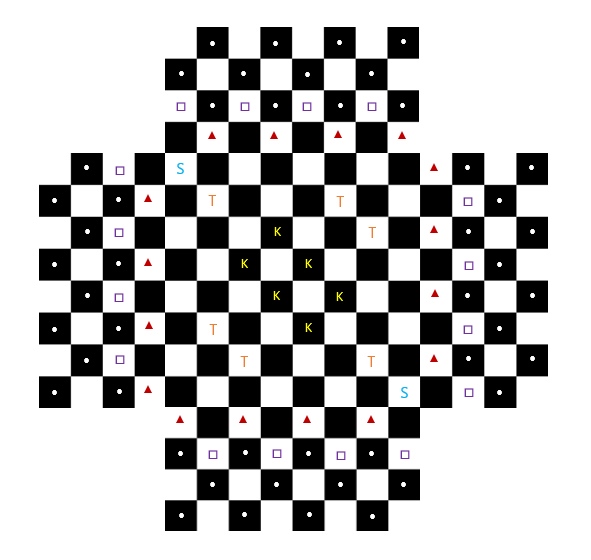
Checker Wars

Project Description Report

**Group 16: Alex Choi, Andrew Macatangay, Sam Alammar, Luke Austin**



**CS 440**

**University of Illinois Chicago**

**October 2020**

**Table of Contents**

[List of Figures 6](#_Toc58080756)

[List of Tables 7](#_Toc58080757)

[I Project Description 8](#_Toc58080758)

[1 Project Overview 8](#_Toc58080759)

[2 The Purpose of the Project 8](#_Toc58080760)

[2a The User Business or Background of the Project Effort 8](#_Toc58080761)

[2b Goals of the Project 8](#_Toc58080762)

[2c Measurement 8](#_Toc58080763)

[3 The Scope of the Work 8](#_Toc58080764)

[3a The Current Situation 8](#_Toc58080765)

[3b The Context of the Work 9](#_Toc58080766)

[3c Work Partitioning 10](#_Toc58080767)

[3d Competing Products 10](#_Toc58080768)

[4 The Scope of the Product 11](#_Toc58080769)

[4a Scenario Diagram(s) 11](#_Toc58080770)

[4b Product Scenario List 12](#_Toc58080771)

[4c Individual Product Scenarios 12](#_Toc58080772)

[5 Stakeholders 13](#_Toc58080773)

[5a The Client 13](#_Toc58080774)

[5b The Customer 13](#_Toc58080775)

[5c Hands-On Users of the Product 13](#_Toc58080776)

[5d Other Stakeholders 14](#_Toc58080777)

[5e User Participation 14](#_Toc58080778)

[6 Mandated Constraints 14](#_Toc58080779)

[6a Solution Constraints 14](#_Toc58080780)

[6b Implementation Environment of the Current System 14](#_Toc58080781)

[6c Partner or Collaborative Applications 15](#_Toc58080782)

[6d Off-the-Shelf Software 15](#_Toc58080783)

[6e Anticipated Workplace Environment 15](#_Toc58080784)

[6f Schedule Constraints 15](#_Toc58080785)

[6g Budget Constraints 15](#_Toc58080786)

[7 Naming Conventions and Definitions 15](#_Toc58080787)

[7a Definitions of Key Terms 15](#_Toc58080788)

[7b UML and Other Notation Used in This Document 16](#_Toc58080789)

[7c Data Dictionary for Any Included Models 16](#_Toc58080790)

[8 Relevant Facts and Assumptions 16](#_Toc58080791)

[8a Facts 16](#_Toc58080792)

[8b Assumptions 16](#_Toc58080793)

[II Requirements 16](#_Toc58080794)

[9 Product Use Cases 16](#_Toc58080795)

[9a Use Case Diagrams 16](#_Toc58080796)

[10 Functional Requirements 17](#_Toc58080799)

[11 Data Requirements 18](#_Toc58080800)

[12 Performance Requirements 19](#_Toc58080801)

[12a Speed and Latency Requirements 19](#_Toc58080802)

[12b Precision or Accuracy Requirements 19](#_Toc58080803)

[12c Capacity Requirements 19](#_Toc58080804)

[13 Dependability Requirements 20](#_Toc58080805)

[13a Reliability and Availability Requirements 20](#_Toc58080806)

[13b Robustness or Fault-Tolerance Requirements 21](#_Toc58080807)

[14 Maintainability and Supportability Requirements 21](#_Toc58080809)

[14a Maintenance Requirements 21](#_Toc58080810)

[14b Supportability Requirements 21](#_Toc58080811)

[14c Adaptability Requirements 22](#_Toc58080812)

[14d Scalability or Extensibility Requirements 22](#_Toc58080813)

[14e Longevity Requirements 22](#_Toc58080814)

[15 Security Requirements 22](#_Toc58080815)

[15a Access Requirements 23](#_Toc58080816)

[15b Integrity Requirements 23](#_Toc58080817)

[15c Privacy Requirements 23](#_Toc58080818)

[16 Usability and Humanity Requirements 23](#_Toc58080819)

[16a Ease of Use Requirements 23](#_Toc58080820)

[16b Personalization and Internationalization Requirements 25](#_Toc58080821)

[16c Learning Requirements 26](#_Toc58080822)

[16d Understandability and Politeness Requirements 27](#_Toc58080823)

[16e Accessibility Requirements 28](#_Toc58080824)

[16f Training Requirements 29](#_Toc58080825)

[17 Look and Feel Requirements 30](#_Toc58080826)

[17a Appearance Requirements 30](#_Toc58080827)

[17b Style Requirements 30](#_Toc58080828)

[18 Operational and Environmental Requirements 31](#_Toc58080829)

[18a Expected Physical Environment 31](#_Toc58080830)

[18b Requirements for Interfacing with Adjacent Systems 32](#_Toc58080831)

[18c Productization Requirements 32](#_Toc58080832)

[18d Release Requirements 33](#_Toc58080833)

[19 Cultural and Political Requirements 34](#_Toc58080834)

[19a Cultural Requirements 34](#_Toc58080835)

[19b Political Requirements 35](#_Toc58080836)

[20 Legal Requirements 36](#_Toc58080837)

[20a Compliance Requirements 36](#_Toc58080838)

[20b Standards Requirements 36](#_Toc58080839)

[21 Requirements Acceptance Tests 37](#_Toc58080840)

[21a Requirements – Test Correspondence Summary 37](#_Toc58080841)

[21b Acceptance Test Descriptions 37](#_Toc58080842)

[III Design 38](#_Toc58080843)

[22 Design Goals 38](#_Toc58080844)

[23 Current System Design 38](#_Toc58080845)

[24 Proposed System Design 38](#_Toc58080846)

[24a Initial System Analysis and Class Identification 38](#_Toc58080847)

[24b Dynamic Modelling of Use-Cases 39](#_Toc58080848)

[24c Proposed System Architecture 40](#_Toc58080849)

[24d Initial Subsystem Decomposition 40](#_Toc58080850)

[25 Additional Design Considerations 41](#_Toc58080851)

[25a Hardware / Software Mapping 41](#_Toc58080852)

[25b Persistent Data Management 41](#_Toc58080853)

[25c Access Control and Security 41](#_Toc58080854)

[25d User Interface 41](#_Toc58080855)

[26 Final System Design 42](#_Toc58080856)

[27 Object Design 43](#_Toc58080857)

[IV Project Issues 46](#_Toc58080858)

[28 Open Issues 46](#_Toc58080859)

[29 Off-the-Shelf Solutions 47](#_Toc58080860)

[29a Ready-Made Products 47](#_Toc58080861)

[29b Reusable Components 48](#_Toc58080862)

[29c Products That Can Be Copied 48](#_Toc58080863)

[30 New Problems 48](#_Toc58080864)

[30a Effects on the Current Environment 48](#_Toc58080865)

[30b Effects on the Installed Systems 49](#_Toc58080866)

[30c Potential User Problems 50](#_Toc58080867)

[30d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product 50](#_Toc58080868)

[30e Follow-Up Problems 51](#_Toc58080869)

[31 Migration to the New Product 51](#_Toc58080870)

[32 Risks 51](#_Toc58080871)

[33 Costs 53](#_Toc58080872)

[34 Waiting Room 54](#_Toc58080873)

[35 Ideas for Solutions 55](#_Toc58080874)

[36 Project Retrospective 56](#_Toc58080875)

[V Glossary 57](#_Toc58080876)

[VI References / Bibliography 58](#_Toc58080877)

[VII Index 58](#_Toc58080878)

### ****List of Figures****

[Figure 1: Game Environments Venn Diagram 9](#_Toc58080449)

[Figure 2: Match Use Case Diagram 17](#_Toc58080450)

[Figure 3: Initial Class Diagram 38](#_Toc58080451)

[Figure 4: Move Piece Sequence Diagram 39](#_Toc58080452)

[Figure 5: Victory Scenario Sequence Diagram 39](#_Toc58080453)

[Figure 6: Client-Server Model for CheckerWars.com 40](#_Toc58080454)

[Figure 7: Checker Wars GUI 42](#_Toc58080455)

[Figure 8: Final System Design (UML) 43](#_Toc58080456)

[Figure 9: CheckerWarsGame and Gameboard classes 43](#_Toc58080457)

[Figure 10: Scenario and Player/AI Classes 44](#_Toc58080458)

[Figure 11: Tile and Piece Classes 45](#_Toc58080459)

*On a related note, the references in the paragraph above, “( see* ***Error! Reference source not found.*** *on page* ***Error! Bookmark not defined.*** *)” include cross-references to the Figure and page number that will adjust automatically when other Figures or pages are added or removed. This is done with the “cross-reference” button in the “Captions” section of the “References” tab in MS Word.* ***Note: Remove this instructional paragraph.***

### ****List of Tables****

*If a document contains a large number of tables, then it is appropriate to include a list of tables at the beginning of the document, following the table of contents. Each table should include a title, and be numbered in a consistent logical fashion. The following list of tables was automatically generated from table captions ( see below), and can be automatically updated by right-clicking on the table below and selecting “Update Field”. This feature is located in the “Captions” section of the “References” tab in MS Word.* ***Note: Remove this instructional paragraph.***

[Table 1: Development Schedule 10](#_Toc58080460)

[Table 2: Requirements - Acceptance Tests Correspondence 37](#_Toc58080461)

# Project Description

## Project Overview

Checker Wars is a new, virtual take on the game of Checkers. It is being developed by Bohn Jell Entertainment to entertain game players and provide a new game for online competitive play. The game is a multidimensional version of Checkers with extra pieces and challenging elements that provide a more challenging experience with greater replayability.

## The Purpose of the Project

### The User Business or Background of the Project Effort

The video game industry is enormous. It has soared past the television and film business to become the biggest and fastest growing category of the entertainment business.

Checker Wars hopes to use this continual growth to establish a community of game players who play for fun and compete against each other in Checker Wars.

### Goals of the Project

We wish to provide a new source of entertainment and competition to the growing field of online game playing and, specifically e-sports.

### Measurement

We will measure the success of Checker Wars by the popularity of the game, both in number of downloads, and hopefully, streamers playing the game against each other via services such as twitch.tv.

## The Scope of the Work

### The Current Situation

Checker Wars is the first game developed by Bohn Jell Entertainment. The game will be the first product and therefore only source of revenue for the company.

### The Context of the Work

*SV: Define the boundary between what is included in “the work” and what is not. It also defines what external entities “the work” must interact with and what those interactions entail.* ***The following example diagram should be replaced with one appropriate to this project.***

*Content*

*The work context diagram identifies the work that you need to investigate to be able to build the product. Note that it includes more than the intended product. Unless we understand the work that the product will support, we have little chance of building a product that will fit cleanly into its environment.*

*The adjacent systems on the context diagram (e.g., Weather Forecasting Service) indicate other subject matter domains (systems, people, and organizations) that need to be understood. The interfaces between the adjacent systems and the work context indicate why we are interested in the adjacent system. In the case of Weather Forecasting Service, we can say that we are interested in the details of when, how, where, who, what, and why it produces the District Weather Forecasts information.*

*Motivation*

*To clearly define the boundaries for the study of the work and requirements effort. Without this definition, we have little chance of building a product that will fit seamlessly into its environment.*

*Examples*

*Considerations*

*The names used on the context diagram should be consistent with the naming conventions and data dictionary definitions presented in section 5. Without these definitions, the context model lacks the required rigor, and it may be misunderstood. Relevant stakeholders must agree to the definitions of the interfaces shown on the context model.*

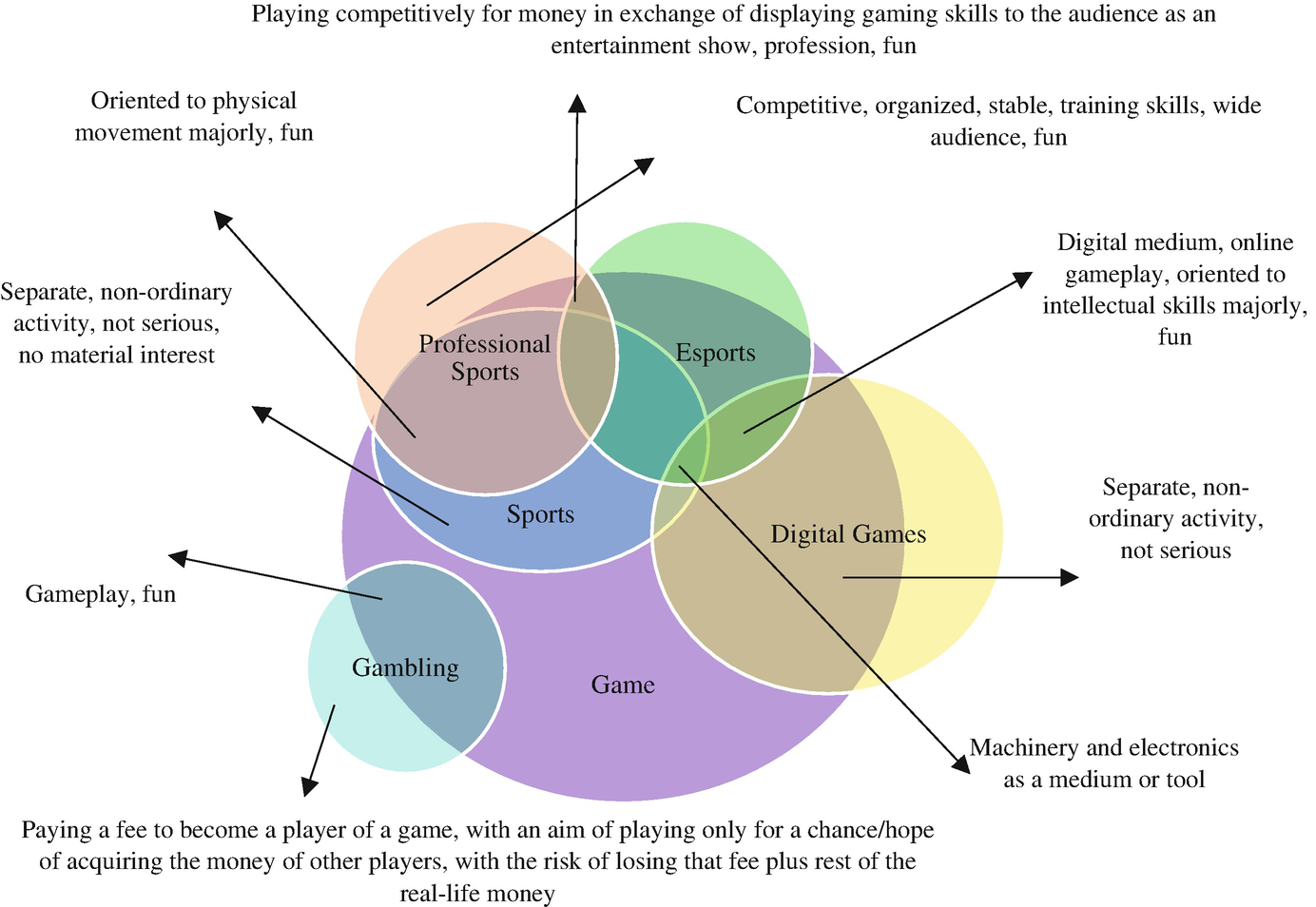


Figure : Game Environments Venn Diagram

The competitive gaming market is made up of many different disciplines. Originally physical, gameplay has moved into the virtual world, and e-sports are a large part of what encourages new players and keeps fueling the gaming industry. Without understanding where Checker Wars fits among these different sports and competitive markets, Checker Wars would be doomed to obscurity.

### Work Partitioning

*SV: “The work” is often large and complex, with many different activities and concerns. One good way to break this down and organize it for analysis is to identify the different events to which the business must respond. A “business event” is an external stimulus which causes the business to take a series of actions in response.*

*Content*

*A list showing all business events to which the work responds. Business events are happenings in the real world that affect the work. They also happen because it is time for the work to do something—for example, produce weekly reports, remind nonpaying customers, check the status of a device, and so on. The response to each event is called a business use case; it represents a discrete partition of work that contributes to the total functionality of the work.*

*The event list includes the following elements:*

*● Event name*

*● Input from adjacent systems (identical with name on context diagram)*

*● Output to adjacent systems (identical with name on context diagram)*

*● Brief summary of the business use case (This is optional, but we have found it is a very useful first step in defining the requirements for the business use case—you can think of it as a mini-scenario.)*

*Motivation*

*To identify logical chunks of the system that can be used as the basis for discovering detailed requirements. These business events also provide the subsystems that can be used as the basis for managing detailed analysis and design.*

*Example*

*Business Event List*

*Event Name Input and Output Summary*

|  |  |  |
| --- | --- | --- |
| *1. Weather Station transmits reading* | *Weather Station Readings (in)* | *Record the readings as belonging to the weather station.* |
| *2. Weather Service forecasts weather* | *District Weather Forecast (in)* | *Record the forecast.* |
| *3. Road engineers advise changed roads* | *Changed Road (in)* | *Record the new or changed road. Check that all appropriate weather stations are attached.* |
| *4. Road Engineering installs new Weather Station* | *New Weather Station (in)* | *Record the weather station and attach it to the appropriate roads.* |
| *5. Road Engineering changes Weather Station* | *Changed Weather Station (in)* | *Record the changes to the weather station.* |
| *6. Time to test Weather Stations* | *Failed Weather Station Alert (out)* | *Determine if any weather stations have not transmitted for two hours, and inform Road Engineering of any failures.* |
| *7. Truck Depot changes a truck* | *Truck Change (in)* | *Record the changes to the truck.* |
| *8. Time to detect icy roads* | *Road De-icing Schedule (out)* | *Predict the ice situation for the next two hours. Assign a truck to any roads that will freeze. Issue the schedule.* |
| *9. Truck treats a road* | *Treated Road (in)* | *Record the road as being in a safe condition for the next three hours.* |
| *10 Truck Depot reports problem with truck* | *Truck Breakdown (in)*  *Amended Gritting Schedule (out)* | *Reassign available trucks to the previously assigned roads.* |
| *11. Time to monitor road treatment* | *Untreated Road Reminder (out)* | *Check that all scheduled roads have been treated in the assigned time, and issue reminders for any untreated roads.* |
|  |  |  |

*Considerations*

*Attempting to list the business events is a way of testing the work context. This activity uncovers uncertainties and misunderstandings about the project and facilitates precise communications. When you do an event analysis, it will usually prompt you to make some changes to your work context diagram.*

*We suggest you gather requirements for discrete sections of the work. This requires you to partition the work, and we have found business events to be the most convenient, consistent, and natural way to break the work into manageable units.*

Table : Development Schedule

|  |  |  |
| --- | --- | --- |
| **Event Name** | **Input and Output** | **Summary** |
| 1. Game start | User inputs credentials (in) | User logs in with username and password |
| 2. Main Menu | Menu display (out) | Main menu is displayed where user can choose single-player or multiplayer gameplay |
| 3. Single-player game mode | Gameplay board is displayed (out) | User plays against AI opponent to practice in-game skills |
| 4. Multiplayer game mode | Gameplay board is displayed (out) | User connects to server and plays against human opponent |
| 5. Game end | Score displayed (out) | User see’s gameplay results and statistics. Score and rank increase or decrease depending on game result |

### Competing Products

*SV:* ***IF*** *there are other existing products that the client could use instead of the proposed product, then they should be discussed here, along with the reasons why the proposed product is still needed / beneficial.*

*Content*

*Other alternatives that already exist can be described here. Why should we go to all the trouble of creating a new product? What flaws or deficiencies do the existing products have that justify the creation of something new?*

*Motivation*

*Knowing what other choices the customer has to choose from can help us judge whether or not our project is even worth doing, and if so, what we need to do different to be better than the available alternatives.*

*Considerations*

*Note the subtle difference between this item and the “Off the Shelf” solutions documented in sections 0 or 29 below. The latter refers to software that we can buy and incorporate into our solution.*

Competitive single-player (with AI opponents) and multiplayer (both offline and online) games are all competitors of Checker Wars. Online Chess, Go, and similar strategy games are all competitors of Checker Wars. Some of these are standalone games, some of them are web apps, some are sold through services such as Steam or the Epic game store. These products’ successes and failures must be studied and emulated to ensure Checker Wars is a success.

