CLUE-LESS   
Design Document

**November 6, 2016 – Fall 2016**

EN.605.401.76.FA16 Foundations of Software Engineering

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Created By:

Wesley W.

Chris T.

Jameka H.

Christina T.

# Introduction

## -Purpose/Overview

The purpose of our design document is to express the vision of our HailToo game. We will describe its contents, goal and plan for implementation of our HailToo, clue-like product. Our design document will be an instruction-like diagram to show our developers what to build and how to build it. The purpose of the Software Design Document is to provide a full description of the design of our Clue-Less game.

This document will allow our software development team to proceed with building our application. The design document covers exactly how we envision our Clue-Less game to be built and how users will step through and complete the game. The document provides full descriptions and details of how the software will be laid out ad built.

Our design document will show the needs of our project and allow our developers, designers and project managers to see the end product before it is created. It will present our guidelines for creating the various parts of our product. This design document is based on the consensus of our team. We have gotten all of our ideas together, eliminated vague points and put together a product we believe our users will enjoy.

## -Audience

The intended audience for this design document is all the stakeholders involved in this project. Our stakeholders include our employees (developers, project manager, etc.), our customers, game players, etc. The design document means different things to different stakeholders.

This Software Design Document was created to provide documentation to aid in the development of our software application. It provides details of how the software application will be built. Within the document, we will cover graphical documentation and narrative information to aid all stakeholders in understanding the application from design to implementation. The document includes sequence diagrams, use case models and other supporting requirement information.

## -Scope

This software Design Document provides a base level system document that shows a proof of concept as to how our application will work. For the purpose of this design document, it focuses on how each of the stakeholders will interact with our system and what the expected outcome of the interaction will be. This design document will walk stakeholders through from start to finish and explain how our application will benefit the users.

## -Goals and Guidelines:

The goal of this document is to help design an interactive, fun and easy-to-use application that users will enjoy. Upon launching the application, the user will be authenticated and the game board will open. The user will have the option to launch a saved game, start a new game or join a game already in progress by entering a unique code sent by another player. Once the user has decided what they want to do, the application will run a Clue-Like game experience and interact with the user along the way.

# Design Overview

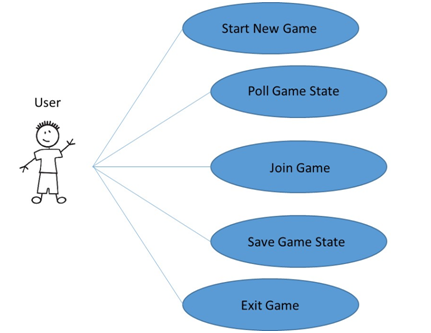
## -Introduction

We are basing our Clue-Less game off of the popular board game CLUE. Clue was first released in 1949 in England. It is a popular murder mystery based game. The design overview will give the stakeholders an overview on how our system interacts with its users. Below are some high level diagrams which should what the user can do using the Clue-Less application. Also shown is the system architecture in which the server and client interact.

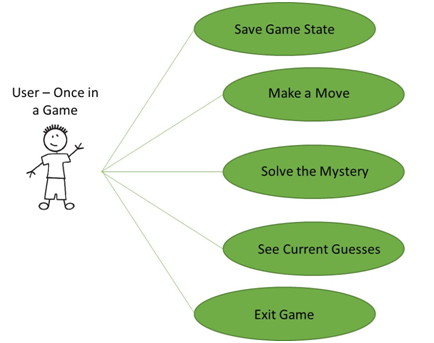
The HailToo team has decided to develop a one of a kind Clue-Like experience without using any Commercial off-the-shelf software products. We decided to use Amazon Web services to host and store the game and game components. The user will authenticate in a browser session and be sent to the game in the cloud over an internet connection.

## -High Level System Architecture

Below is a high level system architecture diagram. When the user opens the application, they have a choice between starting a new game, polling the state of a saved game, joining an active online game, saving their game or exiting their game. The user will have plenty of choices on what to do when interacting with our Clue-Less application. The HailToo team will incorporate all of this functionality when developing the game.



Below shows what the user is capable of doing once they are in a game. The user will have options to save their game state, make a move in the game, attempt to solve the mystery, view other users guesses thus far in the game or exit the game. The user will be fully engaged while using the Clue-Less application.



## -System architecture

HailToo is designed around the *Client-Server* architecture where each client device runs a thin application with visual rendering ability and all actions and business logic occur on a centralized server. This model ensures the integrity of concurrent game play amongst all users rather than performing game-logic on every client device, this also allows for a portable client application. By designing a web-browser based application we are able to support more devices across multiple platforms with a uniform codebase.

