**Software Design**

**for**

**Campus Management Application**

**at**

**World Skill Center (WSC)**

**Design Feature - Students Management Module**

Logo

Description automatically generated

**Sustainable Outreach And Universal Leadership (SOUL) Limited**

**Version no 0.1**

**VERSION HISTORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version No.** | **Created By** | **Updated Section** | **Updated By** |
| 20-01-2022 | Version 0.1 | Priytesh Shah |  |  |

Contents

1. Introduction
2. Platform Overview
   1. Platform requirements
      1. Software Stack
      2. Hosting requirement

3. Architecture

4. Design Overview

4.1 Overview

4.2 Description of Problem

4.3 Dependency

4.4 Flow Chart

4.5 Use Case Diagram

4.6 Form Elaboration

4.7 Methods

4.8 Screen Access

5. Deployment (Build & release)

1. **Introduction**

Campus management project for WSC aims at providing an efficient, secure, and user-friendly platform to manage and automate the various processes. It is an integrated system that provides all the necessary tools to manage the various aspects of running a University. This web-based solution enables administrators and faculty to easily track and manage activities related to student registration and enrollment, faculty and staff management, financial management, facility management, and more. Additionally, it also provides features such as reporting and analytics, notification to students and parents using emails.

Scope of project:

1. Students Management Modules
2. Academic System
3. Procurement & Inventory Management
4. Human Resources Management System
5. Finance and Accounting System
6. Training and Placement
7. Infrastructure
8. **Platform Overview**

**2.1 Platform requirements**

**2.1.1 Software Stack**

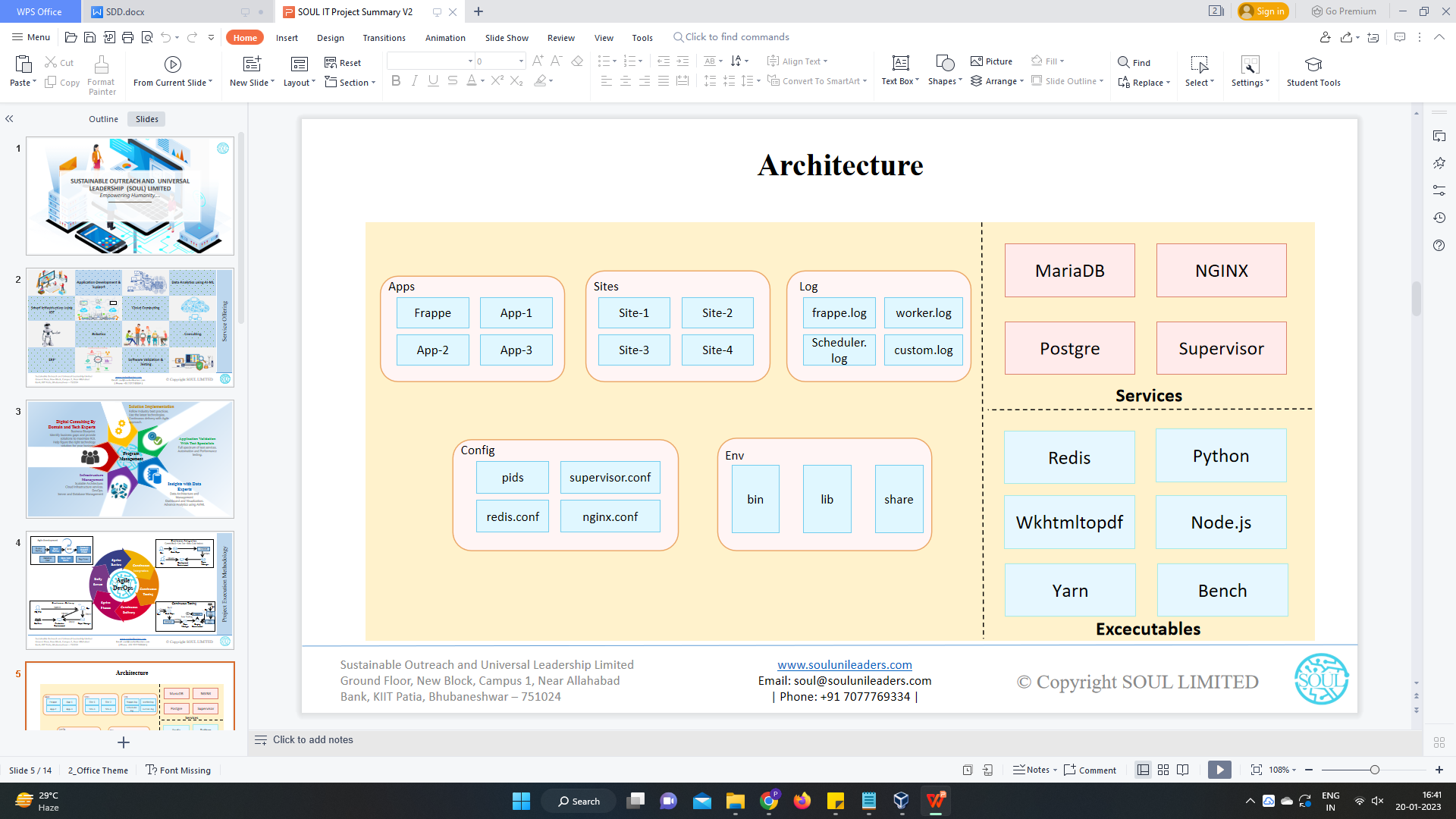
Our application use Frappe framework as the major software stack which is an open-source full stack web framework written in Python and JavaScript that is used to develop and maintain web applications. It is designed to be simple, easy to use, and fast. It is built on top of the popular Django web framework. Frappe provides a powerful set of tools for building web applications with a focus on scalability and extensibility. It provides a powerful ORM, a powerful templating system, and a comprehensive set of APIs for building web applications. Frappe is also designed for ease of use, with a focus on reducing development time and building an intuitive admin interface. It also features a seamless integration with other open-source projects such as Redis, MongoDB, and MySQL, as well as a host of APIs and other tools.