## The Scope of the Product

*SV: This section describes the proposed product as a set of short stories ( scenarios ) providing examples of how the product would be used in practice. This effectively documents what is and what is not included in the product, and who/what would interact with it in what ways. The opening paragraph briefly states what subset of “the work” is to be handled by the proposed product.*

### Scenario Diagram(s)

*SV: The scenario diagram acts as an illustrated list of the scenarios to be presented, showing the boundary of the system and what external “actors” are involved in each of the scenarios.*

*Scenarios are somewhat informal stories describing how the end users would use the product once it is completed. They take the form of narratives and may involve specific individuals and examples. A scenario diagram is a graphical illustration of a number of scenarios, showing their relationships to each other and to external entities. It is similar in nature to use-case diagrams, except for scenarios. ( Note that while many scenarios will eventually evolve into full-blown use-cases, some will not, and additional use-cases will be created by the time the project develops to that stage, so there will not in general be a one-to-one correspondence between the scenario diagram(s) shown here and the use-case diagrams to be developed later. )*

*See Section 9a for a discussion of and examples of use-case diagrams.*

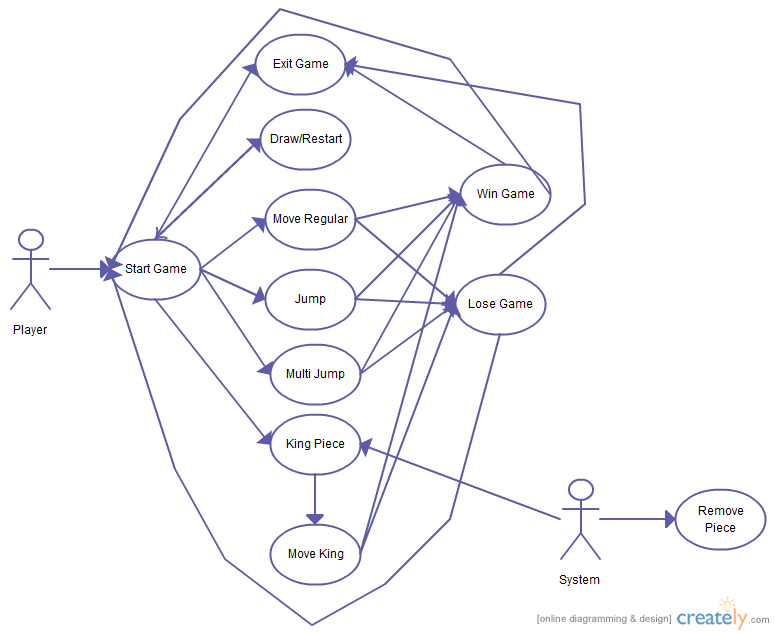


Figure 2: Checker Wars Scenario Diagram

### Product Scenario List

*SV: A table listing the scenarios by name, external actors involved, and possibly other information if relevant.*

*The product scenario list is quite simply a list of the product scenarios that will appear in the next section. It is a good idea to either number or name each scenario for later reference, and it can also be a good idea to organize the list so that related scenarios appear together. ( Depending on the naming / numbering scheme, they can be grouped into sections and subsections, etc. )*

Game Mode Select Scenario

Move Scenario

Move Pawn Scenario

Move Guardsman Scenario

Move Noble Scenario

Upgrade to King Scenario

Move King Scenario

Summon Space Scenario

Tower Space Scenario

Victory Scenario

### Individual Product Scenarios

*SV: This section contains the actual scenarios, the stories of the product being used.*

*Product scenarios are written in a natural narrative fashion, easily understood by clients and other non-technical stakeholders. Each one tells a story of how the end users are expected to eventually use the finished product. For example:*

***Monthly Reports:*** *At the end of every month Mary has to generate the monthly reports, and distribute copies to all the managers and sub-managers. The first thing she has to do is to make sure that all the end-of-the-month tests have been run, and that everyone else is logged off of the system. Then she selects the date range and the specific information she wants included in her reports, selects either the long or short format, and selects a printer. Depending on how busy the month has been, it may take as long as fifteen minutes, during which time no one else can use the system. She only prints one copy on the computer, and then makes all the rest of the copies she needs on the copy machine.*

**Game Mode Select Scenario**: On the menu select screen, the user may select to play single player mode against an AI opponent or multiplayer mode against a human opponent via the internet.

**Move Scenario**: Each player’s turn is determined by the roll of two die. The player with the highest roll goes first. This player gets to decide whether the order of turns goes clockwise or counterclockwise from them. Each player receives 5 turns per round. Each round ends when every player has exhausted their turns. Though, similar to checkers, each attack takes up one turn, and if an attack is chained, each chained attack will not take up another turn.

**Move Pawn Scenario**: The user opts to move a pawn piece on the game board. Pawns may only move to black spaces on the board.

**Move Guardsman Scenario**: The user opts to move a guardsman piece on the game board. Guardsman pieces may only move on white spaces on the board.

**Move Noble Scenario**: The user opts to move a noble piece on the game board. noble pieces may only move on white spaces on the board and cannot attack other pieces.

**Move King Scenario**: The user opts to move a King piece on the game board. The king pieces can move through either white or black spaces on the board. When a king transitions from one space color to another, that uses a turn and that king may not be able to move until the next round. When a king transitions from one color space, white or black, to another, it will do so on a space adjacent to it.

**King Space Scenario**: A pawn that lands on a King space is upgraded to a King piece for that player.

**Summon Space Scenario**: If a king and a noble lands on a summon space and is able to summon a new pawn piece on an adjacent black square. A summon space used by a noble or a king can only be used once per turn.

**Tower Space Scenario**: If a king or a noble lands on a tower space, they occupy that tower. If three towers are occupied by a single player, that player is victorious.

**Attack Scenario**: The user moves to attack an opponent’s piece. Guardsman can attack all other pieces on white spaces. Nobles cannot attack other pieces. King’s can attack any other piece. All pieces attack in a manner similar to the game of checkers, in that they “jump” over the piece to “take” them, and land on the other side of the opponents piece. A piece can only successfully attack another piece if its movement is unobstructed (can successfully land on the other side of the enemy piece.

**Victory Scenario**: A win is determined when either three towers are captured by one army or if the other players have no more pieces. When one player is completely defeated, they are out for the rest of the game and the game will continue with the other players. If for some reason, no moves exist for all players, the game ends in a draw.

## Stakeholders

*SV: Stakeholders include all persons or entities that have an interest in the proposed product or its development, either directly or indirectly.*

### The Client

*SV: The client pays up front for the product to be developed, and provides guidance or other input for its development. Some projects do not have an external client, in which case the developing organization acts as the client.*

*Content*

*This item gives the name of the client. It is permissible to have several names, but having more than three negates the point.*

*Motivation*

*The client has the final say on acceptance of the product, and thus must be satisfied with the product as delivered. You can think of the client as the person who makes the investment in the product. Where the product is being developed for in-house consumption, the roles of the client and the customer are often filled by the same person. If you cannot find a name for your client, then perhaps you should not be building the product.*

*Considerations*

*Sometimes, when building a package or a product for external users, the client is the marketing department. In this case, a person from the marketing department must be named as the client.*

This product is being designed, produced, and commissioned by Bohn Jell Entertainment. Bohn Jell Entertainment has designed the specifications for the game but is outsourcing development and graphics.

### The Customer

*SV: The customer is the person or entity who will buy the product after it has been completed. Some projects do not have an external customer, if they are to be used in-house or for the client’s use only.*

*Content*

*The person intended to buy the product. In the case of in-house development, the client and the customer are often the same person. In the case of development of a mass-market product, this section contains a description of the kind of person who is likely to buy the product.*

*Motivation*

*The customer is ultimately responsible for deciding whether to buy the product from the client. The correct requirements can be gathered only if you understand the customer and his aspirations when it comes to using your product.*

Checker Wars is aimed at teens and young adult players, however the game appeals to players of all ages due to its similarities with the game of checkers. The main focus will be on players who use streaming services such as twitch.tv.

### Hands-On Users of the Product

*SV: These are the people who will actually use the product in practice, and who may be separate from the customer or client. For example, educational software may be purchased by the school system ( customer ) and used by students ( hands-on users. )*

*Content*

*A list of a special type of stakeholder—the potential users of the product. For each category of user, provide the following information:*

*● User name/category: Most likely the name of a user group, such as schoolchildren, road engineers, or project managers.*

*● User role: Summarizes the users’ responsibilities.*

*● Subject matter experience: Summarizes the users’ knowledge of the business. Rate as novice, journeyman, or master.*

*● Technological experience: Describes the users’ experience with relevant technology. Rate as novice, journeyman, or master.*

*● Other user characteristics: Describe any characteristics of the users that have an effect on the requirements and eventual design of the product. For example:*

*Physical abilities/disabilities*

*Intellectual abilities/disabilities*

*Attitude toward job*

*Attitude toward technology*

*Education*

*Linguistic skills*

*Age group*

*Gender*

*Motivation*

*Users are human beings who interface with the product in some way. Use the characteristics of the users to define the usability requirements for the product. Users are also known as actors.*

*Examples*

*Users can come from wide variety of (sometimes unexpected) sources. Consider the possibility of your users being clerical staff, shop workers, managers, highly trained operators, the general public, casual users, passers-by, illiterate people, tradesmen, students, test engineers, foreigners, children, lawyers, remote users, people using the system over the telephone or an Internet connection, emergency workers, and so on.*

**User**: Game Players

**a.** **Role**: both serious, competitive gamers and casual gamers will enjoy playing Checker Wars. Games usually take 5 to 10 minutes to start and finish, allowing students or workers who want a quick mental break from their study or work to fit in a quick game.

**b.** **Subject Matter Experience**: All players will begin as novice, and if they continue to play and learn the ins and outs and meta of the game, over time they will progress to journeyman and then master.

**c.** **Technological Experience**: Users technological experience is only required at a novice level. The game is very easy to install, and the gameplay is very intuitive and easy to pick up. The game is easy to start playing, and hard to master.

**d.** **Other User Characteristics**:

**i.** **Age**: All ages

**ii. Gender**: All Genders

**iii.** **Motivation**: Entertainment or competition.

*SV: Describe users that will install, maintain, update, and otherwise service the product as needed. May not apply to all projects.*

*Content*

*Maintenance users are a special type of hands-on users who have requirements that are specific to maintaining and changing the product.*

*Motivation*

*Many of these requirements will be discovered by considering the various types of maintenance requirements detailed in section 14. However, if we define the characteristics of the people who maintain the product, it will help to trigger requirements that might otherwise be missed.*

Your text goes here . . .

### Other Stakeholders

*SV: This section is a catch-all for all other stakeholders not previously mentioned. Note that some stakeholders may be negatively impacted by the proposed project, for example if their work duties change or are eliminated.*

*Content*

*The roles and (if possible) names of other people and organizations who are affected by the product, or whose input is needed to build the product.*

*Examples of stakeholders:*

*● Sponsor*

*● Testers*

*● Business analysts*

*● Technology experts*

*● System designers*

*● Marketing experts*

*● Legal experts*

*● Domain experts*

*● Usability experts*

*● Representatives of external associations*

*For a complete checklist, download the stakeholder analysis template at* [*www.volere.co.uk*](http://www.volere.co.uk)*.*

*For each type of stakeholder, provide the following information:*

*● Stakeholder identification (some combination of role/job title, person name, and organization name)*

*● Knowledge needed by the project*

*● The degree of involvement necessary for that stakeholder/knowledge combination*

*● The degree of influence for that stakeholder/knowledge combination*

*● Agreement on how to address conflicts between stakeholders who have an interest in the same knowledge*

*Motivation*

*Failure to recognize stakeholders results in missing requirements.*

Because the game is focused around the competitive gaming and the e-sports scene, other potential stakeholders will be computer gear companies such as Razer or Logitech for sponsorship of the game or competitive players of the game.

### User Participation

*SV: To what extent can we expect users to participate during the development of the product?*

*Content*

*Where appropriate, attach to the category of user a statement of the participation that you think will be necessary for those users to provide the requirements. Describe the contribution that you expect these users to provide—for example, business knowledge, interface prototyping, or usability requirements. If possible, assess the minimum amount of time that these users must spend for you to be able to determine the complete requirements.*

*Motivation*

*Many projects fail through lack of user participation, sometimes because the required degree of participation was not made clear. When people have to make a choice between getting their everyday work done and working on a new project, the everyday work usually takes priority. This requirement makes it clear, from the outset, that specified user resources must be allocated to the project.*

There will be an open beta that players can opt in to join (for free) in order to help the developers test the gameplay mechanics.

## Mandated Constraints

*SV: Mandated constraints are requirements that are set in stone by the client before the project is really even started, and before the full set of requirements are determined. Note that not all of these sections will apply to every project, and that some constraints could be placed equally well in more than one section ( but should not be duplicated. ).*

*This section describes constraints on the eventual design of the product. They are the same as other requirements except that constraints are mandated, usually at the beginning of the project. Constraints have a description, rationale, and fit criterion, and generally are written in the same format as functional and nonfunctional requirements.*

### Solution Constraints

*SV: These are general constraints on the product to be developed or the manner in which it is to be developed that are not covered elsewhere.*

*Content*

*This specifies constraints on the way that the problem must be solved. Describe the mandated technology or solution. Include any appropriate version numbers. You should also explain the reason for using the technology.*

*Motivation*

*To identify constraints that guide the final product. Your client, customer, or user may have design preferences, or only certain solutions may be acceptable. If these constraints are not met, your solution is not acceptable.*

*Examples*

*Constraints are written using the same form as other atomic requirements (refer to the requirements shell for the attributes). It is important for each constraint to have a rationale and a fit criterion, as they help to expose false constraints (solutions masquerading as constraints). Also, you will usually find that a constraint affects the entire product rather than one or more product use cases.*

*Description: The product shall use the current two-way radio system to communicate with the drivers in their trucks.*

*Rationale: The client will not pay for a new radio system, nor are any other means of communication available to the drivers.*

*Fit criterion: All signals generated by the product shall be audible and understandable by all drivers via their two-way radio system.*

*Description: The product shall operate using Windows XP.*

*Rationale: The client uses XP and does not wish to change.*

*Fit criterion: The product shall be approved as XP compliant by the MS testing group.*

*Description: The product shall be a hand-held device.*

*Rationale: The product is to be marketed to hikers and mountain climbers.*

*Fit criterion: The product shall weigh no more than 300 grams, no dimension shall be more than 15 centimeters, and there shall be no external power source.*

*Considerations*

*We want to define the boundaries within which we can solve the problem. Be careful, because anyone who has experience with or exposure to a piece of technology tends to see requirements in terms of that technology. This tendency leads people to impose solution constraints for the wrong reason, making it very easy for false constraints to creep into a specification. The solution constraints should only be those that are absolutely non-negotiable. In other words, however you solve this problem, you must use this particular technology. Any other solution would be unacceptable.*

**Description**: The game will be on a computing device such as a PC, Mac, or iPad.

**Rationale**: The game will need Internet connection to play against opponents via the internet.

**Fit Criterion:** The product will be connected to Internet using the user’s connection to the Bohn Jell Entertainment’s servers.

**Description**: The game will most likely be developed in an object-oriented language.

**Rationale:** The game will be played on PC, Mac, and Android operating systems, and will have objects that need to be manipulated.

**Fit Criterion:** All these platforms support development in object-oriented programming languages and coding style.

**Description**: The game will be played by players all around the world.

**Rationale:** The server of the game will be located in United States, but users from other countries will be able to use these game servers.

**Fit Criterion:** Any potential player with an internet connection will be able to download and play the game wherever they are located.

**Description:** The game will require the user to create an account.

**Rationale:** Users will need to create an account in order to track their stats and update their rank on both the local and global leaderboards.

**Fit Criterion:** The user will complete the registration form to sign up.

### Implementation Environment of the Current System

*SV: This section deals with the physical and technical environment in which the proposed product will operate, such as hardware, operating system, and communications issues.*

*Content*

*This describes the technological and physical environment in which the product is to be installed. It includes automated, mechanical, organizational, and other devices, along with the nonhuman adjacent systems.*

*Motivation*

*To describe the technological environment into which the product must fit. The environment places design constraints on the product. This part of the specification provides enough information about the environment for the designers to make the product successfully interact with its surrounding technology.*

*The operational requirements are derived from this description.*

*Examples*

*Examples can be shown as a diagram, with some kind of icon to represent each separate device or person (processor). Draw arrows to identify the interfaces between the processors, and annotate them with their form and content.*

*Considerations*

*All component parts of the current system, regardless of their type, should be included in the description of the implementation environment.*

*If the product is to affect, or be important to, the current organization, then include an organization chart.*

The game is to be played on Windows, Mac, iOS, and Android devices. The easiest way to accommodate this is to have the app developed as a web application, wherein a user is required to log in through a website to play the game. This will be the first step in development, and local clients for each platform will be developed once the web application is completed.

### Partner or Collaborative Applications

*SV: This section documents external applications with which this product must be compatible, such as the ability to read and write Microsoft Excel format data files.*

*Content*

*This describes applications that are not part of the product but with which the product will collaborate. They can be external applications, commercial packages, or preexisting in-house applications.*

*Motivation*

*To provide information about design constraints caused by using partner applications. By describing or modeling these partner applications, you discover and highlight potential problems of integration.*

*Examples*

*This section can be completed by including written descriptions, models, or references to other specifications. The descriptions must include a full specification of all interfaces that have an effect on the product.*

*Considerations*

*Examine the work context model to determine whether any of the adjacent systems should be treated as partner applications. It might also be necessary to examine some of the details of the work to discover relevant partner applications.*

Checker Wars currently has no partner or collaborative applications.

### Off-the-Shelf Software

*SV: This section describes commercial off-the-shelf ( COTS ) software that* ***MUST*** *be included in the final product.*

*Content*

*This describes commercial, open source, or any other off-the-shelf software (OTS) that must be used to implement some of the requirements for the product. It could also apply to nonsoftware OTS components such as hardware or any other commercial product that is intended as part of the solution.*

*Motivation*

*To identify and describe existing commercial, free, open source, or other products to be incorporated into the eventual product. The characteristics, behavior, and interfaces of the package are design constraints.*

*Examples*

*This section can be completed by including written descriptions, models, or references to supplier’s specifications.*

*Considerations*

*When gathering requirements, you may discover requirements that conflict with the behavior and characteristics of the OTS software. Keep in mind that the use of OTS software was mandated before the full extent of the requirements became known. In light of your discoveries, you must consider whether the OTS product is a viable choice. If the use of the OTS software is not negotiable, then the conflicting requirements must be discarded.*

*Note that your strategy for discovering requirements is affected by the decision to use OTS software. In this situation you investigate the work context in parallel with making comparisons with the capabilities of the OTS product. Depending on the comprehensibility of the OTS software, you might be able to discover the matches or mismatches without having to write each of the business requirements in atomic detail. The mismatches are the requirements that you will need to specify so that you can decide whether to satisfy them by either modifying the OTS software or modifying the business requirements.*

*Given the spate of lawsuits in the software arena, you should consider whether any legal implications might arise from your use of OTS. You can cover this in the section on Legal Requirements.*

*Note the subtle difference between this section and section 29 below. This section documents OTS solutions that* ***must be*** *included in the final solution, and the latter offers suggestions for OTS that could be included.*

Checker Wars is being programmed from the ground up, and so does not contain any off-the-shelf software.

### Anticipated Workplace Environment

*SV: This section deals with human factors regarding the environment in which the product will be used, such as noisy environments or mobile applications.*

*Content*

*This describes the workplace in which the users are to work and use the product. It should describe any features of the workplace that could have an effect on the design of the product, and the social and culture of the workplace.*

*Motivation*

*To identify characteristics of the workplace so that the product is designed to compensate for any difficulties.*

*Examples*

*The printer is a considerable distance from the user’s desk. This constraint suggests that printed output should be deemphasized.*

*The workplace is noisy, so audible signals might not work.*

*The workplace is outside, so the product must be weather resistant, have displays that are visible in sunlight, and allow for the effect of wind on any paper output.*

*The product is to be used in a library; it must be extra quiet.*

*The product is a photocopier to be used by an environmentally conscious organization; it must work with recycled paper.*

*The user will be standing up or working in positions where he must hold the product. This suggests a hand-held product, but only a careful study of the users’ work and workplace will provide the necessary input to identifying the operational requirements.*

*Considerations*

*The physical work environment constrains the way that work is done. The product should overcome whatever difficulties exist; however, you might consider a redesign of the workplace as an alternative to having the product compensate for it.*

Both single player and multiplayer gameplay will require an internet connection initially while the web application is the only source of gameplay. Further down the line when the standalone applications for the game are developed for each system, users will be able to play single player games against an AI opponent offline.

### Schedule Constraints

*SV:* ***When*** *things must be done, or when they may be most/least beneficial.*

*Content*

*Any known deadlines, or windows of opportunity, should be stated here.*

*Motivation*

*To identify critical times and dates that have an effect on product requirements. If the deadline is short, then the requirements must be kept to whatever can be built within the time allowed.*

*Examples*

*To meet scheduled software releases.*

*There may be other parts of the business or other software products that are dependent on this product.*

*Windows of marketing opportunity.*

*Scheduled changes to the business that will use your product. For example, the organization may be starting up a new factory and your product is needed before production can commence.*

*Considerations*

*State deadline limitations by giving the date and describing why it is critical. Also, identify prior dates where parts of your product need to be available for testing.*

*You should also ask questions about the impact of not meeting the deadline:*

*● What happens if we don’t build the product by the end of the calendar year?*

*● What is the financial impact of not having the product by the beginning of the Christmas buying season?*

As previously discussed, the web application of the game will be developed first. Depending on reception and popularity of the game, development further into standalone Mac, Windows, iOS, and Android clients will continue, as will general development to the underlying codebase of the game. We aim to have the game developed and ready for launch by the Christmas holiday season.

### Budget Constraints

*SV: Limitations on the funds and other resources available for this project.*

*Content*

*The budget for the project, expressed in money or available resources.*

*Motivation*

*The requirements must not exceed the budget. This limitation may constrain the number of requirements that can be included in the product.*

*The intention of this question is to determine whether the product is really wanted.*

*Considerations*

*Is it realistic to build a product within this budget? If the answer to this question is no, then either the client is not really committed to building the product or the client does not place enough value on the product. In either case you should consider whether it is worthwhile continuing.*

Our first goal is to find an investor or sponsor to join us in funding the project for development. Sponsors can include computer and peripheral companies such as Razer, who can market their products beside the game, and when the game is complete, sponsorship of professional players of the game.

In-app purchases will take the form of cosmetic additions to the gameplay, as we want to create a fair competitive environment for all players of the game regardless of funds available to player. Thus, it is important that we find investors and sponsors who share this vision with us. The scope of the game is not massive, so we hope that we will not need copious funds to complete development of the game.

## Naming Conventions and Definitions

### Definitions of Key Terms

*SV: Define* ***words*** *that may have special or multiple meanings.*

*All Terms, Including Acronyms and Abbreviations, Used in the Project must be defined at some point. List the most important ones here, and refer the reader to the glossary on page 90 for a complete list. ( Note: that page number is a cross-reference, and will automatically be updated whenever the glossary moves. )*

*Content*

*A glossary containing the meanings of all names, acronyms, and abbreviations used within the requirements specification. Select names carefully to avoid giving a different, unintended meaning.*

*This glossary reflects the terminology in current use within the work area. You might also build on the standard names used within your industry.*

*For each term, write a succinct definition. The appropriate stakeholders must agree on this definition.*

*Avoid abbreviations, as they introduce ambiguity, require additional translations, and could potentially lead to misinterpretation in the mind of anyone who is trying to understand your requirements. Ask your requirements analysts to replace all abbreviations with the correct term. This is easily done with word processors.*

*Acronyms are acceptable if they are completely explained by a definition.*

*Motivation*

*Names are very important. They invoke meanings that, if carefully defined, can save hours of explanations. Attention to names at this stage of the project helps to highlight misunderstandings.*

*The glossary produced during requirements is used and extended throughout the project.*

*Examples*

*Truck: A vehicle used for spreading de-icing material on roads. “Truck” is not used to refer to goods-carrying vehicles.*

*BIS: Business Intelligence Service. The department run by Steven Peters to supply business intelligence for the rest of the organization.*

*Considerations*

*Make use of existing references and data dictionaries. Obviously, it is best to avoid renaming existing items unless they are so ambiguous that they cause confusion.*

*From the beginning of the project, emphasize the need to avoid homonyms and synonyms. Explain how they increase the cost of the project.*

All terms planned for the game are of general use and knowledge or are common to all other games played. These include, for example:

**RANK**: A player’s rank among other players of the game, both locally and globally.

**SCORE**: A player’s individual score at the end of every game

### UML and Other Notation Used in This Document

*SV: Define* ***symbols****,* ***diagrams****, and other* ***notations*** *used. May refer to a standard reference, such as “UML Distilled” by Fowler. ( Include in bibliography. )*

*Content*

*This section should describe the specific meaning of any symbols, punctuation, subscripts, superscripts, etc. used commonly throughout the document. If following published or common standards, then it is acceptable to reference those standards, and list any exceptions.*

*Motivation*

*If the distinction between a hollow arrow and a solid arrow is significant, for example, then everyone must know exactly what the distinctions and meanings are.*

*Considerations*

*If a particular notation is only used in one place, say on a single diagram or in a single section, then it may be more appropriate to document it in that specific location.*

*Example*

*This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [4]. Any exceptions are noted where used.*

This document generally follows the Version 3.0 OMG UML standard. Any exceptions are noted.

