

**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

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

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

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

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

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

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

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

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

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

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

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

[3 Detailed description of components (ONLY 2 ARE REQUIRED) 3](#_heading=h.26in1rg)

[3.1 X Component (or Class or Function ...) 3](#_heading=h.lnxbz9)

[3.2 Y Component (or Class or Function ...) 4](#_heading=h.35nkun2)

[4 Reuse and relationships to other products 4](#_heading=h.1ksv4uv)

[5 Design decisions and tradeoffs 4](#_heading=h.44sinio)

[6 Pseudocode for components 4](#_heading=h.2jxsxqh)

[7 Appendices (if any) 4](#_heading=h.z337ya)

[Software component template for section 3 4](#_heading=h.3j2qqm3)

# 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

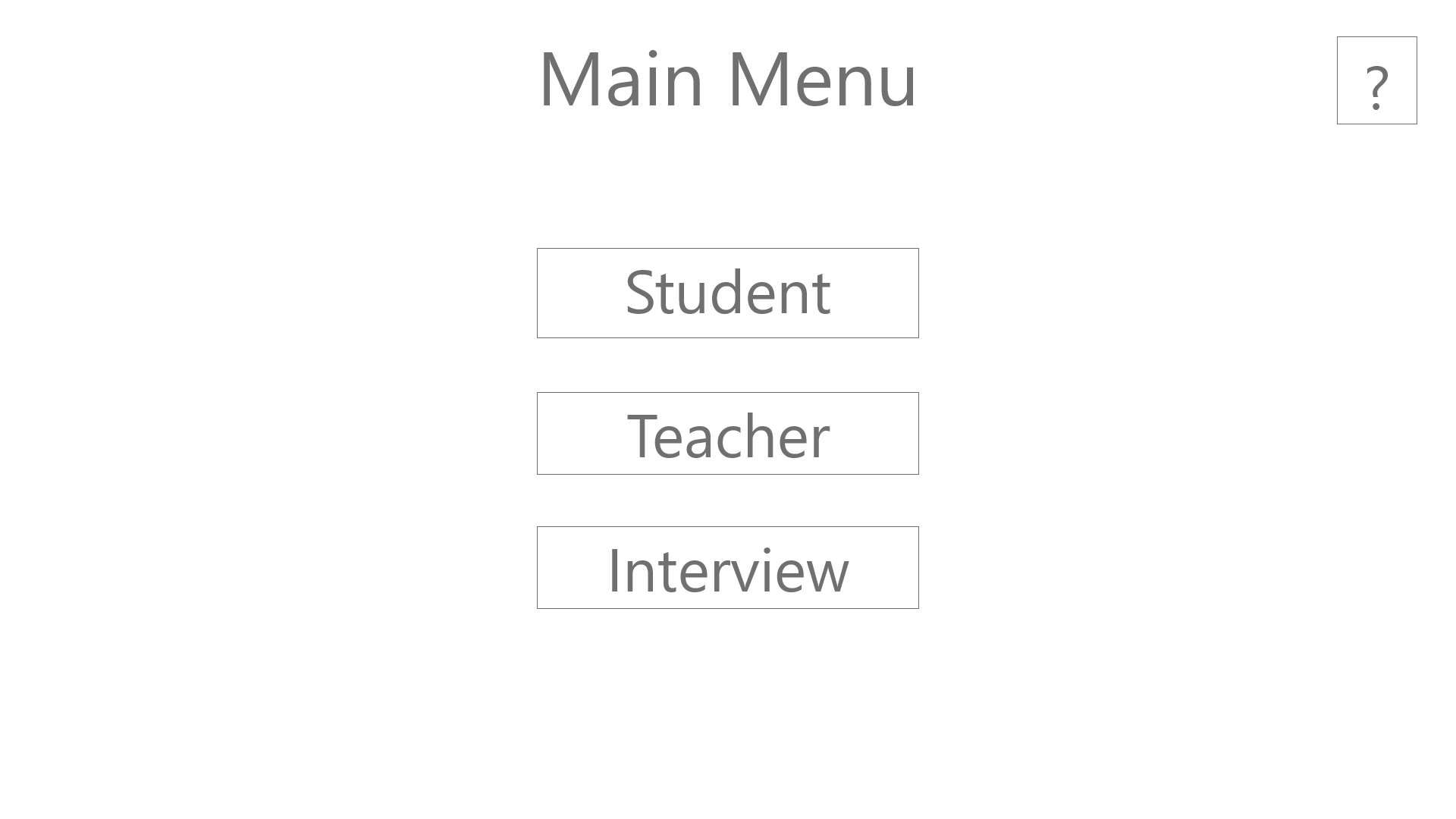
This subsection will introduce the various components and subsystems.

