Phase 2: Specification/Analysis Modeling and Design Modeling

Note: *It is expected that all members of the group will make a reasonable contribution in writing parts of this report, that way one group member does not get overstretched. Furthermore, it ensures all members get practice of writing technical reports.*

Specification and design precisely describe what the product must do with any constraints.
In your project, you are asked to utilize the semi-formal techniques to write the specification i.e., you would utilize below two modeling techniques:

1) **Scenario-based modeling** – use case diagram for your entire system.
2) **OO Modeling in Agile**, i.e., CRC Model and UML Class Diagram

Thus, in your project, writing the specification document involves 5 activities:

1. Create a high level UML Use-case diagram for your ***entire system*** *(based on all user stories from your project backlog)*.
2. Create a database schema diagram (in case your project involves a database of any form).
3. Create a CRC Model for your ***first sprint*** from your user stories.
4. Construct UML Class Diagrams (include any sub-diagrams) from CRC Model for your ***first sprint***.

You may utilize any graphical tool to draw diagrams. MS Visio, genmymodel.com, lucidchart.com, draw.io, Libre Office (Draw), etc.

If needed I shall give qualitative feedback either face to face in Lab or via Canvas.

Format for the documentation is as follows:

Specification/Analysis and Design Modeling

1. Introduction

1.1 Purpose

<This subsection should Delineate the purpose of this document . It should also include information such as class name, semester, group name, members, contributions of each member listed individually to this phase of project and to this report, etc.>

Purpose: *The Specification/Analysis and Design Modeling document describes the outline of the project as well as providing an overview and illustration of the Use-Case Diagram, CRC Model, and UML Class Diagrams.*
Class Name: *Comp 380 Software Engineering*

Semester: *Spring 2023*
Group name*: Webcraft*
Members: *Elliot Fayman, Ryan Kao, Christina Mourad*
Contribution:

Elliot*: Configured Twilio to post to a specific Flask endpoint and added a Flask endpoint that processes a Twilio post.*
Ryan*: Added a method to the flask API that sends a SMS message to a specific phone number and also create a method that allows the receiver to receive the SMS message that was sent from another user.*
Christina: *Implemented a new view for displaying messages and improving the app. I will also be working on creating a new Flutter UI that will allow the user to type and send text messages.*

1.2 Definitions, Acronyms, or Abbreviations

<This subsection should provide the definitions of any acronyms, and abbreviations you use and required to properly interpret the Specification (S). This information may be provided by reference to one or more appendixes in the S or by reference to other documents.>

CodePage: *a separate page for the user to send and receive messages*
HomePageFloatingActionButton*: a clickable button located on the homepage of the app that takes you to the "codepage"*

1.3 References

<For any external or internal references used in your report this subsection should

a) Provide a complete list of all documents referenced in the S;
- Jira
- GitHub

b) Identify each document by title, report number (if applicable), date, and publishing organization;
c) Specify the sources from which the references can be obtained.
This information may be provided by reference to an appendix or to another document.>

- ***Jira source link***
    - *https://webcraft.atlassian.net/jira/software/projects/TEX/boards/1*

- ***GitHub source link***
    - *https://github.com/elliotfayman/TextPy*

1.4 Overview

<This subsection should
a) Describe what the rest of the S contains;

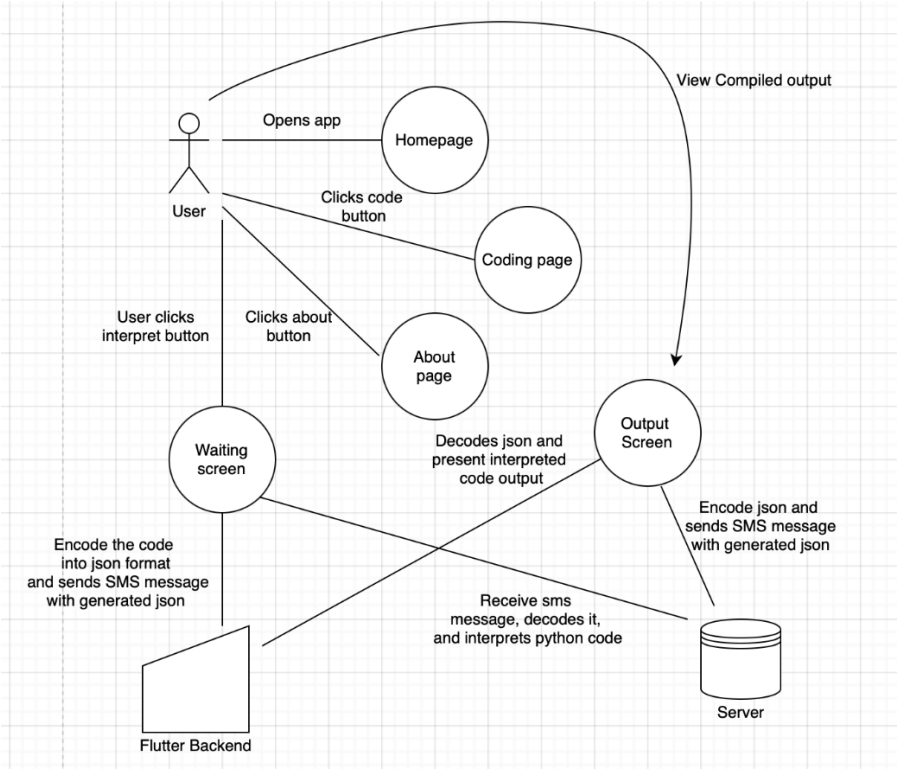
- ***2. Use-Case Modeling diagram based on user stories.***
- ***3.1 CRC Model of the project first sprint from user stories***
- ***3.2 UML Class Diagrams of the project first sprint***

b) Explain how the S is organized.>

- ***2. Use-Case Modeling:*** *The Use-Case Diagram uses a conceptual illustration of actors, which in our case were user, flutter backend, and server, as well as their respected actions and overall roles.*
- ***3.1 CRC Model:*** *The CRC model consist of CRC cards that are organized by the different classes within the project app. Each card contains the class name, description, responsibilities, and collaborator(s).*
- ***3.2 UML Class Diagram:*** *The UML Class Diagram contains classes that are represented by boxes that are organized by name, attributes of the class, and operations/methods.*

2. Use-Case Modeling

Draw a high level use-case diagram for your entire system (based on your project backlog all user stories) based on the user stories for the project.