### Data Dictionary for Any Included Models

*SV: Define* ***data structures*** *and* ***data properties*** *relative to this project, such as the contents of an employee record or the fact that student GPA ranges from 0.0 to 4.0 corresponding to letter grades of F to A. Data file formats may be referenced to documented standards, such as jpg or pdf.*

*Content*

*Dictionary definitions of all information flows and stores used in models. Particular consideration should be given to defining the data attributes of all flows shown the context models (see sections 7 and 8).*

*This section should also contain any technical specifications for interfaces shown on the context models.*

*Motivation*

*The context diagram provides an accurate definition of the scope of the work being studied or the scope of the product to be built. This definition can be completely accurate only if the information flows bordering the scope have their attributes defined.*

*Examples*

*Road de-icing schedule = issue number + {road section identifier + treatment start time + critical start time + truck identifier} + depot identifier*

*As you progress through the requirements specification, define each of the elementary terms in detail.*

*Considerations*

*The dictionary provides a link between the requirements analysts and the implementers. The implementers add implementation details to the terms in the dictionary, defining how the data will be implemented. Also, implementers add terms that are present because of the chosen technology and that are independent of the business requirements.*

Players may export and share their configuration (.conf) files for other players to load and experience what other players’ personal gameplay UI looks like.

## Relevant Facts and Assumptions

### Facts

*SV: Factual information relevant to the project, such as census data.*

*Content*

*Factors that have an effect on the product, but are not mandated requirements constraints. They could be business rules, organizational systems, or any other activities that have an effect on this product. Facts are things you want the reader of the specification to know.*

*Motivation*

*Relevant facts provide background information to the specification readers, and might contribute to requirements. They will have an effect on the eventual design of the product.*

*Examples*

*One ton of de-icing material will treat three miles of single-lane roadway.*

*The existing application is 10,000 lines of C code.*

The game will take up a small amount of space when developed for individual platforms, but will initially all be hosted on Bohn Jell Entertainment’s servers, and loaded (from Bohn Jell Entertainment’s servers or local user cache) whenever a user logs on through the website. The game will be under 20MB in size.

### Assumptions

It is assumed that the end user is using a sufficiently powerful computing device to play the game (2GB+ of RAM, modern ARM, Intel, or AMD processor).

# Requirements

*SV: Sections 9 and 10 deal with functional requirements. Sections 11 to 20 are a very thorough list of possible non-functional requirements, not all of which apply to every project. You should think carefully about each of these, form requirements if applicable, or write “Not Applicable” otherwise. See section 10 for the format of individual requirements. Section 21 documents the acceptance tests planned to verify the requirements – See that section for further details, and be aware that every requirement needs at least one verifying acceptance test ( though some tests may verify more than one requirement. )*

## Product Use Cases

This section describes in more specific detail the steps the system takes during its performance. These use cases more specifically define the system and user requirements and their boundaries, as well as functional requirements, and to organize work.

*SV: Product Use Cases are very similar to Product Scenarios, but in more formal detail. They serve as a first step towards developing functional requirements, and can aid in organizing requirements according to the use case(s) from which they were developed. See the CS 440 web site for a sample use-case form, with instructions.*

*This section begins to describe in more specific and precise detail exactly what steps the system takes in the course of its performance. Use cases serve not only to more specifically define the system ( and its boundaries ), but also to identify functional requirements, to identify initial objects / classes, and to organize the work.*

### Use Case Diagrams

The use case below shows the interactions between players, the current match, and both the game and player servers. Data generated by the match and each player’s moves are sent to the player server to record history of play, and the game server to update the board as a result of moves. The diagram may appear simple, but all of the mechanics of each move within the game as described in previous sections is handled within the game server and sent back to the current game state to update the board.

*SV: Use case diagrams list the use cases developed for a system, mark the boundary of what is internal or external to the system to be developed, and indicate which external entities ( actors ) are associated with each use case.*

*Use Case diagrams serve two purposes: As a form of graphical table of contents listing the individual use-cases, and also to define the boundary of what is included as part of the proposed system and what is not included.*

*A use case diagram identifies the boundaries between the users (actors) and the product. You arrive at the product boundary by inspecting each business use case and determining, in conjunction with the appropriate stakeholders, which part of the business use case should be automated (or satisfied by some sort of product) and what part should be done by the user. This task must take into account the abilities of the actors (section 3), the constraints (section 4), the goals of the project (section 1), and your knowledge of both the work and the technology that can make the best contribution to the work.*

*The use case diagram shows the actors outside the product boundary (the rectangle). The product use cases are the ellipses inside the boundary. The lines denote usage. Note that actors can be either automated or human.*

*Depending on the complexity of the product it may be necessary to use more than one diagram to list all of the use cases. When more than one diagram is required the use-cases can be divided up several ways: Normal operations versus exceptional cases, or daily tasks versus monthly tasks, or user tasks versus administration tasks, etc.*

*Examples*

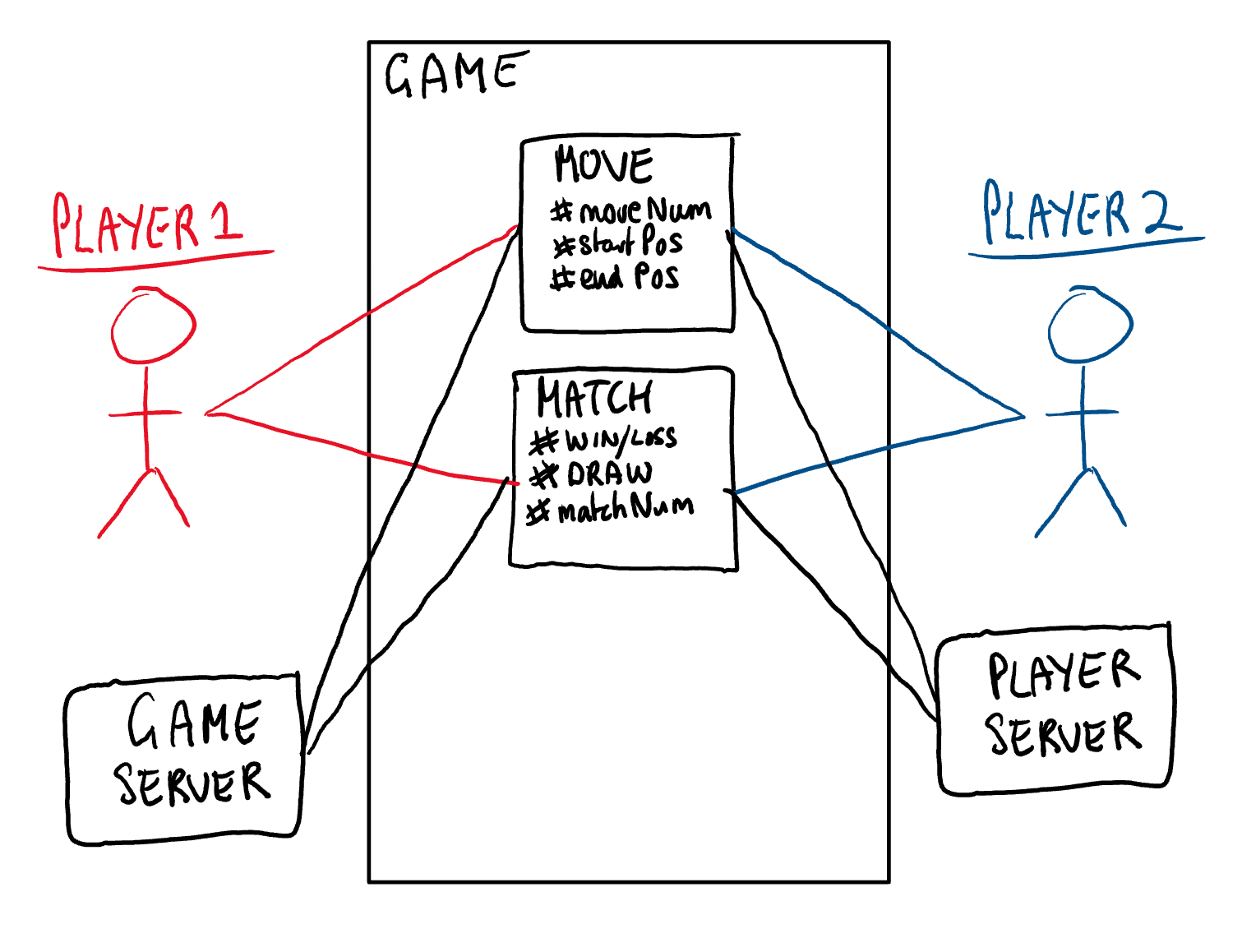


Figure : Match Use Case Diagram

*Derive the product use cases by deciding where the product boundary should be for each business use case. These decisions are based on your knowledge of the work and the requirements constraints.*

### Product Use Case List

*SV: A list ( table ) of use cases is an alternative to the use case diagram, particularly when there are many use cases. There may be additional information in the table not found in the diagram, such as cross referencing to other sections or materials.*

*The use case diagram is a graphical way of summarizing the product use cases relevant to the product. If you have a large number of product use cases (we find 15–20 is a good limit), then it is better to make a list of the product use cases and model or describe each one individually.*

### Individual Product Use Cases

*Use cases are similar to scenarios, in that both tell the story of how the system interacts with the user(s) in response to some business event or while conducting some business task. The difference is that use-cases are much more formal, with certain pre-determined sections for each use-case, and that use-cases indicate clearly what action the system takes in response to what actions taken by the user.*

*SV: The following example was copied from “useCaseFormWithInstructions.docx”, available on the CS 440 web site. ( There is also a blank version available. )*

|  |
| --- |
| Use case ID: Name:  pre-conditions:  post-conditions:  Initiated by:  Triggering Event:  Additional Actors: |
| Sequence of Events:  Initiating event or action should be step 1, taken by initiating actor.   1. System response follows, indented right.   All external action steps are aligned with step 1. ( "stimulus" style )   1. All system responses are indented right, aligned with step 2. ( "response" style )   All steps should be expressed in the active voice, clearly indicating **who** performs each action   1. The sequence of events should show a back-and-forth stimulus-response relationship. |
| Alternatives: These would be normal and expected variations from the base case.  Exceptions: These would be unusual variations from the base case, often caused by problems. |

* *For all of the above, list as NA if not applicable.*
* *The following may be added if relevant, or omitted otherwise:*
  + *related use cases or scenarios*
  + *associated tests, systems, classes, etc.*
  + *revision history*
  + *references to other documents*
  + *author(s) / originator( s )*
  + *notes*
* *Alternatives and Exceptions may be listed either as separate use cases or as notes to a base case, depending on their significance and similarity.*
* *For regularly occurring periodic events, "time" can be listed as the initiating actor.*

*You may also want to view Figure 4.7 from "Object Oriented Software Engineering" by Bruegge and DuToit*

## Functional Requirements

**ID# - FR1 – Save & Quit (Single-Player)**

**Description:** The player must be able to save and quit at any point during the game when playing against an AI opponent.

**Rationale:** Life happens, and users sometimes need to pause what they’re doing and come back later. This is a useful functionality for the game to have for players to enjoy playing the game.

**Fit Criterion:** Automatic test cases, as well as tester/user verified instances of gameplay being paused and quit, then resumed at a later time. Programmed unit tests with every release will test the reliability of the save and quit feature.

**Acceptance Tests:** FR1 – Save & Quit (Single-Player).

**ID# - FR2 – Quit (Multiplayer)**

**Description:** Players may disconnect during online gameplay or may leave the game for other reasons. The system needs to handle all of these instances gracefully. If a player disconnects, there needs to be a grace period that allows them to rejoin, but that grace period should not be too long, or other players will get frustrated.

**Rationale:** To facilitate a good gameplay experience for all users, when an opponent quits, the current player that is still connected needs to be able to wait for a period of time, then disconnect if they choose without penalty.

**Fit Criterion:** Player frustration surveys can be utilized to determine whether the balance between waiting for players to reconnect and moving on to a new game has been reached. Online Softbots will be used to test player disconnection results.

**Acceptance Tests:** FR2 – Quit (Multiplayer).

*SV: Each requirement listed needs to have a unique identifier, a short name, a one- or two-sentence description, a rationale, a fit criteria, and reference to one or more acceptance tests to be used to confirm the completion of this particular requirement. The acceptance tests themselves are documented in section 0- See that section for further details. It is recommended to number the requirements according to their type, such as F-4 for the fourth functional requirement or U-2 for the second usability requirement. Functional requirements specifically deal with the functionality the system must have, and are generally derived directly from the steps the system takes during use cases.*

*Content*

*A specification for each functional requirement. A full explanation is included in this template’s introductory material.*

*Motivation*

*To specify the detailed functional requirements for the activity of the product.*

*Fit Criterion*

*Each functional requirement should have a fit criterion or a test case. In any event, the fit criterion is the benchmark to allow the tester to determine whether the implemented product has met the requirement.*

*Considerations*

*If you have produced an event/use case list (see sections 7b and 8a), then you can use it to help you trigger the functional requirements for each event/use case. If you have not produced an event/use case list, give each functional requirement a unique number and, to help with traceability, partition these requirements into event/use case–related groups later in the development process.*

## Data Requirements

Data requirements of Checker Wars are kept fairly small, but large enough for future product growth. Bohn Jell Entertainment needs to keep just enough data for the gameplay experience to be seamless, and for research purposes for either future games, or to improve Checker Wars AI opponents. All data will be kept on Bohn Jell Entertainment’s off-site servers and will be backed up regularly and/or stored in RAID arrays to avoid any data loss.

**ID# - DR1 – Player Data**

**Description:** Each player’s data, including number of games played, won, lost and drawn, ELO (rank/score), and time played.

**Rationale:** Players need to have ubiquitous access to their data so that no matter which computer or device they are using to play on their account, they can see their stats and record their gameplay history.

**Fit Criterion:** Automatic unit testing, as well as human testing will assure that data is retained and produced to the user in a timely manner, and if there are any issues presented, that a regular backup of all user data is available in case of data loss.

**Acceptance Tests:** FR1 – Save & Quit (Single-Player), FR2 – Quit (Multiplayer), DR1 – Player Data.

**ID# - DR2 – Game Data**

**Description:** All games played will be recorded for posterity on Checker Wars’ servers. This includes the time, move of each player, and win/loss/quit.

**Rationale:** Keeping this data will help Bohn Jell Entertainment when developing better AI opponents for players. Often the creators of games themselves do not anticipate styles of play and the metagame that develops from their creation. There is no way we could program an AI opponent out of the gate that would beat humans forever, and so recording and analyzing human play will help with the development of more and more difficult AI opponents for players to play against.

**Fit Criterion:** Automatic unit testing, as well as human testing will assure that data is retained and produced to the user in a timely manner, and if there are any issues presented, that a regular backup of all user data is available in case of data loss.

**Acceptance Tests:** DR2 – Game Data.

*SV: Data requirements deal with requirements that are somehow related to data, such as the definition of what is included in a “student record” or the acceptable form of an e-mail address or allowable range of certain data items.*

*Content*

*A specification of the essential subject matter, business objects, entities, and classes that are germane to the product. It might take the form of a first-cut class model, an object model, or a domain model. Alternatively, these requirements might be described by defining the terms in the dictionary described in section 5.*

*Motivation*

*To clarify the system’s subject matter, thereby triggering recognition of requirements not yet considered.*

*Example*

*This is a model of the system’s business subject matter using the Unified Modeling Language (UML) class model notation.*

**

*You can use any type of data or object model to capture this knowledge. The issue is to capture the meaning of the business subject matter and the connections between the individual parts, and to show that you are consistent within your project. If you have an established company standard notation, use that, as it will help you to reuse knowledge between projects.*

*Considerations*

*Are there any data or object models for similar or overlapping systems that might be a useful starting point? Is there a domain model for the subject matter dealt with by this system?*

## Performance Requirements

### Speed and Latency Requirements

For the single player experience, this should not be much of an issue, but for multiplayer, different players’ internet speed and connections can have a big effect on the gameplay experience.

To start, we will have servers across America, Europe, and Asia to handle gameplay within these regions. Players will default to their specific region but may opt to play on other regions servers if they so choose.

To facilitate users ability for agency in the game, we will show each player’s current ping updates every 2 seconds above their avatar in-game, so that if one player does lag or time-out, the other player will know that’s the case, and will be able to wait OR decide to disconnect from the game to start a new one.

*SV: Requirements specifying how fast ( or slow ) the product must operate or how much lag is allowable between stimulus and either initial response or task completion. Other timing-related requirements could go in this section.*

*Content*

*Specifies the amount of time available to complete specified tasks. These requirements often refer to response times. They can also refer to the product’s ability to operate at a speed suitable for the intended environment.*

*Motivation*

*Some products—usually real-time products—must be able to perform some of their functionality within a given time slot. Failure to do so may mean catastrophic failure (e.g., a ground-sensing radar in an airplane fails to detect an upcoming mountain) or the product will not cope with the required volume of use (e.g., an automated ticket-selling machine).*

*Examples*

*Any interface between a user and the automated system shall have a maximum response time of 2 seconds.*

*The response shall be fast enough to avoid interrupting the user’s flow of thought.*

*The product shall poll the sensor every 10 seconds.*

*The product shall download the new status parameters within 5 minutes of a change.*

*Fit Criterion*

*Fit criteria are needed when the description of the requirement is not quantified. However, we find that most performance requirements are stated in quantified terms. The exception is the second requirement shown above, for which the suggested fit criterion is*

*The product shall respond in less than 1 second for 90 percent of the interrogations. No response shall take longer than 2.5 seconds.*

*Considerations*

*There is a wide variation in the importance of different types of speed requirements. If you are working on a missile guidance system, then speed is extremely important. By contrast, an inventory control report that is run once every six months has very little need for a lightning-fast response time.*

*Customize this section of the template to give examples of the speed requirements that are important within your environment.*

**ID# - SLR1 - Latency**

**Description:** Latency must always be minimized to the smallest extent possible in order to facilitate good user experience.

**Rationale:** All interactions with the environment need to have quick and obvious reactions from the system so that the user has timely feedback and knows the program is running correctly.

**Fit Criterion:** All responses handled by our servers will respond in a timely manner and be tested regularly for speed and accuracy. All responses given good internet connections should be in less than 2 seconds but have a 95% rate of response in less than 100ms.

**Acceptance Tests:** SLR1 – Latency.

### Precision or Accuracy Requirements

In Checker Wars, precision and accuracy are paramount. The entire credibility of the game relies on users’ movements being executed in exactly the manner they specify. Thus, there are no tolerances for failure of precision or accuracy.

Player score will be accurate to the integer level and not rounded.

Graphical precision and accuracy of board moves will display accurately within 50ms once the move has been registered with our servers.

*SV: Self-explanatory. How accurate or precise must the system be.*

*Content*

*Quantification of the desired accuracy of the results produced by the product.*

*Motivation*

*To set the client’s and users’ expectations for the precision of the product.*

*Examples*

*All monetary amounts shall be accurate to two decimal places.*

*Accuracy of road temperature readings shall be within ±2°C.*

*Considerations*

*If you have done any detailed work on definitions, then some precision requirements might be adequately defined by definitions in section 5.*

*You might consider which units the product is intended to use. Readers will recall the spacecraft that crashed on Mars when coordinates were sent as metric data rather than imperial data.*

*The product might also need to keep accurate time, be synchronized with a time server, or work in UTC.*

*Also, be aware that some currencies have no decimal places, such as the Japanese yen.*

### Capacity Requirements

Because Checker Wars is a new game, we are not sure what the capacity requirements of our servers will need to be. Based on the success of other online multiplayer games, we may need a large capacity, but Checker Wars is also not a very data intensive game, and so will not require the large bandwidth that other online (FPS, strategy) games require.

As a base, Checker Wars will need to be able to handle 10,000 consecutive games being played at launch. A team of data analysts will monitor base and peak loads and will update management as needed to determine when more resources need to be utilized.

**ID# - CR1 – Server Capacity Requirement**

**Description:** At launch Checker Wars needs enough capacity to handle possible influxes of new players.

**Rationale:** Imperative to a good player experience is a seamless gameplay experience. If players are unable to log on, join up, or if it even takes greater than a few seconds for them to join and play, they will opt to play something else instead.

**Fit Criterion:** Server availability needs to be kept at +20% of peak server capacity. Once a new peak is reached, new server capacity needs to be prepared.

**Acceptance Tests:** SLR1 - Latency,CR1 – Server Capacity Requirement.

*SV: Requirements regarding the largest “thing” the system must be able to handle, or perhaps how many things it can handle ( at once. ) Note: Requirements regarding how many things it can handle in a given time period would be a speed requirement, covered in section 12a above.*

*Content*

*This section specifies the volumes that the product must be able to deal with and the amount of data stored by the product.*

*Motivation*

*To ensure that the product is capable of processing the expected volumes.*

*Examples*

*The product shall cater for 300 simultaneous users within the period from 9:00 a.m. to 11:00 a.m. Maximum loading at other periods will be 150 simultaneous users.*

*During a launch period, the product shall cater for a maximum of 20 people to be in the inner chamber.*

*Fit Criterion*

*In this case, the requirement description is quantified, and thus can be tested.*

## Dependability Requirements

### Reliability and Availability Requirements

Gameplay and server reliability are extremely important to gameplay and user experience. Though there always needs to be some tolerance of bugs, faults, and exploits, they need to be pre-empted and handled appropriately. Meaning quickly and gracefully, without impacting player experience for long or in a negative manner. If a player experiences a bug or crash, the data from that crash should be sent to Bohn Jell for investigation to fix the bug that caused it.

