COLLEGE ADMISSION MANAGEMENT SOFTWARE

PROJECT REPORT 2025-2026

Submitted By

24590069 ABINAYA

24590084 VISHNUPRIYA

24590074 MANIMOZHI

24500416 ASIN

Under the guidance of

Mrs. S.DEVI .B.E.,CSE

LECT/IT

Diploma in Information Technology

of the directorate of Technical Education, Government of Tamil Nadu

DEPARTMENT OF INFORMATION TECHNOLOGY

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202.

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202

Department of Information Technology

BONAFIDE CERTIFICATE

This is certified that this project work entitled **College Admission Management Software** has been submitted **ABINAYA, VISHNUPRIYA, MANIMOZHI, ASIN** in the partial fulfilment of the requirements for the award of Diploma in Information Technology during the academic year 2025-2026, who carried out the project work under our supervision.

Project Guide	**Head of the Department**
Mrs. S.DEVI B.E.,(CSE)	**Mrs. C.SRIDEVI B.E.,(CSE)**
LECT- IT	**HOD-COMPUTER & IT**

This is to certify that_____
was examined for the project work viva-voce held on_____

INTERNAL EXAMINER EXTERNAL EXAMINER

Acknowledgement

We express our sincere and profound thanks to our chairman **Mr.A.K.T.Mahendiran B.Com.,** for creating this magnificent edifice of learning by providing all necessary facilities to complete diploma engineering course successfully.

We express our gratitude to our principal **Mr.P.K.Kabilar M.E.,MBA.,** for constant encouragement and facilities provided towards the successful completion of our project work.

At the same time we would like to express our heartfelt thanks to our head of the department **Mrs.C.Sridevi.,B.E.,(CSE).,** and also our project guide **Mrs. S.Devi B.E.,(CSE)**for guiding and needful advices and time to complete this project.

We would like to show our gratitude to our department staffs for all instructions and guidance given throughout.

Our sincere thanks and affection to our parents and friends who gave hand in all our steps.

CONTENTS

ABSTRACT

The College Admission Management Software is a web-based application designed to automate and streamline the entire college admission process. Traditional admission systems often involve manual paperwork, long processing times, and a higher risk of data errors. This project aims to eliminate these challenges by providing a centralized, efficient, and user-friendly digital platform.

The system is developed using the Laravel framework for the backend and MariaDB as the database, ensuring high performance, scalability, and security. It is deployed on a VPS (Virtual Private Server) to provide reliable hosting, data accessibility, and continuous system availability.

The software allows students to register, fill out application forms, upload required documents, and track their admission status online. On the administrative side, it provides tools for managing applications, verifying documents, allocating seats based on merit, and generating reports. The system also supports automated notifications, reducing manual communication efforts.

Key features include user authentication, application management, document verification, merit list generation, and real-time status updates. The use of modern web technologies ensures a responsive and accessible interface across multiple devices.

Overall, this project enhances efficiency, reduces administrative workload, improves data accuracy, and offers a seamless admission experience for both students and college administrators.

1. INTRODUCTION

n the digital era, educational institutions are increasingly adopting technology to improve operational efficiency and provide better services to students. One of the most critical and time-consuming processes in any institution is the college admission process, which traditionally involves manual form filling, document verification, and physical submission of applications. These conventional methods are often prone to errors, delays, data redundancy, and lack of transparency.

The College Admission Management Software is designed to overcome these limitations by providing a fully automated and centralized system for managing admissions. This web-based application enables students to apply for admission online, upload necessary documents, and track their application status in real time. It eliminates the need for physical presence during the initial stages of admission, making the process more convenient and accessible.

The system is developed using the Laravel framework, which offers a robust and secure backend architecture, and MariaDB as the database management system to ensure efficient data storage and retrieval. The application is hosted on a Virtual Private Server (VPS), providing high availability, scalability, and better control over server resources.

From the administrative perspective, the system simplifies tasks such as application review, document verification, merit list generation, seat allocation, and reporting. It also includes features like automated notifications and dashboards, which help administrators monitor the entire admission process effectively.

This project aims to enhance transparency, reduce manual workload, minimize errors, and provide a seamless user experience for both students and college staff. By

integrating modern web technologies, the system ensures reliability, security, and efficiency in managing college admissions.

2. LITERATURE SURVEY

The process of college admission has evolved significantly with the advancement of information technology. Various research studies and existing systems have focused on digitizing and automating admission procedures to improve efficiency, transparency, and accessibility.

Traditional admission systems are primarily manual, involving paper-based applications and in-person verification. According to several studies, such systems are time-consuming, prone to human error, and lack proper data management. They often result in delays in processing applications, difficulties in tracking student records, and challenges in communication between applicants and institutions.

With the introduction of web-based technologies, many institutions have adopted online admission systems. These systems allow students to submit applications electronically and enable administrators to manage large volumes of data efficiently. Research indicates that online systems significantly reduce paperwork, improve data accuracy, and enhance user convenience. However, earlier implementations often lacked advanced features such as real-time tracking, automated notifications, and intelligent decision-making.

Recent developments have incorporated framework-based architectures such as Laravel, which provide secure authentication, modular design, and scalability. Studies highlight that using modern frameworks improves system maintainability and reduces development complexity. Additionally, the use of relational databases like MariaDB/MySQL ensures structured data storage, fast query processing, and reliable transaction handling.

Cloud and VPS-based hosting solutions have also gained importance in modern systems. Research shows that deploying applications on Virtual Private Servers (VPS) offers better performance, uptime, and security compared to shared hosting. It also

enables institutions to handle increasing numbers of users without compromising system speed.

Some advanced systems integrate features like automated merit list generation, document verification, and notification services through email or SMS. These enhancements improve transparency and reduce administrative workload. However, many existing solutions still lack full automation, user-friendly interfaces, and seamless integration of all admission-related activities in a single platform.

Based on the review of existing systems and technologies, it is evident that there is a need for a comprehensive, scalable, and efficient admission management system. The proposed system addresses these gaps by leveraging Laravel, MariaDB, and VPS hosting to deliver a secure, user-friendly, and fully automated solution for college admissions.

3. PROPOSED SYSTEM

The proposed **College Admission Management Software** is a fully automated, web-based solution designed to simplify and digitize the entire admission process. The system integrates modern technologies to provide a secure, scalable, and user-friendly platform for both students and administrators.

The application is developed using the **Laravel framework** for backend processing, ensuring robust architecture, security, and modular development. The **MariaDB database** is used for efficient data storage and management, while deployment on a **VPS (Virtual Private Server)** ensures high availability, performance, and reliability.

Overview of the Proposed System

The system provides an end-to-end solution starting from student registration to final admission confirmation. It eliminates manual paperwork and reduces the need for physical interaction by enabling all processes to be handled online.

Students can create accounts, fill out admission forms, upload required documents, and monitor their application status. Administrators can manage applications, verify documents, generate merit lists, allocate seats, and communicate with applicants through automated notifications.

Key Features of the Proposed System

- **Online Student Registration and Login**
  Secure authentication system allowing students to register and access their admission dashboard.
- **Digital Application Form Submission**
  Students can fill and submit application forms online with proper validation and error handling.
- **Document Upload and Verification**
  Upload of certificates, ID proofs, and other documents with admin-side verification.

- **Application Tracking System**
  Real-time status updates such as submitted, under review, approved, or rejected.
- **Automated Merit List Generation**
  System generates merit lists based on predefined criteria like marks or entrance scores.
- **Seat Allocation Management**
  Efficient allocation of seats based on merit and availability.
- **Admin Dashboard**
  Centralized control panel for managing students, applications, courses, and reports.
- **Notification System**
  Automated email/SMS alerts for application status, deadlines, and announcements.
- **Report Generation**
  Generation of admission reports, student statistics, and performance summaries.
- **Secure Data Management**
  Ensures data integrity, privacy, and protection using Laravel security features.

