



SOFTWARE DESIGN SPECIFICATION

Web Development  
  
Collegeopedia

|  |  |  |  |
| --- | --- | --- | --- |
| **Created By:** | Prasad Pathare | **Approved By:** | Jonathan Dabre |
| **Created On:** | 05-08-2023 | **Approved On:** | 30-09-2023 |

**INDEX**

[**1** **PURPOSE** 2](#_Toc142418236)

[**2** **PROJECT SCOPE** 2](#_Toc142418237)

[**3** **SYSTEM OVERVIEW** 2](#_Toc142418238)

[**4** **DESIGN CONSIDERATIONS** 2](#_Toc142418239)

[4.1 Requirements 3](#_Toc142418240)

[4.2 Assumptions 3](#_Toc142418241)

[4.3 Dependencies 3](#_Toc142418242)

[**5** **SYSTEM ARCHITECTURE** 3](#_Toc142418243)

[5.1 Architectural Strategies 4](#_Toc142418244)

[5.2 Structure & Relationships 4](#_Toc142418245)

[**6** **DETAILED DESCRIPTION OF COMPONENTS** 4](#_Toc142418246)

[**7** **INTEGRATION** 5](#_Toc142418247)

[**8** **APPENDICES** 1](#_Toc142418248)

[8.1 Appendix A – Detailed Description of Components 1](#_Toc142418249)

# **PURPOSE**

This document is created based on the requirement specification document. The purpose of this Software Design Specification (SDS) Document is to break down the project into components to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also serve as a tool for verification and validation of the final product.

# **PROJECT SCOPE**

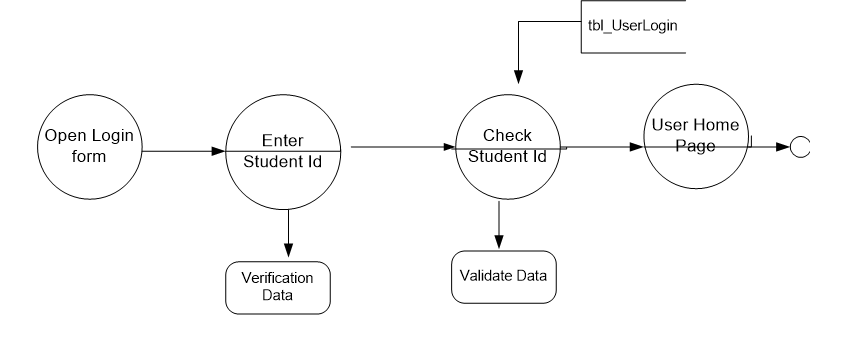
The scope of the Collegeopedia includes its distinct features, its benefits, and its limitations. The system's distinct features allow it to build an appealing web application for students which will help them with accurate and reliable information about colleges, courses offered and contact details. by using HTML, CSS, JavaScript for both front-end and back-end development, the system enables the user to an appealing web application for students which is created due to Unavailability of an efficient and user-friendly online prospective for students to search and discover colleges in India along with detailed information about the courses offered and contact details of the institutions.

# **SYSTEM OVERVIEW**

This section will provide an outline of the various components and subsystems of Collegeopedia

This Centralized College Portal is a comprehensive web-based platform designed to streamline and enhance the overall management and communication processes within a college or university through students. This system provides a centralized hub where students can access

accurate and reliable information about colleges, courses offered and contact details.



# **DESIGN CONSIDERATIONS**

This section describes requirements, assumptions and dependencies to be addressed to devise a complete design solution.

## Requirements

Web Development Framework Database Management System Front-end Technologies (HTML, CSS, JavaScript)

## Assumptions

Availability of college and course data from reliable sources. Adequate user interest and engagement with the platform.

## Dependencies

Ensuring the accessibility of dependable college and course data, along with fostering substantial user interest and active engagement with the platform, constitutes a pivotal objective. To attain this goal, a multifaceted approach is imperative.

# **SYSTEM ARCHITECTURE**

The software system architecture refers to the logical organization of a distributed system into software components. It defines how components of a software system are assembled, their relationship and communication between them. It serves as a blueprint for software application and development basis for developer team. An effective architecture serves as the conceptual glue that holds every phase of the project together for all of its stakeholders, enabling agility, time and cost savings, and early identification of design risks.

The Software architecture:

* Defines structure of a system
* Defines behaviour of a system
* Defines component relationship
* Defines communication structure
* Balances stakeholder’s needs
* Influences team structure
* Focuses on significant elements
* Captures early design decisions

Below some important characteristics which are commonly considered are explained.

**Operational Architecture Characteristics:**

* Availability
* Performance
* Reliability
* Low fault tolerance
* Scalability

**Structural Architecture Characteristics:**

* Configurability
* Extensibility
* Supportability
* Portability
* Maintainability

**Cross-Cutting Architecture Characteristics:**

* Accessibility
* Security
* Usability
* Privacy
* Feasibility

## Architectural Strategies

**Modular and Component-Based Design:**

* Break down the portal into modular components, each responsible for specific functionalities like student services, faculty services, administrative services, and more.
* Use microservices or service-oriented architecture (SOA) to build these components, allowing for independent development, scalability, and maintenance.