**2.2.2 Hosting requirement**

Our application requires a **Linux**-based hosting environment. It requires at least **1GB** of **RAM**, **2GB** of **disk space**, and preferably an **Ubuntu 16.04** or **higher** **operating system**. It also requires a **web server** such as **Apache** or **Nginx**, **MariaDB 10.1** or **higher**, and **Python 2.7.x** or **higher**. Additionally, it requires an SMTP server to send emails and an SSL certificate to enable HTTPS.

1. **Architecture**

Our application is based on a web-based, modular, service-oriented architecture (SOA), which allows for the development of custom applications and services that can be integrated with the core Our application platform. It is built on a combination of JavaScript, Python, and MySQL. The architecture of Our application consists of multiple layers, each with its own set of functions and services. At the highest level are the user interface, user experience, and application logic components. These components provide the user with an interface to the Our application system and the ability to interact with the underlying data and services. Below the user interface layer is the data layer, which is responsible for managing the storage and retrieval of data and services. This layer is powered by a combination of MySQL and PostgreSQL databases. The data layer is also responsible for providing secure access to the data and services. The application layer is responsible for providing access to the data and services, as well as providing the business logic that drives the system. The application layer is powered by a combination of Python and JavaScript, and is responsible for providing the core capabilities of the system, such as authentication, authorization, reporting, and analytics. Finally, the service layer is the layer that connects the application layer to the user interface layer. It provides the ability to integrate the Our application system with external systems and services, such as third-party APIs and web services. The service layer is powered by a combination of web services and web sockets. Following is a diagram showing a representation of the architecture of our application:



**Fig**: Architecture of Our Application

1. **Design Overview:**
   1. **Student**
      1. **Overview**

**The Student document will hold all the data of any Student in your Academy like their Personal Information, Photo, Date of Birth, Address, etc.**