Advantages of the Proposed System

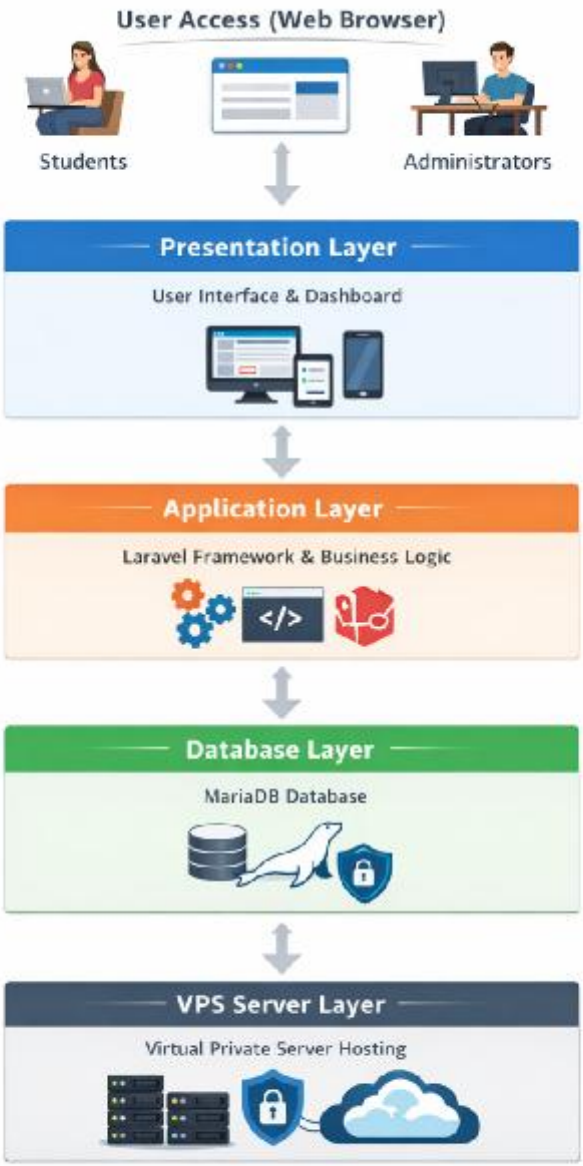
- Reduces manual workload and paperwork
- Improves accuracy and minimizes human errors
- Enhances transparency in the admission process
- Provides faster processing and real-time updates
- Ensures secure and centralized data management
- Accessible anytime and anywhere via web interface

System Workflow (Brief)

1. Student registers and logs into the system
2. Fills and submits the application form

3. Uploads required documents
4. Admin verifies application and documents
5. System generates merit list
6. Seat allocation is performed
7. Student receives notification and confirms admission

3.1 ARCHITECTURE



The **College Admission Management Software** follows a **three-tier web application architecture** that separates the system into different layers for better performance, security, maintainability, and scalability. This architecture ensures that the software can

efficiently handle student applications, administrative tasks, and database operations in a structured manner.

The project is developed using **Laravel** as the backend framework, **MariaDB** as the database, and is deployed on a **VPS Server** for reliable hosting and continuous access.

1. Presentation Layer

The presentation layer is the front-end interface through which students and administrators interact with the system. It is responsible for displaying web pages, forms, dashboards, notifications, and status updates.

This layer includes:

- Student registration and login pages
- Admission application form
- Document upload interface
- Student dashboard for tracking application progress
- Admin dashboard for managing admissions
- Report and notification views

The front end is designed to be responsive and user-friendly so that users can access the system from desktops, laptops, tablets, or mobile devices.

2. Application Layer

The application layer is the core logic layer of the system. It handles all processing activities, validations, workflows, and communication between the front end and the database.

In this layer, Laravel manages:

- User authentication and authorization
- Admission form processing
- Document validation and storage handling
- Merit calculation logic
- Seat allocation logic
- Status update workflows
- Notification generation
- Report generation
- Admin controls and role-based access

This layer acts as the bridge between user requests and the stored data, ensuring that all actions are performed securely and correctly.

3. Database Layer

The database layer stores all system data in an organized manner using **MariaDB**. It manages student records, application details, uploaded documents, merit data, course information, seat availability, and administrative records.

The database stores:

- Student personal details
- Academic qualification details
- Course and department information
- Uploaded certificates and documents
- Admission application status
- Merit and ranking records
- Seat allocation data
- Notification logs
- Admin login and activity records

This structured storage allows quick retrieval, secure transactions, and efficient reporting.

4. Server Hosting Layer

The application is hosted on a **VPS (Virtual Private Server)**, which provides a dedicated environment for deploying the Laravel application and MariaDB database. The VPS hosting layer ensures:

- Better speed and uptime
- Improved security
- Dedicated resource allocation
- Scalability for growing admission data
- Secure backup and recovery options
- Better control over server configuration

This hosting environment is more suitable than shared hosting for a professional college admission system because it can handle higher traffic and sensitive student data more securely.

5. System Flow in Architecture

The architecture works in the following sequence:

1. The student or admin accesses the system through a web browser
2. The request is received by the presentation layer
3. The application layer processes the request using Laravel controllers and business logic
4. The application layer interacts with the MariaDB database to store or retrieve data
5. The processed result is returned to the user interface
6. Notifications, merit generation, and seat allocation are handled through the same logic flow

Architectural Benefits

The proposed architecture provides several advantages:

- **Modularity** – each layer works independently, making development and maintenance easier
- **Security** – sensitive data is protected through Laravel authentication and database security
- **Scalability** – VPS hosting and layered architecture support future expansion
- **Performance** – organized request handling and optimized database operations improve speed
- **Maintainability** – code updates and feature enhancements can be managed efficiently
- **Reliability** – centralized hosting and structured design reduce system failure risks

3.2 MODULES DESCRIPTION

1. Student Registration and Authentication Module

This module is responsible for allowing new students to create an account and securely log in to the system. It collects basic information such as student name, email address, mobile number, password, and course preference during registration. After registration, the system verifies the credentials and grants secure access to the student dashboard.

This module also manages password encryption, login validation, forgot password functionality, and session control. Laravel authentication features help ensure data security and prevent unauthorized access. This is one of the most important modules because it is the entry point for every applicant using the platform.

Functions included:

- New student registration
- Secure login and logout
- Password reset and recovery
- Session management
- Account verification

2. Student Profile Management Module

This module allows students to enter and manage their personal, academic, and contact details. It acts as the central profile repository where all applicant information is stored before and after admission.

Students can update details such as date of birth, gender, address, guardian name, category, previous academic marks, and other required information. Administrators can also view these profiles during the admission screening process. Proper profile management ensures that student data remains complete, accurate, and easily accessible.

Functions included:

- Personal information entry
- Academic details management
- Address and contact details

- Guardian information
- Profile update and correction

---

3. Admission Application Form Module

This module enables students to fill out and submit the online admission application form for their desired course or department. The form includes fields for educational qualification, marks obtained, preferred course, reservation category, and other required admission details.

The system validates all form fields before submission to reduce incomplete or incorrect applications. Once the form is submitted, the data is stored in the MariaDB database and forwarded for administrative review. This module removes manual form-filling and makes the admission process faster and more organized.

