Software Requirements Specification for SFWRENG 4G06: Dice Duels: Duel of the Eights

Team 9, dice_devs
John Popovici
Nigel Moses
Naishan Guo
Hemraj Bhatt
Isaac Giles

October 8, 2024

Contents

Revision History

Table 1: Revision History

Date	Developer(s)	Change
2024-09-27	John Popovici	Set up formatting
2024-09-30	Nigel Moses	Added three blank sections (Commonalities related sections)
2024-10-04	Nigel Moses	Added content to section 11, 12, 13
2024-10-05	Hemraj Bhatt	Added content to section 2 and 3
2024-10-05	John Popovici	Added content to section 4.1 and 4.1.1
2024-10-05	John Popovici	Fixed compiling issues and formatting in section 3
2024-10-07	John Popovici	Added goals and stretch goals to section 4.1.3
...	...	...

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°C	temperature	centigrade
J	energy	joule
W	power	watt ($\mathrm{W} = \mathrm{J\,s^{-1}}$)

[Only include the units that your SRS actually uses. —TPLT]

[Derived units, like newtons, pascal, etc, should show their derivation (the units they are derived from) if their constituent units are in the table of units (that is, if the units they are derived from are used in the document). For instance, the derivation of pascals as $\mathrm{Pa} = \mathrm{N\,m^{-2}}$ is shown if newtons and m are both in the table. The derivations of newtons would not be shown if kg and s are not both in the table. —TPLT]

[The symbol for units named after people use capital letters, but the name of the unit itself uses lower case. For instance, pascals use the symbol Pa, watts use the symbol W, teslas use the symbol T, newtons use the symbol N, etc. The one exception to this is degree Celsius. Details on writing metric units can be found on the NIST web-page. —TPLT]

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	unit	description
A_C	$\mathrm{m^2}$	coil surface area
A_{in}	$\mathrm{m^2}$	surface area over which heat is transferred in

[Use your problems actual symbols. The si package is a good idea to use for units. —TPLT]

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
SFWRENG 4G06	[put an expanded version of your program name here (as appropriate) —TPLT]
TM	Theoretical Model

[Add any other abbreviations or acronyms that you add —TPLT]

1.4 Mathematical Notation

[This section is optional, but should be included for projects that make use of notation to convey mathematical information. For instance, if typographic conventions (like bold face font) are used to distinguish matrices, this should be stated here. If symbols are used to show mathematical operations, these should be summarized here. In some cases the easiest way to summarize the notation is to point to a text or other source that explains the notation. —TPLT]

[This section was added to the template because some students use very domain specific notation. This notation will not be readily understandable to people outside of your domain. It should be explained. —TPLT]

[This SRS template is based on ???. It will get you started. You should not modify the section headings, without first discussing the change with the course instructor. Modification means you are not following the template, which loses some of the advantage of a template, especially standardization. Although the bits shown below do not include type information, you may need to add this information for your problem. If you are unsure, please can ask the instructor. —TPLT]

[Feel free to change the appearance of the report by modifying the LaTeX commands. —TPLT]

[This template document assumes that a single program is being documented. If you are documenting a family of models, you should start with a commonality analysis. A separate template is provided for this. For program families you should look at ??. Single family member programs are often programs based on a single physical model. General purpose tools are usually documented as a family. Families of physical models also come up. —TPLT]

[The SRS is not generally written, or read, sequentially. The SRS is a reference document. It is generally read in an ad hoc order, as the need arises. For writing an SRS, and for reading one for the first time, the suggested order of sections is:

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

—TPLT]

[Guiding principles for the SRS document:

- Do not repeat the same information at the same abstraction level. If information is repeated, the repetition should be at a different abstraction level. For instance, there will be overlap between the scope section and the assumptions, but the scope section will not go into as much detail as the assumptions section.

—TPLT]

[The template description comments should be disabled before submitting this document for grading. —TPLT]

[You can borrow any wording from the text given in the template. It is part of the template, and not considered an instance of academic integrity. Of course, you need to cite the source of the template. —TPLT]

[When the documentation is done, it should be possible to trace back to the source of every piece of information. Some information will come from external sources, like terminology. Other information will be derived, like General Definitions. —TPLT]

[An SRS document should have the following qualities: unambiguous, consistent, complete, validatable, abstract and traceable. —TPLT]

[The overall goal of the SRS is that someone that meets the Characteristics of the Intended Reader (Section 2.3) can learn, understand and verify the captured domain knowledge. They should not have to trust the authors of the SRS on any statements. They should be able to independently verify/derive every statement made. —TPLT]

2 Introduction

Do you know the game Yahtzee? Played with dice, it's like Poker but more varied, requiring different skills, without betting. It uses 5 dice, with the usual dots representing the numbers 1 to 6. In rolling the dice, players try to create specific formations, thereby scoring points. At the end of a set number of rounds of dice rolls, the player with the higher score wins.

