****

**Web based Chat application for Dagoo mail system**

**National Application Center Department**

july, 2016

Elias G/sellassie

**Contents #**

**1. Introduction……………………………………………………………………………………2**

1.1Mission Statement**……………………………………………………………………………2**

1.2 Project Overview……………………………………………………………………………...2

1.3 Benefits……………………………………………………………………………………….3

1.4 Challenges…………………………………………………………………………………….3

1.5 Development Methodology…………………………………………………………………..4

1.6 Success Criteria………………………………………………………………………………4

1.7 System Overview/Architectural Diagram of chat application………………………………..5

**2. Background……………………………………………………………………………………6**

2.1 Impressing Beginnings……………………………………………………………………….6

2.2 Instant messaging in 2000s……………………………………………………………………6

2.3 Social Media Chat…………………………………………………………………………….7

2.4 Some features of chat application……………………………………………………………8

**1. Introduction**

The chat application would be accomplished on a Server-Client Architecture within a LAN. The client– server model of computing is a distributed application that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. The Server Side would be a continuously running service listening to the different Clients asking its services. The chat application would be accessed on every communicating client. A Database of users would be maintained by the Server. When a client PC logins to the application, the Server authenticates the user of the client PC. Once the user is authenticated the IP address of the client is registered to the Server and it sends the list of online user friends and

other relevant data to the Client. When the user wishes to chat to some other user, his IP address along with a Port address would be sent to the other user and vice versa. Thus a connection would be established and the two client PCs would be able to chat together.

* 1. **Mission Statement**

The project aims to develop a chat application for Dagoo mail application, which mainly enhances communication of each member of Dagoo mail users. It is a web based system which will available over the LAN/Internet.

* 1. **Project Overview**

The project will work on top of the Dagoo email application which currently give a messing service, using the proposed application users can communicate online using the chat application.

The application will have two main components the Server and Client logic in addition to the database or the active directory of users. Currently the chat application will have only text messaging feature.

**Why this project?**

* To facilitate communication between users of Dagoo mail application.
* Web based, no software installation on client

**What technologies are used?**

JavaScript, HTML, AJAX(Asynchronous JavaScript language and XML), Java

**The principle of the system is:**

* Keep it Simple
* Extensibility
* Showcase of current trend in web development
* Portability (develop once, run on all platform)
  1. **Benefits**

The proposed system is expected to enhance the communication of Dagoo mail users

* **From the user perspective:**

**Communicate independently, more flexible**

“Independently” here means once the user has user id and password of Dagoo mail, being free from the influence, guidance, or control of the working environment. The only requirements are a connection to the server (internet), and a web browser.

No download, no installation and no manual configuration.

* **From the administrator’s perspective**

**Portability, suit the current infrastructure**

To install the proposed system, no change need to be made on the current infrastructure, the system will suit and integrate most infrastructures automatically. The system will run on a number of major operating system and architectures, including Windows and Linux.

**Maintenance cost reduced**

In the proposed system, software will only be installed on the central server, and nothing needs to be installed on the client side.

In the proposed system, most maintenance work can be done on server and client side.

**Customizable**

The proposed system will be customizable to add feature of chat application other than the current implemented feature.

**1.4. Challenges**

Since the proposed system will be develop from scratch it will have challenges, even it is more popular application now a days .In this section, the challenging aspect of the project will be stated.

Most difficulties of this project are from the client and server side, namely the HTML and JavaScript. Additionally, different browsers will render the same JavaScript code differently. This makes writing a cross browser JavaScript program a very difficult task.

From server side perspective building network application will be more challenging and expects knowing to communicate with low level network protocols, and finally the integration of the application with the existing Dagoo mail will be a third challenge.

To sum up, the project is a challenging project because of the limiting knowledge of JavaScript, and JavaScript web frameworks and knowing how to communicate with network protocols.

**1.5. Development Methodology**

A phased development methodology will be used in this project. In the process, various versions of the system will be produced. This methodology can give user a chance to work with the system sooner, and give chance to identify additional requirements.

Firstly, the background of the system will be studied. The system with similarity will be examined, and their advantages and disadvantages will be listed. Secondly, the analysis and design will be carried out. Before implementation, only the analysis and design of the core function will be carried out. As the nature of phased development, it will bring function to final users more quickly to identify additional requirement. Therefore, before implementation of each part of the system, the analysis and design will be reviewed. Implementation phase will be divided into three parts according to the structure of the system: Client Side development, Server side development and Database development. A prototype system will be developed. The final stage is testing and documentation writing. A test report will be produced. The test is composed of Unit testing and Integration testing. The unit test contains black box testing and white box testing. The integration testing contains the use case testing, use interface testing and interaction testing. Furthermore, program documentation will be produced. Documentation includes developer manual which guides any further development, user manual which gives explanation on every function, and tutorials which will guide the usage of the system.