## 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

There are no error messages, any errors will be consumed by the browser.

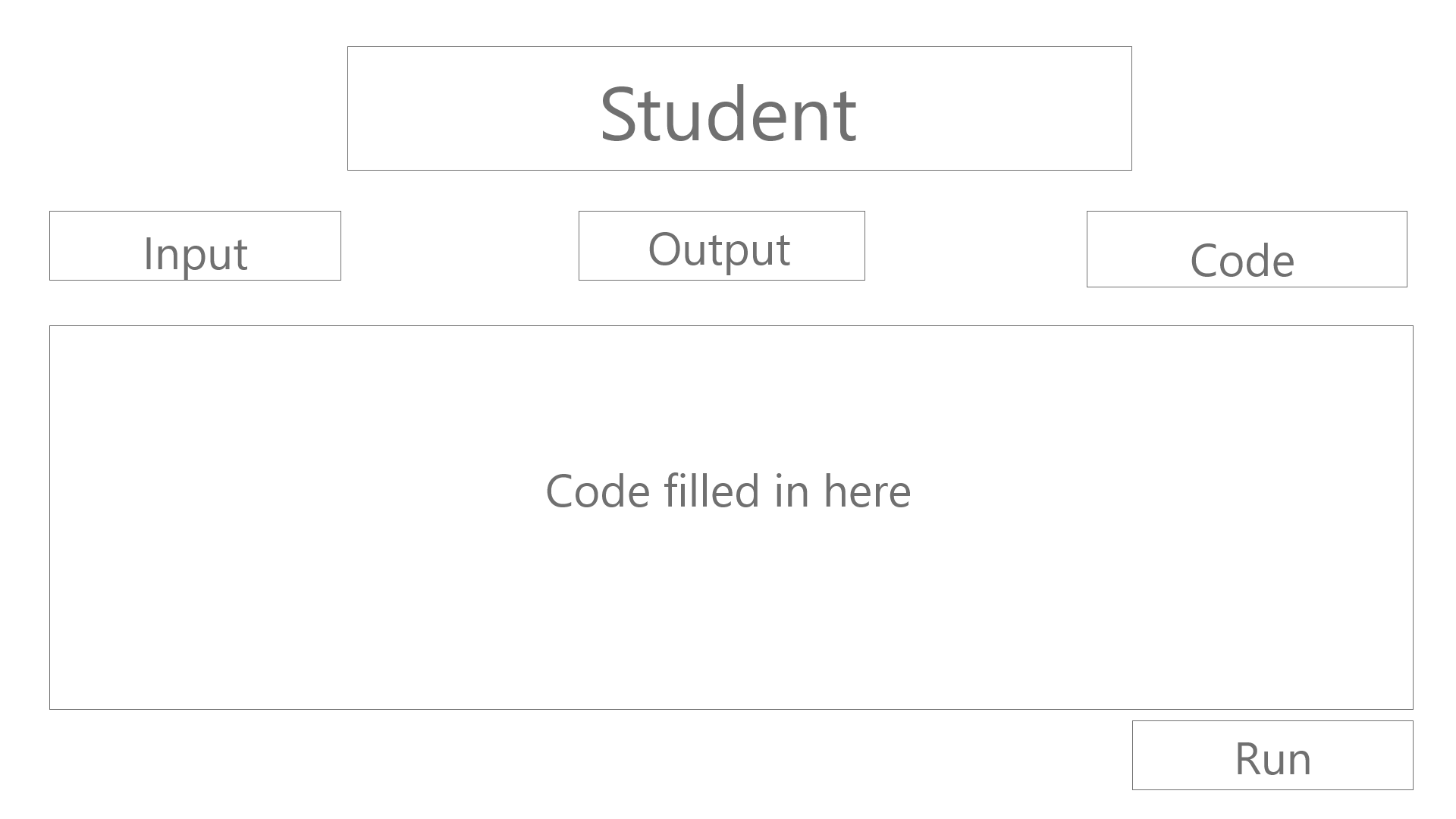
******

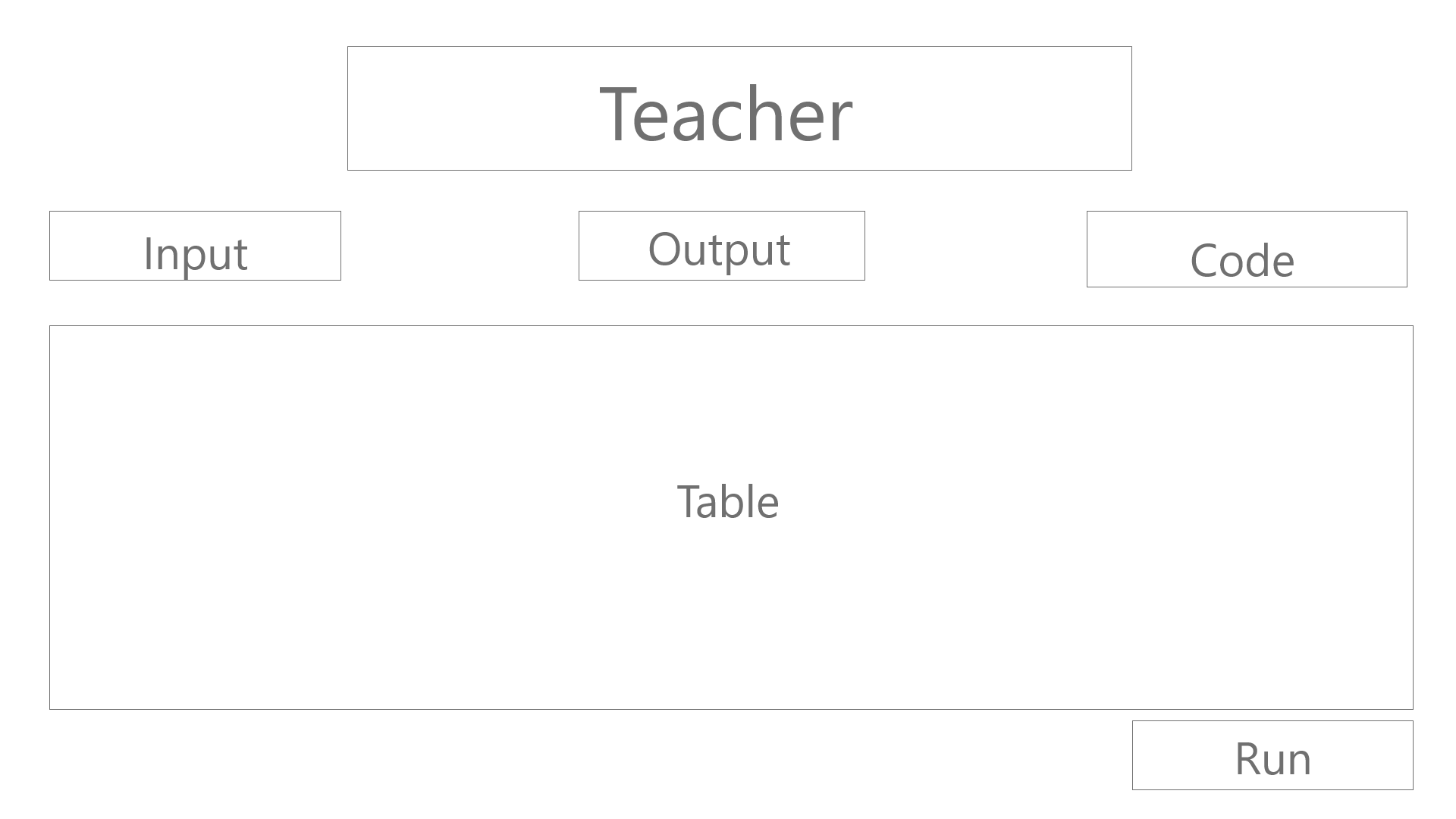
For the main menu the user will select:

Student if they are a student, will lead to the student view (see below)

Teacher if they are a teacher, will lead to the teacher view (see below)

Interview if this tool is being used as a tool during an interview, will lead to the interviewer view (see below)



******

Input - Pulls up a file select model (from the browser), expects an input file

Output - Pulls up a file select model (from the browser), expects an output file

Code - Pulls up a directory select model (from the browser), expects a code file

Run - Tests the code, returns an excel with a file dialogue asking where to save the results