But the cube isn't the only possibility for dice as the octahedron, with 8 sides could be used. Scoring could likewise be done each round. The number of dice could be changed. These suggest an expanded version of Yahtzee.

This project creates an online multiplayer game platform that allows for the creation of custom Yahtzee-like games and variants. This family of games will come with some presets such as classic Yahtzee, Dr. Paul's octahedron version, and more, but will allow for the users to set their own variables such as number of dice, what kind of dice, and some elements of scoring, and then play that game. Kinds of dice would include cubes and octahedrons, among other multi-sided dice, and scoring could be calculated at the end of the game, as in classic Yahtzee, or on a per-round basis, where hands go in a head-to-head matchup.

This section includes a general overview of the entire SRS document providing descriptions of all sections. It also outlines of the areas of knowledge needed to grasp the documentation accurately.

2.1 Purpose of Document

[This section summarizes the purpose of the SRS document. It does not focus on the problem itself. The problem is described in the "Problem Description" section (Section 4.1). The purpose is for the document in the context of the project itself, not in the context of this course. Although the "purpose" of the document is to get a grade, you should not mention this. Instead, "fake it" as if this is a real project. The purpose section will be similar between projects. The purpose of the document is the purpose of the SRS, including communication, planning for the design stage, etc. —TPLT]

The Software Requirements Specification (SRS) document aims to clearly define the functional and non-functional needs of the project in order so that all parties involved have a common understanding of the objectives and requirements of the project. Throughout the whole software lifecycle, the SRS will serve as the development team's fundamental guide, aiding in the phases of design, implementation, and testing. In addition, it facilitates effective communication between stakeholders and creates a framework that ensures the end result satisfies user requirements and is in line with the project's pre-determined goals.

2.2 Scope of Requirements

[Modelling the real world requires simplification. The full complexity of the actual physics, chemistry, biology is too much for existing models, and for existing computational solution

techniques. Rather than say what is in the scope, it is usually easier to say what is not. You can think of it as the scope is initially everything, and then it is constrained to create the actual scope. For instance, the problem can be restricted to 2 dimensions, or it can ignore the effect of temperature (or pressure) on the material properties, etc. —TPLT]

[The scope section is related to the assumptions section (Section 4.2.4). However, the scope and the assumptions are not at the same level of abstraction. The scope is at a high level. The focus is on the "big picture" assumptions. The assumptions section lists, and describes, all of the assumptions. —TPLT]

[The scope section is relevant for later determining typical values of inputs. The scope should make it clear what inputs are reasonable to expect. This is a distinction between scope and context (context is a later section). Scope affects the inputs while context affects how the software will be used. —TPLT]

The scope of the project involves developing a modified Yahtzee game that features adjustable numbers of playable dice and various game attribute variations. This game will be available for both offline and online play, offering single-player and multiplayer. The scope does not include the implementation of advanced graphics, advanced computer opponents and other board games.

2.3 Characteristics of Intended Reader

[This section summarizes the skills and knowledge of the readers of the SRS. It does NOT have the same purpose as the "User Characteristics" section (Section 3.2). The intended readers are the people that will read, review and maintain the SRS. They are the people that will conceivably design the software that is intended to meet the requirements. The user, on the other hand, is the person that uses the software that is built. They may never read this SRS document. Of course, the same person could be a "user" and an "intended reader." —TPLT]

[The intended reader characteristics should be written as unambiguously and as specifically as possible. Rather than say, the user should have an understanding of physics, say what kind of physics and at what level. For instance, is high school physics adequate, or should the reader have had a graduate course on advanced quantum mechanics? —TPLT]

The intended readers of this SRS document are mainly software developers and game designers with expertise in 3D game development and design using game engines, in this case Godot. They should possess a solid understanding of game mechanics, particularly those related to dice games, and have experience with 3D modelling and physics simulations. Additionally, experience with C# and .NET is necessary for readers due to the development being done using C# on a.NET framework. A university-level undergraduate understanding of probability theory is also crucial for comprehending the game's underlying mechanics and probabilities due to the differing number of playable dice. It is assumed that readers are well-versed in these areas.

2.4 Organization of Document

[This section provides a roadmap of the SRS document. It will help the reader orient themselves. It will provide direction that will help them select which sections they want to read, and in what order. This section will be similar between project. —TPLT]

This SRS document is structured to provide a clear roadmap for readers. The sections are as follows:

- **Introduction**
  - Outlines the purpose and scope of the SRS.
- **General System Description**
  - Outlines an overview of the project, along with user characterstics, system constraint and the interfaces between the system and its environment.
- **Specific System Description**
  - Indepth system description containg high-level probelem and goal description, along with solutions characteristics, assumptions, definiations and instance models detailed functionalities and features of the system.
- **Requirements**
  - Outlines both functional and non-functional requirements of the system.
- **Likely Changes**
  - Outlines anticipated modifications to the system.
- **Unlikely Changes**
  - Outlines ascpects of the system that will remain static.
- **Traceability Matrices and Graphs**
  - Tracking of requirements throughout the development process.
- **Development Plan**
  - Outlines the general timeline and milestones for the project.
- **Values of Auxiliary Constants**
  - Outlines constant parameters used in report.
- **Commonalities**

  - Outlines commonalities between differnt aspects of the system.

