

**Software Design Document Specification Template**

The Software Design Specification (SDS) sections provide you with guidelines related to the structure and the contents of SDS document. The Software Design Specification document includes at least these sections.

For the project, your team may have good reasons for wanting to deviate from this proposed outline. If a section is not applicable in your case, do not delete it; instead, give the topic heading and write "Not applicable".

You will note that there is some overlap in the content between different documents (i.e. the User Requirements Specification Document and the Software Design Specification Document.) This redundancy allows each document to stand on its own.

***ONLY THE SECTION TITLES COLORED IN ORANGE ARE REQUIRED TO BE COMPLETED.***

***DO NOT DELETE THE SECTIONS YOU ARE NOT COMPLETING AS THEY ARE A PART OF THE DOCUMENT***

Contents

[Introduction](#_heading=h.gjdgxs) **3**

[Purpose of this document](#_heading=h.30j0zll) 3

[Scope of the development project](#_heading=h.1fob9te) 3

[Definitions, acronyms, and abbreviations](#_heading=h.3znysh7) 3

[References](#_heading=h.2et92p0) 3

[Overview of document](#_heading=h.tyjcwt) 3

[System architecture description](#_heading=h.3dy6vkm) **3**

[Overview of modules / components](#_heading=h.1t3h5sf) 3

[Structure and relationships](#_heading=h.4d34og8) 3

[User interface](#_heading=h.17dp8vu) 3

[User interface issues](#_heading=h.3rdcrjn) 3

[Detailed description of components (ONLY 2 ARE REQUIRED)](#_heading=h.26in1rg) **4**

[X Component (or Class or Function ...)](#_heading=h.lnxbz9) 4

[Y Component (or Class or Function ...)](#_heading=h.35nkun2) 6

[Reuse and relationships to other products](#_heading=h.1ksv4uv) **8**

[Design decisions and tradeoffs](#_heading=h.44sinio) **8**

[Pseudocode for components](#_heading=h.2jxsxqh) **9**

[Appendices (if any)](#_heading=h.z337ya) **9**

[Software component template for section 3](#_heading=h.3j2qqm3) **9**

# Introduction

## Purpose of this document

Full description of the main objectives of the SDS document.

## Scope of the development project

This will be similar to what was written in the SRS.

## Definitions, acronyms, and abbreviations

Be sure to alphabetize!

## References

This section will include technical books and documents related to design issues. Be certain that the references you give are complete and in the appropriate format.

## Overview of document

A short description of how the rest of the SDS is organized and what can be found in the rest of the document. This is not simply a table of contents. Motivate and briefly describe the various parts!

# System architecture description

## Overview of modules / components

This subsection will introduce the various components and subsystems.

## Structure and relationships

Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

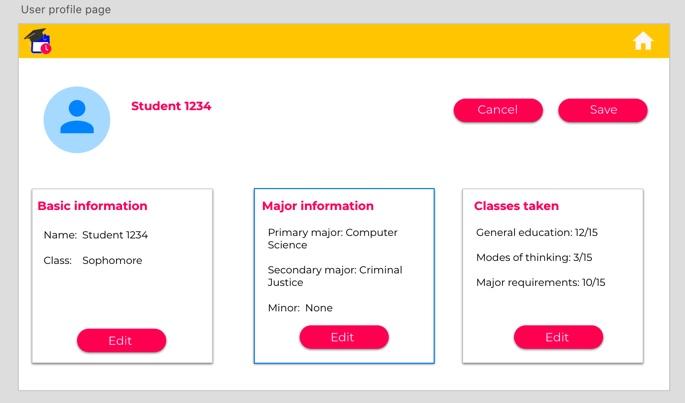
## User interface

<https://xd.adobe.com/view/1f1b7865-d992-47bb-7df0-0b4bbbc3f484-692a/>

## User interface issues

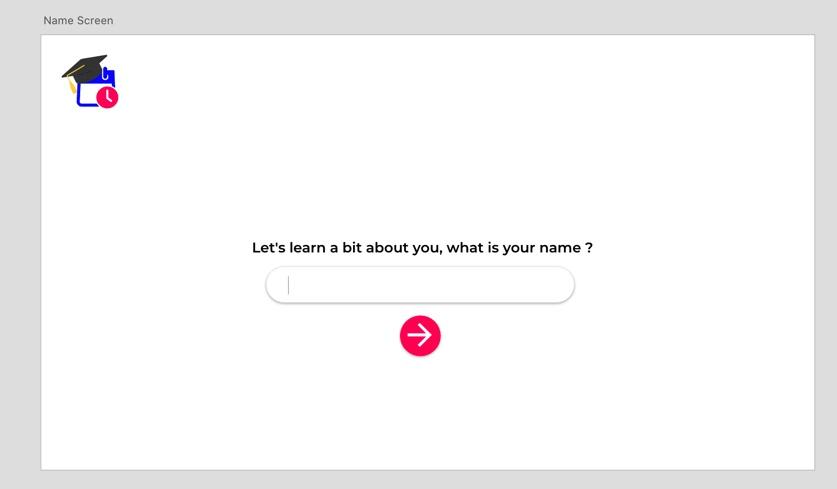
**2.4.1 If the user has a 3rd major**

If the user has a third major they will not be able to add it to the application as it is out of scope for the application. As seen in figure 1, the user settings only have a spot for the users primary, secondary major and minor as this is most common for students.

*(figure 1)*

**2.4.2 User does not enter typed information**

During the start of the application the user is asked several questions, with the drop down menus they are required to select an option or the application will default to the first option. However, where there is a blank box (figure 2) where the student has to type in information if they leave it blank will result in an error.

