**Final Documentation Template**

You will need to provide appropriate documentation with your project. Although the construction of appropriate javadoc documentation for the code is required, this is not the same as the project documentation. Remember that you can use javadocs through the design phase to document the design. The material that follows will indicate what your project documentation should contain.

*Cover Page*

The cover page should indicate the name of your project, the members of your team, that this is a course requirement and the date of the release of the document.

*Intellectual property*

Provide a simple assignment of a copyright and intellectual property rights. For example, Copyright 2022 AUTHORS.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3, (3 November 2008) or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found at [https://www.gnu.org/licenses/fdl.html.](%20https://www.gnu.org/licenses/fdl.html.)

*Table of Contents*

Each section should be included along with the page number. Example:

Table of Figures ……………………………………………. i

1.0 Project Description ………………………………………1

1.1 TBD ……………………………………………...1

2.0 Project Management ……………………………………..2

*Table of Figures*

Provide a **list** of the figures in your document. These page numbering use the i, ii, iii format. Prior to the document pages.

***Followed by an INTENTIONALLY LEFT BLANK page before PAGE 1 of the Document.***

*1.0 Project description*

This section should provide your basic description of the project. These will be very brief, an overview of the project's purpose and main features.

*2.0 Project management*

This is the description of the work that went into the project.

*History*

This section will provide a description of what you have done during the semester. You should provide a brief description of the accomplishments made over the semester. Include a project time line with milestones.

*Personnel*

Identify the members of your team. Briefly indicate the educational background of the member and any additional professional or experiential detail, if relevant. If a member of the team had the lead on a particular part of your project, clearly indicate this.

*Effort*

Indicate approximately how much time the team as a whole as spent on the project. Also, indicate the number of meeting that occurred during the semester.

*3.0 Use Cases*

Place your principle use cases here. You should describe these appropriately (see slide deck 2a) so that they can be easily understood. Number them for reference in other sections.

*4.0 Requirements*

The description of the requirements are based on your project description and the use cases.

*Functional requirements*

If you use special words with specific meanings, define them first. This should be from the user or client point of view. Use the use cases as a guide. These will be listed in summary form in the *Summary* section below.

Describe:

* what the functions of the software are, not how these are done
* what is expected from the program, not what the program looks like
* Your use cases probably highlight specific requirements that your system must implement.

Each individual requirement should be numbered in a standard way. These will be referenced in your testing section. Example:

R*1* The software shall allow for the selection of a game from a collection of games.

R*2* The software will keep an accurate accounting of the user’s available money.

*User interface requirements (diagrams/sketches of the envisioned UI are very useful here)*

If you use special words with specific meanings, define them first.

Describe:

* what the program should look like
* what the input elements are
* what the output elements are

Remember to discuss how invalid user input is handled, i.e., what happens if the information supplied by the user is incorrect. Do not describe how the error handling is actually realized, just indicate what follows if some input is invalid.

*Future modification and extensions (Optional in Requirement Phase, Required in Final Project)*

Describe how the program could be extended in the near future. These are the things that will not appear in your implementation, but you will take these things into consideration in your design. You may use additional scenarios, window sketches, etc., to illustrate the future extension.

*Summary*

This should be a list of statements that indicate what the software shall do. These should have a clear numbering and be clearly linked to the scenarios. For example, “The software shall accept a user selection of direction and produce the appropriate scene as presented in Use Case 1.4”. You might consider a table since you will need to show how the requirements are linked to the scenarios. Consider this to be the general contract for your system. You will satisfy the contract if you satisfy each requirement. This does not mean that you have satisfied the contract well or that the contract was interesting. These last two items will also be used in evaluating your performance

*Associated tests (Optional in Requirements Phase, Required in Final Project Document)*

In this section describe the test you will use to determine whether the requirement is satisfied. There will probably be a bit of text but there should also be a clear summary (perhaps a table) that indicates the requirement and the test.

*5.0 Design*

This section describes the static and dynamic elements of your system and the relation of the model to the user interface (views and controls). Descriptions of the final CRC cards may be used here. The actual CRC cards should be placed in an appendix. At least some of the design should be documented with appropriate UML class diagrams (This section should include your UML static CLASS DIAGRAM and your example SEQUENCE DIAGRAMS). Focus on areas that have classes and interactions specific to your project. In other words, no not provide diagrams for common data structures or utilities.

Since the MVC architecture is used in the project, describe your data model, the views, and the controls. Indicate the communication between the parts. You should, for example, describe your overall data model. For each class, describe it, define its purpose and how its state is maintained/updated/initialized. For the view, describe each class, its purpose and how it interacts with objects in the model, and so on. This section is about design and not implementation.

*Model design*

What is in the model and what are the ways in which it can change?

*View design*

What are the views of the data and how are they generated?

*Control design*

What are the elements by which the user can request a change to the model?

*6.0 Implementation*

This section provides descriptions of how the design is implemented. The details should be in the javadocs and other code comments. You should indicate which, design patterns you have implemented.

*Packages and classes*

Describe the packages, classes, components, interfaces, and the relationships that implement the design. Make sure you explicitly indicate how a package, class or group of classes relates to or satisfies a design specification in the previous section. Describe each class. The method details for each function should be in the javadocs comments.. For each public class indicate in what package it is contained. For each public class clearly indicate if it is abstract or concrete, whether it extends another class, and whether it implements an interface. For each public class describe the API for that class.

*Utility classes and packages*

Describe the packages and classes that are used to satisfy design specifications but are not in the actual design. This would be the placed to indicate the way in which you satisfied the four component requirements that are not explicitly examined above. You can consider the packages and classes that deal with the views and controls as satisfying the GUI component.

*Tested functionality*

Describe functionalities and the outcome of the tests, i.e., list functionalities that have been tested and present the results of the tests. Note that all the use cases/functionality listed in the requirements document should be tested.

*Untested functionality*

Describe the functionalities that have not been tested, and include an explanation of why the tests have not been completed.

*7.0 Discussion*

Provide a discussion of any items that you think are important. This should include anything that you want to say about trade-offs that you have made, particular implemented features that you want to highlight, and any lessons learned.