**ID# - RR1 – Server Reliability and Availability**

**Description:** Checker Wars’ servers need to have a reliability that facilitates user expectations and provides a good experience.

**Rationale:** Imperative to a good player experience is a seamless gameplay experience. If players are unable to log on, join up, or if it even takes greater than a few seconds for them to join and play, they will opt to play something else instead.

**Fit Criterion:** 99.9% server uptime for player and game servers.

**Acceptance Tests:** SLR1 - Latency,CR1 – Server Capacity Requirement, RR1 – Server Reliability.

*SV: Reliability relates to how frequently the system fails, ( either by shutting down or by delivering erroneous results ), and the consequences of those failures. These requirements may also address the conditions under which it is allowed to fail ( or not. ), See also availability and robustness in the following sections.*

*Content*

*This section quantifies the necessary reliability of the product. The reliability is usually expressed as the allowable time between failures, or the total allowable failure rate.*

*Motivation*

*It is critical for some products not to fail too often. This section allows you to explore the possibility of failure and to specify realistic levels of service. It also gives you the opportunity to set the client’s and users’ expectations about the expected frequency and significance of potential failures.*

*Examples*

*The product shall not fail more than once per day.*

*No data shall be lost or damaged in the event of a failure. ( This is an example of a* ***fail-safe*** *requirement, which states that the product is allowed to fail, but it must do so safely. )*

*Considerations*

*Consider carefully whether the real requirement for your product is that it is available for use or that it does not fail at any time.*

*Consider also the cost of reliability and availability, and whether it is justified for your product.*

### Robustness or Fault-Tolerance Requirements

**ID# - FTR1 – Graceful Crashing**

**Description:** Crashes are an unfortunate reality but can be dealt with. There is the possibility of logic, graphical, and connection issues while playing Checker Wars, and these instances need to be handled in a positive way for the user and for stopping these same crashes from happening again in the future.

**Rationale:** Without handling these errors and fixing the for later, players will lose interest in the game out of frustration.

**Fit Criterion:** Crashes must resolve within 2 seconds of happening and must also be reported to the game or player servers so that they can be analyzed and patched.

**Acceptance Tests:** SLR1 - Latency, CR1 – Server Capacity Requirement, RR1 – Server Reliability,FTR1 – Graceful Crashing.

*SV: This section deals with the system’s ability to provide at least partial functionality in the face of failures or resource shortages, such as operating in offline mode when network connectivity is unavailable. See also reliability and availability.*

*Content*

*Robustness specifies the ability of the product to continue to function under abnormal circumstances.*

*Motivation*

*To ensure that the product is able to provide some or all of its services after or during some abnormal happening in its environment.*

*Examples*

*The product shall continue to operate in local mode whenever it loses its link to the central server.*

*The product shall provide 10 minutes of emergency operation should it become disconnected from the electricity source.*

*Considerations*

*Abnormal happenings can almost be considered normal. Today’s products are so large and complex that there is a good chance that at any given time, one component will not be functioning correctly. Robustness requirements are intended to prevent total failure of the product.*

*You could also consider disaster recovery in this section. This plan describes the ability of the product to reestablish acceptable performance after faults or abnormal happenings.*

### Safety-Critical Requirements

*SV: These requirements address potential harm to health, safety, or property, and may refer to relevant standards such as OSHA compliance.*

*Content*

*Quantification of the perceived risk of damage to people, property, and environment. Different countries have different standards, so the fit criteria must specify precisely which standards the product must meet.*

*Motivation*

*To understand and highlight the damage that could potentially occur when using the product within the expected operational environment.*

*Examples*

*The product shall not emit noxious gases that damage people’s health.*

*The heat exchanger shall be shielded from human contact.*

*Fit Criterion*

*The product shall be certified to comply with the Health Department’s standard E110-98. It is to be certified by qualified testing engineers.*

*No member of a test panel of [specified size] shall be able to touch the heat exchanger. The heat exchanger must also comply with safety standard [specify which one].*

*Considerations*

*The example requirements given here apply to some, but not all, products. It is not possible to give examples of every variation of safety-critical requirement. To make the template work in your environment, you should customize it by adding examples that are specific to your products.*

*Also, be aware that different countries have different safety standards and laws relating to safety. If you plan to sell your product internationally, you must be aware of these laws. A colleague has suggested that for electrical products, if you follow the German standards, the largest number of countries will be supported.*

*If you are building safety-critical systems, then the relevant safety-critical standards are already well specified. You will likely have safety experts on your staff. These experts are the best source of the relevant safety-critical requirements for your type of product. They will almost certainly have copious information that you can use.*

*Consult your legal department. Members of this department will be aware of the kinds of lawsuits that have resulted from product safety failure. This is probably the best starting place for generating relevant safety requirements.*

**ID# - Name**

**Description:** Your description here . . .

**Rationale:** Your rationale here . . .

**Fit Criterion:** Your fit criteria here . . .

**Acceptance Tests:** List ID# and/or names here . . .

## Maintainability and Supportability Requirements

### Maintenance Requirements

As with all products and services, maintenance is required and important to ensure performance and dependability. Monthly maintenance will be scheduled for the first Monday night of each month from 2:00AM to 6:00AM. This schedule will impact the least amount of players from launch, but data analysts may be able to provide useful insights as to another time that will affect less players once we have established a regular player base.

**ID# - MR1 – Monthly Maintenance**

**Description:** Server maintenance will ensure players run into less issues as they play Checker Wars.

**Rationale:** Maintenance is required and important to ensure performance, dependability, and trust in Checker Wars.

**Fit Criterion:** Monthly maintenance is completed, and logs are kept of what hardware/software was updated, and what needs to be updated at the next monthly maintenance period.

**Acceptance Tests:** MR1 – Monthly Maintenance.

*SV: This section deals with the ease with which the system can be maintained, and possibly who will perform system maintenance and under what conditions. The ease of evolving the system into future versions may also be addressed here, or in a separate section ( not included in this template ) if that is a major concern.*

*Content*

*A quantification of the time necessary to make specified changes to the product.*

*Motivation*

*To make everyone aware of the maintenance needs of the product.*

*Examples*

*New MIS reports must be available within one working week of the date when the requirements are agreed upon.*

*A new weather station must be able to be added to the system overnight.*

*Considerations*

*There may be special requirements for maintainability, such as that the product must be able to be maintained by its end users or by developers who are not the original developers. These requirements have an effect on the way that the product is developed. In addition, there may be requirements for documentation or training.*

*You might also consider writing testability requirements in this section.*

### Supportability Requirements

Checker Wars’ support will be intuitive to users who play it. Tutorial levels will be provided for new users, and a video explaining gameplay will also be available. No explicit supportability requirements except for the above are listed, as the needs for supportability will change as the product evolves.

### Adaptability Requirements

Refer to: [6b Implementation Environment of the Current System](#_Implementation_Environment_of)

### Scalability or Extensibility Requirements

As listed in 12c Capacity requirements, Checker Wars will need to be able to handle 10,000 consecutive games being played at launch. Games can have a small player base and suddenly explode in popularity, needing extra server capacity in a very short amount of time.

There are services available today that allow the server capacity to expand as demand expands, but these are often expensive and leave our users data in the hands of other companies. Ideally Checker Wars will host their own servers and increase capacity as needed, but services that auto-expand with demand should be kept in mind as a possibility if needed.

**ID# - SER1 – Scalability Requirement**

**Description:** Game popularity can fluctuate greatly from day to day, week, to week, and sometimes even minute to minute. If a large influx of new players suddenly wants to join the Checker Wars world, we need to be able to handle them joining up all at once.

**Rationale:** Games can have a small player base and suddenly explode in popularity, needing extra server capacity in a very short amount of time.

**Fit Criterion:** As listed in CR1, server capacity needs to always be 20% above peak server loads. A contingency plan that includes the ability to quickly access and use new game and player servers within 12 hours needs to be in place. Even if these servers are temporary until more permanent servers can be installed, losing out on new players because of server capacity issues can be a death knell for games.

**Acceptance Tests:** SLR1 - Latency,CR1 – Server Capacity Requirement, SER1 – Scalability Requirement.

*SV: The ease of expanding the system to a larger capacity as the business grows.*

*Content*

*This specifies the expected increases in size that the product must be able to handle. As a business grows (or is expected to grow), our software products must increase their capacities to cope with the new volumes.*

*Motivation*

*To ensure that the designers allow for future capacities.*

*Examples*

*The product shall be capable of processing the existing 100,000 customers. This number is expected to grow to 500,000 customers within three years.*

*The product shall be able to process 50,000 transactions per hour within two years of its launch.*

### Longevity Requirements

No specific timeframe is suited beyond the initial launch +1 year. If the game hasn’t garnered a regular user base by then, the game will need to be reassessed and altered, or the company needs to pivot to a new IP.

*SV: This specifies the expected lifetime of the product.*

*Content*

*This specifies the expected lifetime of the product.*

*Motivation*

*To ensure that the product is built based on an understanding of expected return on investment.*

*Examples*

*The product shall be expected to operate within the maximum maintenance budget for a minimum of five years.*

## Security Requirements

Because Checker Wars does not log or record any sensitive data (we do not log IP addresses or any sensitive information other than that given to us by the users) our security requirements are minimal.

### Access Requirements

Only those with direct needs for the data will have access to it. This includes programmers or researchers using data to improve products.

*SV: These requirements address who has access to what ( data or functionality ) and under what conditions or restrictions.*

*Content*

*Specification of who has authorized access to the product (both functionality and data), under what circumstances that access is granted, and to which parts of the product access is allowed.*

*Motivation*

*To understand the expectations for confidentiality aspects of the system.*

*Examples*

*Only direct managers can see the personnel records of their staff.*

*Only holders of current security clearance can enter the building.*

*Fit Criterion*

*System function name or system data name.*

*User roles and/or names of people who have clearance.*

*Considerations*

*Is there any data that management considers to be sensitive? Is there any data that low-level users do not want management to have access to? Are there any processes that might cause damage or might be used for personal gain? Are there any people who should not have access to the system?*

*Avoid stating how you will design a solution to the security requirements. For instance, don’t “design a password system.” Your aim here is to identify the security requirement; the design will then come from this description.*

*Consider asking for help. Computer security is a highly specialized field, and one where improperly qualified people have no business. If your product has need of more than average security, we advise you to make use of a security consultant. Such consultants are not cheap, but the results of inadequate security can be even more expensive.*

### Integrity Requirements

Checker Wars shall prevent incorrect data from being introduced. For further information, refer to [section 13 Dependability Requirements.](#_Dependability_Requirements)

**ID# - IR1 – Data Accuracy**

**Description:** Data will be assessed as to whether it’s correct before being introduced to the game or player servers within realistic constraints.

**Rationale:** Accurate data is imperative. If incorrect data is introduced, it can destroy the credibility of Checker Wars or even John Bell Entertainment as a company. So, our customers can trust us, we must ensure accurate, reliable recording and storage of data.

**Fit Criterion:** Data will be audited every 6 months by our data scientists to ensure we are not recording incorrect or erroneous data within a 99.99% tolerance range (0.01% of data can be corrupted in some way)

**Acceptance Tests:** IR1 – Data Accuracy.

*SV: These requirements address the protection of data(bases) from intentional or accidental corruption, loss, or theft.*

*Content*

*Specification of the required integrity of databases and other files, and of the product itself.*

*Motivation*

*To understand the expectations for the integrity of the product’s data. To specify what the product will do to ensure its integrity in the case of an unwanted happening such as attack from the outside or unintentional misuse by an authorized user.*

*Examples*

*The product shall prevent incorrect data from being introduced.*

*The product shall protect itself from intentional abuse.*

*Considerations*

*Organizations are relying more and more on their stored data. If this data should be come corrupt or incorrect—or disappear—then it could be a fatal blow to the organization. For example, almost half of small businesses go bankrupt after a fire destroys their computer systems. Integrity requirements are aimed at preventing complete loss, as well as corruption, of data and processes.*

### Privacy Requirements

By design and to avoid any potential legal issues, Checker Wars will not record any personally identifiable information. The value of that kind of data is not worth the risk of holding on to it. Some information (account name, email address) that users give us will follow GDPR guidelines worldwide to protect user privacy.

If a customer opts to delete their account, it will be permanent. We will only keep data for active customers, and if a customer deletes their account and later wants to rejoin, they will have to create an entirely new account.

*SV: These requirements address data that must remain confidential, such as medical records or other personally identifiable data. Laws often apply. (See also section 20.)*

*Content*

*Specification of what the product has to do to ensure the privacy of individuals about whom it stores information. The product must also ensure that all laws related to privacy of an individual’s data are observed.*

*Motivation*

*To ensure that the product complies with the law, and to protect the individual privacy of your customers. Few people today look kindly on organizations that do not observe their privacy.*

*Examples*

*The product shall make its users aware of its information practices before collecting data from them.*

*The product shall notify customers of changes to its information policy.*

*The product shall reveal private information only in compliance with the organization’s information policy.*

*The product shall protect private information in accordance with the relevant privacy laws and the organization’s information policy.*

*Considerations*

*Privacy issues may well have legal implications, and you are advised to consult with your organization’s legal department about the requirements to be written in this section.*

*Consider what notices you must issue to your customers before collecting their personal information. A notice might go so far as to warn customers that you intend to put a cookie in their computer. Also, do you have to do anything to keep customers aware that you hold their personal information?*

*Customers must always be in a position to give or withhold consent when their private data is collected or stored. Similarly, customers should be able to view any private data and, where appropriate, ask for correction of the data.*

*Also consider the integrity and security of private data—for example, when you are storing credit card information.*

## Usability and Humanity Requirements

### Ease of Use Requirements

At first glance, the user must be able to utilize all controls with minimal difficulty. In this way, the user will solely concentrate on the content of the game rather than the frustration behind the controls or the lack of knowledge of the controls. As such, the controls must be intuitive and approachable. To achieve this, we will focus on several criteria, as defined by the client:

**Efficiency of Use:** The control scheme must be clearly defined in a way such that at first glance, the user will already be able to figure out which controls are being used. Since the game may end up being multi-platform, the user must know whether the game will be utilizing touch controls, certain buttons on a keyboard, or a control pad. Upon playing a tutorial or reading an instruction manual, the user must be able to connect the context of what will occur in game, to the intended user input. The game must be able to react immediately to a user input and in a way such that the user predicts or immediately observes the in-game change.

**Ease of Remembering:** The game will be unique in its design and have limitless replayability. Although the game does contain some complex design, at its core-functionality, it is easy to replay, and the mechanics are easily remembered. Various options will be added to allow the player to customize the game in various ways and multiplayer and AI would be utilized to allow players to play with each other or by themselves. In this way, the player will be able to approach the game and have a unique experience every time.

**Error Rates:** In order to minimize the amount of errors that the user commits, the controls of the product will be reactive and yet, during certain movements in the game, the game will request for the user’s confirmation of action. In this way, the user may be able to plan a move by inputting in action, and yet will not observe any consequence until the user confirms that action.

**Overall Satisfaction:** The user must not feel intimidated while playing the game, nor must they feel frustrations from the controls of the game. The mechanics are already unique to the game and so the feel of the game will follow suit. As such, the game has a wide demographic and already stands out from the market. The controls of the game are intuitive and certain design elements will be utilized in order to attract the attention of the user, such as color palette and art style.

**Feedback:** While playing the game, the user will be able to determine their action prior to the confirmation of that action. If a player wished to move a piece, the possible movements will be telegraphed to the player before the player confirms the action. This would also apply to attacking or utilization of a special space on the game board. In that way the player will be able to determine what they can or cannot do. Also, during certain sequences of the game, if the user hovers over an option with their controls, that option will be highlighted in order to allow the player to see the option that they will possibly select.

Upon the satisfaction of all these requirements, the product shall be easily utilized by a user with minimal technology experience at first glance, and it shall be easy for all, within the target demographic, to enjoy with minimal frustration.

**ID# - EOU1 – Ease of Use Requirements**

**Description:** The product shall be able to be used with only basic knowledge of the device that it resides on.

**Rationale:** To ensure that any user, regardless of technological experience, will be able to enjoy the product without any frustration pertaining to the mechanics of the product thereof.

**Fit Criterion:** Utilizing a test group of individuals with various technological experiences and of a multitude of backgrounds, at least 80 percent of the test group must be able to play the game successfully without any complaints or suggestions.

**Acceptance Tests:** EOU1 – Ease of Use Requirements

### Personalization and Internationalization Requirements

**Menu Options and Accessibility:** On the menu screen, there is a button which will allow the user to change up various aspects of the program. The user can configure various graphics options or the resolution. The graphics options allow a user to play on computer systems of varying performance. The resolution of the window and the orientation of the window is also customizable so that the program can accompany monitors of varying sizes and will be allowed to be put into a windowed or full-screen mode. The sound settings are also able to be changed in this menu which will allow for various volume adjustments of music or sound effect volumes. There is a sound setting for the orientation of the user’s output device. This setting appeals to whether the user is using headphones, 2.1 speaker setups, or a surround setup. Various theme changes are also available to the player in the options menu. The user can change the look of the various pieces or game board through the cosmetic purchases made, the default themes and color palettes, or through a user-generated theme.

**Options for Users with Disabilities:** The menu options screen feature additional options to include users that have disabilities. There is a colorblind mode which will change the color palette in a way such that colorblind players will have no issue distinguishing objects from each other on a screen. The product shall also feature adaptive gamepad support. This is support for non-conventional gamepads that are tailored towards users with certain motor-function disabilities. Another option that was implemented is text-to-speech and subtitles which will cater to users with visual and hearing impairment.

**Game Configuration Options:** In order to maximize the target demographic, the game will include many different customization options. For example, the AI difficulty shall be modular in that the player may be able to select a difficulty level that appeals to their playstyle or experience. Also, various aspects of each game are able to be customized. For instance, the level design may change from game to game, a timer can be implemented in order to speed up player choices, or the amount of actions may change depending on the user’s choice to speed up gameplay. By doing this, not only will we create a game which can appeal to a wider audience of players, we will also be satisfying the conditions that are outlined in section 16a by allowing flexibility of gameplay and making the game endlessly playable. The game board layout and the number of players per game will also be customizable in that different hazards may be added to the game board and the board shape may change to accompany such players. Another implementation would involve team play in which players may choose or have their teams randomly selected in a way such that players can play with each other in order to achieve victory. Another idea that will be implemented, called “Deus Ex Mode”, will allow an AI to oversee the progress in a game and intervene with the current game in different ways by changing the game board layout, swapping the positions of units, removing certain units from the game, swapping player teammates, or helping out the player that is doing the worst. Tournament play will be allowed in that the players can play multiple rounds of games against different players based upon their success rate and a “fog of war” is a customization option which will block the players view of the entire board and only limit their view to their a specific range around their pieces.

**Localization:** The game is not localized to specifically United Stated users. The game has numerous customization options which will allow for different languages. If a player chooses a specific language, the entire UI will cater to that language user. Also any cosmetic purchases that are made in-game will be localized to the country of purchase meaning that if a person is in the U.S., the currency used will be dollars, whereas a person in South Korea would encounter a store in which the currency will be South Korean Won. This game should not be bound by any country’s laws since this game will appeal to a general audience. As such, the game will not have any restrictions being released in any other country since this game does not involve any controversial topics, nor does it have any depictions of adult themes.

**ID# - PIR1 – Personalization and Internationalization Requirements**

**Description:** The product shall allow the user to configure their preferred settings and cater to users with a multitude of language backgrounds.

**Rationale:** Users located in different countries and with different device configurations will be able to access any settings to tailor their own user experience to their individual preferences.

**Fit Criterion:** The product must be able to work with all common languages with exceptional translation and through a panel of testers, must be customizable with all testers’ devices.

**Acceptance Tests:** PIR1 – Personalization and Internationalization Requirements

### Learning Requirements

This product will be able to be picked up by any person, regardless of the person’s experience with the product or lack thereof and be instantly familiar in terms of navigation of the products various menus and controls. The mechanics of the game itself will be simple enough that a person who plays the tutorial will automatically recognize how to play the game and all the rules associated with it. The game shall provide instant feedback towards all the player’s decisions which will help internalize the rules of the game.

Within a short amount of time, the player shall be familiar with the controls and mechanics of the game, but in terms of mastery of the game and game mechanics, it will involve the amount of games the player has played. A more proper estimate suggests that the game can be efficiently played by a player within 20 minutes or within the time frame in which the user completes the tutorial.

**ID# - LKR1 – Learning and Knowledge Requirements**

**Description:** The product shall be accessible to any person with basic knowledge of their device and be effectively played upon completion of the tutorial.

**Rationale:** A game with little learning curve will allow users that are intimidated by extensive game mechanics, to immediately enjoy the game and develop their own approach to their preferred playstyle.

**Fit Criterion:** A user with no prior knowledge of the game must be able to pick up the game, play through the tutorial, and be able to successfully play the game within a time frame of 20 minutes.

**Acceptance Tests:** LKR1 – Learning and Knowledge Requirements, EOU1 – Ease of Use Requirements.

### Understandability and Politeness Requirements

A player with a minimal understanding of the language supported should have no trouble understanding any of the messages or their context thereof. All the messages within the product are made in a way such that they are clear and concise, and all messages are consistent in their fluidity and style and feature virtually no colloquialisms of the languages in which they reflect. All the context-based messages accurately and precisely interject in such a way that the user understands what is always going on. The product shall be understandable in such a way that a general user of a language may be able to approach the product with no ambiguity of understanding.

**ID# - UPR1 – Understandability and Politeness Requirements**

**Description:** The product shall include messages and responses which are concise and intuitive and minimize all ambiguity of meaning of any object used in-game.

**Rationale:** Any user with a basic understanding of their own language should be able to approach the game and immediately understand the context of the game and its underlining tool tips.

**Fit Criterion:** Users that fit into an age group of between 3 – 10 should be able to recite the meaning of all messages shown with 75 percent accuracy.

**Acceptance Tests:** UPR1 – Understandability and Politeness Requirements, EOU1 – Ease of Use Requirements, PIR1 – Personalization and Internationalization Requirements.

### Accessibility Requirements

Approximately 20 percent of all males are red-green colorblind, and as such it would be a loss if no accessibility options were available for users with such disabilities. As seen in section 16b options for users with disabilities there will be various options in the settings menu found on the title screen to cater to these individuals.

There are options for users with common disabilities such as colorblindness, an option to accompany adaptive gamepads suited for users with motor function disabilities, subtitle options for users with hearing disabilities, and text-to-speech options for users with visual impairments.