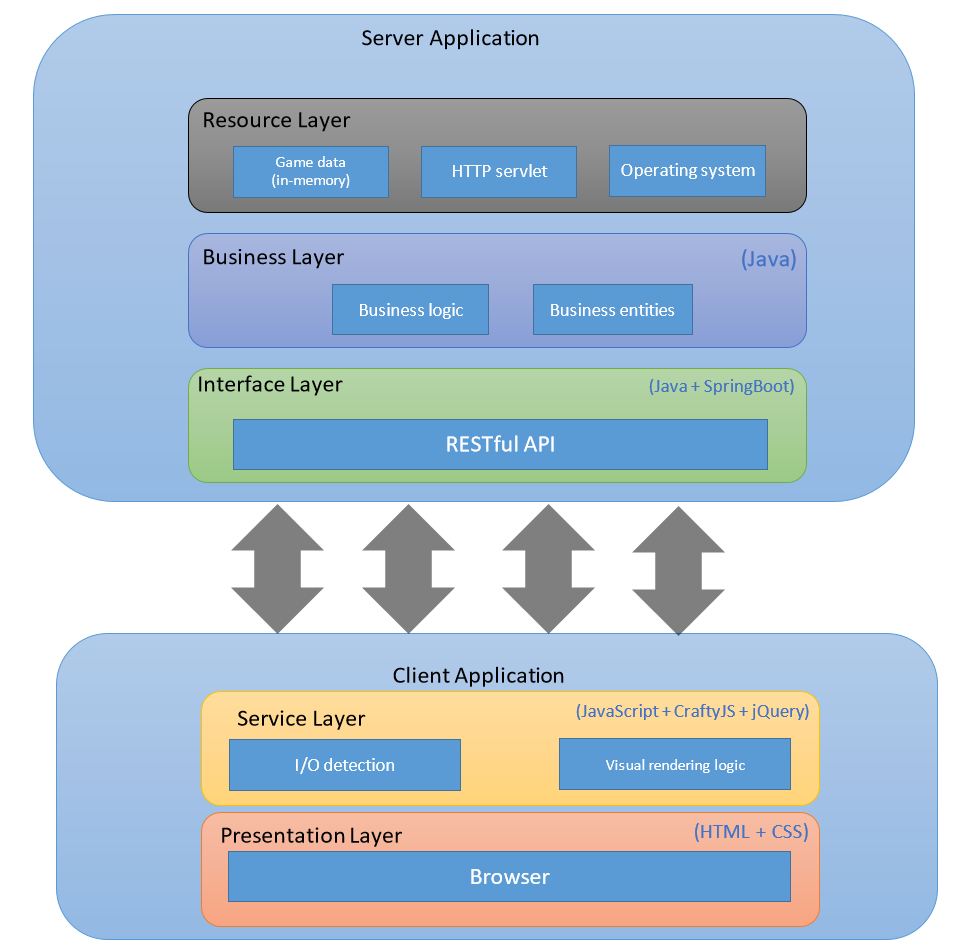


Figure 1 - High-level software architecture

Visual renderings will occur in the background on client applications through JavaScript code and libraries while frequently polling the server application for the current game-state. When the game-state is received on the client, the details on rendered on screen so that all users share the same experience within a game. The client and server interact through a REST interface which allows for object manipulation through HTTP POST and PUT requests from the client (after proper authentication using JSON Web Tokens (JWT)), and game-state retrieval through HTTP GET requests.

The server application will be composed through the Java programming language utilizing the SpringBoot framework for web-server boilerplate implementations and establishing concise, simple RESTful endpoints. Game-state will be stored on the server; the initial implementation will persist this data in memory locally with abstracted logic to allow for easy portability to a NoSQL-flavored database in the future if necessary.

## -Security Features

For now, the HailToo development team has incorporated basic user authentication. We send all data in plaintext because it does not currently include any personal data besides a first name and an email address. Once we start asking users to authenticate with other personal information, we will begin to encrypt transactions between the client and server.

## -Records Management

For now, the HailToo development team has incorporated basic user authentication. We send all data in plaintext because it does not currently include any personal data besides a first name and an email address. Once we start asking users to authenticate with other personal information, we will begin to encrypt transactions between the client and server.

# Objects and Actors

## -Actor/object descriptions

Before we get into the games objects and actors, we will cover the different game players, weapons and rooms. Our game will be based off a two dimensional graphical user interface that will be interactive with its users.