Functions included:

- Dynamic admission form
- Course selection
- Form validation
- Save and submit option
- Application data storage

4. Document Upload and Verification Module

This module allows students to upload all required admission documents in digital format. These may include mark sheets, transfer certificate, community certificate, ID proof, passport-size photo, and signature.

Uploaded files are stored securely on the server, and administrators can review and verify them through the admin panel. The module helps reduce physical paperwork and makes the verification process faster and more transparent. File type validation, file size control, and secure storage are key features of this module.

Functions included:

- Upload academic certificates
- Upload ID proof and photo
- File validation and storage

- Admin-side document review
- Verification approval or rejection

5. Course and Seat Management Module

This module is used to manage available courses, departments, intake capacity, and seat distribution. Administrators can define course names, eligibility criteria, total seats, reserved seats, and admission rules.

It helps the institution maintain accurate information about seat availability and prevents over-allocation. This module plays a critical role in managing the admission structure of the college and supports fair distribution of seats based on predefined conditions.

Functions included:

- Add and update courses
- Define seat capacity
- Department-wise seat management
- Eligibility rule settings
- Reserved category seat allocation setup

6. Application Review and Approval Module

This module is designed for administrators to review submitted applications and verify whether students meet the required eligibility criteria. The admin checks student profile details, uploaded documents, academic scores, and course preferences.

Based on verification, the application can be marked as approved, pending, under review, or rejected. The module improves transparency in the admission process because each application is processed systematically and status updates are recorded clearly.

Functions included:

- View submitted applications
- Verify eligibility criteria
- Check uploaded documents
- Approve or reject applications

- Update review remarks and status

7. Merit List Generation Module

This module automatically generates merit lists based on predefined admission criteria such as academic marks, category rules, reservation policy, or entrance score. It reduces manual calculation errors and ensures that seat selection is fair and transparent.

The system retrieves applicant data, applies the merit calculation logic, ranks students accordingly, and prepares the final merit list. This module is especially useful when handling a large number of applicants.

Functions included:

- Merit calculation based on marks
- Rank generation
- Category-based filtering
- Course-wise merit preparation
- Final merit list publication

8. Seat Allocation Module

This module allocates seats to students according to merit rank, eligibility, reservation category, and seat availability. Once the merit list is prepared, the system can automatically or semi-automatically assign seats in the selected course.

The module ensures that allocation is done without exceeding available capacity. It also updates seat balances in real time and helps administrators track filled and vacant seats. This module is very important for final admission confirmation.

Functions included:

- Automatic seat allotment
- Merit-based course allocation
- Seat availability tracking
- Reservation-based allocation
- Admission confirmation record

9. Notification and Communication Module

This module manages communication between the college and applicants. It sends alerts and updates regarding registration, application submission, approval, rejection, merit list publication, seat allotment, and admission confirmation.

Notifications may be sent through email, SMS, or dashboard alerts. This module helps reduce manual follow-up work and ensures that students stay informed throughout the admission process.

Functions included:

- Application status notification
- Merit list announcement alerts
- Seat allotment notification
- Email and SMS communication
- Dashboard messages and reminders

10. Admin Dashboard and Report Generation Module

This module provides administrators with a complete control panel to monitor and manage the admission system. It gives access to student records, application statuses, course data, seat allocation information, and document verification results.

The module also generates reports such as total applications received, department-wise admissions, approved and rejected applications, seat utilization reports, and merit statistics. These reports help management make informed decisions and maintain proper admission records.

Functions included:

- Admin dashboard overview
- Application statistics
- Student admission reports
- Department-wise report generation
- Download and print reports

4. IMPLEMENTATION

The implementation of the AI Medical Report System is carried out using a combination of web technologies, database management, and artificial intelligence services to create a complete and practical healthcare application. The system is developed using PHP for server-side programming, MySQL for data storage, HTML, CSS, JavaScript, and Bootstrap for designing the user interface, and OpenAI API for extracting and understanding information from discharge summaries.

The implementation begins with the development of a secure login system for administrators and staff users. This includes user authentication, session handling, password encryption, and role-based dashboard access. Once the user logs in successfully, the system displays a professional dashboard with a left-side navigation menu to access modules such as report upload, patient management, follow-up alerts, and AI advice.

The next stage of implementation focuses on the discharge summary upload feature. The user uploads a discharge summary in image or PDF format through the dashboard. The system validates the file type, file size, and format before storing it on the server. After successful upload, the file is sent to the AI extraction module, where OpenAI-powered processing reads the contents of the report and identifies important details such as patient name, age, diagnosis, doctor name, hospital name, medicines, discharge date, and follow-up date.

Once the medical details are extracted, the data validation and structuring module processes the extracted content. This step ensures that the output is clean, formatted, and stored in the correct database fields. Invalid dates, missing values, and improperly extracted medicine details are corrected or flagged for review. After validation, the information is inserted into the MySQL database across related tables such as patients, reports, doctors, hospitals, medications, follow-ups, and advice records.

The patient management module is then implemented to allow administrators to search, edit, and manage patient profiles. Each patient record is linked with uploaded discharge summaries, extracted diagnosis details, prescriptions, and follow-up history. The doctor and hospital module is integrated to store and manage clinical references from the extracted reports.

The follow-up reminder module is implemented to compare stored follow-up dates with the current date and generate alerts for upcoming or overdue patient visits. These alerts are displayed on the dashboard so that staff can take timely action. In future expansion, this module can also be integrated with SMS or email notifications.

The AI advice modules are implemented to generate personalized recommendations based on the extracted diagnosis and discharge instructions. Using OpenAI, the system produces patient-specific food advice, activity guidance, medication-related precautions, and general recovery suggestions. These recommendations are stored in the database and displayed in the patient's detail page.

For frontend implementation, Bootstrap is used to build a responsive and professional user interface. The dashboard includes cards, tables, forms, modals, and navigation menus for easy interaction. JavaScript is used for dynamic behaviors such as live search, alert display, file preview, and form validation.

The backend implementation in PHP handles routing, database interaction, business logic, API communication, session control, and report generation. MySQL tables are properly normalized to reduce redundancy and improve data consistency. The system can also include audit logs and status tracking for uploaded files and generated advice.

Security is a major part of the implementation. Uploaded files are validated and stored securely, user passwords are encrypted, sessions are protected, and database queries are executed using prepared statements to prevent SQL injection. Access to patient data is restricted based on user roles.

Overall, the implementation of the AI Medical Report System transforms manual discharge summary handling into an intelligent digital workflow. It combines secure record management, automated data extraction, patient monitoring, and AI-driven post-discharge guidance into one integrated platform. This implementation improves efficiency, reduces human effort, and enhances the quality of healthcare administration.

4.1 SOFTWARE ENVIRONMENT

The **Software Environment** defines the set of software tools, frameworks, technologies, and platforms used to develop, deploy, and run the **College Admission Management Software**. The system is built using modern web technologies to ensure performance, scalability, security, and ease of maintenance.

1. Development Technologies

Frontend Technologies

The front end of the system is responsible for user interaction and interface design. It provides a responsive and user-friendly environment for students and administrators.

- **HTML5** – Used for structuring web pages and forms
- **CSS3** – Used for styling, layout design, and responsiveness
- **JavaScript** – Used for client-side validation and dynamic content updates
- **Bootstrap** – Used for responsive design and UI components

Backend Technologies

The backend handles application logic, data processing, and system workflows.

- **PHP (Laravel Framework)**
  Laravel is used as the backend framework due to its MVC architecture, security features, and scalability. It manages routing, controllers, middleware, authentication, and business logic.
- **Laravel Features Used:**
  - MVC Architecture
  - Blade Templating Engine
  - Eloquent ORM
  - Authentication & Authorization
  - Routing and Middleware
  - Validation and Security

Database Technology

- **MariaDB (MySQL Compatible)**

  MariaDB is used as the relational database management system for storing structured data such as student details, applications, documents, and admission records.

