Red System Requirements

CSC 480-800 / HCI 521-800 - Spring 2018

SUNY Oswego

Version 1.1

Table of Contents :

Section 1 - Introduction:

Pretext

The product at question is the resultant of the State University of New York [SUNY] Oswego's HCI 521-800 / CSC 480-800 Software Design Spring 2018 course; directed and managed by Dr. Bastian Tenbergen. The objective of this course is to apply the concepts of software engineering and software development in a structured group project. The entire class will work together on a project assigned by the instructor(s) and will collaborate throughout the semester.

Our purpose is to sufficiently outline and shape a system that can effectively deliver a unique variation of a word building game to a workstation in the Richard S. Shineman Center of SUNY Oswego. The system will only interact with those close enough to connect a mobile device, laptop, or any device with web capabilities to the hardware.

This document is pertinent to Team Red and will prescribe and describe the unique properties of their implementation of "Oswebble." Additionally, this document encompasses all three fundamental parts of their implementation, GUI, Database, and Engine.

1.1 System Definition Glossary:

[Notice]: See primary requirements document. Place further definitions below.

AI	- (Artificial Intelligence) is a function that pulls information from other classes within the Model package to generate and determine optimal placement of tiles from it's respective hand.
CTV	- (Center Tile Validation) validates the first word is on the center tile.
Dictionary database	- The database used by the backend used to validate played words. This database also checks if a word is "profane", as well as if a word is a bonus Oswego-themed word.
Game database	- The database used by the backend used to persist and track actual game information. This database stores teams as well as every non-AI player. This database is also used to calculate statistics for teams, for players, and for specific words.

OOBV	- (Out Of Bounds Validation) validates the word to be
Oswego Themed	- refers to the use of the color scheme as follows: Oz Green #2a6342, Dark Oz Green #1a3f28, Oz Gold #ffcc33, Dark Oz Gold #c19a24, Grey Black #4d4d4d, White #f3f3f3
Oswego Dictionary	- a select list of words that are based on iconic landmarks, campus buildings, as well as other Oswego appropriate terms. These terms themselves, shall be 'bonus' words if played in a game session.
Server frontend	- the Libgdx implementation including GUI, AI, logic, and communications interfaces.
Server backend	- the Node.js implementation connecting the server frontend, GUI, and database
Word Validation	- various methods to ensure a given word can be played correctly.
Bad Word Validation	- a logic gate to stop inappropriate language from being played.
Placement validation	- a logic gate to ensure that letters and words do not overlap other words/letter.
Adjacency Verification	- a logical gate to make sure words adjacent to each other are scored correctly.

1.2 Misc Glossary:

Official Tournament & Club Word List - NASPAWiki. (n.d.). Official Tournament and Club Word List. Retrieved April 12, 2018, from https://scrabbleplayers.org/w/Official_Tournament_and_Club_Word_List

See primary requirements document :

Cierro, M., Anilonis, M., Sumano, C., Santos, J., Spagnola, J., Downey, A., & LeRoy, N. (2018). System Requirements CSC 480-800 / HCI 521-800 Spring 2018 (Vol. 2.1, Oswebble). Oswego, NY: SUNY Oswego.

Section 2 - Overall Description

2.1 Product Perspective:

Based from the instructor selected topic, ‘arcade’, the class agreed upon developing a SUNY Oswego Themed word building board game. Players will place tiles, in which the tiles must form words in: crossword fashion, read left to right in rows or downwards in columns, and be defined in a standard dictionary or lexicon.

This product will be built in the perspective of Team Red’s vision.

2.2 Product Functions:

The game itself will consist of a game session with four players--be them all AI players, all human players, or a combination of the two. Each player will take their turn in placing randomly selected tiles from their hand--any amount of tiles so long as it does not exceed the amount of tiles the player is currently holding--onto the game board; in an attempt to stem valid words off of one another until the game session runs into a end-game scenario. The end-game scenario can be one of two: (1) the game board is filled to a point where no possible moves are available, or (2) the players in the present game session ‘pass’ a given amount of times, such that they activate end-game scenario (1).

The players will be able to shuffle the tiles within their hand, place tiles onto the game board, exchange a given amount of tiles for new ones, access a ‘help’ button that shall explain the rules of the game and how to operate the game.