### -Equipment:

The equipment for our application will consist of the gam board which will show the rooms, corridors and secret passage ways, a set of original Clue characters, a set of murder weapons and an interactive GUI that allows a user to save their game, enter a new one, make moves, talk smack and make an educated guess as to who the murderer was.

### -Suspects:

Our application will consist of the original six characters. The board game has actual pewter characters while our application has moveable sprites. Our character list is shown below:

* **Ms. Vivienne Scarlet** - the sultry and beautiful actress of the game.
* **Col. Michael Mustard** - the militant and athletic colonel of the game.
* **Mrs. Blanche White** - the intrusive and kindly maid of the game.
* **Rev. Jonathan Green** - the conniving and religious priest(mobster) of the game.
* **Mrs. Elizabeth Peacock** - the sinister and political senator of the game.
* **Prof. Peter Plum** - the uptight and intelligent professor of the game.

### -Rooms and Corridors:

The original game has nine rooms so that is what we are sticking to. The nine rooms will be laid out in a rectangular fashion. Each pathway will connect the nine rooms together. Each of the four corner rooms may contains a secret passage way (if time permits) that leads to the room on the opposite diagonal corner of the map. The center room (often referred to as the Cellar, or Stairs) is inaccessible to the players, but contains the solution envelope, and is not otherwise used during game play.

Here are the original rooms of Boddy Estate:

* + Kitchen
  + Ballroom
  + Conservatory
  + Billiard Room
  + Library
  + Study
  + Hall
  + Lounge
  + Dining Room
  + Cellar

# User Interface Components

## -Game Board

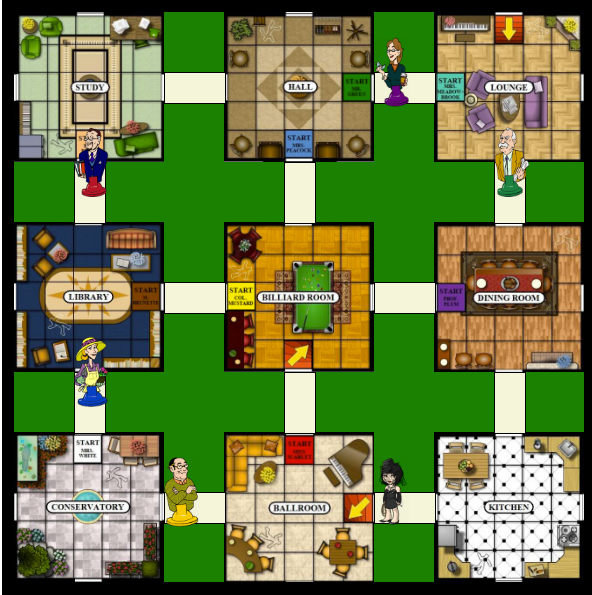


Figure 2 - Actual game board implemented

The user interface (UI) will render a visual representation of the Clue game board and should have nine rooms with adjacent hallways. When a player joins a game they will be prompted to choose the character (game piece) to represent them on the board, their starting location is dependent upon their choice. Each character’s image will be overlaid on a sprite and will move to adjacent areas during the player’s turn, or be summoned to an area due to another player’s guess.

Outside of the game board will be a notification panel which will update as the game progresses, as well as an information panel displaying the other players participating in the same game. The notification panel may provide historic data of the game to aid in the current player’s ability to solve the mystery (so that they do not duplicate a hypothesis which has already been debunked).



Figure 3 - Mock up prompt for player to move/guess

Users will interface with the game board when it is their turn to do so. The user will be notified of their turn and be prompted to choose “move” or “guess”, upon the selection to move, the user will click (or touch) on the area which they would like to move to. When the user chooses to make a guess to solve the mystery a modal dialog will be displayed allowing the user to pick the suspect, weapon and area based on graphic representations of the objects in a wizard-like format (1. Choose a suspect; 2. Choose a weapon; 3. Choose a room).



Figure 4 - Mock up of player making move.