- **Variabilities**
  - Outlines variables between differnt aspects of the system.
- **Parameters of Variations**
  - Outlines the different variations of aspects of the system.
- **Appendix — Reflection**
  - Indepth insights into the development process and lessons learned.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints. [This text can likely be borrowed verbatim. —TPLT]

[The purpose of this section is to provide general information about the system so the specific requirements in the next section will be easier to understand. The general system description section is designed to be changeable independent of changes to the functional requirements documented in the specific system description. The general system description provides a context for a family of related models. The general description can stay the same, while specific details are changed between family members. —TPLT]

3.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs → Calculations → Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

Figure 1: System Context

[For each of the entities in the system context diagram its responsibilities should be listed. Whenever possible the system should check for data quality, but for some cases the user will need to assume that responsibility. The list of responsibilities should be about the inputs and outputs only, and they should be abstract. Details should not be presented here. However, the information should not be so abstract as to just say “inputs” and “outputs”. A summarizing phrase can be used to characterize the inputs. For instance, saying “material properties” provides some information, but it stays away from the detail of listing every required properties. —TPLT]

- User Responsibilities:
  -
- SFWRENG 4G06 Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  -

[Identify in what context the software will typically be used. Is it for exploration? education? engineering work? scientific work?. Identify whether it will be used for mission-critical or safety-critical applications. —TPLT] [This additional context information is needed to determine how much effort should be devoted to the rationale section. If the application is safety-critical, the bar is higher. This is currently less structured, but analogous to, the idea to the Automotive Safety Integrity Levels (ASILs) that McSCert uses in their automotive hazard analyses. —TPLT]

[The —SS]

3.2 User Characteristics

[This section summarizes the knowledge/skills expected of the user. Measuring usability, which is often a required non-function requirement, requires knowledge of a typical user. As mentioned above, the user is a different role from the "intended reader," as given in Section 2.3. As in Section 2.3, the user characteristics should be specific an unambiguous. For instance, "The end user of SFWRENG 4G06 should have an understanding of undergraduate Level 1 Calculus and Physics." —TPLT]

This section summarizes the knowledge and skills expected of the user:

- The end user of the Yahtzee 3D game should have an understanding of the Yahtzee game, including its rules and strategies.
- Users should have a undergraduate level of understanding of probability theory, as it is important for comprehending the game's mechanics and making informed decisions during gameplay.
- A foundational knowledge of 3D environments and interactions may enhance the user's experience but is not strictly required.

3.3 System Constraints

[System constraints differ from other type of requirements because they limit the developers' options in the system design and they identify how the eventual system must fit into the world. This is the only place in the SRS where design decisions can be specified. That is, the quality requirement for abstraction is relaxed here. However, system constraints should only be included if they are truly required. —TPLT]

4 Specific System Description

This section presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

Games are a staple of entertainment and are used to bring people together for both competition and fun. There can often be a desire to share the experience of playing a game with someone but meeting can be hard or impossible in-person and so having an online version of popular games allows for such opportunities. One such game is the game of Yahtzee, which while it does have online versions, are limited to the classic rule-set and do not allow for variants to be designed and played. *Dice Duels: Duel of the Eights* looks to solve the issue of not having online access to Yahtzee and the ability to create custom versions of the game and play them.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- Dice rolls can be given in the form AdX where A and X are variables representing the number of dice and number of sides, respectively. When A is 1, its inclusion is optional. For example 4d6 represents rolling 4 6-sided dice, and a d12 represent 1 die with 12 sides. These terms can also be used in contexts such as "we are playing with d8s".
- Since a fundamental mechanic of *Dice Duels: Duel of the Eights* is that of re-rolling dice, an additional variable will be added to the notation to denote the number of rolls, AdXrB. For example, typical Yahtzee with 3 rolls would be denoted as 5d6r3.
- Dice typically take the forms of platonic solids, meaning where the dice faces are congruent regular polygons. The five such polyhedra are:
  - A tetrahedron has four faces (d4)
  - A cube has six faces (d6)
  - An octahedron has eight faces (d8)
  - A dodecahedron has twelve faces (d12)
  - An icosahedron has twenty faces (d20)

Other die shapes can be used such as a pentagonal trapezohedron with ten faces (d10) or even dice meant to be rolled lengthwise such as a triangular prism which despite having five total faces, is used as having three (d3), or has caps to prevent rolling an unintended face.