There are no visuals that would require a epilepsy warning either and a health and safety warning intro screen is already implemented which will direct the user to either the instruction booklet or a website which underlines possible health risks associated with extended play of the game. As such, this product is compliant with the Americans with Disabilities Act and the International Game Developers Association (IGDA) Special Interest Group on Game Accessibility guidelines.

**ID# - AR1 – Accessibility Requirements**

**Description:** The product shall be approachable by any user with a common disability.

**Rationale:** Certain precautions and options must be taken into consideration in order to allow the product to be approachable by players with disabilities and in some countries, to be compliant with current disability laws.

**Fit Criterion:** 100 percent of users with common disabilities must be able to play the game with little to no difficulty.

**Acceptance Tests:** AR1 – Accessibility Requirements, UDR1 - User Documentation Requirements.

Upon purchase of the product, the user will be given various paperwork pertaining to the usage of said product. All paperwork would be in the language that is native to the country of purchase. This includes an instruction manual, which details the game’s rules, mechanics, possible health warnings as previously mentioned in section 16e, and the various available options. This manual will be written in such a way that the user will be able to understand all of the mechanics of product and will be able to quickly find important information quickly in the event that they need to review material about the mechanics of the game. This will also include recommended and minimum specifications for a PC user even if this product will be able to be played on most mainstream computer configurations including Apple, Linux, or Windows based systems.

Another section of the instruction manual will include installation instructions even if the game will be primarily found on DRM (Digital Rights Management) systems such as google play store or steam. There will be virtual documentation which will need to be reviewed and signed off by the user. These will involve terms and services for the online play of the game and for cosmetic purchases. These will outline the various rules and define possible instances that may result in a ban from online services or suspension of purchases. The instruction booklet will not be updated; however, the terms of service may need to be updated to accommodate future additions to the product.

**ID# - UDR1 - User Documentation Requirements**

**Description:** The product shall include terms-of-service and an instructional manual for the user.

**Rationale:** To legally protect the company and underline the consequences of a breach of contract to players that utilize the cosmetic store or play online. Also, to inform the users of the mechanics of the game and its installation and health and safety information that might be pertinent to certain players.

**Fit Criterion:** At any point of time, 100 percent of users must be able to reference the instruction manual and review the solutions that they might have regarding the mechanics of the product. Also, all users utilizing any of the online content must agree to the terms-of-service.

**Acceptance Tests:** UDR1 - User Documentation Requirements.

### Training Requirements

The user will not need any formal training in order to play the game. A basic understanding of device mechanics will be needed. For example, a user playing with a touch-capacitive screen must be able to know how to operate a touch-capacitive screen and a user operating a desktop computer must be able to operate their IO devices such as keyboards, mice, or gamepads. Installation is covered in the instruction manual, but most of the installation should be straightforward and defined through the DRM system that the user is utilizing. Besides that, all the technical training of the game will be covered in the tutorial provided in-game. If the user has basic literacy, context of all buttons used in the game are clear and concise. The product shall provide instruction to user’s in-game if the user has a basic understanding of the device that the game is installed on.

**ID# - TR1 -Training Requirements**

**Description:** The product shall not require any formal training from the user apart from the included tutorial from first-time users.

**Rationale:** The installation and operation of the product will be simple enough such that the user will feel no frustration in proceeding to play the game past the tutorial.

**Fit Criterion:** The user will be able to play the game within 20 minutes or less upon the completion of the tutorial and installation of the game.

**Acceptance Tests:** TR1 -Training Requirements.

## Look and Feel Requirements

### Appearance Requirements

The appearance of the product shall not incorporate any controversial images or involve and seizure-inducing images or animations. The images and colors shown must not affect gameplay in any sort of capacity. The design and color palette must be acceptable and attractive to users. Upon starting of the game, Bohn Jell Entertainment’s logo is displayed using animations, as well as, other companies that were involved in the development and the planning of the game. The client’s corporate branding is also included in the intro screen. The colors and animations used to display each corporation are consistent with each company’s signature colors and do not, in any way, deface any of the corporate brand’s image.

Different design elements and the colors come together to form the main theme of all the game screens which must not hinder gameplay. Animations between menu screens and on the game board are made to be fluid so that they will not hinder user gameplay. A prototype of all the colors, images, themes, and animations shall be compliant with the client’s requirements as well as any of the user purchasable cosmetic items.

**ID# - APR1 - Appearance Requirements**

**Description:** All design elements of the product shall not hinder the gameplay of the user and shall appeal to corporate branding standards.

**Rationale:** In order to maximize user amusement, the user must be able to focus on the gameplay in and of itself and the corporate branding of the client and associated development companies must satisfy their needs and not tarnish their brand as to ensure future client relations.

**Fit Criterion:** All associated companies must approve the image in which their brand is portrayed, and a test panel of users must accept the design of the product with a percentage of no less than 80 percent.

**Acceptance Tests:** APR1 - Appearance Requirements.

### Style Requirements

The style of the UI (User Interface) must appeal to the user and be acceptable for users of all ages. The style of the product and its packaging will reflect the theme and feel of the game, in that it must be playful and friendly, and yet exude the feeling of an incoming conflict. It must be approachable and yet reflect the ominous and dramatic feeling of an upcoming battle. With that being said, the style of the product must also tread carefully in that it should not be too overbearing as to dishearten a potential user from playing the game.

The packaging shall be professional and yet sprightly in such a way that the user will feel enticed to read it. The packaging for the physical copy of the game will reflect the overall feel and mechanics of the game and will be conservative in its sizing in order to adhere to other country’s specification requirements. As for users looking to download the game from a DRM system, the store pages will show images and videos as well as a summary which reflects the product’s gameplay. As a result, the product’s style must be made in such a way that at least 60 percent of potential users who see the packaging or the store page will be enticed to learn more about the product just at a first glance.

**ID# - STR1 - Style Requirements**

**Description:** The style of the product shall not affect the user’s experience and shall not overbear the user in any way.

**Rationale:** To ensure that the user will return to the product and feel a sense of immersion which will enhance the gameplay of the player.

**Fit Criterion:** 90 percent of all surveyed users must feel satisfaction and respond with relevance in terms of the style of the product.

**Acceptance Tests:** STR1 - Style Requirements, APR1 - Appearance Requirements

## Operational and Environmental Requirements

### Expected Physical Environment

The product will exist on whatever platform the user downloaded or installed the game onto. This may be any mobile device, desktop, or laptop computer systems. As such, this game will be able to be played while on the move or while the user is stationary in an indoor environment. Based upon this, the product will be utilized in any environment which is satisfactory for the device that is being used. This product shall be used in any sort of environment if it is within an acceptable boundary for the device. This may involve a lowlight situation or a completely controlled environment such as inside a household.

**ID# - EPE1 - Expected Physical Environment**

**Description:** The product shall be completely usable in any environment in which the device it resides, will properly function.

**Rationale:** The software should be able to work in any environment so long that the device that it is being played on, works without issue. That way the user can play the product wherever they feel like within the device’s environmental constraints.

**Fit Criterion:** So long as the device functions, the game must also be able to be played on 100 percent of the devices.

**Acceptance Tests:** EPE1 - Expected Physical Environment

### Requirements for Interfacing with Adjacent Systems

This product shall work on any legacy operating systems as well as on the most modern operating systems. This product will be compatible with older versions of windows from Windows XP to Windows 10 and will be available for all versions of modern-day versions of macOS and Linux. The product will also be compatible with all versions of android and IOS.

As a result of these compatibilities, the application being developed will run smoothly on any of these platforms and will be able to run with any overlays that are available from other programs. The operating system that will house the product will not only provide storage but will also allow the product to communicate with other user’s that have the product and with the online store that allows for cosmetic purchases. The data sent between users will solely be from online multiplayer play and will contain no data such that the communication will compromise the user’s personal information. Also, the communication with the store will only include data that is pertinent to the main user such as credit card information and prior and available purchases for the current user.

The multiplayer information will be updated during every round and the store information will be updated upon opening of the store or purchase of an item. Multiplayer and store information will be sent over a wireless internet connection and will be unavailable to users that play offline. Upon purchase of a cosmetic item on the online store, this data will be saved on the user’s computer so that the user will be able to access their cosmetics during offline play.

**ID# - RIAS1 - Requirements for Interfacing with Adjacent Systems**

**Description:** The product shall be compatible with all modern operating systems, current, relevant mobile devices, between users on different devices. and with any DRM software that it is being distributed from.

**Rationale:** This will allow more users to be able to enjoy the product from a multitude of devices and to be able to take advantage of the product’s features with little frustration.

**Fit Criterion:** 100 percent of all users, regardless of device, must be able to play the product with each other with no difficulty.

**Acceptance Tests:** RIAS1 - Requirements for Interfacing with Adjacent Systems.

### Productization Requirements

The product will be distributed in both a physical and virtual copy. The physical copy will be distributed using a CD, catering to users that do not have access to a reliable internet or for users who prefer to collect physical CDs. The physical CDs will only be supported by users that are using a standard computer platform. The downloadable version will be available to users of mobile devices and for computer users.

The installation process for the physical copy will be easy to understand as the user will only need to run the CD. From there, the InstallShield Wizard® installation utility will be activated and will guide the user through the installation process. While this is occurring, no additional input is needed for the user during the rest of the installation. For the downloadable version, the user will need to have an appropriate DRM software. This software allows for the management of digital licenses and will help prevent unauthorized redistribution of the product. Examples of supported platforms for DRM services are Steam on Windows, Linux, and MacOS computers, Google Play store on Android-based devices, and the Apple Appstore for Apple mobile devices. There are other examples which will be utilized such as Epic Game Store, GOG, and Uplay.

These services have their own installation processes which are relatively pain-free. The services will automatically allow for distribution of the software and save a library of games associated with the user. At any moment of time, the user can select which game they want to install where a download and installation process will automatically start. At the completion of installation, the user will be able to instantly access the game with no issue. The DRM services also allow for game developers to host their game stores which will attach to the in-game purchases made. These services will automatically keep track of the user’s purchases and will automatically update and install the user’s newly purchased cosmetic items. The product shall be accessible via a physical or a downloadable content and will feature redistribution protection through DRM systems. The product installation shall be simple and involve minimal input from an untrained user. As such, the product must be able to be installed within 5 minutes of the user purchasing the product.

**ID# - PRODR1 - Productization Requirements**

**Description:** The product shall require minimal user input to install onto the device of their choice and be distributable through physical and downloadable copies.

**Rationale:** By achieving this, the user will not be concerned with the installation at all and will instead focus on the content of the product.

**Fit Criterion:** The product must be able to be installed within 5 minutes of the user purchasing the product.

**Acceptance Tests:** PRODR1 - Productization Requirements.

### Release Requirements

The product shall be consistently updated anytime a bug fix is required. Also, the product shall be updated with new content in a period of 5 years or earlier depending on whether the client feels that the product has successfully served its purpose. During this period, any new findings on bugs or stability issues will be addressed in a timely manner. This will be done through a ticketing system which will keep track of any complaints or issues from users where a dedicated team of developers will suggest actions to be included in the next update. This may include, but not be limited to, any issues with compatibility of preexisting systems, any new possible issues with compatibility with a new operating system update, any compatibility with aforementioned gamepads, errors in code that may have been overlooked, and possible issues with performance due to a lack of optimization of the product with the platform it is running on. In terms of new content, new content will be added yearly or biyearly, in the form of new cosmetic shop items or new game additions. This will be primarily determined by the client. Any and all updates to the game shall not interfere with previous released version and will be thoroughly tested in a manner such that each issue that is being addressed will be completely rectified without affecting the game in other aspects.

**ID# - RELR1 - Release Requirements**

**Description:** The product shall be supported by the development team for a time period designated by the client, in which issues pertaining to the programming of the product will be addressed and additional content will be added.

**Rationale:** With regular updates, the game will not get stale to the user in that any issue that is brought up will be promptly fixed and will not affect the user’s experience in any way. All additional content will ensure that the player’s support for the game will be longer-lived.

**Fit Criterion:** Within a period of 5 years or otherwise outlined by the client, the product needs to be updated yearly or more occasionally if needed.

**Acceptance Tests:** RELR1 - Release Requirements.

## Cultural and Political Requirements

### Cultural Requirements

Regardless of gender, race, culture, religion, or sexual orientation, this product shall not offend anyone. Any newly added content will also take this requirement into effect. This product will keep all users in mind and is developed in such a way that any historical or modern association will be purely coincidental. This product is not associated with any controversial themes or groups and therefore will appeal to any audience of different countries.

Depending on which country the game is purchased, the game may consider events that involve popular, but not controversial, holidays that the country may observe. Any cosmetic purchases will also take this into heavy consideration and will be carefully developed in such a way that they will in no way be considered offensive. Any user-created content shared online will be reviewed and if for any reason a user-created theme is considered offensive, strict and rapid action will be given which will notify the user of why their content may be offensive and will be terminated.

**ID# - CULTR1 - Cultural Requirements**

**Description:** The product shall not offend any user regardless of the user’s gender, race, culture, religion, or sexual orientation.

**Rationale:** By achieving this, the product will not be subject to negative attention and will appeal to a larger target demographic.

**Fit Criterion:** The game will be accepted by 100 percent of users regardless of the user’s gender, race, culture, religion, or sexual orientation.

**Acceptance Tests:** CULTR1 - Cultural Requirements.

### Political Requirements

This product will not include any material borrowed from any other media properties unless authorized and intentions are clearly put on paper. As such, unless the other party has consented, this product will not infringe on any other company’s intellectual properties. The source code of the project will only be accessible to the development team and certain other individuals within the company of development. During development, all software developments pertaining to the product will solely belong to the company of development, Bohn Jell Entertainment.

The product must appeal to all regulations enacted by the countries of localizations and must not break any laws that may reference software and intellectual property laws. The game shall be reviewed and hopefully accepted by that country’s software rating board, such as ESRB (Entertainment Software Rating Board), before release of the product in that country. If any of the employees associated with the development of the project may be involved in controversial events, that employee will be subject to possible consequence which may involve termination of such employee. The company will swiftly react to any negative publicity with appropriate and professional responses.

**ID# - POLR1 - Political Requirements**

**Description:** The product shall satisfy the ethical code of conduct, company policy on employee conduct, and a country’s copyright and labor laws.

**Rationale:** This will avoid negative press about the game and will avoid legal issues that may arise with the development of the product.

**Fit Criterion:** The product must be compliant with 100 percent of company guidelines on conduct.

**Acceptance Tests:** POLR1 - Political Requirements, CULTR1 - Cultural Requirements.

## Legal Requirements

### Compliance Requirements

None of the images, animations, mechanics, or general content shall involve any contentious figures. All software relating to this product will comply with any of the laws of countries of sale and shall be reviewed by any pertinent media rating boards. This product will be required to comply with the Data Protection Act and all associated content of the product will be explicitly reviewed in order to not infringe on the intellectual properties of any other companies. Any of the content created for this game will be unique for this game.

All labor laws will be complied with. Employees working on the development of this project are subject to certain requirements that are outlined by country’s labor laws, such as the Fair Labor Standards Act. Any paperwork will be reviewed by an appropriate lawyer and any legal issues will be dealt with quickly. All work done with additional content and user-generated content will be subject to heavy scrutiny to minimize any potential legal issues. Copyright laws for every country of sale will be reviewed and taken into consideration.

**ID# - COMPR1 - Compliance Requirements**

**Description:** The product shall be compliant with the Data Protection Act, copyright laws, and the Fair Labor Standards Act, and shall be devoid of any controversial elements.

**Rationale:** To avoid lawsuits and controversy.

**Fit Criterion:** A legal team must 100 percent approve of the development and distribution of said product.

**Acceptance Tests:** COMPR1 - Compliance Requirements.

### Standards Requirements

This product shall be required to be approved by certain parties including the client and the project manager. Upon approval, the product shall be tested thoroughly and adhere to the company’s standard of quality. The product must be in working condition within a certain timeframe and must be devoid of any apparent issues in programming. Before distribution, the product must be ESRB compliant and must be of enough working order. The product must also be compliant with our own ethical code of conduct

**ID# - STANDR1 - Standards Requirements**

**Description:** The product shall adhere to the company’s standards of quality.

**Rationale:** This will ensure that Bohn Jell Entertainment will be seen in a positive light to existing and potential clients.

**Fit Criterion:** The product must comply with the company’s 10 standards of quality.

**Acceptance Tests:** STANDR1 - Standards Requirements.

## Requirements Acceptance Tests

Refer to the Fit Criterion of each ID# for the required acceptance tests.

### Requirements – Test Correspondence Summary



Table 2: Requirements - Acceptance Tests Correspondence

### Acceptance Test Descriptions

Refer to the fit criterion of the required ID# above for the description of the acceptance tests.

# Design

## Design Goals

*SV: Identify the important design goals that are to be optimized in the proposed design.*

*Content*

*Design goals are important properties of the system to be optimized, and which may affect the overall design of the system. For example computer games place a higher priority on speed than accuracy, and so the physics engine for a computer game may make some rough approximations and assumptions that allow it to run as fast as possible while sacrificing accuracy, whereas the physics calculations performed by NASA must be much more rigorously correct, even at the expense of speed.*

*Note an important difference between design goals and requirements: Requirements include specific values that must be met in order for the product to be acceptable to the client, whereas design goals are properties that the designers strive to make "as good as possible", without specific criteria for acceptability. ( Note also that the same property may appear in both a requirement and a design goal, so a design goal may be to make the system run as fast as possible, with a requirement that says any speed below a certain specified threshold is unacceptable. )*

Though most computer games place a priority on speed over accuracy, accuracy is actually the most important part of the equation for an online strategy game like Checker Wars. Our design goals include both, but the goal emphasizes accuracy of recording and graphical representation.

## Current System Design

*SV:* ***IF*** *the proposed new system is to replace an existing system, then the current system should be described here. Otherwise insert a brief statement that there is no pre-existing system.*

There is currently no existing system, as this is a novel online game.

## Proposed System Design

*This section will make heavy use of class diagrams, and also sequence diagrams where noted. However don’t overlook finite state, activity, communication, or other diagram types as needed for effective communication.*

This section will make heavy use of UML, class diagrams, and also sequence diagrams where needed.

### Initial System Analysis and Class Identification

*SV: Perform grammatical and similar analyses to identify the most import and obviously needed classes, and to organize them into an initial class structure. An initial class diagram is appropriate, containing few if any internal details.*

What will be needed to create the game are a CheckerWarsGame class to hold the Gameboard (which holds the players and towers for the game). This will extend from the tile class, which extends from the Tiles class(es). The Scenario class will contain the different available moves (Attack, Move, Victory), which is connected to the Player/Enemy class. This class is connected to the Piece class, which extends from the different Pieces (Pawn, Guardsman, Noble, and King). These connections are detailed in brief illustration below, which is further refined in section 27.

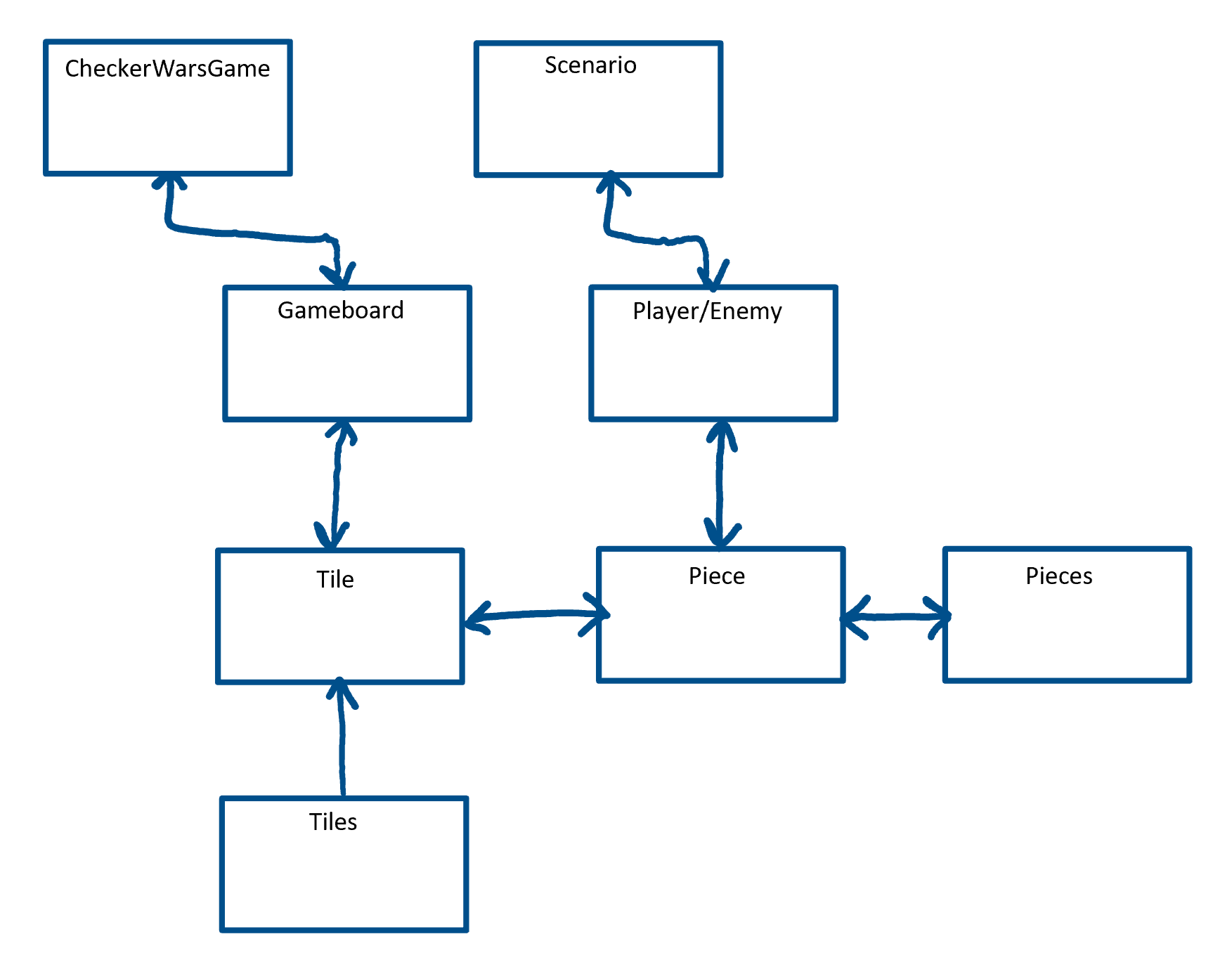


Figure : Initial Class Diagram

### Dynamic Modelling of Use-Cases

*SV: Insert sequence diagrams of (at least the most important) use-cases, as a means of identifying other needed classes.*

*Content*

*Include sequence diagrams of each important use-case here. This is a first step towards identifying preliminary objects. ( If the sequence diagram would be too big to fit, then it can either be broken down into pieces or a communication diagram can be used in its place. )*

The two main moves that will be happening within the game are the move scenario, and the victory scenario. It is important that the game server checks for the victory scenario after every turn is played, so that players will immediately know who has won, and also so in the future players will be able to receive updates of the potential moves that would cause a win or a loss (for training mode).