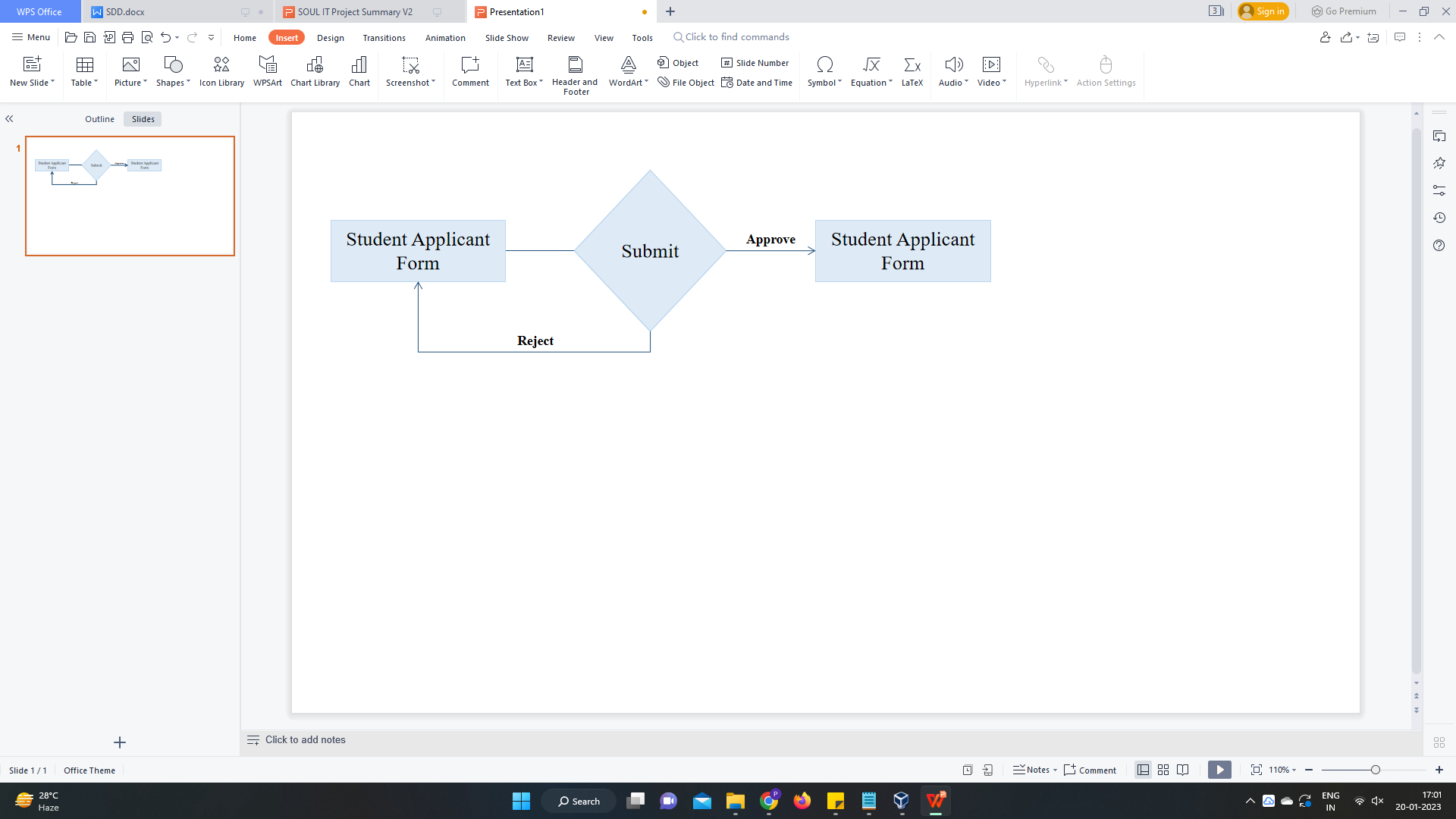
* + 1. **Description of Problem**

Create a page containing all Information about a student enrolled in the system including Personal details, Past educational information, Current educational details, etc.