- Dice rolls indicate a value integer when rolled. These can be represented by the numeric value of the integer or by representing the integer value as dots, called pips.
- In typical Yahtzee some patterns for scoring have terms. These will have to be added to and abstracted for a game with a different number of dice, but for a 5d6r3 these would be the scoring opportunities:
  - Rolling for aces, twos, threes, fours, fives, or sixes is rolling for as many of the number.
  - Chance is any combination of dice as a sum of all dice values.
  - A yahtzee is rolling all five of five dice with matching faces.
  - A three of a kind is having at least three of five dice match. For four of a kind, it is having at least four dice the same.
  - A straight is a set of sequential dice values. This can come in the small straight variant with four sequential values, or a large straight where all five dice are part of a sequence.
  - A full house is having two dice of a kind and three of another.

4.1.2 Physical System Description

[The purpose of this section is to clearly and unambiguously state the physical system that is to be modelled. Effective problem solving requires a logical and organized approach. The statements on the physical system to be studied should cover enough information to solve the problem. The physical description involves element identification, where elements are defined as independent and separable items of the physical system. Some example elements include acceleration due to gravity, the mass of an object, and the size and shape of an object. Each element should be identified and labelled, with their interesting properties specified clearly. The physical description can also include interactions of the elements, such as the following: i) the interactions between the elements and their physical environment; ii) the interactions between elements; and, iii) the initial or boundary conditions. —TPLT]

[The elements of the physical system do not have to correspond to an actual physical entity. They can be conceptual. This is particularly important when the documentation is for a numerical method. —TPLT]

The physical system of SFWRENG 4G06, as shown in Figure ?, includes the following elements:

PS1:

PS2: ...

[A figure here makes sense for most SRS documents —TPLT]

4.1.3 Goal Statements

These primary goals should be achieved in the development of our system, providing criteria for completeness. We have additionally organized stretch goals for further development, but they are not to be a metric for system completeness
Given the [inputs —TPLT], the goal statements are:

GS1: Enjoyable game. The project is more than just a capstone, and we need the game to be an enjoyable experience.

- User feedback collection. Implement a simple feedback system such as rating or comments for use in interviews or surveys to gather player insight.
- Testing iteration. Conduct at least two rounds of user testing to identify and iteratively improve the system based on user experience.
- Quality assets. Ensure that graphics, animations, and sounds are of a quality to add positively to the overall enjoyment.

Measurement: Based on user feedback, a minimum of 75% consider the experience as enjoyable.

GS2: Online multiplayer functionality. We need to be able to connect two concurrent players to play the game together.

- Connection setup. Develop a server-client connection system where two players can connect.
- Game state synchronization. Each player's actions are to be reflected to both user outputs using real-time synchronization.
- Disconnection handling. Design a method to handle disconnections.

Measurement: Two players can connect such that both players can affect the game state and both players are notified of the updates.

GS3: Customizable game settings. Core game elements must be modifiable to create custom Yahtzee variants. As a goal we would need these options to be implemented:

- Dice quantity option. Create an interface element for users to select from at least three options for number of dice.
- Dice type option. Create an interface element for users to select from at least three options for type of dice. These different types of dice will have different number of faces.

- Scoring method option. Offer at least two scoring systems that can be used.
- Timer feature. Implement a timer that can be turned on or off, providing a countdown for turn time.

Measurement: The above options are implemented and are compatible with each another.

GS4: Preset game settings. By having some preset game configurations, it would allow players to more quickly learn the game or jump into an environment that has been tested.

- Create presets. Develop at least three preset configurations and test them for gameplay balance.
- Preset selection menu. Create a simple way to select a preset.
- Preset names and description. Give each available preset a name and a brief description to help players understand them.

Measurement: At least three preset game configurations would be available for players to load up and play.

GS5: 3D dice rolling. Rolling the dice will need to be or look to be three dimensional to recreate the tactile feel of the original game.

- Dice 3D models. Develop 3D dice models, or the appearance of such.
- Dice interaction. Allow for user interaction with rolling dice through clicks or drags.

Measurement: Dice will have the appearance of the preset die shape, and of being rolled, based on a minimum of 75% of user feedback considering it so.

Additional stretch goal statements are the following, and can additionally be considered for when looking to add to the system's complexity and to better fulfill the intended goal of being an enjoyable game. Given the [inputs —TPLT], the stretch goals are:

SG6: Local multiplayer. This would allow for players to play together on a single computer, but would require a different user interface and allow for different user interactions.

- Player input methods. Develop a system to handle inputs from multiple players on the same device.
- User interface adjustments. Design a UI layout suitable for two players sharing a single screen.

Measurement: The ability for two players to play together using a single interface and game instance without an internet connection.

SG7: Singleplayer variants. A singleplayer game could be achieved either through a computer-run opponent in a game, or through a custom designed experience that could leverage the different environment.