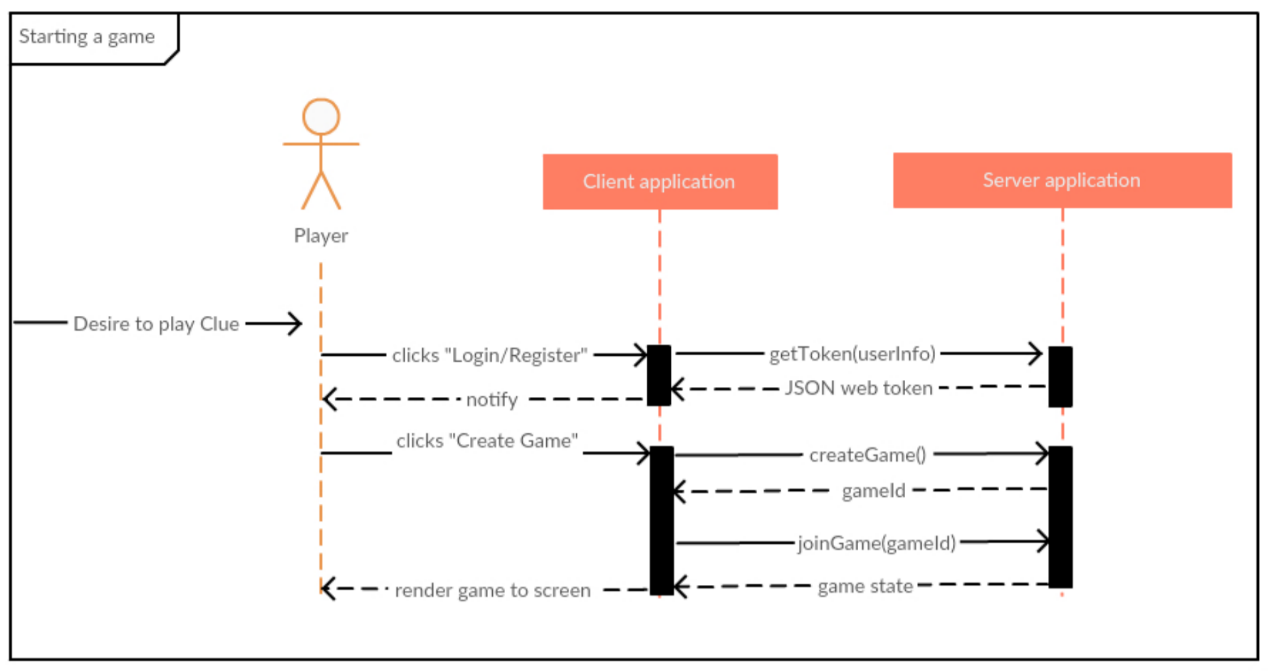
Figure 5 - Mock up of player making guess to solve the mystery.

## Use Cases

## -Use Case 1: Starting a Game

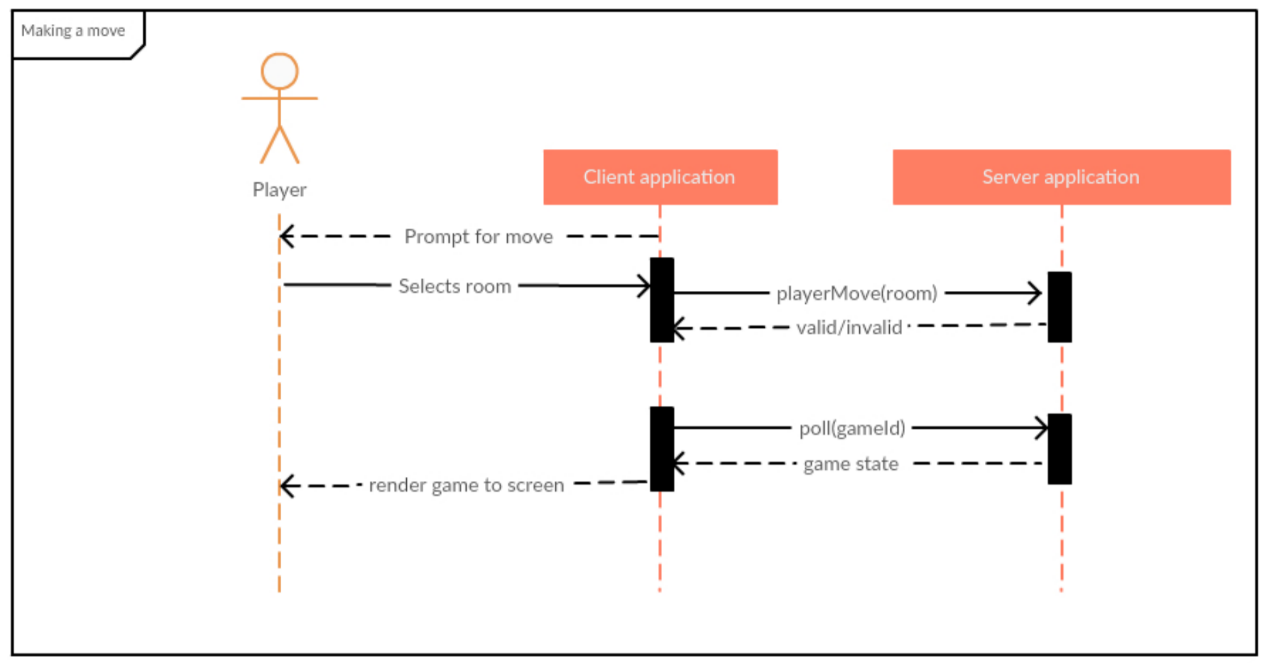
Initiating a game relies on a single actor (player), who seeks to begin a game of Clue, logging in and registering through the application interface. Once the user has established login credentials, they may login through the client which sends user info to the service. The service then notifies the client that login credentials have been validated, and the player may request to start a game.