*(figure 2)*

# Detailed description of components (ONLY 2 ARE REQUIRED)

## X Component (or Class or Function ...)

Use exactly the template shown at the end of the document.

|  |  |
| --- | --- |
| Identification | Course Database |
| Type | Database |
| 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.  Function and Performance Requirements:   * Allows the application to reference the database for required courses for a specified major/minor. * Allows the application to generate suggested courses using the database. * Allows the application save mock schedules between sessions * Allows the application to save student profile information between sessions. * Allows the application to update the courses offered each semester. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.   * The database component saves application data between sessions * Data Items   + Course Data Table     - Course Name     - Course Number     - Course Section     - Professor Name     - Semester     - Timeslot     - Room Number     - Building Name   + Student Profile Table     - Student Name     - Student Number     - Major/Minor (Student can have up to 3 majors)     - Advisor Name   + Completed Courses Table   + Mock Schedule Table |
| 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.  The functions of the database depends on the functionality of the student view, and the ability for the database to obtain all course information for a specified semester.  Functional Dependencies:   * StudentView   + Collect student profile information   + Collect completed course information   + Import available courses for the current semester   + Save mock schedules created by the user   + Save suggested courses created by the application |
| Interfaces | Detailed descriptions of all external and internal interfaces as well as of any mechanisms 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.  Internal Interfaces:   * Select completed courses   + Users will be prompted with a screen to select which classes have already been completed.   + The course numbers selected will be added to the completed courses table in the database. * Search for offered courses   + When students are searching for classes through the studentView, the studentView will query the database depending on the field searched. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The 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. This information will probably be recorded in the data dictionary. |

## Y Component (or Class or Function ...)

Client application

|  |  |
| --- | --- |
| Identification | StudentView |
| Type | Client application |
| 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.  The primary functional requirement for this component is to provide a user interface for the application. The user will be able to use this component to accomplish the goals of the application. The performance requirements should consist of timely responses and low latency. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.   1. This component allows the user to interact with the application. It provides intuitive controls that lets the user accomplish his/her goals in the app. 2. Transformation process:    1. Inputs: Student Info, Major/Minor Info, Courses Taken    2. Outputs: Suggested Courses, Mock Schedule    3. The user inputs information about him/herself for course qualifications based on prerequisites (Ex. Only seniors can take senior seminar).    4. The user inputs major/minor info in order to tailor the courses taken page    5. The user inputs what courses he/she has taken to determine what further courses should be suggested and what courses can be taken based on prerequisites.    6. The program will use this information to offer course suggestions for the user’s tentative schedule.    7. The program uses the tentative schedule to export it to a PDF file. |
| 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.  This component requires a database of all courses at DeSales and another database of offered courses for a semester in order to work correctly. This component is used directly by the user, and the student’s generated schedule is dependent on the functionality of the StudentView. |
| Interfaces | Detailed descriptions of all external and internal interfaces as well as of any mechanisms 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.   * Choose Major/Minor Screen   + Screen will begin with one fragment allowing student to select major   + A drop-down menu will be used to select major   + Buttons underneath the fragment will allow the user to add a major or minor   + Button on fragment to delete a major/minor   + Button to submit info * Select Completed Courses   + Series of survey screens divided by graduation/major requirement section   + List of classes with a checkbox next to each one   + Forward and back arrow button to transition between screens * Home Screen   + Window with three separate panels:     - A quick view of the schedule the student is compiling     - Buttons that manipulate the schedule (Edit, View, Export, Create New)     - Settings options * Schedule Editor Screen   + Panel that has a visual display of the student’s current schedule   + Panel with 3 course suggestions that can be easily added to the schedule   + Buttons that allow the user to export or save the schedule * Settings Screen   + 3 panels that display information about: the user, the major, and class taken   + Buttons to edit any piece of this information   + Buttons that allow the user to save or cancel work |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The 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. This information will probably be recorded in the data dictionary. |

...

Examples of a component are:

1. Database
2. Server
3. Client application
4. Search Application
5. Etc.

# Reuse and relationships to other products

For teams doing enhancement work, reuse is an important issue. Most enhancement work should focus on extending, rather than replacing, the design and product development from earlier semesters. For teams doing new development, reuse can also be an important strategy. In some cases, there is freeware that could be incorporated. In other cases, there are existing modules or classes that could be adapted. Another possibility is the use of special tools that produce open source results and thus permissible under the terms of this course.

This section should include the following subsections as appropriate:

* How reuse is playing a role in your product design
* How reuse is playing a role in your product implementation (and the motivation for changes)
* If you are not reusing material that is available, then give motivation for why it is being thrown out.

# Design decisions and tradeoffs

* Basic requirement page

On the basic requirements page we initially were going to have a checkbox given menu that is broken up by sections from the major curriculum requirement paper. However, we found that this made the page too crowded and there was not enough room to fit all the classes nicely to resolve this we decided to go with a drop down menu that the user will check so that there is enough room on the page.

* Prerequisite error page

At first this page was not going to allow for the users to add the class anyway however, we decided to add this button as some users could eventually get a waiver to take the case.

* Colouring

In order to keep our bright colours we decided to place all the main information on white cards so that it remains easy to read and accessible

.

# Pseudocode for components

Utilize the use cases to create pseudocode for components.

# Appendices (if any)

# Software component template for section 3

The template given below suggests a reasonable structure for giving a thorough description of each component described in Part 3 of the SDS. The specific information depends in part on the design approach. Your team must adapt this template to your needs and describe it in section 3.1 of your SDS.

|  |  |
| --- | --- |
| Identification | The unique name for the component and the location of the component in the system. |
| Type | A module, a subprogram, 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. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified. |
| 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 descriptions of all external and internal interfaces as well as of any mechanisms 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. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The 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. This information will probably be recorded in the data dictionary. |