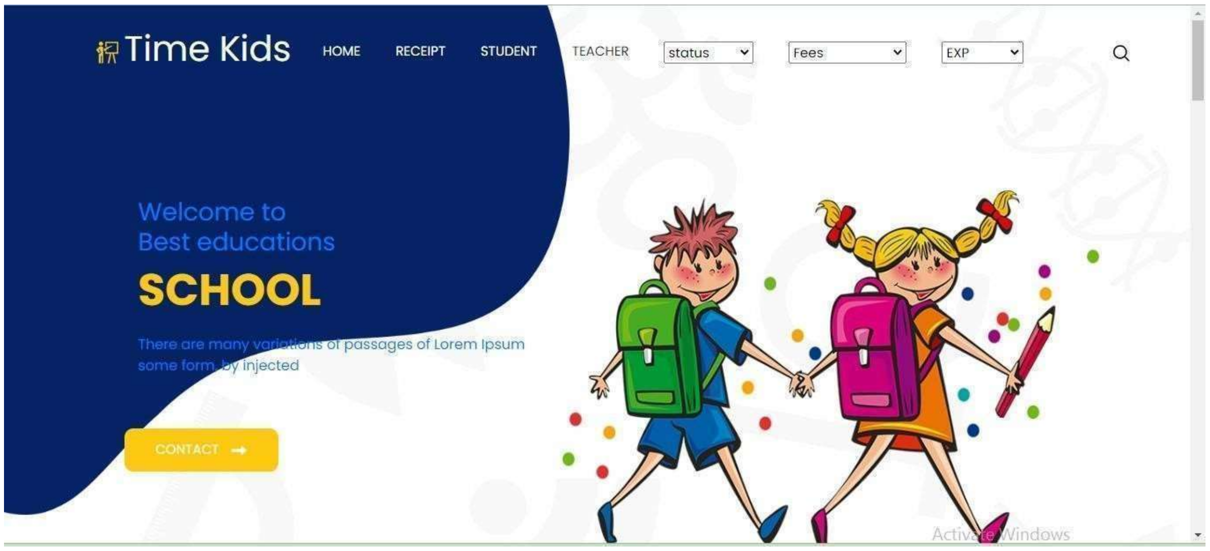
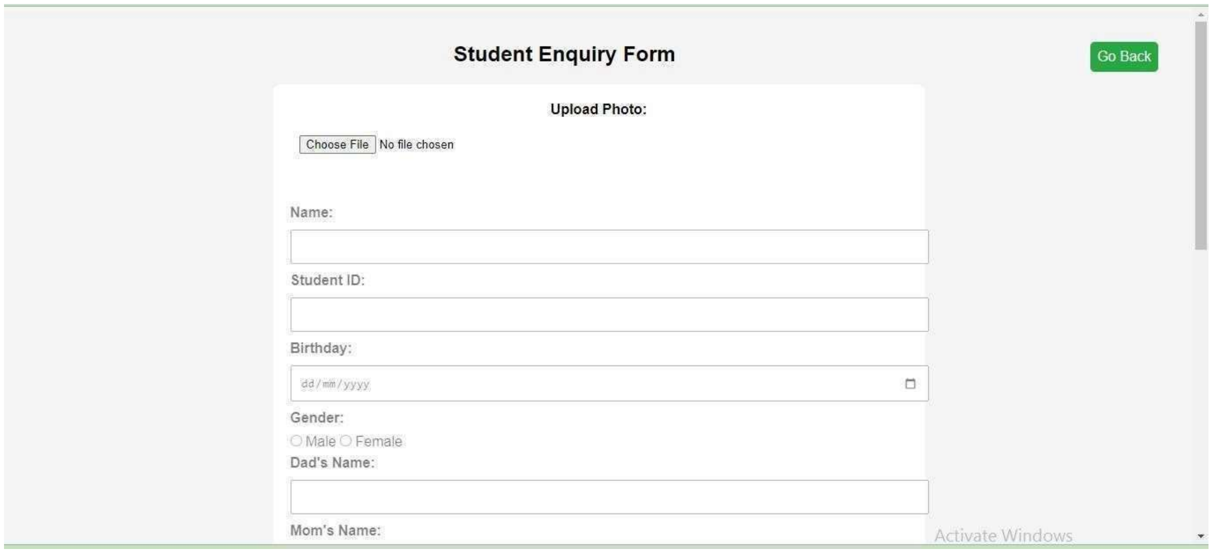


FIG 6.2 HOME PAGE



Student Enquiry Form

Upload Photo:

No file chosen

Name:

Student ID:

Birthday:

dd/mm/yyyy

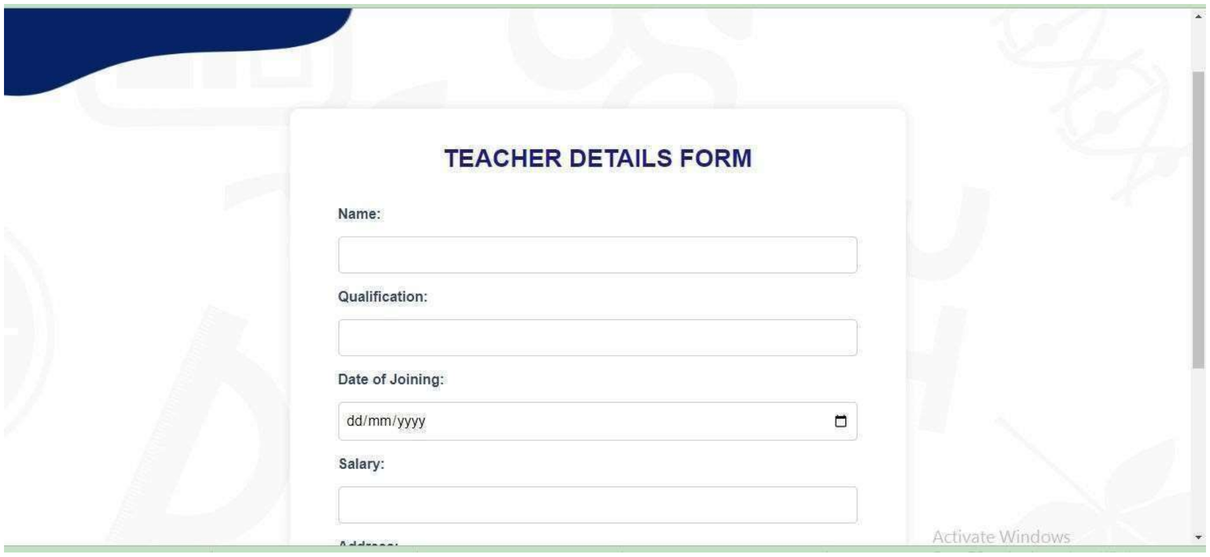
Gender:

○ Male ○ Female

Dad's Name:

Mom's Name:

FIG 6.3 STUDENT ENQUIRY



TEACHER DETAILS FORM

Name:

Qualification:

Date of Joining:

dd/mm/yyyy

Salary:

FIG 6.4 TEACHER DETAILS