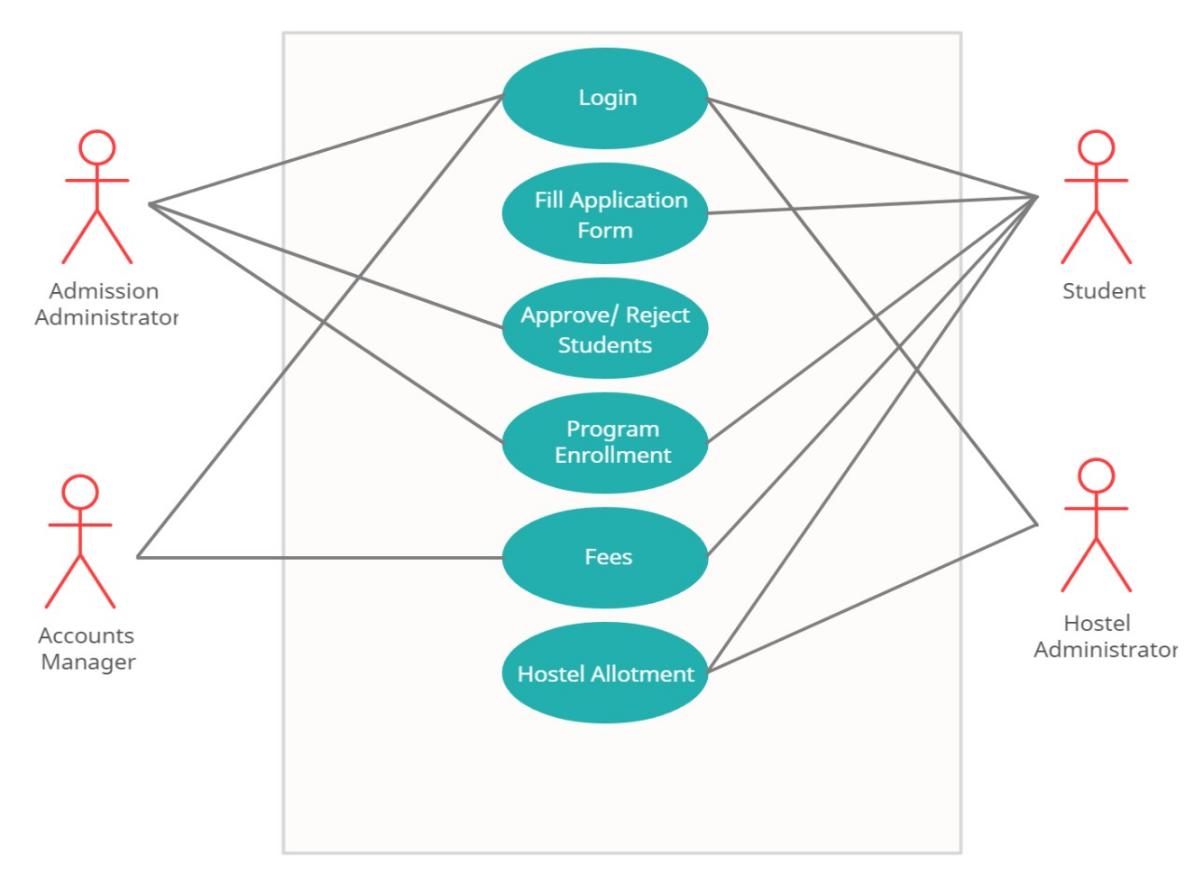
* + 1. **Dependency**

Dependent on Approval of Student Application form which is filled by student

* + 1. **Flow Chart**



* + 1. **Use Case Diagram**



**Brief Description -**

Admission Administrator: Able to create admission for a particular academic year for all the classes, Approve/Reject Application forms, Does Program Enrollment for approved students

Accounts manager: Charges fees for all approved students after Program Enrollment

Student: Fill in the Application form for admission, opts for Hostel, Pays Fees

Hostel Administrator: Provides Hostel to students who have opted for it.

* + 1. **Form Elaboration**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Type** | **Mandatory** |
| First Name | Data | 1 |
| Middle Name | Data | 0 |
| Last Name | Data | 0 |
| Program | Link | 1 |
| Student Category | Link | 0 |
| LMS Only | Check | 0 |
| Paid | Check | 0 |
|  | Column Break | 0 |
| Naming Series | Select | 0 |
| Application Status | Select | 0 |
| Application Date | Date | 0 |
| Academic Year | Link | 0 |
| Academic Term | Link | 0 |
| Student Admission | Link | 0 |
| Image | Attach Image | 0 |
| Personal Details | Section Break | 0 |
| Date of Birth | Date | 0 |
| Gender | Select | 0 |
| Blood Group | Select | 0 |
|  | Column Break | 0 |
| Student Email Address | Data | 0 |
| Student Mobile Number | Data | 0 |
| Nationality | Data | 0 |
| Home Address | Section Break | 0 |
| Address Line 1 | Data | 0 |
| Address Line 2 | Data | 0 |
| Pin-code | Data | 0 |
|  | Column Break | 0 |
| City | Data | 0 |
| State | Data | 0 |
| Guardian Details | Section Break | 0 |
| Guardians | Table | 0 |
| Sibling Details | Section Break | 0 |
| Siblings | Table | 0 |

* + 1. **Methods**

Pseudo code containg programming logic for all methods for the doctype

* + 1. **Screen Authorization**

|  |  |
| --- | --- |
| **User** | **Type of access** |
| Education Administrator | Read/ Write/ Modify |
| Student | View |

1. **Deployment**

Details abut product deployment