- AI opponent development. Create an AI that can play against a human player.
- Difficulties. Design and allow players to adjust the AI opponent difficulty.
- Solo game modes. Design at least one unique solo challenge or mode that provides a self-contained experience.

Measurement: A single person can play at least one variant made specifically to be singleplayer without requiring a second human player to update the game state.

SG8: Online matchmaking. The game would provide users with the option to connect to another concurrent user based on a matchmaking score.

- Player rating system. Develop a rating system to categorize players by experience and skill.
- Matchmaking algorithm. Implement a system to match players.

Measurement: A player can connect to another unknown concurrent player who was selected as a compatible opponent.

SG9: Saving custom game setting. Having this ability would allow for a user who created a custom game variant to save them for the ability to replay it without the need to recreate those specific settings.

- Save and load system. Build a system for players to save custom setting to a file or local storage.
- Edit and delete options. Allow players to edit or delete saved custom settings for better management.

Measurement: A custom game variant, as per the "Customizable game settings" goal and "More game setting customization" stretch goals, can be saved locally and loaded up to be played.

SG10: More game setting customization. Besides the options in the goals section, some additional game customization options would include:

- Scorecard customization. Provide methods for players to adjust what options and hands appear on the scorecards.
- Scoring points options. Offer players the ability to modify the scores of scoring options.

- Additional scoring mechanisms. Include options for different methods of round or game scoring.
- Gambling mechanism. Add additional ways to act on probabilities such as wagering on specific rolls.
- Feedback-based features. Gather ideas through user feedback and testing to further expand available customization options.

Measurement: Additional game options outside the ones listed in the "Customizable game settings" goal would be available.

SG11: Dice customization. Dice could be made to appear differently, either as a means for personalization or for aiding with different impairments. An example could be a dice with pips versus a dice with a numbered faces.

- Dice colour options. Create at least three different dice appearances.
- Dice number representation. Create different ways to represent dice face values such as traditional pips, numbers, and symbols.
- Personalization menu. Implement a menu to allow for easy selection of dice appearance.

Measurement: At least five different dice appearance variants players can choose from, that would appear in the game.

SG12: Post game statistics. This could allow for players to analyze a game after completion in a more quantitative manner, aiding in better understanding statistical probabilities.

- Key statistic tracking. Track important game stats during play.
- Post-game summary screen. Present collected stats on a summary screen after a game.

Measurement: A post-game summary showing at least three key game stats, available after each game.

SG13: Multi-platform support. While most gaming experiences are for windows, this would allow for the game to be run on more than just the Windows operating system, allowing for a wider audience.

- Compile for systems. Compile the created system for other operating systems.
- Platform testing. Test the game on multiple operating systems.
- Cross-platform functionality. Verify the features such as online multiplayer work across different operating systems.

Measurement: The game can be run on operating systems other than Windows.

SG14: Dice highlighting. This would aid in determining what dice are used when scoring.

- Automatic dice highlighting. Implement a system to automatically highlight the dice that contribute to a player's score.
- Optional setting. Allow players to enable or disable the feature.

Measurement: Dice used in scoring will be highlighted when appropriate.

4.2 Solution Characteristics Specification

[This section specifies the information in the solution domain of the system to be developed. This section is intended to express what is required in such a way that analysts and stakeholders get a clear picture, and the latter will accept it. The purpose of this section is to reduce the problem into one expressed in mathematical terms. Mathematical expertise is used to extract the essentials from the underlying physical description of the problem, and to collect and substantiate all physical data pertinent to the problem. —TPLT]

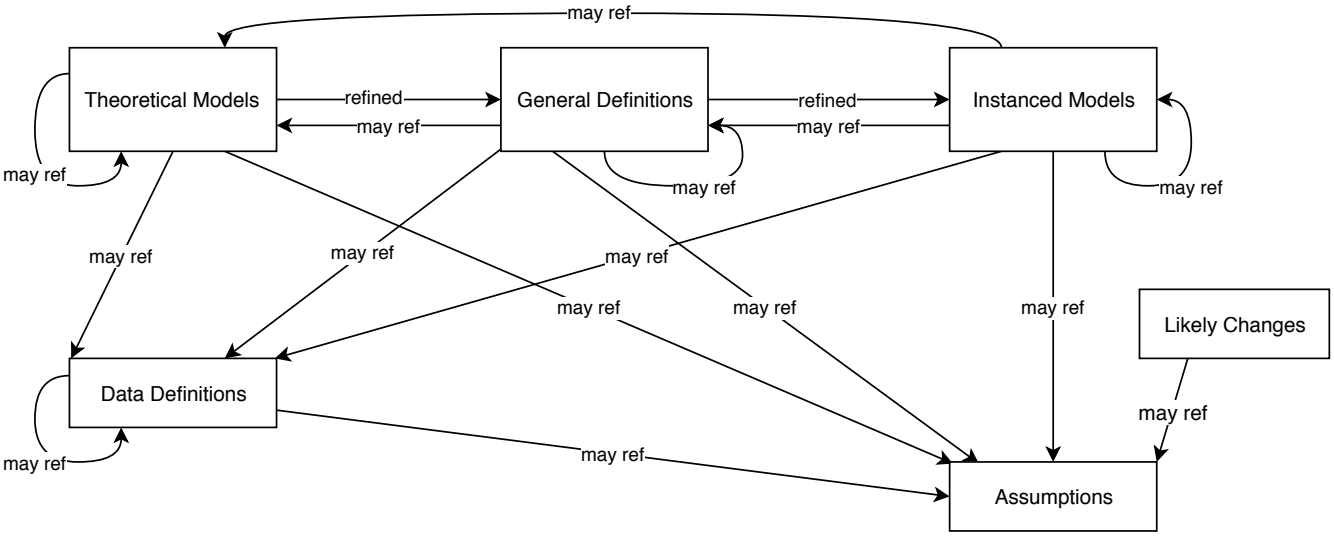
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]