The request is sent to the backend, the backend responds with a gameId presented to the player, and the player can choose to join a game with the provided gameId. Once the game has been created, the client renders a game to the screen based on the game state established by the server. Any additional players who wish to join the same game must register (if necessary) and login, and provide the correct gameId when requesting to join a game (whereas the initial player was required to create a game).



## -Use Case 2: Making a Move

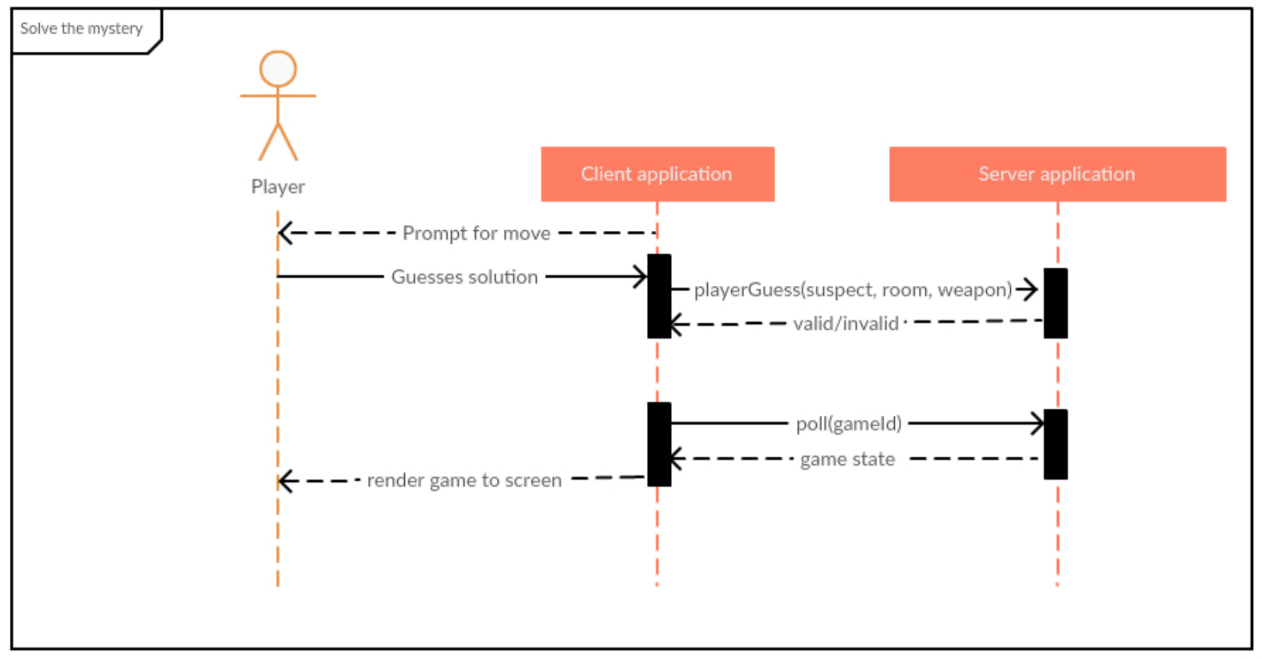
Once a game has been created and game state is verified to be in a valid state of play, the actor (player) is prompted to make a move. The player may request to move their avatar to a room, and the request is sent from the client to the server, which then determines if the requested move is valid. If the move is invalid, the player is notified to make a valid move. If the move is deemed valid, the server notifies the client, updates the game state, and the client renders the updated game state reflecting the player's avatar relocation on the game board.