- **Key Features:**
  - High performance and reliability
  - Structured data storage
  - ACID compliance

4.2 SYSTEM REQUIREMENTS

1. Hardware Requirements

Development System Requirements

These are the minimum hardware requirements for developing and testing the project:

- **Processor:** Intel Core i3 / i5 or higher
- **RAM:** 8 GB minimum
- **Hard Disk:** 256 GB SSD or higher
- **Monitor:** 14-inch or above
- **Keyboard and Mouse:** Standard input devices
- **Internet Connection:** Stable broadband connection

These specifications are sufficient for coding, database handling, local server testing, and UI development.

Server Requirements (VPS Hosting)

For live deployment, the application requires a **Virtual Private Server (VPS)** with the following configuration:

Component	Specification
Server Type	Virtual Private Server (VPS)
CPU	8 Core Processor
RAM	32 GB
Storage	300 GB NVMe SSD
Bandwidth	High-speed / Unlimited preferred
Operating System	Ubuntu / CentOS / AlmaLinux
Web Server	Apache or Nginx
Database Server	MariaDB
Control Panel	WHM / cPanel

Component	Specification
Backup Support	Daily / Weekly Backup Recommended
SSL Certificate	Required for secure access

4.3 SAMPLE CODING

Login

```
@extends('auth.layouts.master')
@section('title', __('auth_login'))
@section('content')

<!-- Start Content-->
<div class="card">
   <div class="card-body text-center">
      <div class="mb-4">
         <i class="feather icon-unlock auth-icon"></i>
      </div>
      <h3 class="mb-4">{{ __('auth_login_title') }}</h3>

      <!-- Form Start -->
      <form method="POST" action="{{ route('login') }}">
      @csrf
         <div class="input-group mb-3">
            <input id="email" type="email" value="admin@mail.com" class="form-
control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}"
required autocomplete="email" placeholder="{{ __('field_email') }}" autofocus >

            @error('email')
               <span class="invalid-feedback" role="alert">
                  <strong>{{ $message }}</strong>
               </span>
            @enderror
         </div>
         <div class="input-group mb-4">
```

```
            <input id="password" type="password" class="form-control
@error('password') is-invalid @enderror" name="password" required
autocomplete="current-password" placeholder="{{ __('field_password') }}"
value="admin1234">

            @error('password')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
        <div class="form-group text-left">
            <div class="checkbox checkbox-fill d-inline">
                <input type="checkbox" name="remember" id="remember" {{
old('remember') ? 'checked' : '' }}>

                <label class="cr" for="remember">
                    {{ __('field_remember') }}
                </label>
            </div>
        </div>
        <input type="submit" class="btn btn-primary shadow-2 mb-4" name="submit"
value="{{ __('auth_login') }}">
    </form>
    <!-- Form End -->

    @if (Route::has('password.request'))
        <p class="mb-2 text-muted">
            <a href="{{ route('password.request') }}">
                {{ __('auth_forgot_password') }}
            </a>
```

```
            </p>
        @endif

        @if (Route::has('register'))
        <p class="mb-0 text-muted">
            {{ __("auth_dont_have_account") }}
            <a href="{{ route('register') }}">
                {{ __('auth_register') }}
            </a>
        </p>
        @endif

        @isset($setting->copyright_text)
        <p class="mb-0 text-muted">&copy; {!! strip_tags($setting->copyright_text,
'<a><b><br>') !!}</p>
        @endisset
    </div>
</div>
<!-- End Content-->

@endsection
```

Mater file

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>

        @include('admin.layouts.common.header_script')

        <style type="text/css" media="screen">
```

```
        h3 {
          font-size: 18px;
        }
        .auth-logo {
          position: absolute;
          left: 40px;
          top: 10px;
          overflow: hidden;
        }
        .auth-logo img {
          max-height: 100px;
          max-width: 100px;
        }

        @media screen and (max-width: 767px) {
          .auth-logo img {
             max-height: 70px;
          }
        }
    </style>

</head>
<body>

    <div class="auth-wrapper">

        @if(isset($setting))
        @if(is_file('uploads/setting/'.$setting->logo_path))
        <a href="#" class="auth-logo">
          <img src="{{ asset('uploads/setting/'.$setting->logo_path) }}" alt="logo">
        </a>
```

```
        @endif
        @endif

        <div class="auth-content">
            <div class="auth-bg">
                <span class="r"></span>
                <span class="r s"></span>
                <span class="r s"></span>
                <span class="r"></span>
            </div>

            <!-- Start Content-->
            @yield('content')
            <!-- End Content-->

        </div>
    </div>

    @include('admin.layouts.common.footer_script')
  </body>
</html>
```

Application

```
<?php

namespace App\Http\Controllers\Web;

use App\Http\Controllers\Controller;
use Flasher\Laravel\Facade\Flasher;
use Illuminate\Support\Facades\DB;
use App\Models\ApplicationSetting;
use Illuminate\Http\Request;
use App\Traits\FileUploader;
use App\Models\Application;
use App\Models\Province;
use App\Models\Program;
use Carbon\Carbon;

class ApplicationController extends Controller
{
    use FileUploader;

    protected $title, $route, $view, $path;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        // Module Data
```

```
        $this->title    = trans_choice('module_application', 1);
        $this->route    = 'application';
        $this->view     = 'admin.application';
        $this->path     = 'student';
    }

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
        $data['title']  = $this->title;
        $data['route']  = $this->route;
        $data['view']   = $this->view;
        $data['path']   = $this->path;


        $data['programs'] = Program::where('status', '1')->orderBy('title', 'asc')->get();
        $data['provinces'] = Province::where('status', '1')->orderBy('title', 'asc')->get();
        $data['applicationSetting']    =    ApplicationSetting::where('slug',    'admission')-
>where('status', '1')->firstOrFail();

        return view($this->view.'.create', $data);
    }

    /**
     * Store a newly created resource in storage.
     *
```

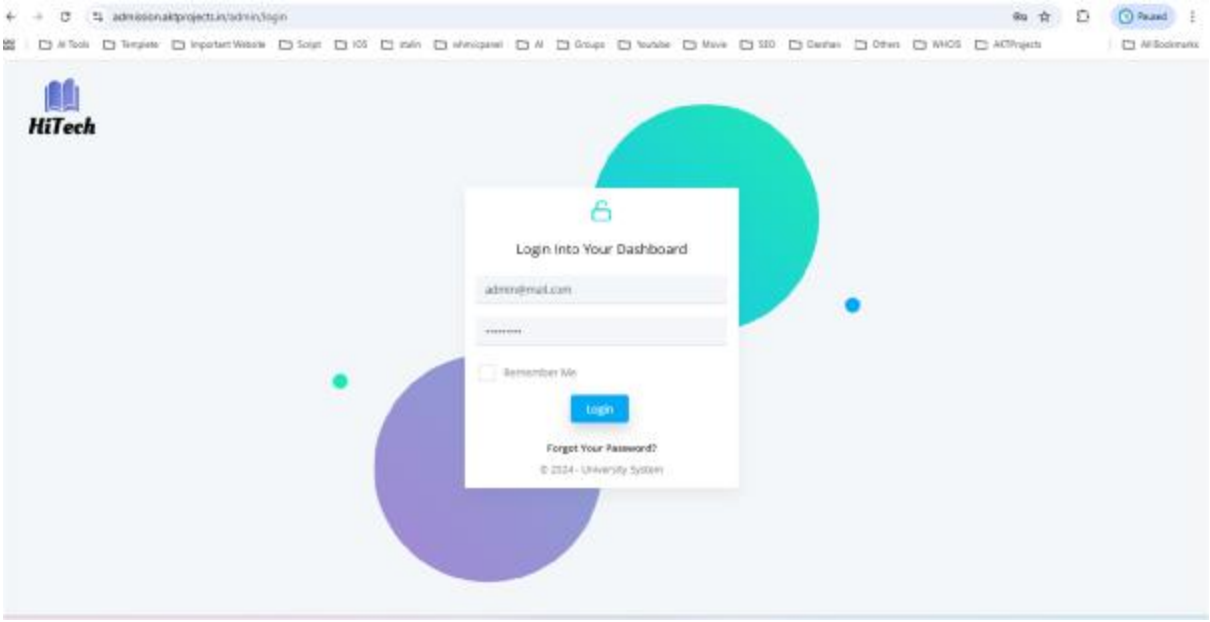
```
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    // Field Validation
    $request->validate([
        'program' => 'required',
        'first_name' => 'required',
        'last_name' => 'required',
        'email' => 'required|email|unique:applications,email',
        'phone' => 'required',
        'gender' => 'required',
        'dob' => 'required|date',
        'photo' => 'nullable|image',
        'signature' => 'nullable|image',
    ]);