2.3 User Characteristics:

C1.0 SUNY Oswego Student(s)

- Visits Shineman Center frequently: has some course in Shineman
- Knows that the game exists and how to access it

C1.1 Education Level

- C1.1a : English as primary language : has a fluent level of English vocabulary
- C1.1b : English as secondary language : has a basic level of English vocabulary

C1.2 Game Experience

- C1.2a : Has played some variation of Scrabble™ prior

- C1.2b : Has never played some variation of Scrabble™ prior

C2.0 SUNY Oswego Faculty

- Knows that the game exists and how to access it

C2.1 Visits Shineman Center frequently

- C2.1a : Has some course in Shineman
- C2.1b : Has an office in Shineman

C2.2 Education Level

- C2.2a : English as primary language : has a fluent level of English vocabulary
- C2.2b : English as secondary language : has a basic level of English vocabulary

C2.3 Game Experience

- C2.3a : Has played some variation of Scrabble™ prior
- C2.3b : Has never played some variation of Scrabble™ prior

2.4 Constraints:

There is a kiosk stationed at the main entrance of Richard S. Shineman Science Center of SUNY Oswego, where two monitors will be garrisoned. The primary monitor will be projecting a live imitation of the game board, as well as the player's hand, all placed words on the game board, and the current game scores between the two teams. The secondary monitor will project game statistics--i.e. Each team's cumulative scores, team's best game scores, highest yielding word score, most frequently used word, etc.

The players are able to interact with the game via any device that has the ability to access the internet--such as smart devices, laptops, tablets, etc.

2.5 Assumptions and Dependencies:

1. User Assumptions:
    a. The User has a device that can connect to the target hardware.
    b. The User is able to use their device correctly to play the game.
    c. The User is literate in the English language.
    d. The User understands how to connect and disconnect the game.

2. Software Assumptions and Dependencies:
    a. The Ubuntu operating system, on the target hardware, will be stable.
    b. The server will be constantly running.
3. Hardware Assumptions and Dependencies:
    a. The monitor hardware will be free of error.
    b. The CPU will successfully to execute instructions, on the target hardware.
4. Other Assumption and Dependencies:
    a. All libraries used during development are reliable and consistent.
    b. The interface will be understandable to the players.

2.6 Design and Implementation Constraints:

implementation

---

(1) The Chosen Dictionary database
    (a) English language
    (b) Must not contain any proper nouns
        (i) The Oswego Themed dictionary database is the only exception to this constraint
    (c) Must not contain any word that is only a suffix, abbreviation, or prefix
    (d) Must not include, nor contain, any word that requires a hyphen or apostrophe
    (e) Shall not contain words of profanity

(2) Word Validation
    (a) Words placed on the front end will not be considered valid until backend validation has occurred. This is to prevent, not only cheating, but rather maintain a consistent board state between the backend and the visual representation on the front end.

(3) Connection
    (a) When connecting to the game server users will be prompted for an alias in which they can identify themselves with.
    (b) If there is a spot open, a connected user will replace an AI and wait until it is their respective turn before being allowed to play a word.
    (c) If there is no spot open the user will be prompted a message indicating the game is full. ~[TBD]
    (d) When leaving the game, user stats will be recorded and sent to the database and then an AI will replace the respective player.

Section 3 - System Requirements:

These are natural language requirements pertinent to the entire “Oswebble” game system, with respect to Team Red’s implementation.

Section 3.1 - External Interface Requirements:

These requirements specify hardware, software, and database elements that Red Team’s system interacts with.

3.1.1 User Interfaces

This section describes any interface in which the User will interact with the system.

ID	Requirement Type	Requirement
RUI1.0	Functional	The primary monitor shall also display the rules; as described in UI1.0-UI1.2 from the Solution Neutral Document (SND).
RUI2.0	Functional	The GUI will use the “Oswego Themed” color palette as defined in the System Definition Glossary.
RUI3.0	Functional	The gameboard shall be an 11x11 square.
RUI4.0	Functional	All monitors will use tiles that are dark gray squares with bold white lettering and a numeric value in the lower left-hand corner.
RUI5.0	Functional	The multipliers will follow the “Oswego Themed” color palette using Oz Green for double-letter multipliers and Oz Gold for double-word multipliers.

