Software Requirements Specifications

COS 301: REDIRECTION

2018

Client: Retro Rabbit

Team Members

Stephen Teichert - u16254661

Kyle Wood - u16087993

Russell Dutton - u16016612

Jeffrey Russell - u16010648

Justin Grenfell - u16028440

Byron Antak - u16039689

Contents

1 Project Overview

1.1 Purpose

The system should allow a user to be able to create custom bot scripts for the game DOTA 2, to ensure that the bots follow a particular strategy so that the user may practise against that strategy and learn how to play against such a strategy.

1.2 Project Scope

The aim of the project is to design system that gives users the ability to configure bot behaviour on a micro and macro level (using an intuitive and user-friendly interface), where this behaviour is described into LUA files according to the DOTA 2 Scripting API to ensure that bots perform as desired.

The system is still under development and no name has been decided upon for the product. The system will provide a simple interface to customize bot behaviour according to a predefined set of behaviours and actions. The system, however, will **NOT** allow the user to specify exact behaviour (such as always moving in zigzags). Rather, it will allow the user to specify generic behaviour. We are not sure exactly how far we will get with this system and as a result we have decided that we will have the following as non-negotiable.

For team level options, the following will be possibilities:

- The user will be able to create a pool of heroes, so they can specify exactly what heroes they want the bots to select from.
- The user will be able to set the push desires for each individual lane, and edit the desires based on specific triggers
- The user will be able to set the defend desires for each individual lane, and edit the desires based on specific triggers
- The user will be able to set the farm desires for each individual lane, and edit the desires based on specific triggers
- The user will be able to set the Roshan desire, and edit the desires based on specific triggers
- The user will be able to set the roam desire, and edit the desires based on specific triggers
- The user will be able to specify each heroes ability progression
- The user will be able to specify each heroes item progression content...

1.3 Definitions, acronyms and abbreviations

Terminology and Concepts Dota/Dota 2 - Defense of the Ancients 2, the game for which we are designing the bot configuration system.

Ancient - The core building of each team which must be protected at all costs. Losing this building loses the game.

Bot - A computer controlled hero in a team in Dota.

Radiant - The team on the bottom left hand side of the Dota map.

Dire - The team on the top right hand side of the Dota map.

HP - Hit Points, the amount of sustained damage a hero can take.

MP - Mana Points, used for casting spells.

Lane - One of three paths on the Dota map. The lanes are located on the Left/Top side of the map, Middle and Bottom/Right side of the map and are referred to as top, mid and bottom or bot lanes, respectively. Lanes can also be classified into 3 types: Safe Lane, Mid Lane and Off lane. The Safe Lane has its Tower placed furthest from the base to provide additional support for the heroes in that lane. For Radiant, Safe Lane is the bot lane. For Dire, it is top lane. Radiant and Dire have their mid towers placed mostly equally apart on each side of the map, providing a very equal match-up between the heroes in the lane. Offlane is the hardest lane, because the tower is closest to the base and the activity in the lane is likely futher away from this tower in this lane than the other team. For Radiant, Offlane is top lane and for Dire it is bottom lane. Note that a safe lane for one team faces the offlane of the opposing team.

River - The river that divides the map into the Radiant and Dire sides.

Jungle - The areas on the map that are not a lane and not in the river.

Objective - An entity in the Dota game that is one of many steps in achieving victory. An objective can be a Tower, a Barracks or Roshan.

Roshan - A massive ancient creep that grants bonuses to the heroes that kill him, including Aegis of Immortals, Cheese and a Refresher Shard. The Aegis will grant any hero who wields it a second life; the cheese will instantly restore 2000 HP and MP, and the Refresher Shard is a one-time use item that instantly refreshes all cooldowns of a hero.

Tower - A defensive turret that can be destroyed by the opposing team. There are 3 turrets per lane, each placed more or less the same distance apart in the

lane, but distances do vary. These towers are referred to as Tier x towers, where x is either 1, 2 or 3 with a Tier 1 tower being furthest from base and Tier 3 being closest to base. There are an additional 2 towers in front of the team's Ancient, called Tier 4 towers.

Teamfight - A big fight involving nearly all team members of both teams in any arbitrary location on the map. The victor of the teamfight is determined by the number of losses, and if it is even, it is then decided by the number of core heroes still alive, and if it is still even, then it is either decided by the net gold change of a team, or it is dubbed a draw. Victory in a teamfight will likely result in an objective being taken, whereas defeat will result in a very defensive stance being taken by the team.

Creeps - Little minions that spawn and run down a lane to meet the opposing team's creeps, also called a creep wave. There are also neutral creeps that spawn in the jungle areas of the map that give gold and experience when killed. If a team destroys a barracks of the enemy team, then the creeps of the team that destroyed the barracks will start spawning stronger, upgraded creeps in that lane, called super creeps. Once all barracks are destroyed, the creeps are further upgraded to mega creeps. An upgraded creep has more HP, and gives less gold and experience.