[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]



The instance models that govern SFWRENG 4G06 are presented in Subsection 4.2.9. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Types

[This section is optional. Defining types can make the document easier to understand. —TPLT]

4.2.2 Scope Decisions

[This section is optional. —TPLT]

4.2.3 Modelling Decisions

[This section is optional. —TPLT]

4.2.4 Assumptions

[The assumptions are a refinement of the scope. The scope is general, where the assumptions are specific. All assumptions should be listed, even those that domain experts know so well that they are rarely (if ever) written down. —TPLT] [The document should not take for granted that the reader knows which assumptions have been made. In the case of unusual assumptions, it is recommended that the documentation either include, or point to, an explanation and justification for the assumption. —TPLT] [If it helps with the organization and understandability, the assumptions can be presented as sub sections. The following sub-sections are options: background theory assumptions, helper theory assumptions, generic theory assumptions, problem specific assumptions, and rationale assumptions —TPLT]

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to TM, GD, DD etc., using commands like dref, ddref etc. Each assumption should be atomic - that is, there should not be an explicit (or implicit) "and" in the text of an assumption. —TPLT]

4.2.5 Theoretical Models

[Theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section "Physical System Description" (Section 4.1.2). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. —TPLT]

[Optionally the theory section could be divided into subsections to provide more structure and improve understandability and reusability. Potential subsections include the following: Context theories, background theories, helper theories, generic theories, problem specific theories, final theories and rationale theories. —TPLT]

This section focuses on the general equations and laws that SFWRENG 4G06 is based on. [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

RefName: **TM:COE**

Label: Conservation of thermal energy

Equation: $-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$

Description: The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\mathrm{J\,kg^{-1}\,{}^{\circ}C^{-1}}$) and density ρ ($\mathrm{kg\,m^{-3}}$), where $\mathbf{q}$ is the thermal flux vector ($\mathrm{W\,m^{-2}}$), g is the volumetric heat generation ($\mathrm{W\,m^{-3}}$), T is the temperature (°C), t is time (s), and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.

Notes: None.

Source: http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm

Ref. By: GD**??**

Preconditions for TM:COE: None

Derivation for TM:COE: Not Applicable

["Ref. By" is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if TM1 is referenced by GD2, that means that GD2 will explicitly include a reference to TM1. —TPLT]

4.2.6 General Definitions

[General Definitions (GDs) are a refinement of one or more TMs, and/or of other GDs. The GDs are less abstract than the TMs. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the TM could be Newton's

Law of Cooling stated abstracting. The GD could take the general law and apply it to get a 1D equation. —TPLT]

This section collects the laws and equations that will be used in building the instance models.

[Some projects may not have any content for this section, but the section heading should be kept. —TPLT] [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	GD1
Label	**Newton's law of cooling**
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q(t) = h\Delta T(t)$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings. $q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$). h is the heat transfer coefficient, assumed independent of T (A**??**) ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$). $\Delta T(t) = T(t) - T_{\mathrm{env}}(t)$ is the time-dependent thermal gradient between the environment and the object (°C).
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

[This may be necessary when the necessary information does not fit in the description field. —TPLT] [Derivations are important for justifying a given GD. You want it to be clear where the equation came from. —TPLT]

4.2.7 Data Definitions

[The Data Definitions are definitions of symbols and equations that are given for the problem. They are not derived; they are simply used by other models. For instance, if a problem depends on density, there may be a data definition for the equation defining density. The DDs are given information that you can use in your other modules. —TPLT]

[All Data Definitions should be used (referenced) by at least one other model. —TPLT]