    // Insert Data
    try{
        DB::beginTransaction();

        $student = new Application;
        $student->program_id = $request->program;
        $student->apply_date = Carbon::today();

        $student->first_name = $request->first_name;
        $student->last_name = $request->last_name;
        $student->father_name = $request->father_name;
        $student->mother_name = $request->mother_name;
```

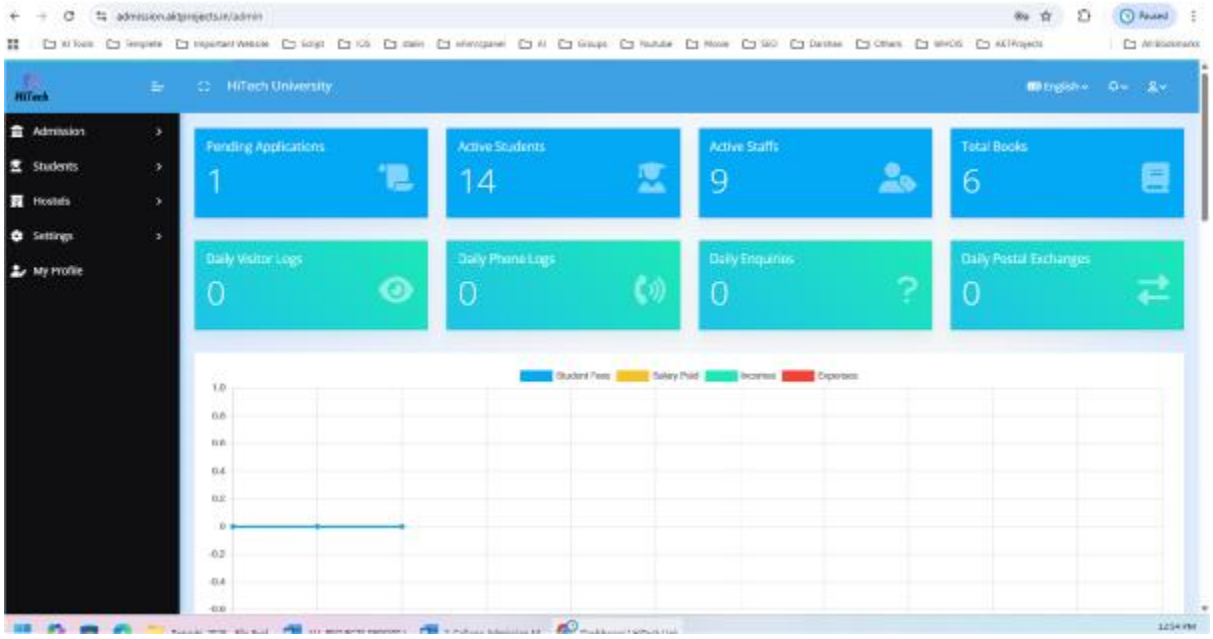
```
$student->father_occupation = $request->father_occupation;
$student->mother_occupation = $request->mother_occupation;

$student->country = $request->country;
$student->present_province = $request->present_province;
$student->present_district = $request->present_district;
$student->present_village = $request->present_village;
$student->present_address = $request->present_address;
$student->permanent_province = $request->permanent_province;
$student->permanent_district = $request->permanent_district;
$student->permanent_village = $request->permanent_village;
$student->permanent_address = $request->permanent_address;

$student->gender = $request->gender;
$student->dob = $request->dob;
$student->email = $request->email;
$student->phone = $request->phone;
$student->emergency_phone = $request->emergency_phone;

$student->religion = $request->religion;
$student->caste = $request->caste;
$student->mother_tongue = $request->mother_tongue;
$student->marital_status = $request->marital_status;
$student->blood_group = $request->blood_group;
$student->nationality = $request->nationality;
$student->national_id = $request->national_id;
$student->passport_no = $request->passport_no;

$student->school_name = $request->school_name;
$student->school_exam_id = $request->school_exam_id;
$student->school_graduation_year = $request->school_graduation_year;
```

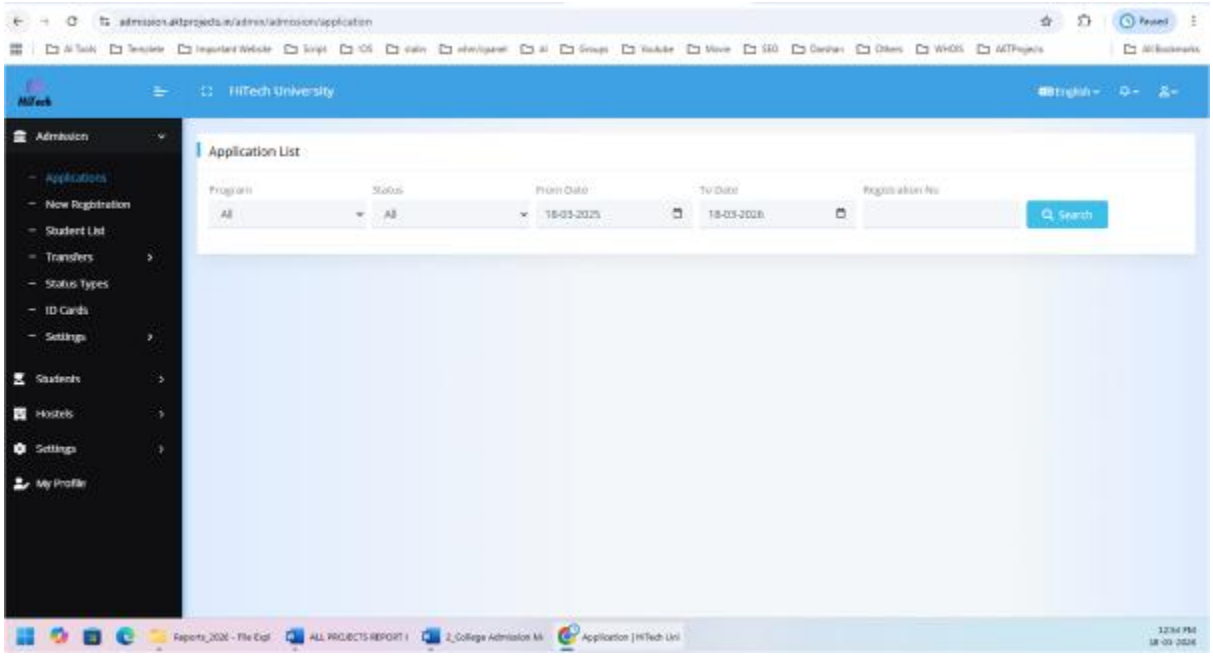
```
        $student->school_graduation_point = $request->school_graduation_point;
        $student->collage_name = $request->collage_name;
        $student->collage_exam_id = $request->collage_exam_id;
        $student->collage_graduation_year = $request->collage_graduation_year;
        $student->collage_graduation_point = $request->collage_graduation_point;
        $student->school_transcript = $this->uploadMedia($request,
'school_transcript', $this->path);
        $student->school_certificate = $this->uploadMedia($request,
'school_certificate', $this->path);
        $student->collage_transcript = $this->uploadMedia($request,
'collage_transcript', $this->path);
        $student->collage_certificate = $this->uploadMedia($request,
'collage_certificate', $this->path);
        $student->photo = $this->uploadImage($request, 'photo', $this->path, 300,
300);
        $student->signature = $this->uploadImage($request, 'signature', $this->path,
300, 100);
        $student->status = '1';
        $student->save();