3.1.3 Software Interfaces

This section describes any software used within the system in which the Red Team’s implementation of the system must interact with.

ID	Requirement Type	Requirement
RSI1.0	Constraint	The server backend and server frontend event communication must use Socket.IO.
RSI2.0	Constraint	The server frontend must use Unirest.
RSI3.0	Constraint	The server frontend must use the Libgdx Framework.
RSI4.0	Constraint	The server backend must use Node.js.
RSI5.0	Constraint	The server backend must use the express framework.
RSI6.0	Functional	The server backend shall run on port 3000.
RSI7.0	Constraint	The server backend must use the Jasmine Test Framework for unit testing.
RSI8.0	Constraint	The server frontend must use JUnit 4 for unit testing.
RSI9.0	Constraint	The server backend must be a REST API web service.
RSI10.0	Constraint	The server backend must use H2 Database Engine.
RSI11.0	Constraint	The server backend must use Spring Framework.
RSI12.0	Functional	The dictionary database shall run on port 8090.
RSI13.0	Functional	The game database shall run on port 8091.

3.1.4 Communications Interfaces

This section describes the different interfaces the Red Team uses in which the system must communicate with.

ID	Requirement Type	Requirement
RCI1.0	Functional	The server front end connects with sockets.
RCI2.0	Functional	The mobile front end connects sockets and REST-calls
RCI3.0	Functional	Static interactions with the server backend will occur over HTTP.

RCI4.0	Functional	The server uses a custom web-socket to mimic socket.io
RCI5.0	Functional	Communications with the server backend will be done in JSON format.
RCI6.0	Functional	When the server frontend receives a “wordPlayed” socket event, the frontend shall check to see if the current player is an AI.
RCI7.0	Functional	When a user joins the game session, a socket connection shall be established.
RCI8.0	Functional	When a word is played, it is first sent to the Dictionary Database.
RCI8.1	Functional	The Dictionary Database replies to the server with three boolean values.
RCI8.1.1	Functional	The three boolean values shall be whether the word is valid, whether the word is profane, and whether the word is an Oswego-themed word.
RCI9.0	Functional	Point values for letters shall be pulled from the Dictionary Database at the beginning of a game.
RCI10.0	Functional	Statistics on player’s, team’s, and played words shall be pulled from the Game Database to be used for the secondary monitor.
RCI11.0	Functional	The Dictionary Database shall be able to accept an array of words from a GET request.
RCI12.0	Functional	When a word is played, each word shall be sent as an array to the scores database.
RCI13.0	Functional	Statistics on letters shall be pulled from the scores database to be used for the secondary monitor.

Section 3.2 - Functional Requirements:

These are requirements which pertain to the basic and necessary functional requirements for the system.

3.2.1 Performance Requirements

This section describes the overall quickness, reliability, and consistency. See primary requirements document for overarching requirements.

ID	Requirement Type	Requirement
RP1.0	Quality	The socket connection between the backend and the frontend shall be less than 15 seconds.
RP1.1	Quality	If the connection between the backend and the frontend takes more than 15 seconds the session will be logged and flagged.
RP2.0	Quality	The lag between tile placement and word validation shall be under 2 seconds.
RP2.1	Quality	If the lag between tile placement and word validation lasts longer than 2 seconds the session will be logged and flagged
RP3.0	Quality	The lag between tile placement and animation of placing the tile shall be less than 3 seconds.
RP3.1	Quality	If the lag between tile placement and animation of placing the tile is more than 3 seconds the session will be logged and flagged.

3.2.2 Design Constraints

See primary requirements document for overarching requirements.

ID	Requirement Type	Requirement
RDC1.0	Quality	The AI will have a delay of 5 seconds between the decision and play of a word.
RDC1.1	Quality	The AI will be notified if it takes more than 5 seconds between decision and play of a word.
RDC2.0	Functional	The AI will have a local knowledge-base of words.

RDC2.1	Functional	The AI will store its knowledge-base in the form of a text file.
RDC3.0	Functional	The AI will pick the best word available in its custom dictionary list.
RDC4.0	Constraint	The valid word dictionary used shall be the Official Tournament and Club Word List as referenced in section 1.2.