**Centralized Data Repository:**

* Implement a centralized database or data warehouse to store all essential data, such as student records, course information, and administrative documents.
* Use a relational database management system (RDBMS) depending on the data structure and scalability requirements.

**API-Driven Architecture:**

* Develop robust APIs (Application Programming Interfaces) to facilitate communication between different components and external systems.

**User Authentication and Authorization:**

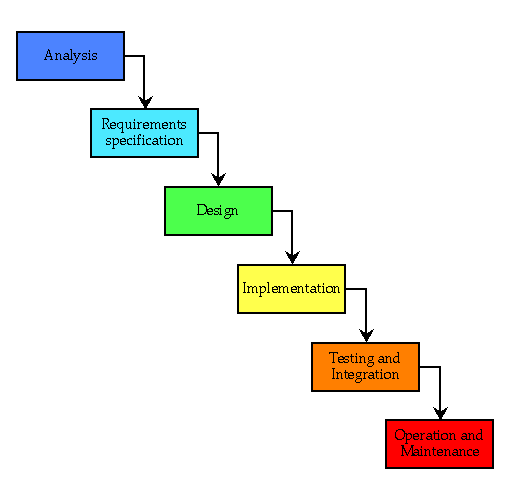
* Implement a single sign-on (SSO) mechanism to allow users to log in once and access various parts of the portal seamlessly.
* Use role-based access control (RBAC) to define user roles and permissions for different portal features.

**Responsive and Mobile-First Design:**

* Ensure that the portal's user interface is responsive, making it accessible and user-friendly on various devices, including desktops, tablets, and smartphones.
* Prioritize a mobile-first design approach to meet the needs of users on mobile devices.

**Scalability and Load Balancing:**

* Design the portal to be horizontally scalable, allowing for the addition of more resources (e.g., servers) to handle increased user traffic.
* Implement load balancing to distribute incoming requests evenly across multiple servers for improved performance and fault tolerance.



## Structure & Relationships

<Create a flowchart to show the structure and relationships between the components listed above >

**Modular and Component-Based Design**

**Centralized Data Repository**

**API-Driven Architecture**

**User Authentication and Authorization User Authentication and Authorization:**

**Responsive and Mobile-First Design**

**Scalability and Load Balancing**

# **DETAILED DESCRIPTION OF COMPONENTS**

For detailed description of the components, please refer **Appendix A – Detailed Description of Components**

The below template will be used to specify the details of all the components

**Table 1: Detailed Design Specification Template**

|  |  |
| --- | --- |
| **Identification** | The unique name for the component and the location of the component in the system. |
| **Type** | A module, a subprogram, a form, a data file, a control procedure, a class, etc. |
| **Purpose** | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS - but are implied or adjunct to formally stated SDS requirements. |
| **Subordinates** | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| **Dependencies** | How the component’s function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| **Interfaces** | Detailed description of all external or internal interfaces as well as of any mechanism for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| **Resources** | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. |
| **Processing** | A full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| **Data** | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. |

# **INTEGRATIONS**

# **Library and Email:** Integrate library management and email systems to search for resources and receive college communications.

# **Authentication (SSO):** Implement single sign-on for a seamless user experience across multiple services.

# **Calendars and Maps:** Include academic calendars, campus maps, and event listings.

# **Admissions and Support:** Streamline admissions processes

# **Content Management:** Use a content management system for easy updates.

# **External Services:** Collaborate with external providers for additional services like career boards

# **APPENDICES**

## Appendix A – Detailed Description of Components

|  |  |
| --- | --- |
| **Identification** | **Login screen** |
| **Type** | Form |
| **Purpose** | purpose of a login page is to verify the identity of student who want to access our secure system or application. Students provide their credentials, such as a username and password, to prove that they are authorized to use the system. |
| **Subordinates** | A registration page allows new users to create accounts by providing necessary information, such as username, email, and password. |
| **Dependencies** | Our login page relies on a database to store user information, including usernames, passwords (usually hashed and salted for security), and additional user attributes (e.g., email addresses, roles, permissions). |
| **Interfaces** | login page includes elements like a header with branding, fields for entering a username/email and password, options for "Remember Me" and "Forgot Password," a "Sign-In" button, and links for "Sign Up" and error messages. |
| **Resources** | essential resources like web development frameworks , UI libraries, authentication libraries, database systems |
| **Processing** | The processing of a login page involves user input validation, client-side feedback, and server-side authentication. Users enter credentials, which are validated both on the client side and server side. |
|  |  |

|  |  |
| --- | --- |
| **Identification** | **College search** |
| **Type** | Class |
| **Purpose** | primary purpose of a search bar is to provide students with a convenient and efficient way to search for specific college, course, or items within a website, application, or database. |
| **Subordinates** | It offers an alternative means of navigating a website or app, especially in cases where the list of colleges is vast, making it easier for users to find what they're looking for |
| **Dependencies** | Search results can often be filtered and sorted to further refine and organize the displayed content based on criteria such as relevance, date, or category. |
| **Interfaces** | A well-implemented search bar contributes to a positive user experience by offering a user-friendly and familiar way to interact with a college website |
| **Resources** | essential resources like web development frameworks , UI libraries, authentication libraries, database systems |
| **Processing** | By facilitating content discovery and making information readily accessible, a search bar can increase user engagement and retention on websites and applications. |
|  |  |