The move Scenario and the Victory Scenario are laid out as sequence diagrams below. The details are left up to the developers on how to implement the checking for each case. For the move scenario there are many different possible moves, so a lot will have to be done to check this condition.

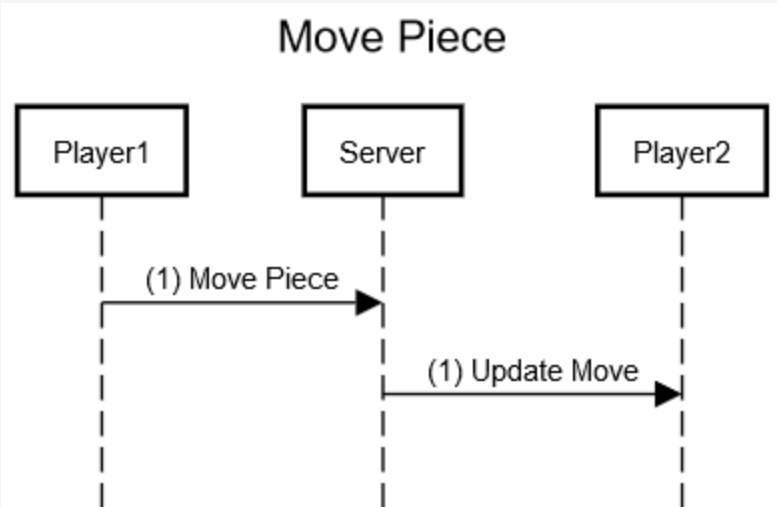


Figure : Move Piece Sequence Diagram

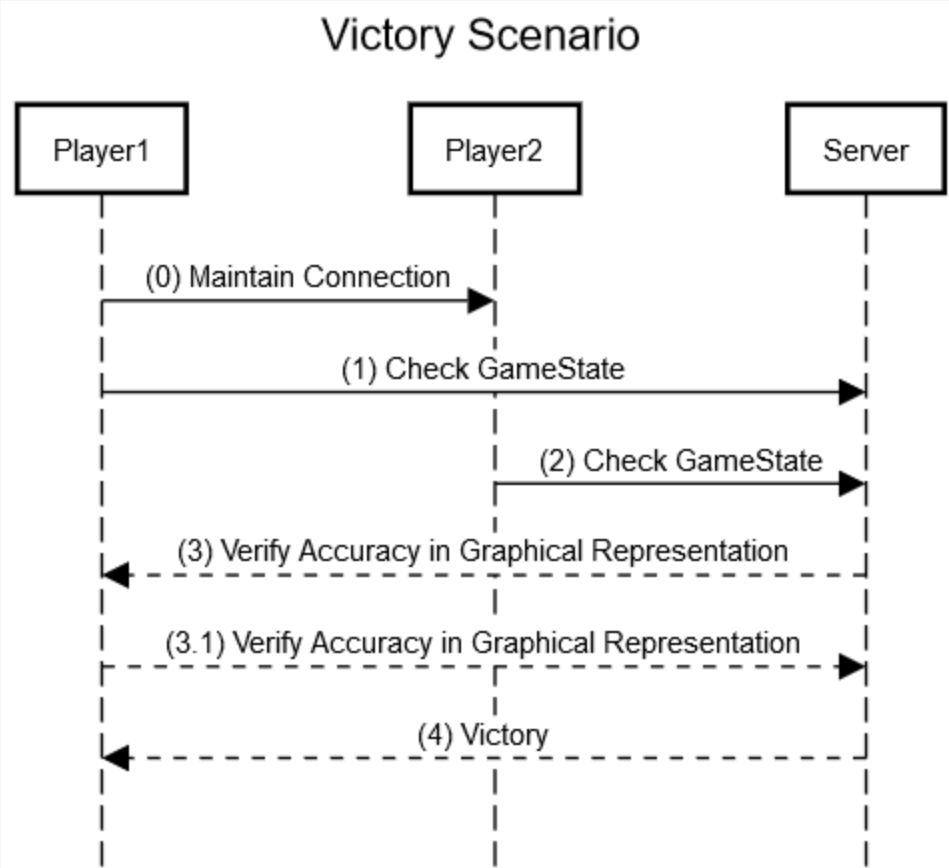


Figure : Victory Scenario Sequence Diagram

### Proposed System Architecture

*SV: Identify the Software Architecture to be applied to this project, such as Client-Server, Repository, MVC, etc., along with justification for the choice.*

We will use the client-server model for Checker Wars. In this model, the client connects to the web server (through CheckerWars.com or their respective game client), which controls the GUI layer, the Application Layer (which holds the logical representation of the game), and the Data Layer which gets data from the database and from web services.

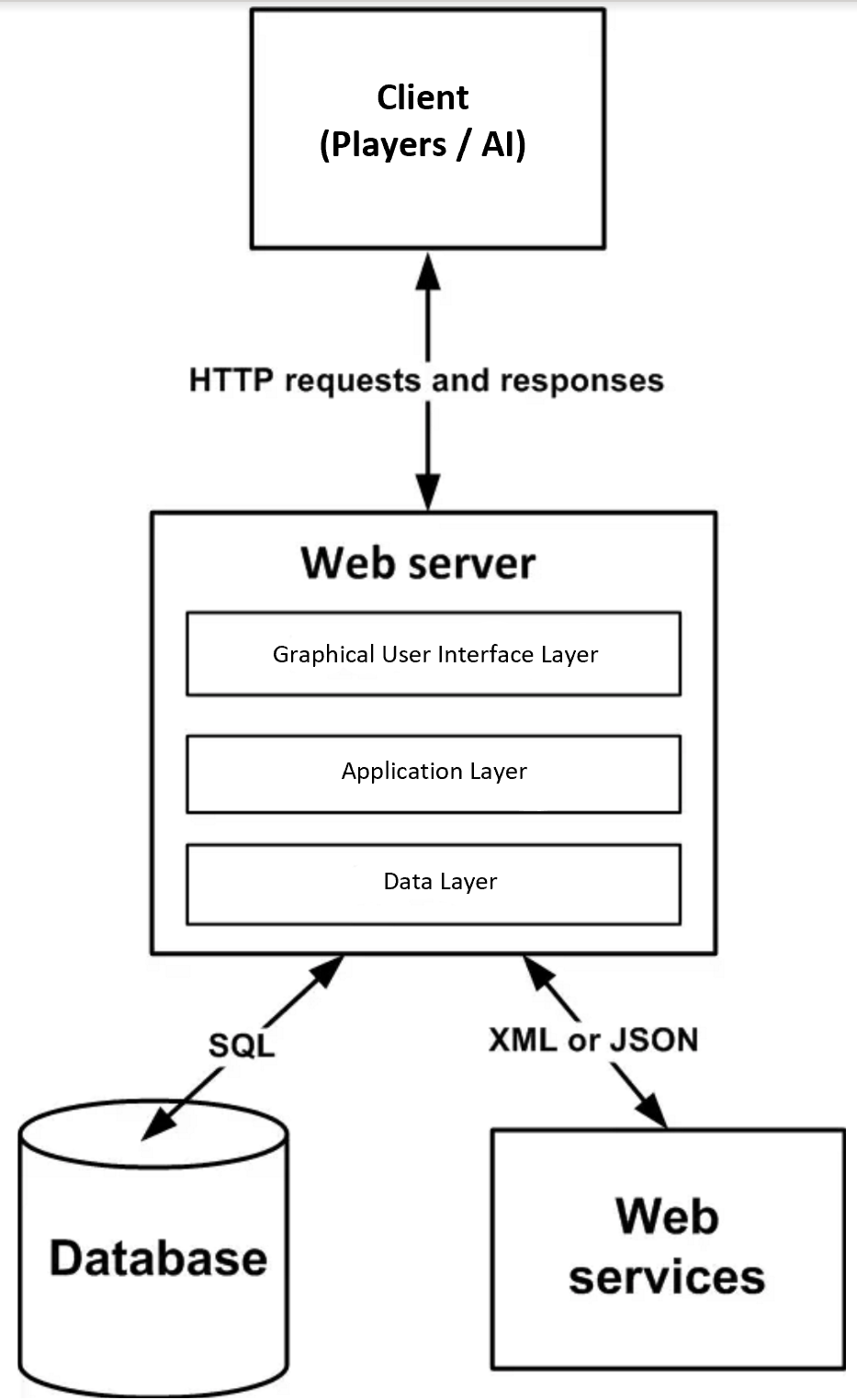


Figure : Client-Server Model for CheckerWars.com

### Initial Subsystem Decomposition

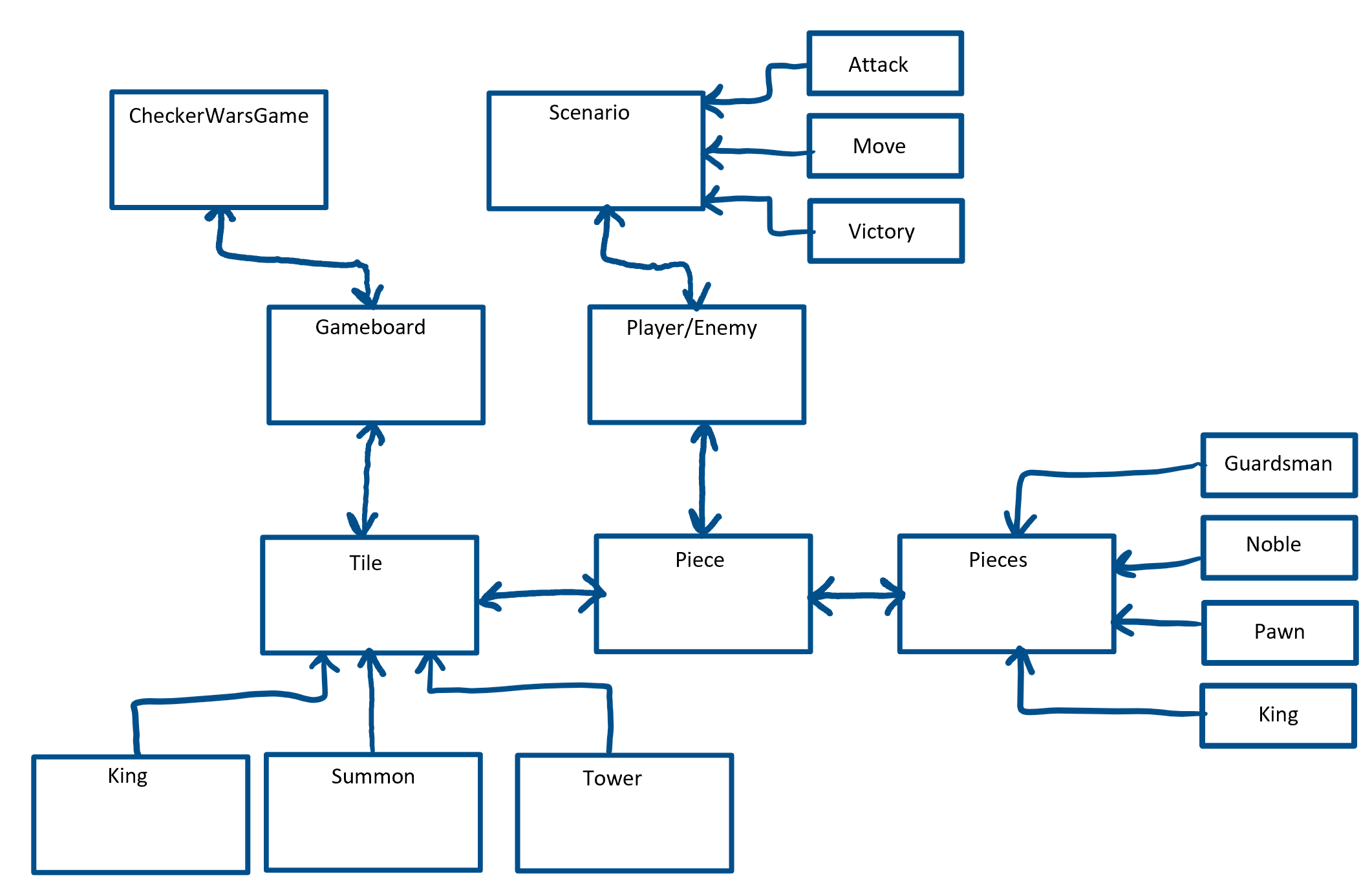
*SV: A slightly more detailed class diagram, showing the classes identified in sections 24a, 24b, and 0 above, partitioned into subsystems. For each subsystem provide a brief description of the subsystem, including its key responsibilities. There should still be few if any internal details.*

Scenario is now split up into the three possible moves that extend from it: Attack, Move, and Victory. The Scenario class is in charge of taking the move executed by the Player/Enemy class and executing that move on the server.

Pieces now extends Guardsman, Noble, Pawn, and King pieces. The Pieces class determines which pieces move and are represented on the gameboard for both users.

The Tile class now extends King, Summon, and Tower spaces. The tile piece is responsible for determining which spaces are occupied, which spaces are special tile spaces, and communicating this information with the server.

The CheckerWarsGame and Gameboard classes hold information about the games played by each of the users, and the state of the current gameboard.



## Additional Design Considerations

*SV: The sections listed here do not need to be presented in the order given, and may not all be relevant for any particular project. Those that are relevant can help identify additional classes that are needed as a result.*

### Hardware / Software Mapping

*SV: This is particularly important for distributed systems, such as those employing a client-server architecture. Use a deployment diagram to indicate which subsystems are mapped onto which piece(s) of hardware, and what communication subsystems need to be added to the system as a result.*

No hardware will be needed (except for the server infrastructure, which will be rented initially so that we do not have a big outlay on server costs before we know if the game will be successful.)

### Persistent Data Management

*SV: Document the classes and perhaps subsystems necessary to store persistent data when the system shuts down, and to restore that data when the system starts back up again.*

*Reiterate key data structures and information as necessary for the understanding of this design phase. Refer the reader back to the data dictionary in section I7c above to avoid undue repetition, while reviewing only the most relevant items here.*

The serves that hold all the data for the game will be located in data warehouses and will be online 24/7. The system will run in a distributed fashion, so that a single server failure will not result in the loss of data.

### Access Control and Security

*SV: Identify the access control and security concerns for this system, and the new classes and/or subsystems that must be added to handle those concerns.*

Users will never have direct access to the data on the servers. Only the developers will have access to that data, so access control and security is not a big concern. Standard industry security protocols will be sufficient.

### User Interface

*SV: Include a preliminary user interface design here, possibly as a rough sketch or other mockup, in order to identify additional classes needed to implement the interface.*

*The final user interface design will normally be developed by appropriate experts in that area. However it is appropriate to include an initial design here, including possibly a low- or high- fidelity sketch/mockup, in order to identify key classes necessary to implement the user interface, such as forms and dialog windows. It may also go towards addressing usability and/or look-and-feel requirements, and/or identifying other overlooked components.*

Below is a diagram of the layout of the board:

* T represent the Tower tiles.
* K represent the King tiles.
* S represent the Summon tiles.
* Red triangle represent the Guardsman.
* Purple square represent the Nobles.
* White dots represent the Pawns.

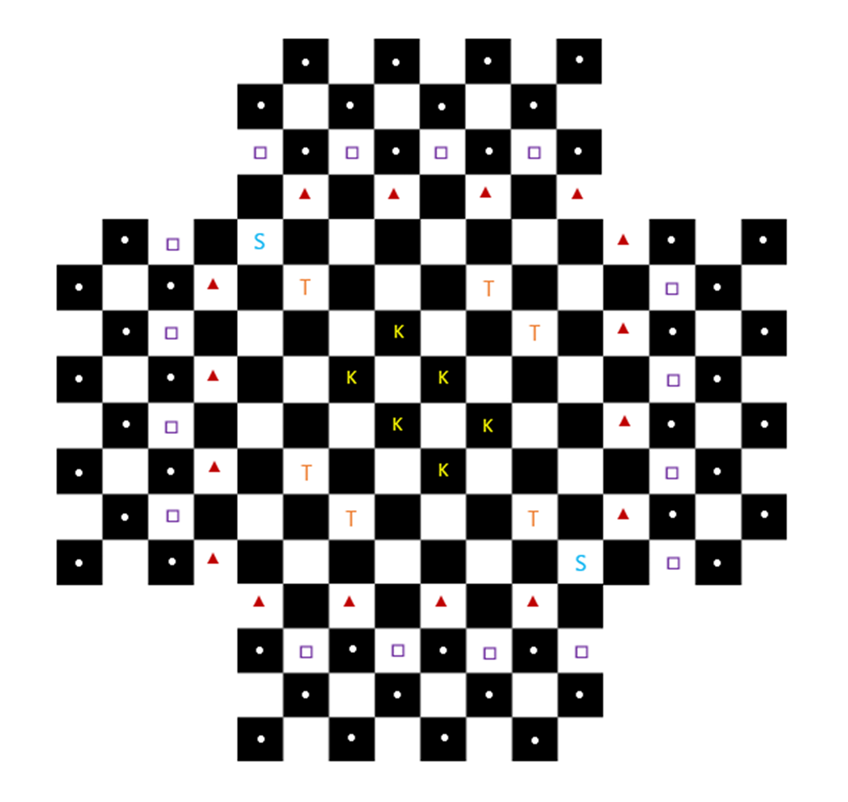


Figure : Checker Wars GUI

## Final System Design

*SV: Include here the final version of the overall system design, incorporating all the subsystems and classes added as a result of additional design considerations. Multiple diagrams may be needed, possibly starting with an overall package diagram showing all the different subsystems and the ( important ) classes contained within each one. Still not a lot of internal details.*

Below is the completed UML class diagram of the game detailing how all the pieces, players, and gameboard are connected together.

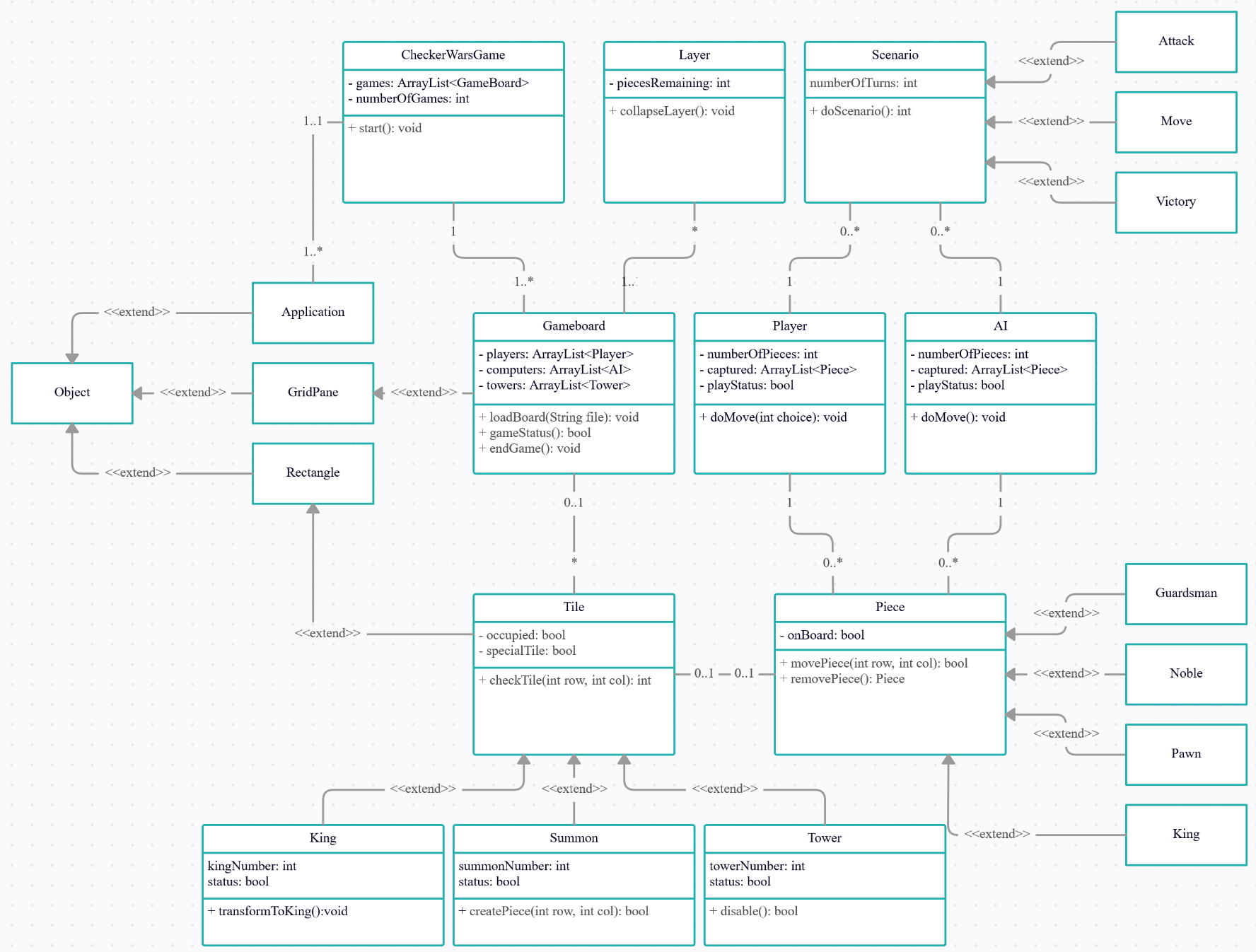


Figure : Final System Design (UML)

## Object Design

*This section documents the internal details of each class, to the extent that they can be designed at this time. Included should be the class interfaces ( public method signatures and responsibilities ) and constraints. It is probably best to break this section up into subsections corresponding to subsystems as documented above, and/or by ( Java ) packages if those are designed. It may also be appropriate to address additional design pattern considerations here, but not to the point of being redundant of previous documentation.*

*Certain methods, such as simple getters, setters, and constructors are not always documented, unless there is something special about them such as in the Singleton or Factory Method design patterns.*