3.2.3 Software System Attributes

This section describes the overall functionality and behavior of the software system has incorporated. See primary requirements document for overarching requirements.

ID	Requirement Type	Requirement
RSA1.0	Functional	Each player's tiles will be displayed on the primary monitor surrounding the gameboard on all four sides.
RSA2.0	Functional	The client interface shall have the following functionalities
RSA2.1	Functional	The client interface shall display the gameboard.
RSA2.2	Functional	The client interface shall display the player's tiles.
RSA2.3	Functional	The client interface shall display the timer for the turn.
RSA2.4	Functional	The client interface shall display the team's score.
RSA2.5	Functional	The client interface shall display the opposing team's score.
RSA2.6	Functional	The client interface shall display the shuffle button.
RSA2.7	Functional	The client interface shall display the exchange button.
RSA3.0	Functional	[Moved to UI Requirements]
RSA4.0	Functional	The gameboard shall contain twelve triple-word and twelve double-word multipliers arranged evenly.
RSA4.1	Functional	The multipliers shall be denoted a 'TW' for triple-word multipliers or 'DW' for double-word multipliers.
RSA4.2	Functional	[Moved to UI Requirements]

RSA5.0	Functional	The gameboard multiplier's location shall remain static for each game session.
RSA6.0	Functional	Players shall be informed of the rules on the client interface the first time they login and join the game.
RSA6.1	Functional	Players shall be informed of the rules on a static display found on the primary monitor that players can reference at any time.
RSA7.0	Functional	A horizontal bar shall be used to depict points above the game-board.
RSA7.1	Functional	This horizontal bar shall correspond to a specific team's color.
RSA8.0	Functional	As a team's score increases, the corresponding horizontal score bar shall expand to the right, depicting an increase in score.
RSA9.0	Functional	The GUI shall provide feedback to indicate a user has performed an action.
RSA10.0	Functional	Players shall be able to exit out of the rules upon joining the game.
RSA11.0	Functional	The rules can be accessible on the client interface.
RSA12.0	Functional	Out of bounds validation shall occur when a word is placed.
RSA12.1	Functional	If the word exceeds the board boundaries, out of bounds validation is false.
RSA13.0	Functional	If a word is played, player turn validation occurs.
RSA15.0	Functional	Valid word validation occurs when a player attempts to place a word.
RSA16.0	Functional	If the Dictionary Database contains the attempted word, word validation is true.
RSA17.0	Functional	Invalid placement validation occurs when a player attempts to place a word.
RSA18.0	Functional	If a different tile is played on the spot of an existing tile, placement validation shall be false.
RSA19.0	Functional	If the tiles placed do not correspond to an existing word within the Dictionary Database, then the Adjacency Verification returns false.

RSA20.0	Functional	“Connected to tiles” validation occurs when a player attempts to place a word connected to another tile by at least one character.
RSA21.0	Functional	If at least one character of the played word connects to an already played tile, “connected to tiles” validation shall be true.
RSA22.0	Functional	“Bad word” validation shall occur when a player places a word flagged as “profane”.
RSA23.0	Functional	The Dictionary Database shall contain three lists of words :...
RSA23.1	Functional	...a list of valid words...
RSA23.2	Functional	...a list of profane words...
RSA23.3	Functional	...and a list of Oswego-themed words.
RSA25.0	Functional	When a player attempts to place a word, the word shall be checked in all three dictionaries in the same query.
RSA26.0	Functional	The game database shall be able to search for a team by its team name.
RSA27.0	Functional	The game database shall consist of three tables: a player table, a team table, and a played word table.
RSA27.1	Functional	The played word table shall keep track of statistics data for each word played.
RSA27.1.1	Functional	Each row of the played word table shall have a relation to the player who played the word.
RSA27.2	Functional	The player table shall keep track of statistics data for each player.
RSA27.2.1	Functional	Each row of the player table shall have a relation to the team that the player is on.
RSA27.3	Functional	The team table shall keep track of statistics data for each team.
RSA27.3.1	Functional	Each row of the team table shall have a relation to a list containing every player that has joined the team.
RSA28.0	Functional	The game database shall calculate persistent statistics for teams.
RSA28.1	Functional	The game database shall calculate persistent statistics for players.