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,°C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.8 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of **?**. —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.9 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.7 to replace the abstract symbols in the models identified in Sections 4.2.5 and 4.2.6.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Energy balance on water to find T_W**
Input	m_W, C_W, h_C, A_C, h_P, A_P, t_{final}, T_C, T_{init}, $T_P(t)$ from IM**??** The input is constrained so that $T_{\text{init}} \leq T_C$ (A**??**)
Output	$T_W(t)$, $0 \leq t \leq t_{\text{final}}$, such that $\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$, $T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??**
Description	T_W is the water temperature (°C). T_P is the PCM temperature (°C). T_C is the coil temperature (°C). $\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s). $\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless). The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where 0^oC and 100^oC are the melting and boiling points of water, respectively (A**??**, A**??**).
Sources	Citation here
Ref. By	IM**??**

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.10 Input Data Constraints

Table 2 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 2 are listed in Table 3.

Table 2: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
L	$L > 0$	$L_{\text{min}} \leq L \leq L_{\text{max}}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 3: Specification Parameter Values

Var	Value
L_{min}	0.1 m

4.2.11 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —TPLT]. [These properties are in addition to the stated requirements. There is no need to repeat the requirements here. These additional properties may not exist for every problem. Examples include conservation laws (like conservation of energy or mass) and known constraints on outputs, which are usually summarized in tabular form. A sample table is shown in Table 4 —TPLT]

Table 4: Output Variables

Var	Physical Constraints
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A**??**)

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

5 Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

5.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing now well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, "The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by

SFWRENG 4G06 shall be described following the procedure given in Section X of the Verification and Validation Plan." A link to the VnV plan would be a nice extra. —TPLT]

NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]

NFR3: **Maintainability** [The effort required to make any of the likely changes listed for SFWRENG 4G06 should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]

NFR4: **Portability** [This NFR is easier to write than the others. The systems that SFWRENG 4G06 should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]

- Other NFRs that might be discussed include verifiability, understandability and reusability.

5.3 Rationale

[Provide a rationale for the decisions made in the documentation. Rationale should be provided for scope decisions, modelling decisions, assumptions and typical values. —TPLT]

6 Likely Changes

LC1: Add other non-dice based probabilities (cards,etc)

- If the scope of our game permits an addition, we may consider adding non-dice based probabilities into our games, such as playing cards, in order to provide more variety for the user.

LC2: the amount and variability of dice options may increase and decrease based on the scope of the project

- As the Scope of our project increase and decreases, we will consider experimenting with different amounts and types of dice. and should circumstances allow, we may choose to add or remove these different options for our game.

LC3: We may expand comparability to other operating systems depending of the scope of the project (Mac OS, etc).

- While we will initially design our system to work on Windows 10/11 devices, should the scope of our project expand, we are likely to adapt our system to work on other OS systems too such as Mac OS, or Linux, allowing our product to reach more players.

LC4: We may change how the scores are calculated.

- As we change and add various options for the players, such as changing the number and types of the dice, we may need to account for the different probabilities that these new options may create, which in turn would require us to modify our scoring system to account for these radically different probabilities.

LC5: We may expand the amount of players that can play the same game from just the initial 2 depending on scope.

- While initialized for two players, it is known that Yatzee can easily support more players, given that the fundamental mechanics of the game don't change when you add them. Thus, when we stabilize our two-player multiplayer, we could expand upon it to allow more then the initial two players to play the same game, allowing more users to enjoy our product.

7 Unlikely Changes

UC1: We will not remove the multiplayer component of the game.

- Yatzee is a social game of chance that involves chance and strategy in an appempt to get the highest score compared to other players, As such Multiplayer is a core component of the game and it must be included to ensure that an important aspect of the game isn't lost.

UC2: We will not remove the dice and it's probabilities

- Dice are a critical component of Yhatzee and it's variations, and the probabilities that the dices rolls provide are a core component in the way that score is calculated in game. Thus, our game will always involve the usage of dice and the calculation of the probabilities involving them.

UC3: We will not switch from our engine Godot for the duration of the project.

- Godot is the engine that most of the team is familiar. Thus, it would be too risky to switch game engines and learn something the team is unfamiliar with.

UC4: We will always have customization between game variants, and not just presets.

- One of the selling points for this project is to have customised settings for our game, allowing the user to tailor their experience to the way they want it. Thus, it is unlikely we will alter our plans to include this feature.

LC5: We will not change the 3D format of our game to 2D

- When playing a physical game like Yahtzee, one of the most engaging aspects is the action of rolling the dice. We wish to recreate the feel of playing the physical game as closely as possible by allowing our player to be able to visually see the dice roll in a way that mirrors the physical experience.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 5 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 6 shows the dependencies of instance models, requirements, and data constraints on each other. Table 7 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	TM??	TM??	TM??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??
TM??												
TM??			X									
TM??												
GD1												
GD??	X											
DD1				X								
DD??				X								
DD??												
DD??								X				
IM1					X	X	X				X	
IM??					X		X		X	X		
IM??		X										
IM??		X	X				X	X	X		X	

Table 5: Traceability Matrix Showing the Connections Between Items of Different Sections

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other.