        $student->registration_no = intval(10000000) + $student->id;
        $student->save();

        DB::commit();


        Flasher::addSuccess(__('msg_sent_successfully'), __('msg_success'));

        return redirect()->route($this->route.'.index')->with('success',
__('msg_sent_successfully'));
    }
```

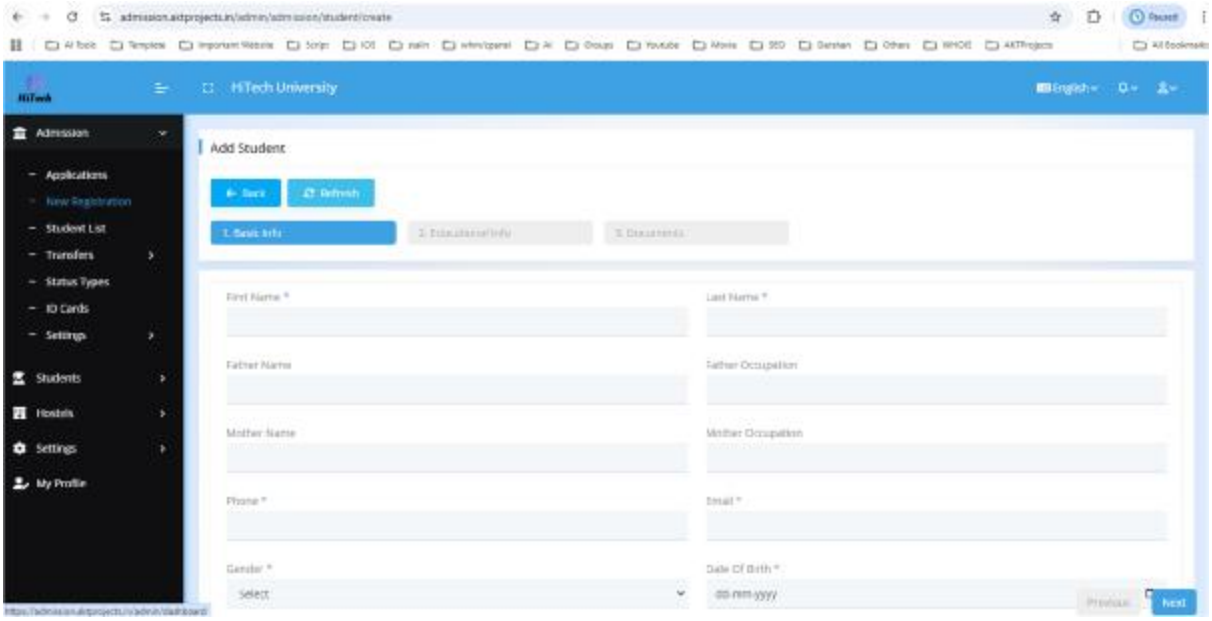
```
        catch(\Exception $e) {

            Flasher::addError(__('msg_created_error'), __('msg_error'));