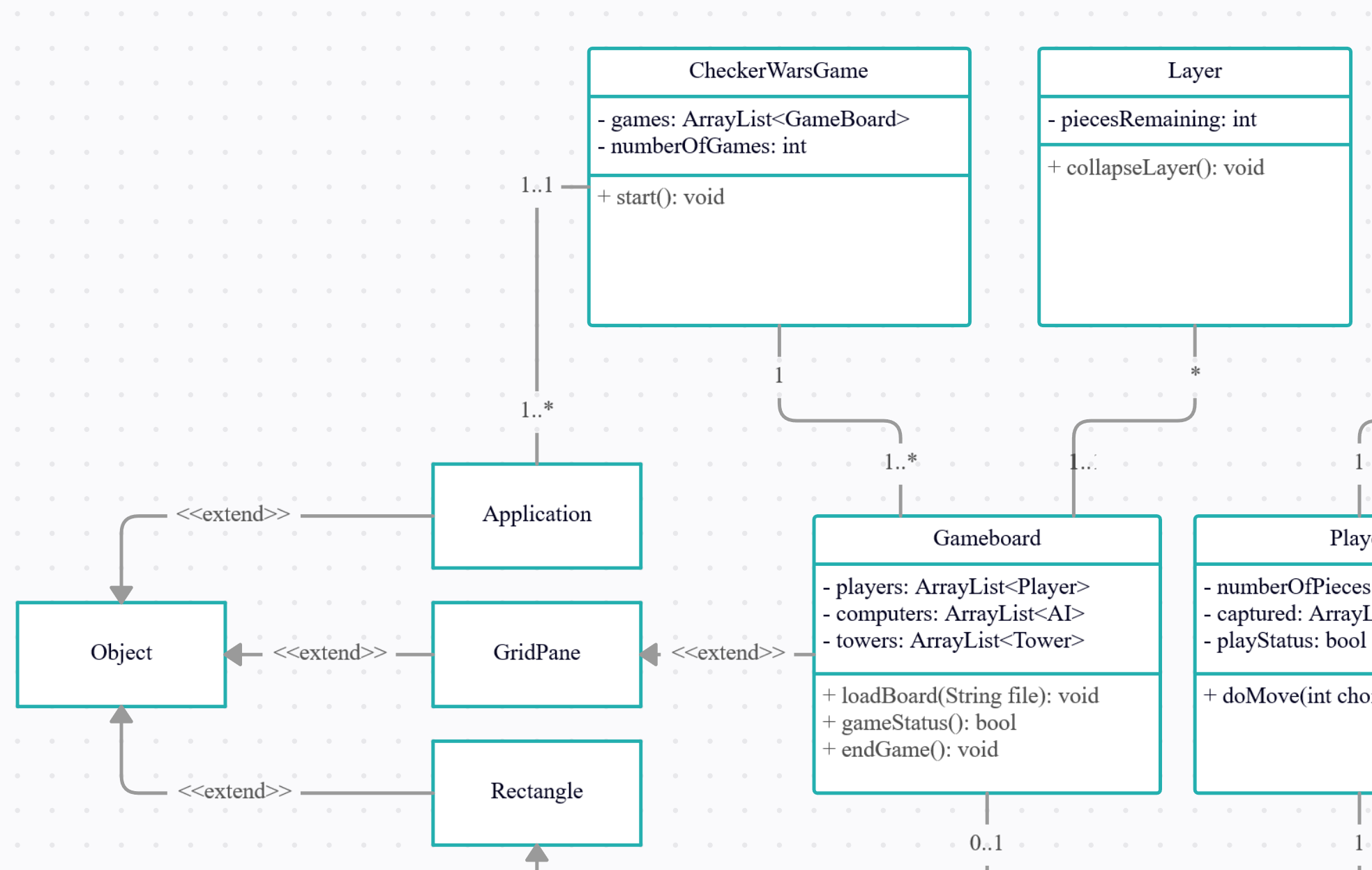


Figure : CheckerWarsGame and Gameboard classes

**CheckerWarsGame Class**: This class contains private members games, which is an ArrayList of GameBoards, and numberOfGames which is an integer counting the number of games in the match. It also contains the public method start, which will begin the game.

**GameBoard Class**: This class contains 3 private members, players which is an ArrayList containing the players of the game, computers which is an ArrayList containing the AI players (if any) of the game. And towers which is an ArrayList containing the Towers in the current gameboard. This class contains public methods, loadBoard, gameStatus, and endgame, which are self-explanatory. This class extends the GridPane class, which extends object. This GridPane represents the gameboard to the users playing the current game.

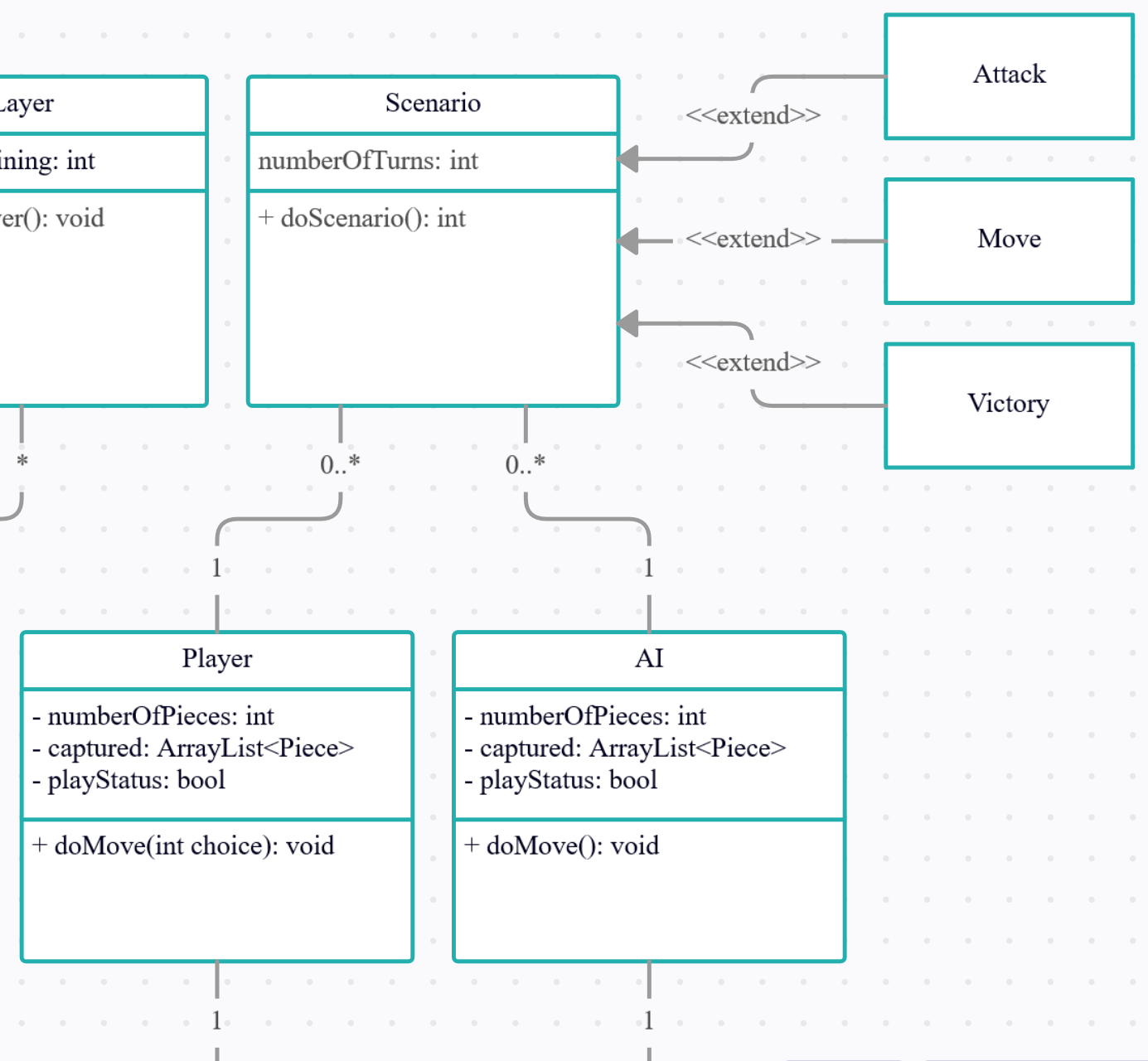


Figure : Scenario and Player/AI Classes

**Scenario Class**: This class contains the private member numberOfTurns which counts the number of turns played so far by each player. It also contains the public method doScenario which extends Attack, Move, and Victory.

**Player/AI Class**: The player and the AI class contains the private members numberOfPieces, captured which is an ArrayList of which pieces have been captured, and playStatus, which is a Boolean value representing which players turn it is in the game. It also contains the public method doMove which represents the move the player is executing.

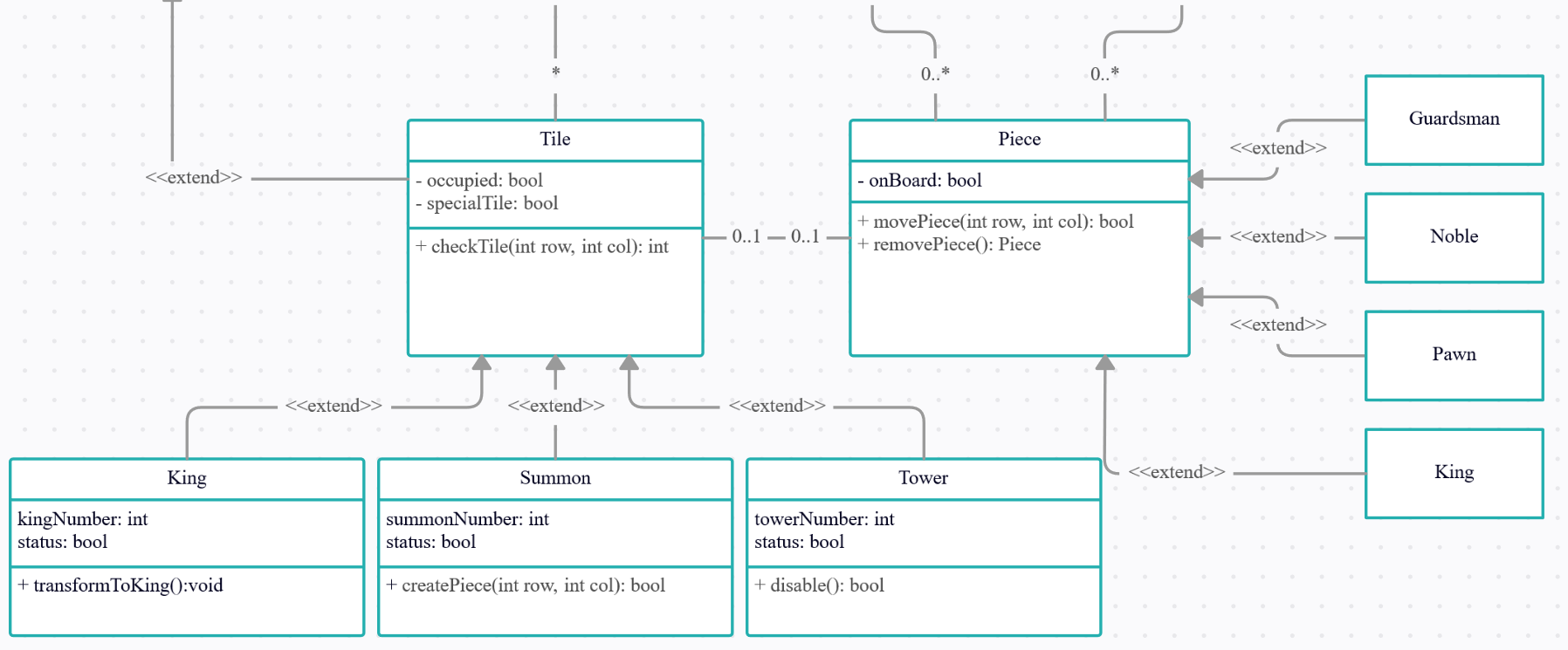


Figure : Tile and Piece Classes

**Tile Class**: The Tile class contains the private members occupied which is a Boolean value showing whether that tile is occupied by a piece, and specialTile, which is a Boolean value indicating whether the tile is a special tile piece (King, Summon, or Tower). The Tile class contains the public method checkTile, which checks what piece is in the current tile.

**King Class**: The King class extends to the Tile Class. It contains the private members kingNumber and status. It contains the public method transformToKing which transforms the piece to a king if the piece lands on the King tile.

**Summon Class**: The Summon class extends to the Tile Class. It contains the private members summonNumber and status. It contains the public method transformToSummon which, if a king or a noble land on a summon space the player is able to summon a new pawn piece on an adjacent black square.

**Tower Class**: The Tower class extends to the Tile Class. It contains the private members towerNumber and status. It contains the public method disable, which determines if the tile is enabled or disabled. If a king or a noble land on a Tower tile, they occupy that tower. If three towers are occupied by a single player, that player is victorious.

**Piece Class**: The piece class contains the private member onBoard which is a Boolean value that indicates if the piece is on the board. It contains the public methods movePiece and removePiece, which are self-explanatory. The Piece class extends the 4 possible pieces, which are Guardsman, Noble, Pawn, and King.

# Project Issues

## Open Issues

*SV: Issues that have been raised and do not yet have a conclusion.*

*Content*

*A statement of factors that are uncertain and might make significant difference to the product.*

*Motivation*

*To bring uncertainty out in the open and provide objective input to risk analysis.*

*Examples*

*Our investigation into whether the new version of the processor will be suitable for our application is not yet complete.*

*The government is planning to change the rules about who is responsible for gritting the motorways, but we do not know what those changes might be.*

*Considerations*

*Are there any issues that have come up from the requirements gathering that have not yet been resolved? Have you heard of any changes that might occur in the other organizations or systems on your context diagram? Are there any legislative changes that might affect your system? Are there any rumors about your hardware or software suppliers that might have an impact?*

When dealing with issues that we have had with our project, we took a bit more of a pragmatic approach. One of the main issues that we have discussed about involved the usage of user accounts and what that may entail. In modern times most games, if not all, have included some type of system dealing with user accounts or a digital rights management system. There are already many pre-established DRM programs that are very widely used such as Steam, Uplay, Battle.net, Epic Games Store, and Origin which may tailor to games from independent studios. On consoles, this is not an issue since every user that plays online on a console requires a user account which can be seamlessly add to the game, but on PC it’s a completely different story. If our game becomes mainly browser-based, if a user wants to keep track of their long-term record or progress in a single-based game should be retained in the case of having a user account. Also, if we decide to publish it on a DRM or non-DRM service, then we need to think of ways to link up all the current user accounts with the players of the games. In certain cases, this user account must be linked to any in-game purchases the user may have. We have not yet come to a consensus on how the design and implementation of the cosmetic purchases will work.

There have also been issues thinking up security concerns as a result. If the user has a user account, then there should be some type of security to protect the user’s data as well as their opponent’s data if they are playing online. Another issue with security would be game piracy. If the game ends up with a DRM service, this is not an issue. However, a browser-based game needs to have certain security measures in order to deter hackers which may alter gameplay and users that will potentially scrub data about the game off the website. Also, if we decide to allow players to save card numbers, we need to think of a way to protect that information from leaking to outside parties.

In terms of legal issues, the game may incorporate an in-game chat. As a result of that there needs to be some way to ban unruly players and potentially block harmful communication between players. At Bohn Jell Entertainment, we respect each and every player’s background regardless of gender orientation, race, or culture and we want to avoid potentially having a community of players that is harmful or antagonistic. We also need to make sure that our game does not use any copyrighted material without the owner’s written permission. This will apply to in-game cosmetics, as well as, any soundtracks or images used within the game.

Another issue to be addressed is how we are going to deal with optimization. Some companies may choose to optimize a game for a specific platform and port that version to another platform. We do not want to have players that are discouraged from playing Checker Wars on any platform that it’s released on just because the game does not run well. For the most part, the game will not consume much of a computer’s performance, however, we do not want to give an unfair advantage to computer players versus console players if we decide to make the game multi-platform, released on different platforms, and with cross-platform multiplayer, users on different platforms will be able to play with each other. As such we need to consider optimizations depending on the operating system or platform and general stability depending on system architecture.

Distribution of updates may be a unique issue as well. If there are general reports of system instability on a specific platform, we need to think about whether that may translate to other platforms as well.

## Off-the-Shelf Solutions

*SV: Discussion of products or components currently available that could either be incorporated into the new solution or simply used instead of developing ( parts of ) the new solution.  The distinction between sections 35 a, b, and c is subtle, and not very important.*

As said in the previous section, any user account information protection or purchase information can be protected if the game is released on a digital rights management program. Since all DRMs require the user to register an account in order to utilize their game purchases, using a DRM would circumvent all of the previous issues with the user-account and most of the issues dealing with security and in-game purchases.

For the most part, consoles utilize their own DRM systems, however on a PC, the user may choose to use an existing DRM service or utilize the browser version. A DRM generally has ways of dealing with hackers as well. If the user decides to use the browser version, instead of having to program all of the security and user account systems, we can outsource that programming to companies such as LEAN Security which specifically deals with online game security. Any security dealing with purchases made in-game on the browser version could also be outsourced to another company such as Norton or PayPal.

### Ready-Made Products

*SV: Products available for purchase that could be used either as part of a solution or instead of ( a part of ) a solution.*

*Content*

*List of existing products that should be investigated as potential solutions. Reference any surveys that have been done on these products.*

*Motivation*

*To give consideration to whether a solution can be bought.*

*Considerations*

*Could you buy something that already exists or is about to become available? It may not be possible at this stage to make this determination with a lot of confidence, but any likely products should be listed here.*

*Also consider whether some products must not be used.*

**User Accounts, In-game Purchases, Security:** Any DRM solution such as Steam, Uplay, Epic Games Store, GOG, Battle.net, Bethesda Games Store, Origin

By using a DRM service for PCs or Macs, this would significantly circumvent many issues. However, DRM services are not a fool-proof way to deter pirating of games, requires first-time users to download a service that they may not otherwise use, and may cause the price of the game to rise due to the fact that a DRM service will take a specific percentage of all game and in-game purchases. Also, by using a DRM service, we are giving up total control of distribution and direction of the game.

**Online Browser Game Security:** Lean Security, Norton, PayPal

By using these systems, this would reduce the need to prepare most programming for in-game purchases and potentially user accounts. However, effectiveness of these services is questionable at best. Most of these services would contain all the user’s data so if that company is involved in some sort of controversy, it could affect Bohn Jell Entertainment’s image and vice versa.

**Console In-game Purchases:** Sony PlayStation Store, Xbox Game Store, Nintendo Store

For the most part, these services are fool proof. They would deal with in-game purchases and these consoles already require a user to create an account. There would additional cost dealing with publishing on the respective consoles much like the DRM services.

### Reusable Components

*SV: Similar to 35a, but for components such as libraries or toolkits instead of fully blown products.*

*Content*

*Description of the candidate components, either bought from outside or built by your company, that could be used by this project. List libraries that could be a source of components.*

*Motivation*

*Reuse rather than reinvention.*

Any program design or libraries that deal with the project’s compatibility with any outside programs such as DRM may be reused in future projects. Also, if instead we design a system for dealing with user accounts, in-game purchases, and security may be reused. If we decide to release a sequel, expansion, or downloadable content for the game, the main games mechanics may be reused with additional features added on.

### Products That Can Be Copied

*SV: Products that could legally be copied would typically be past projects developed by the same development group, provided there were no restrictions that would prevent their reuse.*

*Content*

*List of other similar products or parts of products that you can legally copy or easily modify.*

*Motivation*

*Reuse rather than reinvention.*

*Examples*

*Another electricity company has built a customer service system. Its hardware is different from ours, but we could buy its specification and cut our analysis effort by approximately 60 percent.*

*Considerations*

*While a ready-made solution may not exist, perhaps something, in its essence, is similar enough that you could copy, and possibly modify, it to better effect than starting from scratch. This approach is potentially dangerous because it relies on the base system being of good quality.*

*This question should always be answered. The act of answering it will force you to look at other existing solutions to similar problems.*

At the core, the idea and mechanics of checkers is not copyrighted since the game is thousands of years old so any ideas pertaining to the game is okay to pursue. Also, during previous projects, we have already developed a version of checkers which is working. The issue is we would need to overhaul the core functionality and refine the game to make it appeal to more audiences. Also, the program was written in C++ which would need to be translated into a more appropriate language. Mobile platforms, consoles, and libraries that deal with compatibility of web design programming languages are more compatible with a programming language like Java. While porting over the game from one programming language to another, improvements to the game’s code could be made at the same time.

For the most part, our program does not have any competitors with which to base our programming on since a multiplayer, checkers-based game has not been developed before and as such we cannot borrow any ideas. With that being said, certain ideas can be borrowed from RTS, real-time strategy, games like Warcraft or from grid-based games like Crypt of the Necrodancer.

As for the images and theme of the game, we would have to develop those from scratch even though we could use a game like 3D Tabletop simulator for inspiration.

For the most part, by developing our own assets we do not have to worry about copyright, however, by using other games as inspiration, we could cut the production of the game by up to 20 percent or more.

## New Problems

*SV: The proposed new system certainly has its benefits, but it could also raise new problems.  It is a good idea to identify any such potential problems early on, rather than being surprised by them later.*

### Effects on the Current Environment

*SV: Could the new system have any adverse effects on the working environment, e.g. the way people do their jobs?*

*Content*

*A description of how the new product will affect the current implementation environment. This section should also cover things that the new product should not do.*

*Motivation*

*The intention is to discover early any potential conflicts that might otherwise not be realized until implementation time.*

*Examples*

*Any change to the scheduling system will affect the work of the engineers in the divisions and the truck drivers.*

*Considerations*

*Is it possible that the new system might damage some existing system? Can people be displaced or otherwise affected by the new system?*

*These issues require a study of the current environment. A model highlighting the effects of the change is a good way to make this information widely understandable.*

Since this project is our first project, we expect that this game will present new problems in the workplace. Although we have general plans of how development will go with this project, overall, our process has not been tried and tested. As such we expect that at first, we will have some issues and changes that will be made to the process of development during the actual development of this first product. The workplace will definitely be more stressful because extra effort will need to be taken in order to find out a proper way to maximize efficiency of the workplace.

In terms of budget and budgeting, we are seriously considering hiring a personal accountant and a financial planner. Currently, no one on the team has a solid background in finances and bookkeeping so we may need to outsource all of this to a third party. This should be no issue since our company is still in infancy, but there is still some anxiety in thinking that this game will be our own sole source of income after development. As such we need to plan accordingly preparing for the worst case and cutting our losses and hoping for the best. Our employee pay will be of utmost importance, but there is an inherent risk that if the game does badly, then considering bankruptcy may not be such a bad choice since our financial portfolio is not diversified enough. This is where a competent financial planner and an accountant could rectify this possible issue.

As of right now, we expect that reaching the deadline should be no problem with the current set of employees, however, there may end up being setbacks during development. This could mean that a rather colossal bug may end up being found or that more staff would need to be hired. As a result, we may have to resort to hiring more staff or a specialist if any major issues occur.