## -Use Case 3: Attempting to Solve the Mystery

This use case is very similar to Use Case 2. However, instead of sending a request to relocate an avatar, a player may request to solve the mystery. The player selects the suspect, room, and weapon they believe to be the solution, and the client sends a POST to the service containing suspect, room, and weapon parameters.

The service determines whether or not the parameters are valid or invalid. If one or more of the parameters are incorrect, the server notifies the client and the player is informed. If all three parameters are correct, the game is over and the player has won. The gameId is passed back to the server and the game state is updated and sent to the client reflecting the results of the attempt to solve the mystery by the player.



# Supplementary Documentation

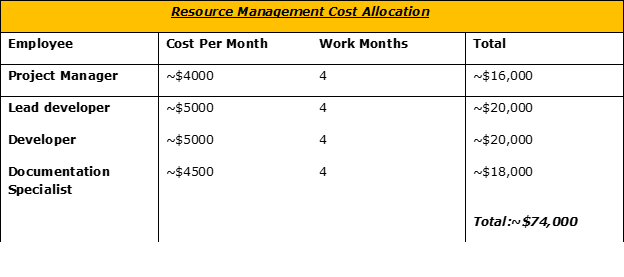
Diagrams in this document were created with Microsoft PowerPoint and Creately.

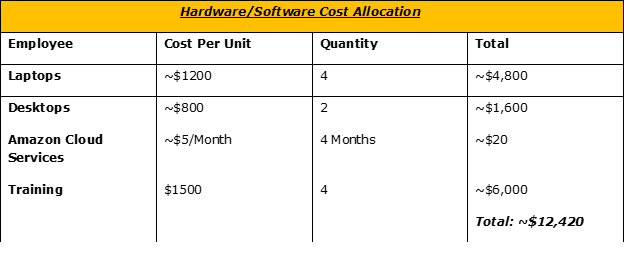
## -Cost and revenue projections:

The cost and revenue projections for HailToo, are listed below. We broke up our costs based on what we have done so far.

## -Resource cost:

Our resource cost is based on the estimated resources within the technical analysis. Employee costs are roughly based on salaries and overhead, which the finance or payroll department would provide in the “real world.” However, ours are just estimated for sake of keeping salaries anonymous. We have used our own personal laptops for the development of HailToo, but we will throw in some new hardware and software costs along with the cost of our cloud services.





## Additional costs (if any):

This section is an assessment of additional costs incurred from licensing, contracting, out-source testing, and so on. We may or may not have any addition costs throughout the development of this application. We will give ourselves an extra $20,000 for incidentals.

# Design Considerations:

## -Assumptions:

The end user must have basic computer operation skills. The end user must have a computer or smart device with a basic operating system and very minimal amount of computing power. We assume that the end user has a basic browser and internet connection of any speed. All storage requirements are taken care of through the Amazon cloud. As long as the end user has the computer, the application and server side portion of the application will store the game state and game itself. No downloading is required.

## -Constraints:

* Hardware limitations
  + Minimal computing power needed.
* End-user environment
  + Mouse and keyboard required
* Availability or volatility of resources
  + Internet connection
* Interoperability requirements
  + Application is compatible with all operating systems and browsers
* Interface/protocol requirements
  + No interface requirements
  + Need browser that supports HTML
* Licensing requirements
  + No licensing needed – free product
* Data repository and distribution requirements
  + All data will be stored in Amazon Cloud
* Security requirements (or other such regulations)
  + User will authenticate with Clue-Less application
* Memory or other capacity limitations
  + Only need enough memory to run OS and browser instance
* Network communications
  + Need minimal speed internet connection
* Verification and validation requirements (testing)
  + Clue-Less will be extensively tested

## -Risks:

The application is fairly simple. We do not foresee any risks. The user will authenticate with the application but no personal data is transmitted other than the saved game state.

# Future Development:

## -Architectural Strategies & Design Decisions:

The plan for the future is to expand game features, speed and storage capacity. With the launching of the application, it will remain free to use for the time being. The plan from the HailToo team is to offer unlocking of more players, weapons, maps and more. Once the HailToo team starts to implement these other features, we may go to a pay-as-you-play model. Users will be able to access the base game at no charge but will be charged for more features.

For now, the HailToo development team will continue to use only four members. As the need for more resources increases, we will evaluate the team, budget and need for these resources and adjust as necessary. We plan to keep the team size small but wide spread in abilities. We also plan to incorporate other security features such as encryption in the future. Currently, we have basic user authentication.