RSA28.2	Functional	The game database shall calculate persistent statistics for words.
RSA29.0	Functional	The scores database shall calculate persistent usage statistics for letters.
RSA30.0	Functional	The scores database shall keep track of how many times each letter in the English alphabet has been used.

3.2.4 Other Requirements

This section provides any requirements that do not meet the description of any other section

ID	Requirement Type	Requirement
ROR1.0	Functional	The name of the game shall be “Oswebble.”
ROR2.0	Functional	Players will have three minutes for their turn during which time they can play in any valid spaces on the game board
ROR2.1	Functional	If the player does not play a valid word before the time limit, the system replaces the human player with an AI and moves to the next player's turns.
ROR3.0	Functional	There shall be a QR Code present for the user on the primary game screen.
ROR3.1	Functional	The QR Code shall give a link when read for the user to connect to game’s login page.
ROR4.0	Functional	There shall be a help screen present for the user located on the top middle of the game.
ROR4.1	Functional	The help screen shall display relevant information to assist the user in using the system.
ROR4.2	Functional	The help screen will be accessible anytime throughout the game.
ROR4.2.1	Functional	The help screen shall be closable anytime while it is open.

Section 4 - System Scenarios:

System scenarios refer to the sequence of steps a user takes when interacting with Team Red's game system.

4.1 Primary Scenarios

These are scenarios depicting "normal" use-cases with the system at hand. More information can be seen in the primary requirements document for overarching requirements.

ID	Actor Action	System Response
RS1.0	A game session end	All respective stats and scores are stored in the database and memory is cleared.
RS2.0	A human player leaves the game	Their respective turn ends and then their spot is replaced with an AI.
RS3.0	A human player places the word	The word is sent from the front end to the backend to be validated. This is then sent to the database for the actual validation, of which the backend then verifies. If all this returns as true the word is played.
RS4.0	The server frontend receives a "wordPlayed" socket event.	The frontend shall check to see if the current player is an AI.

4.2 Alternative Scenarios

These are scenarios depicting "abnormal" or erroneous use-cases with the system at hand. More information can be seen in the primary requirements document for overarching requirements.

ID	Actor Action	System Response
RAS1.0	A human player places a bad word	The word is sent from the front end to the backend to be validated. This is then sent to the database which will deny the word, causing the backend to reject the word played. The player will then have another opportunity to play a word.
RAS2.0	A human player tries to play not on their turn.	The respective player's word's will be unavailable to place. And will indicate the user in some from [TBD].

Section 5 - Diagrams:

These are various diagrams which provide additional clarity and information about Red Team's system under development.

5.1 : Red AI Seq Diagram

- This diagram depicts the Human Player and AI handoff/handshake within a game session. This is in the context of the general schema between Engine, GUI, and a player.

5.2 : Red AI Activity Diagram Diagram

- This diagram depicts the AI, player transfer sequence during an active game session.

5.2.1 : Red AI Activity Diagram

- This is a UML activity diagram which demonstrates the logic behind the AI making a move, as well as providing an insight to how the game updates.

5.3 : Red Backend UML Class Diagram

- This diagram depicts the dependencies, I/O, and associations between the backend and the gameboard's state.

5.4 : Red AI, Backend, Connection Sequence Diagram

- This diagram depicts the the backend, socket, and AI communication which occurs during connection attempts and instantiation.
-

5.5 Red GUI Class Diagram

- This diagram depicts the classes, methods, variables and interactions between the user interface.

5.6 : Red System Sequence Diagram

- This diagram depicts the sequence of steps and interactions when a human player wants to join a game session.

5.7 : Red Database UML Class Diagram

- This diagram depicts how players are associated with a team and how a players word is validated when played

Appendix A - Depreciated Items

This section contains requirements and scenarios that were removed from section 3 and section 4. The original location of these requirements will be marked as such in section 3. Their IDs here are kept for reference purposes.

[As of version 1.1]

ID	Pre-Text	Post-Text
RCI1.0	Functional	The backend will be a REST API web service.
RDC3.0	Functional	The server frontend will use sockets to communicate with the backend.

Appen. - ____

ID	Pre-Text	Post-Text