            return redirect()->back();
        }
    }
}
```

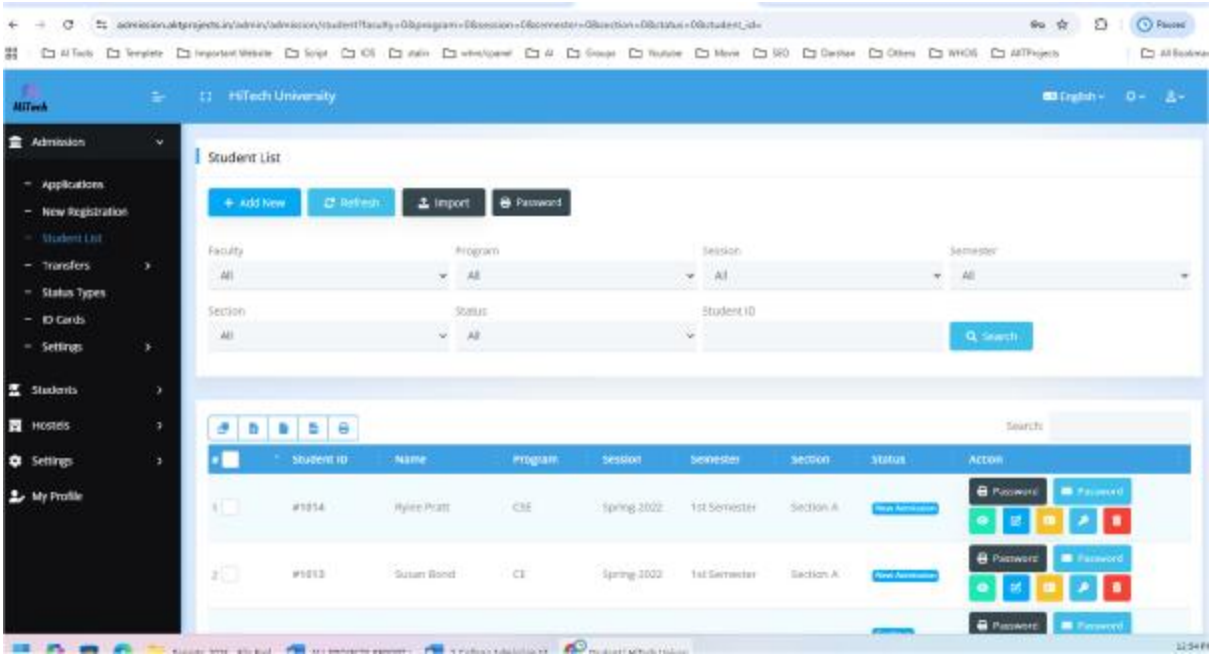
4.4. SCREEN SHOT

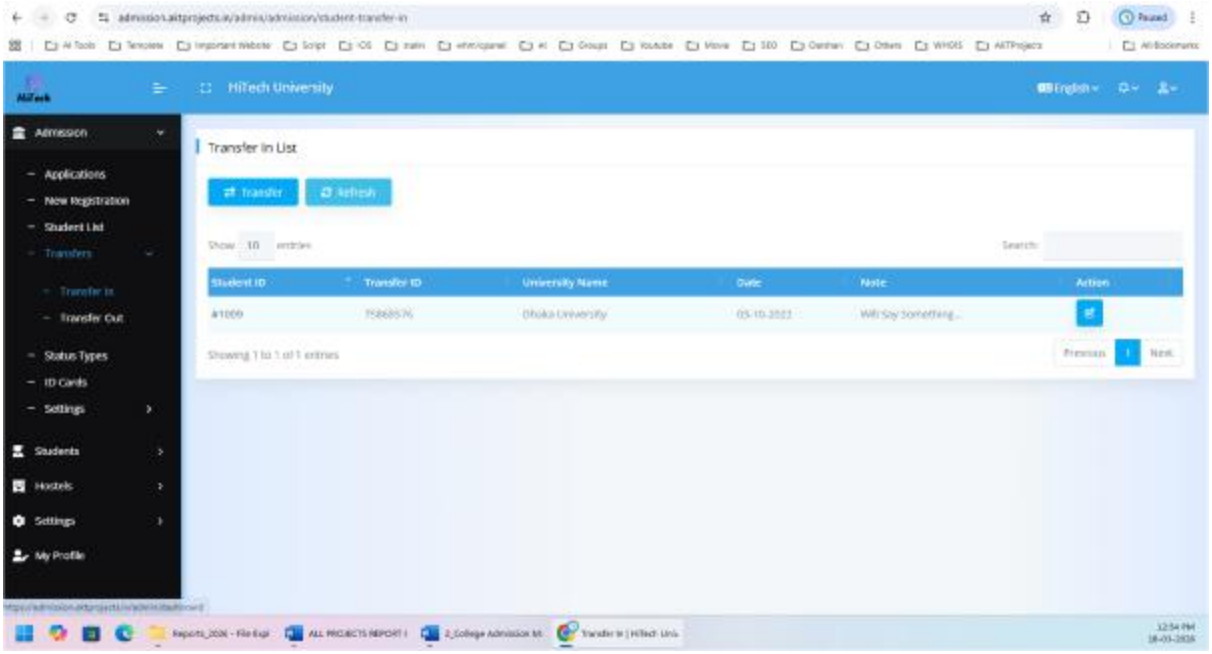


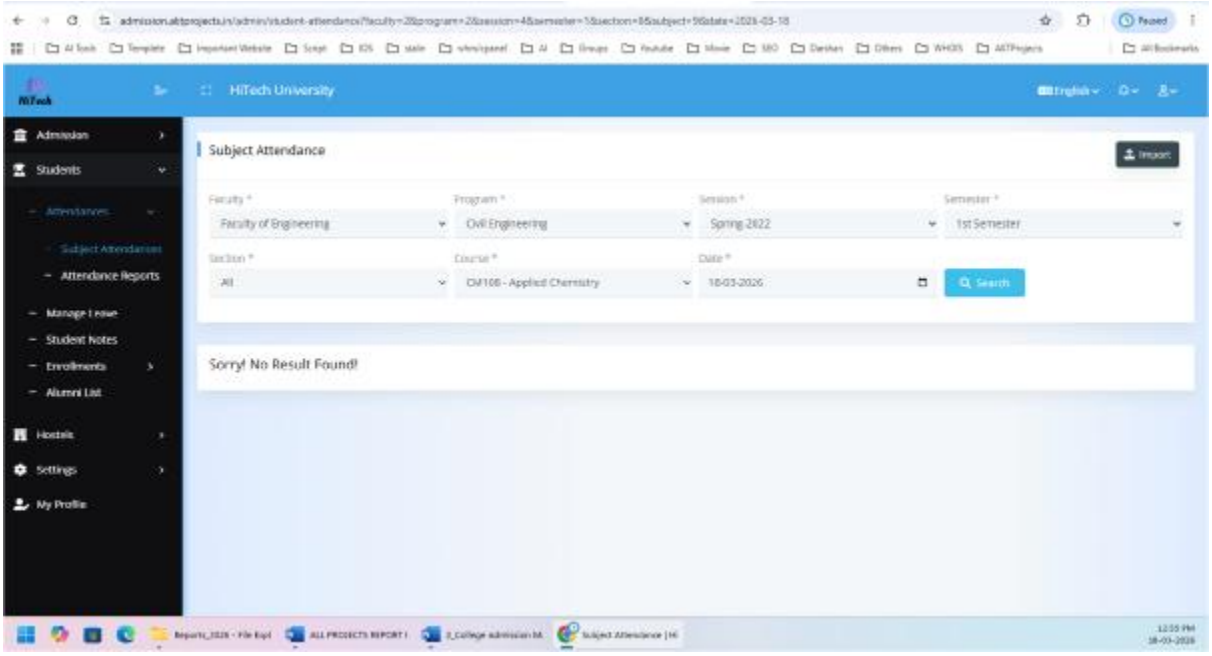

admission.aktprojects.in/admin
HiTech University
Admission
Students
Hostels
Settings
My Profile
Pending Applications
1
Active Students
14
Active Staffs
9
Total Books
6
Daily Visitor Logs
0
Daily Phone Logs
0
Daily Enquiries
0
Daily Postal Exchanges
0

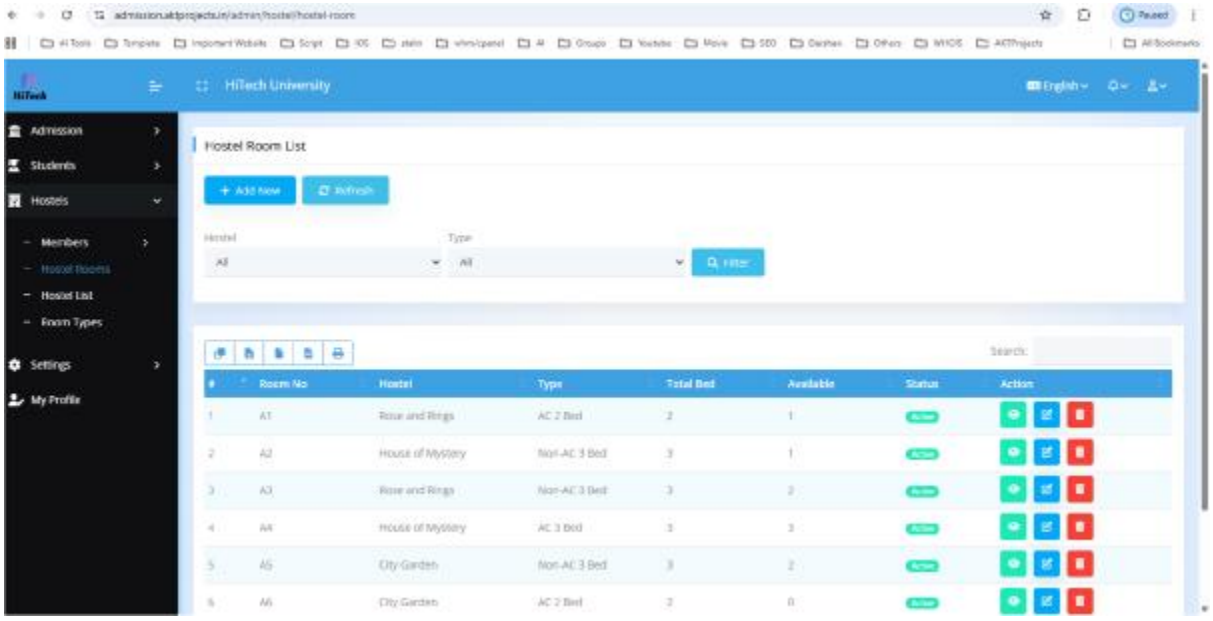

admission.aktprojects.in/admin/admission/application
HiTech University
Admission
Application List
Applications
New Registration
Student List
Transfers
Status Types
ID Cards
Settings
Students
Hostels
Settings
My Profile
Program
Status
From Date
To Date
Search

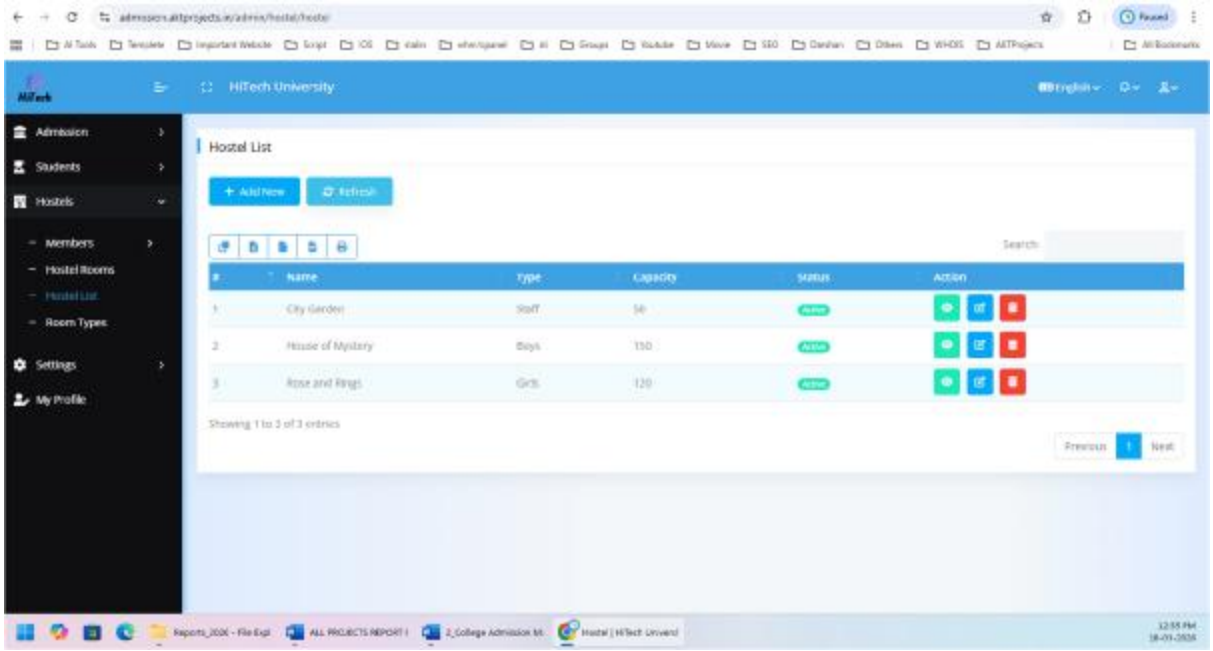

HiTech University
Admission
Applications
New Registration
Student List
Transfers
Status Types
ID Cards
Settings
Students
Hostels
Settings
My Profile
Add Student
Back
Refresh
1. Basic Info
First Name *
Last Name *
Father Name
Father Occupation
Mother Name
Mother Occupation
Phone *
Email *
Gender *
Select
Date Of Birth *
dd-mm-yyyy
Previous
Next


HiTech University
Admission
Applications
New Registration
Student List
Transfers
Status Types
ID Cards
Settings
Students
Hostels
Settings
My Profile
Student List
Add New
Refresh
Import
Password
Faculty
Program
Session
Semester
All
Section
Status
Student ID
Search
Search:
Student ID
Name
Program
Session
Semester
Section
Status
Action
#1014
Rylee Pratt
CSE
Spring 2022
1st Semester
Section A
#1013
Susan Bond
CE
Spring 2022
1st Semester
Section A


HiTech University
Admission
Applications
New Registration
Student List
Transfers
Transfer In
Transfer Out
Status Types
ID Cards
Settings
Students
Hostels
Settings
My Profile
Transfer In List
Transfer
Refresh
Student ID
Transfer ID
University Name
Date
Note
Action


HiTech University
Admission
Students
Attendances
Subject Attendances
Attendance Reports
Manage Leave
Student Notes
Enrollments
Alumni List
Hostels
Settings
My Profile
Subject Attendance
Import
Faculty *
Program *
Session *
Semester *
Faculty of Engineering
Civil Engineering
Spring 2022
1st Semester
Section *
Course *
Date *
All
CM108 - Applied Chemistry
10-03-2026
Search
Sorry! No Result Found!


HiTech University
Admission
Students
Hostels
Members
Hostel Rooms
Hostel List
Room Types
Settings
My Profile
Hostel Room List
+ Add New
Refresh
Hostel
Type
All
All
Filter
Search:
Room No Hostel Type Total Bed Available Status Action
1 A1 Rose and Rings AC 2 Bed 2 1 Active
2 A2 House of Mystery Non-AC 3 Bed 3 1 Active
3 A3 Rose and Rings Non-AC 3 Bed 3 2 Active
4 A4 House of Mystery AC 3 Bed 3 3 Active
5 A5 City Garden Non-AC 3 Bed 3 2 Active
6 A6 City Garden AC 2 Bed 2 0 Active


HiTech University
Admission
Students
Hostels
Members
Hostel Rooms
Hostel List
Room Types
Settings
My Profile
Hostel List
+ Add New
Refresh
Search:
Name Type Capacity Status Action
1 City Garden Staff 50 Active
