Product Design

| **Team** | <Team Number, Name & Team Members Names In **Bold Black**> |
| --- | --- |

***Delete or replace everything in blue. In the end, all of the text should be black.***

***This is a “living document”, meaning its content and format will grow with the implementation of the project. Use it to capture key project concepts and document important design decisions.***

***You should use any drawing tool for your UML diagrams. Look at the schedule page for links to various UML tools. If your diagrams are too big to cut and paste into this document, provide a reference to the external image file(s) [JPG or PNG] where they can be found or segment your image into legible sections.***

# Introduction

Provide an overview and purpose of the document.

# System Overview

Provide a general description and functionality of the software system. Describe some potential real-life scenarios of people actually using your product. How will they use it and what problem would it solve?

# **Design Overview**

## **Architectural design**

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the responsibilities of the system were partitioned and then assigned to subsystems. Identify each high-level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Add diagrams showing the major subsystems and data repositories and their interconnections.

## **System interfaces**

### **User Interface**

*Explain how the user will be able to use your system to complete all the expected features. Add screenshots if possible. If this is not applicable to your project, you may omit it.*

### **APIs**

*Explain what APIs you will be exposed to enable the users to interact with your system. Give the API definitions clearly.* ***If this is not applicable to your project, you may omit it.***

## **Model**

Draw a simple class diagram and describe the classes in the table in this section. This diagram should represent the classes and their relationships. *It is only necessary to show methods that are publically accessible by other classes. Only show an instance variable of a class if it is publically accessible*. The diagram and the table should be consistent with each other.

*Identify the classes (logical groupings of software methods that provide a related set of services). Make sure the design conforms to good design principles.*

*For each class, specify the information it maintains and the functionality it provides. Provide sufficient detail so that the purpose of each class in the design is clear.*

| <Class No. 1> | eClass state   * What information is the class responsible for maintaining? * E.g., a printing subsystem might hold the current status of all the printers it controls as well as the queue of print jobs waiting to be printed.   Class behaviour   * What methods does the class implement? * E.g., classes related to the printing subsystem might support the queuing up of new jobs, estimating the time until a given job completes, or emailing status information at the end of a job. |
| --- | --- |
| <Class No. 2> | Class state   * What information is the class responsible for maintaining?   Class behaviour   * What methods does the class implement? |
| <Class No. 3> add more rows as needed. | Class state   * Etc.   Class behaviour   * Etc. |

.

# **Sequence Diagram(s)**

UML sequence diagram for the agreed significant use cases (about 4 major use cases).

*While you may reference an external file here, most instructors* ***strongly*** *prefer embedded images, or, failing that, external files in generic formats such as JPEG, PNG or PDF.*

# **Design Rationale**

This is a running list of issues that arise as your design process proceeds. This is an important section of the design document as it captures the **thought process** of the product's designers. It includes why or why not (rejected solutions) a design decision was made and supports future changes to the product. It should be updated whenever a design change occurs.

*It is RARELY the case that the first design you consider is the best one you can come up with that meets the requirements and that can be implemented, tested, and delivered on schedule. Your instructor will be looking for signs that you considered at least a few approaches, and that you had a coherent rationale for preferring the design your team eventually adopts.*

*This is the place to record such thoughts – what alternatives did you consider? What are the final design's strengths (and deficiencies) compared to the other alternatives considered? Why did you select the approach you finally chose? This last question should be answered with an eye to the inevitable tradeoffs in creating an appropriate design.*