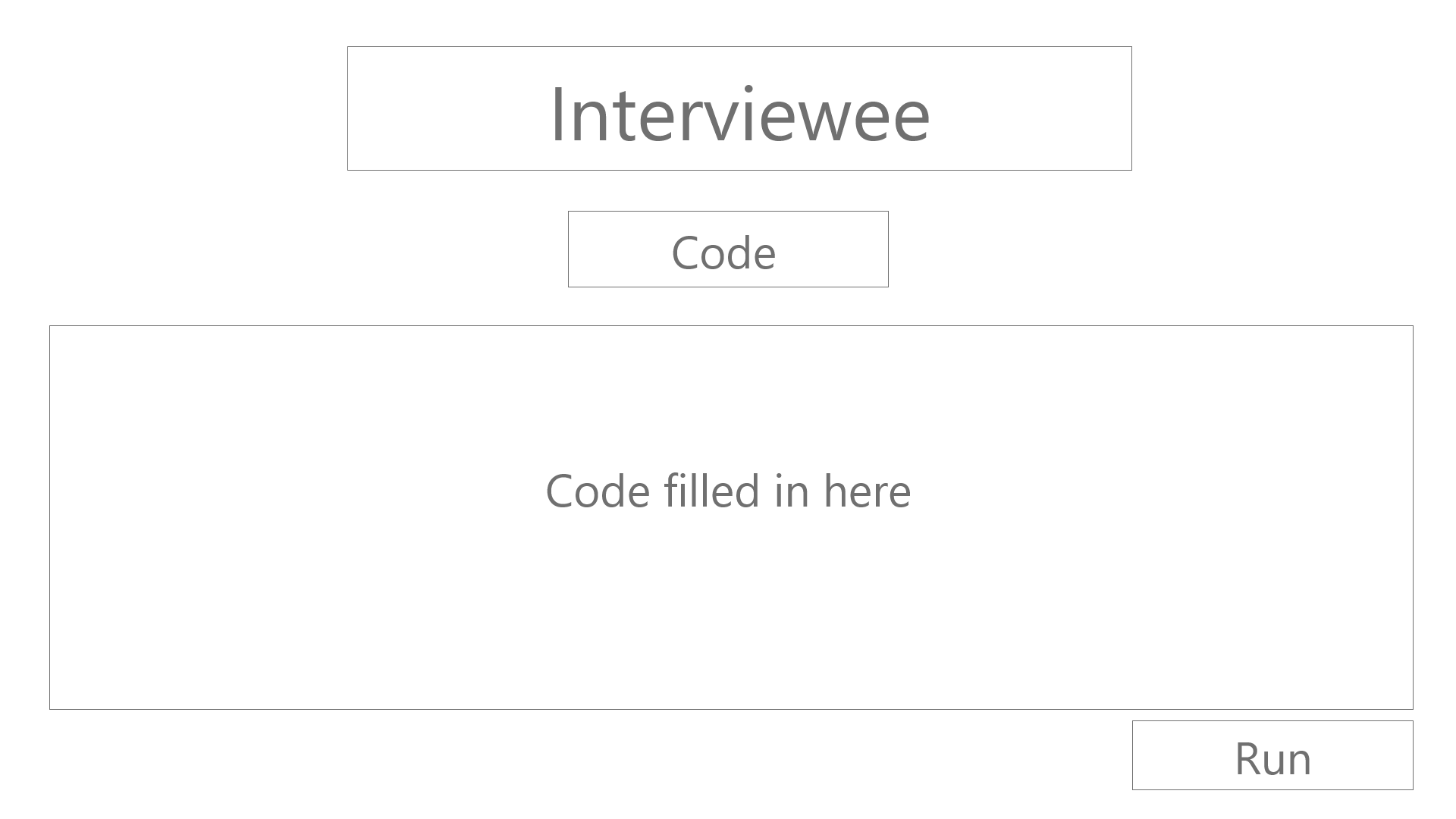
******

Input - Pulls up a file select model (from the browser), expects an input file

Output - Pulls up a file select model (from the browser), expects an output file

Code - Pulls up a file select model (from the browser), expects a code file

Continue - Loads the Interviewee window

******

Code - pulls up a model select so the user can select which file to run

Run - Code is run and a repo will be given

Output Model



Match Percentage - says how close your output is to the expected output

## User interface issues

We aim for simplicity in this effort our application is very bare boned. This limits both the amount of errors that could be present and documentation that must be written. Our UI will be designed around the premise of the least amount of components as possible to assure that our users are never confused.

# Detailed description of components (ONLY 2 ARE REQUIRED)

## File loader

|  |  |
| --- | --- |
| Identification | File Loader |
| Type | A Control Procedure |
| Purpose | To generate the needed inputs and outputs for the application |
| Function | Stores the location of the file in the front end so the server can locate and use its data to generate the student output |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | It requires the application to be hosted, the web browser to be properly functioning and have javascript enabled |
| Interfaces | The user will need to use the UI buttons to upload both the input and output files. The user will then navigate through their system file explorer that is built into their operating system. |
| 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. |

## Code Compiler

|  |  |
| --- | --- |
| Identification | Code Compiler |
| Type | A Control Procedure |
| Purpose | Get a readable output to compare generated output with designated output. |
| Function | Compiles Code (Java) |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | Having Python/Java installed and using a javascript enabled browser. The user will need to have uploaded a code file to run. |
| Interfaces | The user will need to press the run button in the UI after having uploaded their code file. |
| 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. |

# 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

We decided for a more linear design for the user. We wanted to allow for a simple UI that allows for all of the capabilities needed for the user. The simplicity of the UI allows users a greater understanding of our product and an easier way to check their code. Some of the design tradeoffs we made were:

1. Not using a database/not hosting our application

# Pseudocode for components

Utilize the use cases to create pseudocode for components.

.0