Changes in scheduling may be required in order to accommodate sick days or vacations that an employee may require especially with the current pandemic. In order to deal with this, we have made our development process with a much more conservative estimate. Theoretically, we could finish the total development project up to 1 month quicker than our current deadline, but that is assuming that the development process is without any issues.

Even with a conservative deadline with development, there will still be possible issues with distribution. Overall, we project that the majority of game purchases will be made digitally, however, for those wishing to purchase a physical copy, an entire distribution network will need to be sourced in order to get our product on store shelves.

### Effects on the Installed Systems

*SV: Could the new system have any adverse effects on other hardware or software systems?*

*Content*

*Specification of the interfaces between new and existing systems.*

*Motivation*

*Very rarely is a new development intended to stand completely alone. Usually the new system must coexist with some older system. This question forces you to look carefully at the existing system, examining it for potential conflicts with the new development.*

Since we are implementing our product on a variety of platforms, our product needs to be of the utmost quality and consider all users’ computer systems and platforms. In terms of consoles and mobile development, optimization of the game on these platforms should not be an issue. Since over 95 percent of all users are using a more recent system update on these platforms and since generally speaking, backwards compatibility of older updates is generally not an issue, we should not have any issues on these platforms. Website development of the product should be rather straightforward. However, it is a bit of a different story with PC and mac development.

In terms of mac and PC compatibility, looking at other applications, certain applications are compatible with only the newest iterations of the operating systems and others are only compatible with older iterations of operating systems. In order to ensure a maximum coverage of users we will support Mac users with the Mountain Lion macOS update and newer and for windows, windows 7 and later. This would roughly cover all users with updates from 2015 to now which would be a vast majority of users. In terms of Linux users, this application work with all Linux kernels.

For the future, there may be an issue with using the application with an ARM-based CPU, but overall, it should not be an issue. This is brought up because Apple may decide to switch to ARM based CPUs. Since our application will not require heavy performance, working on an ARM architecture CPU should not be an issue despite the extra requirement that the ARM would have to use an x86 emulator. Despite this if ARM chipsets become more widely used, we may have to apply a future patch to optimize the game for this architecture.

### Potential User Problems

*SV: Could the new system have any adverse effects on the users of the software? Could users possibly have a negative response to the new system?*

*Content*

*Details of any adverse reaction that might be suffered by existing users.*

*Motivation*

*Sometimes existing users are using a product in such a way that they will suffer ill effects from the new system or feature. Identify any likely adverse user reactions, and determine whether we care about those reactions and what precautions we will take.*

For the most part, we will try to optimize our game to all intended platforms. This will take extensive and arduous development, but in the end, should end up with no user complaints.

Users need a way to contact a customer service team if any issues arise which could be rectified.

The most important issue that we will try to address is the game refinement. There may be some unseen, potential issues with stability and performance as our sample size during testing is not that major.

Also, we are planning on requiring all users to register an account or have an account on a DRM service that the game is available on. This would mean that we would alienate potential customers that may not want to create an account or download a DRM service in order to play the game.

Another issue is that users may affect each other while playing a game. Certain boundaries should be made to limit the ability of a user to harm another user or obtain another user’s information.

Lastly, a potential issue may arise with cosmetics or with certain graphics. There are over 50 million people in the world with epilepsy and many more that may be adversely affected by flashy images. As such much care must go into the creation of new visuals and a warning screen should be implemented to warn potential users of harm from visuals or long-term play.

### Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

*SV: Are there any ( physical ) limitations in the expected environment that could inhibit the proposed product?  ( e.g. weather, electrical interference, radiation, lack of reliable power, etc. )*

*Content*

*Statement of any potential problems with the new automated technology or new ways of structuring the organization.*

*Motivation*

*The intention is to make early discovery of any potential conflicts that might otherwise not be realized until implementation time.*

*Examples*

*The planned new server is not powerful enough to cope with our projected growth pattern.*

*The size and weight of the new product do not fit into the physical environment.*

*The power capabilities will not satisfy the new product’s projected consumption.*

*Considerations*

*This requires a study of the intended implementation environment.*

As said in a section above, our development process is going to have to remain a bit more fluid since this is our first product and no adamant plan has been made. To mitigate this, we could seek outside help or advising from third parties. Since the game is being made to be online, multiple servers will need to be implemented in order to handle online traffic. Also, internet service stability may or may not be a potential issue in the future. Other companies may plan to have servers in a multitude of states or countries, but as of right now, Bohn Jell Entertainment will not have that luxury, so contingency plans and redundancy may need to be built-in in case internet service or power goes out.

The number of servers to handle the online traffic may not be enough and as such may require outsourcing.

As it stands, there may not be enough employees in order to handle all jobs efficiently.

Distribution of the physical product has not been planned yet and will require extensive thought on which stores we can partner with.

Also, the location of the company may be an issue to certain employees that will commute to company grounds.

### Follow-Up Problems

*SV: Basically any other possible problems that could occur.*

*Content*

*Identification of situations that we might not be able to cope with.*

*Motivation*

*To guard against situations where the product might fail.*

*Considerations*

*Will we create a demand for our product that we are not able to service? Will the new system cause us to run afoul of laws that do not currently apply? Will the existing hardware cope?*

*There are potentially hundreds of unwanted effects. It pays to answer this question very carefully.*

If the product performs well, then we may have a potential issue with online stability due to either server problems or due to the application not working well with too many clients connecting to the server. Especially since we plan on the game allowing for cross-platform play.

DDOS, distributed denial of service, attacks are very common and as such, we may not be able to deal with these without outside help since our staff may or may not have experience with online security.

The advertising budget may not be suitable to get a quick influx of new players. As a result of a lack of advertising, which is often one of the biggest expenses for large scale products, we may be losing out on many potential customers.

As a result of a lack of an advertising budget, most likely, the initial influx of users may be slow and the popularity of the game as a result will slowly rise.

Since this product will be our first and only product for a while, spending will need to be very conservative and as such may bring unintended effects on employee morale.

Localization could also be a potential issue depending on the quality of translation of the product. As a result, certain messages within the product may confuse the users.

## Migration to the New Product

*SV: This section only applies when there is an existing system that is being replaced by a new system, particularly when data must be preserved and possibly translated / reformatted.  Otherwise just write "Not Applicable" under section 38 and remove sections 38a and 38b.*

Not Applicable

## Risks

*SV: Consideration of the potential risks that could cause the project to fail / underperform.*

*All projects involve risk—namely, the risk that something will go wrong. Risk is not necessarily a bad thing, as no progress is made without taking some risk. However, there is a difference between unmanaged risk—say, shooting dice at a craps table—and managed risk, where the probabilities are well understood and contingency plans are made. Risk is only a bad thing if the risks are ignored and they become problems. Risk management entails assessing which risks are most likely to apply to the project, deciding a course of action if they become problems, and monitoring projects to give early warnings of risks becoming problems.*

*This section of your specification should contain a list of the most likely risks and the most serious risks for your project. For each risk, include the probability of that risk becoming a problem. Capers Jones’s Assessment and Control of Software Risks (Prentice-Hall, Englewood Cliffs, N.J., 1994) gives comprehensive lists of risks and their probabilities; you can use these lists as a starting point. For example, Jones cites the following risks as being the most serious:*

*• Inaccurate metrics*

*• Inadequate measurement*

*• Excessive schedule pressure*

*• Management malpractice*

*• Inaccurate cost estimating*

*• Silver bullet syndrome*

*• Creeping user requirements*

*• Low quality*

*• Low productivity*

*• Cancelled projects*

*Use your knowledge of the requirements as input to discover which risks are most relevant to your project.*

*It is also useful input to project management if you include the impact on the schedule, or the cost, if the risk does become a problem.*

* Not enough servers to handle the initial volume of users. Based upon our advertising budget and game genre popularity, probability would be low of it becoming a major issue.
* Filing for bankruptcy as a result of low sales and insufficient budget. Probability would be very high of it becoming a major issue.
* Understaffed as a result of poor planning would have a medium probability of it becoming a major issue since we have already implemented conservative due dates.
* Low quality within the product causing user frustration. Probability would be low because all the quality assurance would be handled by all senior members of the programming teams.
* Slow and gradual influx of users leading to slow revenue. Probability would be low of it becoming a major problem.
* Encountering a large bug that may hinder project progress. The probability of it becoming a major issue may be high or low depending on whether the source of the bug is known or not.
* Encountering a legal issue dealing with copyright would have a low or high probability of it becoming an issue depending on how important the graphic or mechanic is to the overall design of the game.
* Language and translation issues is a potential risk, however, the probability of it becoming a major issue would be low.
* Bad performance due to system incompatibility or due to bad optimization would be a high probability of it becoming a major issue.
* Product instability with the system’s operating system would have a high probability of it becoming a really bad issue.
* A potential hacking would be a low to very high probability of it becoming a major problem. At the bare minimum, one user’s game may be compromised due to an opposing player hacking the gameplay mechanics. At the worst, user information is leaked by the web-based version or physical version of the game.
* Problems in the distribution chain of the product. The probability of this becoming a major problem is low since the digital version should not have any problems at all and we project that most users won’t use the physical version.
* Another company infringing on our own property would be a low to high risk since a larger company would have no issues with distribution and advertisement which would lead to a disadvantage to Bohn Jell Entertainment, and a small company would generally not affect the situation at all.
* Management malpractice and controversy would be a high probability of this becoming an issue because it may cause the understaffing problem to be an extra issue.
* An employee calling in sick or taking a longer absence of leave would be a low probability of becoming an issue since our deadlines already account for this.
* Low productivity would be a high probability of becoming a problem since this would hinder development and also it would contribute to a lesser workplace environment.
* General logistical issues such as scheduling conflicts would have a high probability of becoming a major issue since it would affect productivity and would increase the amount of stress in the workplace.
* Bad budgeting would have a high probability of becoming an issue since it would not only hinder staffing, but it may also cause a large general decrease of efficiency in the workplace.
* Low advertising budget would have a high probability of becoming an issue since much of the target audience would remain untapped since they would not know if Checker Wars even exists. Also, it may give certain users the wrong idea of the game if it is marketed falsely.

## Costs

*SV: An estimate of what it will cost to complete this project.  Think not only in terms of dollars, but also time, resources, lost opportunities, etc.*

*For details on how to estimate requirements effort and costs, refer to Appendix C Function Point Counting: A Simplified Introduction*

*The other cost of requirements is the amount of money or effort that you have to spend building them into a product. Once the requirements specification is complete, you can use one of the estimating methods to assess the cost, expressing the result as a monetary amount or time to build.*

*There is no best method to use when estimating. Keep in mind, however, that your estimates should be based on some tangible, countable artifact. If you are using this template, then, as a result of doing the work of requirements specification, you are producing many measurable deliverables. For example:*

*● Number of input and output flows on the work context*

*● Number of business events*

*● Number of product use cases*

*● Number of functional requirements*

*● Number of nonfunctional requirements*

*● Number of requirements constraints*

*● Number of function points*

*The more detailed the work you do on your requirements, the more accurate your deliverables will be. Your cost estimate is the amount of resources you estimate each type of deliverable will take to produce within your environment. You can create some very early cost estimates based on the work context. At that stage, your knowledge of the work will be general, and you should reflect this vagueness by making the cost estimate a range rather than a single figure.*

*As you increase your knowledge of the requirements, we suggest you try using function point counting—not because it is an inherently superior method, but because it is so widely accepted. So much is known about function point counting that it is possible to make easy comparisons with other products and other installations’ productivity.*

*It is important that your client be told at this stage what the product is likely to cost. You usually express this amount as the total cost to complete the product, but you may also find it advantageous to point out the cost of the requirements effort, or the costs of individual requirements.*

*Whatever you do, do not leave the costs in the lap of hysterical optimism. Make sure that this section includes meaningful numbers based on tangible deliverables.*

* Based upon preliminary results and based upon average costs for independent video games our initial estimate for the total cost of development was $80,000 at the minimum to $160,000 at the higher end.
* We estimated that the total time of development would be 2 months at the maximum which is a rather conservative estimate which would consider any potential major setbacks and employee grievances during development. If we break this estimate down further, we estimate that it would take roughly 3 weeks to work out the main game mechanics, an extra 2 weeks to deal with programming of user account systems and purchase systems, and the rest 3 weeks dealing with distribution, optimization, and general logistics such as advertising, hiring of new employees, and budgeting.
* Since our advertising budget is small compared to the loss of opportunity would cost an estimated $10,000 to $30,000 in profits. This would be due to not hitting the full potential of the target audience.
* Independent of game development, the addition of new servers at the company to accommodate more online players would cost around $15,000 to $10,000.
* In terms of function points, development of the mechanics of the game cost 2500 function points, 1000 for game logic, 1000 for system compatibility and optimization, and 500 for user interface design. The in-game marketplace for cosmetics took an additional 1000 points and the client-server system design took 600 points. Cross-play implementation was 800 points. This led to a total of 4900 function points.
* Pay per employee ranged from $1,000 from the most basic employees up to $10,000 for the software developers involved in the project.
* Additional cost that were dealt with involved rent and utilities costs which averaged between $2,000 to $4,000 per month.
* Costs to utilize DRM systems ranged between $100 for Steam to $20,000 for PSN, PlayStation network, which included a dev-kit for $15,000 and right-to-publish for $5,000. The website version costed approximately $200 to $300 as website creation costed between $100 to $200 and upkeep costed $50 dollars a month. For physical copies, 300 were made and with an average cost of $27 to produce and ship to physical store, ended up costing $8,100.
* To deal with potential legal issues, a third-party legal team was hired which cost around $2,000 to $4,500 which was set at an hourly pay.

## Waiting Room

*SV: This is a place to record ideas or wishes that will not be included in the current release of the product, but which might be worth reconsidering at a later date.*

*Requirements that will not be part of the next release. These requirements might be included in future releases of the product.*

*Content*

*Any type of requirement.*

*Motivation*

*To allow requirements to be gathered, even though they cannot be part of the current development. To ensure that good ideas are not lost.*

*Considerations*

*The requirements-gathering process often throws up requirements that are beyond the sophistication of, or time allowed for, the current release of the product. This section holds these requirements in waiting. The intention is to avoid stifling the creativity of your users and clients, by using a repository to retain future requirements. You are also managing expectations by making it clear that you take these requirements seriously, although they will not be part of the agreed-upon product.*

*Many people use the waiting room as a way of planning future versions of the product. Each requirement in the waiting room is tagged with its intended version number. As a requirement progresses closer to implementation, then you can spend more time on it and add details such as the cost and benefit attached to that requirement.*

*You might also prioritize the contents of your waiting room. “Low-hanging fruit”—requirements that provide a high benefit at a low cost of implementation—are the highest-ranking candidates for the next release. You would also give a high waiting room rank to requirements for which there is a pent-up demand.*

Quite a few ideas for future implementation of the project were considered while creating this product.

* Creating a 3D version of the project with an upgraded graphical interface.
* Allowing user-generated content in order to change certain aspects of the product including the look of Checker Wars and/or the mechanics behind the game.
* Creating a full single-player mode complete with story points, unique scenarios, and potentially specially made heroes which would have their own set mechanics and skills which they can use to change up the game. Possibly with fully voiced dialogue.
* A hero mode in which a multiplayer match may include special characters which have special skills and may or may not be randomly selected by players.
* Possible AI intervention to balance out player versus player games.
* New pieces and tiles that would perform their own special tasks.
* More premade maps which will be balanced and each of different sizes.
* A user-creation mode where a user can create their own levels complete with their own scenarios.
* A randomizer which would completely randomize aspects of the map to allow players to have a unique experience every time they play.
* Loot boxes in which the user can purchase a box in the store to obtain rare cosmetics.
* A marketplace in which users may perform transactions pertaining to rare cosmetics.
* A ranked mode that will keep track of the players overall record and rank them accordingly.
* Special events which will have their own activities to obtain rare cosmetics.
* Maps which may have multiple areas within areas.
* AI-driven enemies which may attack either player.
* Item-based gameplay in which the players can allow items to appear in-game which may change gameplay.
* Unlockable secrets and Easter eggs which will give more incentive to players who play the game longer or for players who discover other aspects of the game.
* Achievements and unlockables for return players of the game.
* Incentives for players who come back to the game after a long period of time.
* Seasons in which game balance may change.
* An RPG, role-playing game, mode where certain pieces may have a level which will increase with each piece that piece removes. These pieces would become stronger or tougher after each level-up.
* Free incentives to long-term players depending on how many games have been played.
* A puzzle-mode which would have scenarios where the pieces are setup and the players would have to determine the best course of action to clear the board.

## Ideas for Solutions

*SV: When developing requirements only, it is not the role of the business analyst to dictate the implementation of the solution.  However they can pass along any ideas they have here as suggestions to the developers.  For CS 440 this report includes system and object design, so this section would make suggestions for implementation and testing that would come after design, such as the use of a particular language, IDE, library, or other tools.*

*When you gather requirements, you focus on finding out what the real requirements are and try to avoid coming up with solutions. However, when creative people start to think about a problem, they always generate ideas about potential solutions. This section of the template is a place to put those ideas so that you do not forget them and so that you can separate them from the real business requirements.*

*Content*

*Any idea for a solution that you think is worth keeping for future consideration. This can take the form of rough notes, sketches, pointers to other documents, pointers to people, pointers to existing products, and so on. The aim is to capture, with the least amount of effort, an idea that you can return to later.*

*Motivation*

*To make sure that good ideas are not lost. To help you separate requirements from solutions.*

*Considerations*

*While you are gathering requirements, you will inevitably have solution ideas; this section offers a way to capture them. Bear in mind that this section will not necessarily be included in every document that you publish.*

Although many issues in developing have been found thus far, certain present and future issues may be addressed.

* Although we are already using a client-server system design, a model-view-controller design or event handler design may be appropriate as well to help develop the multiplayer mode which would wait for input from the user and update as needed.
* A decorator design pattern may be useful for the design of the UI and a factory method design pattern may be useful for spawn spaces.
* In order to get Java to work with other programming languages, other plugins or libraries may need to be used which will seamlessly merge two languages into a single project.
* Object-oriented design will be a given since Java is being used, however, it may also aid in security as Java has quite a few built-in libraries and object-oriented design may help limit the user to only access specific part of the game.
* CRC cards may be used in order to aid in the object-oriented design and to think of new ways to create the methods of the classes.
* JavaFX may work well as a GUI design tool; however, this game could also utilize certain pre-built videogame engines such as the unreal engine.
* Since we have added more servers, a more intensive client-server test should be implemented before release of the product.
* Usage of IntelliJ or Microsoft Visual Studios Pro may aid in the productivity for development.
* Using pair-programming may be a better choice since most programmers at the company are still at a lower experience of programming.
* Creating smaller modules one-at-a-time will increase cohesion and decrease coupling allowing for ease of debugging, flexibility, and reuse of modules.

## Project Retrospective

*SV: At the conclusion of the ( CS 440 ) project, reflect back on what worked well and what didn't, and how the process could be improved in the future.*

*Content*

*At the end of every project you should reflect upon what methods were used that worked out well and should be repeated in the future, and also what methods did not work out well and should be avoided. Any recommendations, suggestions, or ideas for how to do things better in the future should also be documented*

*Motivation*

*To learn from experience, and to continually strive for process improvement.*

*Considerations*

*When things don't go well, it is important to distinguish whether the methods themselves were poor, or simply poorly implemented in this particular case, or whether they just weren't right for this particular project / group of engineers.*

In terms of what worked well and what didn’t right off the bat, it was a bit too risky to base the company’s future off of one product even though the budget for a small company is low. Certain budgeting could have been done such as limiting the platform availability strictly to Mac or PC users. That would allow for a much larger portion of the budget to be allocated to other sections that may be more important such as to advertising. Also, this would allow much of the team resources to be allocated to more important parts since less updates, optimizations, and program porting would have to be dealt with.

With the company being as small as it is, it would still have been nice to hire employees to create and join other teams such as an in-house QA team, advertising team, financial team or legal team. This would greatly reduce the amount of overhead dealing with third parties.

The allocation of the actual programming of the product worked out very well and this sort of architecture could possibly be honed and used again for future project. The idea to setup a conservative schedule was a great one since it reduced stress in the workplace and increased productivity as it allowed employees to be a bit more flexible with how they approached their work.

More employees would have allowed the company to work on multiple projects concurrently which would allow for more work to be done. It would also reduce overwork in the workspace which is very common in the videogame industry today.

Planning out the project’s logistics in advance would have made the situation a lot smoother as the fluid structure of development lacked a solid foundation in which the employees could rely on. As such, the employees needed to be extra flexible in dealing with various issues.

Adding more servers to accommodate more players was definitely a good idea. At the bare minimum, we would have had underutilization of servers which would have been a small issue compared to having the game be unplayable due to not having enough servers.

Overall, the production of Checker Wars was very rocky to put it bluntly due to multiple issues in creating a new company, but this first project acted a solid platform in which to refine the process in building future applications and videogames.

# Glossary

*SV: The glossary is a more complete and inclusive dictionary of defined terms than that found in section I.7.a, the latter of which only covered the most important key terms needed to understand the report.*

*The glossary defines terms that may not be familiar to all readers. This is especially important if the document is expected to reach a wide and varied audience, such as school children. The glossary may be placed at either the beginning or the end of the document.*

***Flotsam:*** *Any part of a ship or its cargo found floating on the water, whether it was deliberately or accidentally lost by its original owners.*

***Jetsam:*** *Any part of a ship or its cargo that is deliberately cast off ( jettisoned ) by its original owners, generally in order to lighten the ship, whether it floats or sinks.*

**AI**: Artificial Intelligence.

**Bohn Jell Entertainment**: The name of the company that developed the game!

**Rank**: A player’s rank among other players of the game, both locally and globally.

**Score**: A player’s individual score at the end of every game.

**GUI**: Graphical User Interface.

**Twitch.tv**: A popular online streaming platform.

**e-sports**: Electronic competitive video games considered sports by the community.

**DRM**: Digital Rights Management.

**IO Devices**: Input/Output devices.

**CD**: Compact Disk for storage of media.

**GOG**: Good Old Games, a digital videogame storefront.

**Uplay**: A digital videogame storefront owned by Ubisoft.

**Epic Game Store**: A digital videogame storefront owned by Epic.

**Steam**: A digital videogame storefront owned by Valve.

**QA**: Quality Assurance.

# References / Bibliography

|  |  |
| --- | --- |
| [1] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |
| [2] | Robertson and Robertson, Mastering the Requirements Process. |
| [3] | B. M. C. L. O. B. Arthur Mezheritskiy, "Dungeons and Dank," 2018. |

# Index

Design 69, 73, 75, 76, 79

GUI 72, 75, 97

legal 91

Legal 66

Requirements 33, 38, 40, 42, 43, 44, 45, 58

UML 31, 76