	IM1	IM??	IM??	IM??	4.2.10	R??	R??
IM1		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 6: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
TM??	X																		
TM??																			
TM??																			
GD1		X																	
GD??			X	X	X	X													
DD1							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 7: Traceability Matrix Showing the Connections Between Assumptions and Other Items

Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

9 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented. In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

10 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

11 Commonalities

This section outlines the shared elements across the different games in the product family, including core mechanics, UI components, and gameplay goals.

11.1 Dice Rolls

A fundamental aspect of the game is its reliance on dice rolls. All variants will incorporate dice rolling as a primary mechanic, with corresponding UI elements to visually represent the action, ensuring consistency across game versions.

11.2 Yahtzee Hands

The scoring system in each game variant will be based on Yahtzee-style hands. While the specific scoring mechanics and available hands may vary slightly, the core objective of rolling a hand that aligns with predefined scoring categories remains consistent throughout all versions.

11.3 Score Sheet Structure

Similar to the Yahtzee Hands mechanic, the structure of the score sheet will be uniform across all game variants. This provides players with a familiar interface and ensures consistency in tracking scores.

11.4 Simultaneous Play

The game is designed for real-time play rather than a turn-based system. Players will always be engaged in the same stage of rolling, preventing any strategic advantage in head-to-head game variants.

11.5 2-Player or Single-Player Only

None of the game variants will support more than two human players. The focus will remain on single-player modes or head-to-head matchups with only two participants.

11.6 Always Playing Against an Opponent

Players will never play in isolation to achieve the highest possible score. Instead, each game will involve competing either against another player or against a predetermined computer score criterion, reinforcing the competitive nature of the gameplay.

12 Variabilities

This section details the variabilities between the different games in the product family, including changes in UI elements, game goals, and core mechanics.

12.1 Number of Dice

The number of dice used in the game can vary between different versions. Some variants may increase or decrease the number of dice to adjust the complexity and strategy of the game.

12.2 Sides on Dice

The number of sides on each dice can differ across game variants. This allows for customization of gameplay, where different variants may feature dice with more or fewer sides, influencing the probability and outcomes of each roll.

12.3 Individual Sides

The specific values or symbols on the sides of each dice can vary between versions. This variability provides an opportunity to introduce different themes or scoring dynamics depending on the game's rules.

12.4 Scoring Calculation

How points are calculated for different hands and the influence of dice rolls on the final score can be customized in different game variants. This allows for changes in how specific hands are valued and how scoring impacts the overall gameplay strategy.

12.5 Time Per Turn

The amount of time a player has to roll the dice and make their selection can be adjusted between variants. Time limits can vary, creating different levels of pressure and pacing in the game.

12.6 Hand Restrictions

Some game variants may restrict certain hands from being scored or may allow hands to be scored multiple times. These variations provide flexibility in game strategy and scoring dynamics.

13 Parameters of Variations

This section details the specific values that the variabilities between the different games in the product family can take, corresponding to the variabilities outlined in the previous section.

13.1 Number of Dice

The number of dice used in the game can range from 4 to 8. This allows for variations in gameplay complexity, depending on the specific variant.

13.2 Sides on Dice

The number of sides on each die can range from 4 to 9. This flexibility in dice sides introduces variability in the probability and potential outcomes for each roll.

13.3 Individual Sides

The values displayed on the individual sides of each die can range from 1 to 9. This variability allows for different numerical or symbolic configurations to match specific game variants.

13.4 Scoring Calculation

Each hand can be assigned any integer score value or modified by factors based on the dice rolls. This provides flexibility in defining how each hand is valued in the different game variants.

13.5 Time Per Turn

The time allotted for each turn can range from 5 seconds to 2 minutes, or be set to unlimited. This parameter affects the pacing of the game, allowing for both fast-paced and more thoughtful gameplay.

13.6 Hand Restrictions

Hands can be restricted by removing up to all but one from play. Additionally, hands can be allowed to be repeated between 1 to 10 times or an unlimited number of times, giving options for different scoring strategies.

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]
[Grammar, flow and LaTeXadvice:

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX and formatting rules
  - Variables are italic, everything else not, includes subscripts (link to document)
    - Conventions
    - Watch out for implied multiplication
  - Use BibTeX
  - Use cross-referencing
- Grammar and writing rules
  - Acronyms expanded on first usage (not just in table of acronyms)
  - "In order to" should be "to"

—TPLT]
[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

1. What went well while writing this deliverable?

   - ...

2. What pain points did you experience during this deliverable, and how did you resolve them?

   - ...

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

   - ...

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

   - ...

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

   - ...

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   - ...