2 House of Mystery Boys 150 Active
3 Rose and Rings Girls 120 Active
Showing 1 to 3 of 3 entries
Previous 1 Next

5. CONCLUSION

The **College Admission Management Software** successfully addresses the challenges associated with traditional admission processes by providing a fully digital, automated, and efficient solution. The system simplifies the entire workflow—from student registration to final seat allocation—thereby reducing manual effort, paperwork, and processing time.

By leveraging modern technologies such as **Laravel**, **MariaDB**, and **VPS hosting**, the system ensures high performance, scalability, and data security. The implementation of features like online application submission, document upload and verification, merit list generation, and real-time status tracking enhances transparency and improves the overall user experience.

The software not only benefits students by offering a convenient and accessible admission process but also assists administrators in managing large volumes of applications with accuracy and ease. The centralized database system ensures proper data management, while automated notifications keep users informed at every stage.

In conclusion, the proposed system provides a reliable, secure, and user-friendly platform for managing college admissions. It demonstrates how digital transformation can significantly improve institutional processes and deliver better services. The system is scalable and can be further enhanced with additional features to meet future requirements, making it a practical and valuable solution for modern educational institutions.

REFERENCES

1. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California.
2. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
3. Welling, L., & Thomson, L. (2017). PHP and MySQL Web Development. Addison-Wesley.
4. Nixon, R. (2018). Learning PHP, MySQL & JavaScript. O'Reilly Media.
5. Laravel Documentation. (2025). Laravel Official Documentation. Available at: https://laravel.com/docs
6. MariaDB Foundation. (2025). MariaDB Documentation. Available at: https://mariadb.org
7. Bootstrap Team. (2025). Bootstrap Documentation. Available at: https://getbootstrap.com
8. Mozilla Developer Network (MDN). (2025). Web Development Documentation. Available at: https://developer.mozilla.org
9. Sommerville, I. (2016). Software Engineering (10th Edition). Pearson Education.
10. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill.
11. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts. McGraw-Hill.
12. Tanenbaum, A. S., & Van Steen, M. (2016). Distributed Systems: Principles and Paradigms. Pearson.
13. OWASP Foundation. (2025). OWASP Web Security Guidelines. Available at: https://owasp.org
14. Red Hat. (2025). Linux Server Administration Guide. Available at: https://www.redhat.com
15. DigitalOcean. (2025). VPS Hosting and Cloud Infrastructure Documentation. Available at: https://www.digitalocean.com