Push - a collective effort from a team to attempt to advance down a lane to attempt to take an objective, a teamfight, or both.

Farm - The simple activity of a hero collecting gold and experience by killing creeps.

Defend - The act of staying close to or even behind your own tower and creep wave when the enemy team is pushing down your lane.

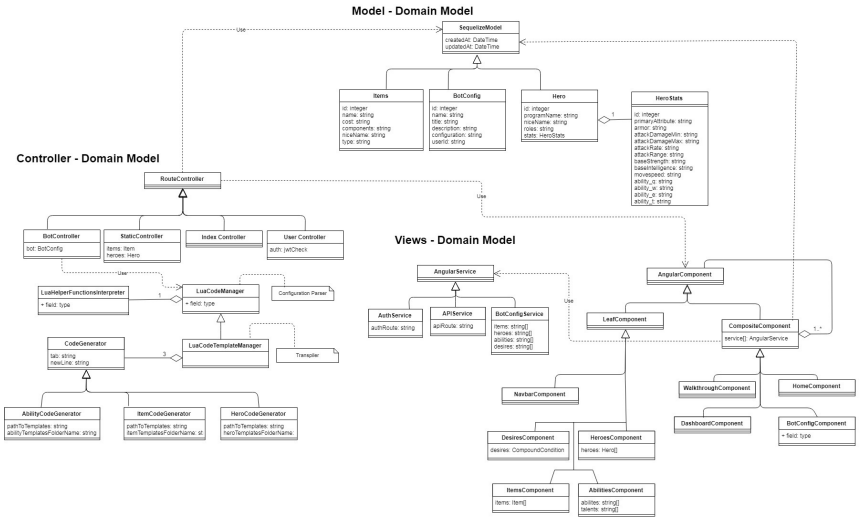
Roaming - the act of not remaining in a lane for too long, but rather moving between lanes to attempt to kill enemy heroes with the other heroes in that lane or in the jungle.

Ganking - The act of a group of team players grouping together to attempt to kill enemy heroes on the map.

1.4 UML Domain model

The system uses a modified MVC pattern. The view allows the user to edit and modify the bots however they may wan to, once the user has done what they wanted to the system sends raw strings to the controller which converts the users input into the custom data format that the controller will understand.

It then sends this converted format to the controller which converts it into LUA, code which is the language used by DOTA 2. Once this is complete it sends the final resulting LUA code to the user for use within the game.



2 Functional Requirements

2.1 Users

We expect have two types of users:

2.1.1 Novice DOTA 2 Player (referred to as novice)

This user is inexperienced in DOTA 2 and knows very little about the game as whole. This user will need to have as much information about the game and system provided to them so that they can understand how to use the system. This user would prefer a user friendly interface so that they are not overwhelmed with the information and so that using the system is easy to understand despite their lack of knowledge. This user will also need to know how to install the bot scripts once generated for them as well as.

2.1.2 Intermediate - Advanced DOTA 2 Player

This user is experienced with DOTA 2 and knows the concepts of the game quite well and hence will not need as much information provided to them. This user would like to use the system but would prefer to have it be as quick as possible while still being intuitive.

2.2 Subsystems

2.2.1 Database Manager

The database manager subsystem controls everything related to the database, anything that gets put in the database will go though this subsystem

2.2.2 Transpiler (Back-end)

The transpiler is the system that converts the object into LUA code to make it compatible with the game DOTA 2. It receives an object of all the user inputs from the frontend.

2.2.3 Intermediate Format Analyser

The intermediate format analyser is the subsystem that receives the raw user input from the front-end and converts it it into an object that the backend systems will understand and then sends it to them for further action

2.2.4 Front-end (Angular)

The front-end is the user interface, this is the system that the user will interact with. It receives all the user input and sends it to the intermediate format analyser for interpretation

2.3 Functional Requirements

1. A user shall be able to login with existing Facebook or Google accounts.
2. A user shall be able to register with the product (via Auth0) by providing specific information (such as First name, last name, user name, password etc.)
3. A user shall be required to verify their email after signing up by following a link sent to them via their email if they chose not to use Facebook or Google accounts
4. A user shall be able to logout of the system
5. A user shall be automatically logged back into their account if they were logged in previously within a specified time period
6. A user shall be able to view all of the bots that they have created and saved thus far
   Precondition: The user has created and saved bots in the past
7. A user shall be able to delete a bot configuration which they have made
   Precondition: The user has created and saved bots in the past

8. A user shall be able to modify the configuration of an already existing bot Precondition: The user has created and saved bots in the past

9. A user shall be able to assemble bots into a team Precondition: The user has created and saved bots in the past

10. A user shall be able to view all teams that they have assembled Precondition: The user has created and saved bots in the past