3. Class Modeling

3.1 Create a CRC Model for your ***first sprint*** from your user stories.

***Class**: HomePage*	
***Description**: A welcome page to the app that contains the app name "TextPy" and a "Click to Continue" button*	
Responsibility:	***Collaborator:***
Greet the user with homepage, contains the button that will allow for the user to access the code page	*HomePageFloatingActionButton*

***Class:** CodePage*	
***Description**: A page that provides a messaging board so that the user can type Python code and receive interpreted Python code*	
Responsibility:	***Collaborator:***
Provide an area for a text field *Provide an area for a response field (text box)* *Continuously refresh code page to check for newly interpreted Python code in the server*	*sendButton, aboutButton, userTextField, userTextBox*

***Class:** HomePageFloatingActionButton*	
***Description**: A clickable button that states "Click to Continue" located at the FirstPage*	
Responsibility:	***Collaborator:***
The button redirects the user from the first homepage to the second page	*CodePage*

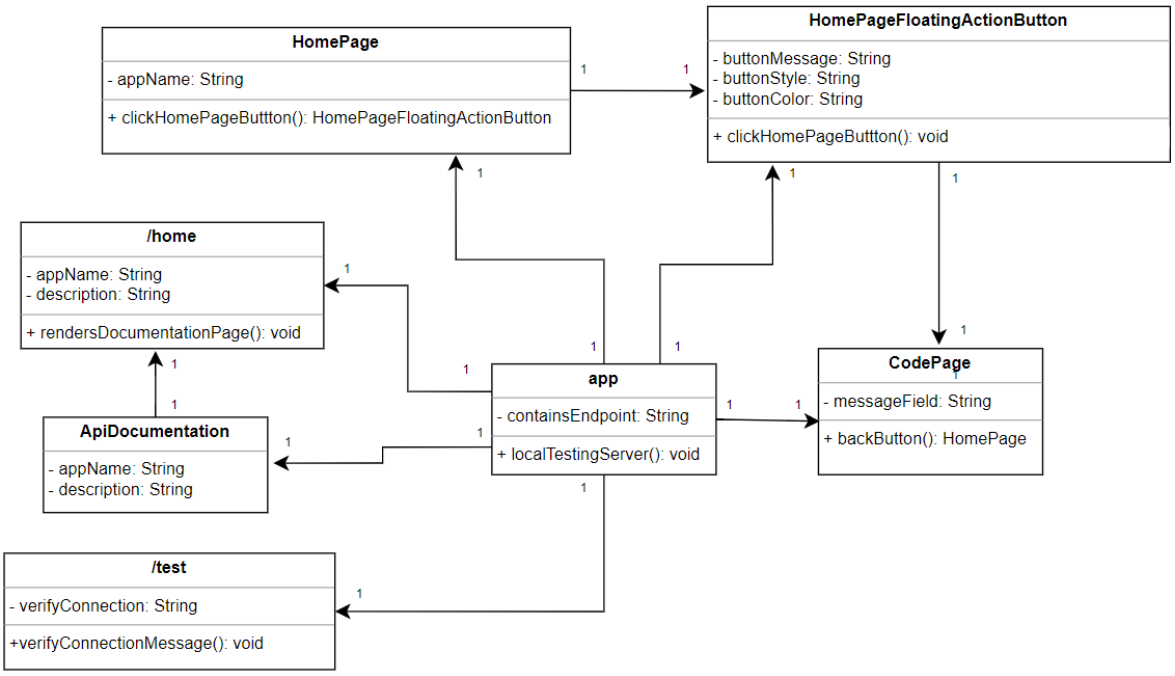
***Class:** ApiDocumentation*	
***Description**: A html page that contains use case information related to the purpose and functionality of the API*	
Responsibility:	***Collaborator:***
Provides information on each API endpoint *Provides information on how to use API endpoint* *Provides point of contact for errors that occur as a result of using the API*	

Class: */home*	
Description*: A get API endpoint that renders the API documentation page that helps provide information about API functionality*	
Responsibility:	***Collaborator:***
Renders API documentation page	*ApiDocumentation, app*

Class: */test*	
Description*: A get API endpoint that allows for the verification of a proper connection to the server*	
Responsibility:	***Collaborator:***
Verifying successful connection, returning successful connection json message	*app*

Class: *app*	
Description*: Containerizes each flask endpoint and allows for the user to launch a test server to test the functionality of the API locally*	
Responsibility:	***Collaborator:***
Containerize each flask endpoint *Provide a platform for a local testing server*	*/home, /test*

3.2 Construct UML Class Diagrams (include any sub diagrams) from CRC Model for your ***first sprint***. Provide all instance data members and methods for your classes.



4. Database schema diagram (in case your project involves a database of any form)
Indicate all relationships, keys, and attributes for each table.
Otherwise:
Show how the data would be organized in the files. How many files are there and how the data fields are notionally linked from one file to the other. Create a diagram for easy viewing.

Our project does not involve a database of any form.