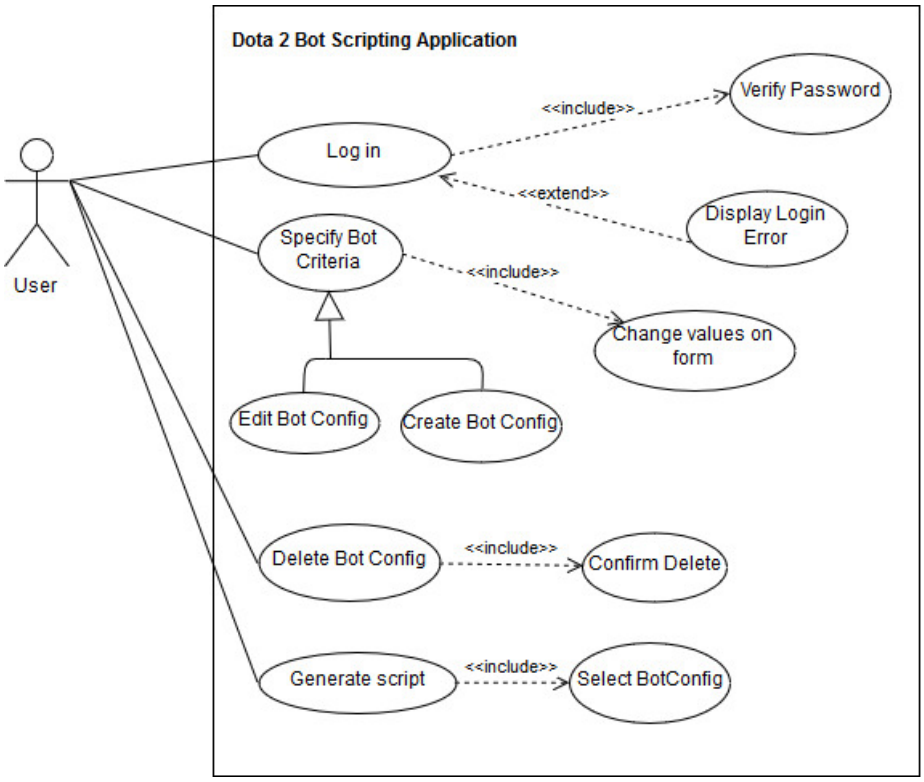
11. A user shall be able to delete any team which they have created Precondition: The user has created and saved bots in the past

12. A user shall be able to interact with a configuration form to determine what generic behaviour the bots should have

13. A user shall be able to save a specific bot configuration to their account

14. A user shall be able to download the configuration as a LUA script(s) with instructions regarding the proper installation of the bot script and how to run it

15. A user shall be provided with a video tutorial on the site to indicate how the product can be used and to guide them through the initial steps to ensure that their DOTA 2 environment is correctly set-up (the installation of DOTA 2 Workshop Tools)



3 Non-Functional Requirements

3.1 Reliability

1. The system should show the correct list of bot configurations and teams created by the user.
2. The system should generate the LUA scripts correctly based on the given bot configuration input.
3. The system should handle errors by displaying appropriate messages without compromising the system's integrity.

3.2 Availability

1. The system should be made available at least 99% of the time, not considering network failures.
2. The application should be able to connect to the internet in order to communicate with the database.

3.3 Security

1. The system should facilitate secure login and authentication using Auth0 and JSON Web Tokens (JWT).

3.4 Maintainability

1. The system should be able to scale well to the increasing number of users on the system.
2. The system structure should be modular to adhere to the concept of low coupling.
3. Coding standards should be adhered to throughout the entire system in order to enhance clarity and readability.

4 System Architecture

4.1 Interfaces

4.1.1 User Interfaces

The system consists of a web application interface which can be accessed through a web browser from a user device. The system accommodates for all forms of devices through a responsive interface.

4.1.2 Software Interfaces

The Angular application represents the client application which handles the front-end interaction by the user. The Express server represents the controlling and information processing of the system, and handles all requests and responses between the Angular application and the database.
The system uses Auth0 for a secure and reliable login/registration system for the user to use. This allows the user to use various login methods (Such as through Facebook, Google or email).

4.1.3 Hardware Interfaces

The system makes use of NodeJS and Expressjs (For NodeJS) as the framework API.
The system also makes use of a mySQL database for storage and retrieval of all data such as saving or retrieving previously created bot scripts.

4.1.4 Communications Interfaces

The client application communicates with the system server when making resource requests. The Express server is responsible for database communication and handles data insertion, modification, and retrieval.

4.2 System Configuration

4.2.1 Server

The system uses a web-server with any OS, so long as NodeJS is installed, to allow for the frontend angular service and the backend ExpressJS service to be run. The system also requires mySQL to be installed on the server since it uses a mySQL database to store all required persistent data.
All application logic takes place within the ExpressJS environment.

4.2.2 User

The User is able to use any Web Browser of their choice to run the system. The system uses angular and bootstrap for design and interactions.
The user will use their mouse and keyboard to interact with the system, which consists of a series of drop-downs with values to modify bot behaviour as well as drag and drop features for hero selection and item progression.