**1.6. Success Criteria**

How this system can be judged as a success? These are the general criteria. For the detail Requirement, please refer to requirement analysis section. Solely with this web based system:

* The chat application is successfully integrated with Dagoo mail system
* User can browse the chat application after login to Dagoo mail system.
* If the chat application populates contacts with their status online/offline of the user.
* User can select a user to chat and if the application pop up a new window.
* User can send chat message to the respective client user.

**1.7 System Overview/Architectural Diagram of chat Application**

****

**Fig 1. System Architecture Over view**

1. **Background:**

**2.1 Impressive Beginnings**

The phrase "instant messaging" entered common usage in the early 1990s, but the concept actually dates back to the mid-1960s. Multi-user operating systems such as the Compatible Time-Sharing System(CTSS, which was created at Massachusetts Institute of Technology (MIT)'s Computation Center in 1961, allowed up to 30 users to log in at the same time and send messages to each other. The system, which is perhaps closer to what we now think of as email, had hundreds of registered users from MIT and other New England colleges by 1965

## 2.2 Instant Messaging in the 2000s

In 2000, Internet users took notice of Jabber a multi-protocol instant messenger that acted as a single gateway for users to chat with friends and access their buddy lists on all of the big networks at the time: AIM, Yahoo and MSN. Jabber.org is the original IM service based on Extensible Messaging and Presence Protocol (XMPP). Most recently, in August 2012, Jabber disabled new registration due to user abuse and denial of service attacks.

Apple developed iChat, or iChat AV, for its Mac OS X operating system in 2002. Mac users could integrate their address books and Apple Mail in a native app compatible with AIM. In 2011, Apple announced iMessage for the updated Mac OS. Both iChat and iMessage were replaced earlier this year by OS X Mountain Lion's Messages allowing users to send unlimited messages to almost any Apple product.

Founded in 2003, Skype allows Internet users to communicate with others through video, voice and instant messaging. The instant messaging aspect of the service, while perhaps not its most popular function compared to video conferences, is used by many. In July 2011, Skype announced integration with Facebook, so users could see Facebook friends on Skype and see Facebook Chat through both services.

Meebo began in 2005 as an instant messaging service accessed via web browser. Before it was acquired by Google last June, it supported Yahoo! Messenger, Windows Live Messenger, AIM, ICQ, and later MySpaceIM, Facebook Chat, Google Talk and others. Meebo had also developed mobile versions for iPhone and Android.

## 2.3 Social Media Chat

In 2005, Google released Google Talk, often referred to as Google Chat or Gchat. Available in various web, native and mobile applications, Google Talk always appears in a Gmail user's window, allowing for easy communication with email contacts. The service includes text-based messaging, voice calls and video conferences. Most recently, Google Talk has been integrated with Google+, allowing users to chat while in the social network.

Myspace developed MySpaceIM in 2006 as an addition to its social platform — the first social network to do so. Users could instant message with friends on their desktops, as well as online starting in 2009, through MySpaceIM for Web. Later the service was integrated with Skype.

Facebook released Facebook Chat in 2008, allowing users to instant message one friend or multiple people through the groups feature while logged into the social network. In 2011, Facebook announced the incorporation of video in Chat — integrated with Skype — and has also released the mobile app Facebook Messenger.

Where will these services go next? Tell us your thoughts on the future of instant messaging in the comments.

**2.4 Some Features of chat application**

* Chat Server

The System that we have designed is based on client-server architecture. So creating a server is necessary. The functions of a server are to maintain user information, maintaining the record of users logged in to the system and providing the IP addresses to the clients who want to chat.

In the Server side a server-socket is created which accepts connection from client those who have logged in. Once any user logs in and makes connection with the server, it is assigned a thread. Input stream and output stream objects are created in the run method. If the user has requested for registration then a new entry is added to the Database. If the request is for sign in then the user is verified and then his flag is set as an

* Painting

Once the user logs in, he can use the application to draw and send images. This has been implemented by using a painting class. The following functionalities have been provided to the user. He can draw any of the following figures like point, line, circle, rectangle and rounded rectangle. Free hand drawing is also possible. Filled circle, filled rectangle and filled rounded rectangle are the other drawing figures available. User can use any of the ten colors for drawing. An eraser has been provided for erasing the drawn figures. These are the functionalities provided to the user. The working of the painting functionality can be explained. The painting class has implemented the action listener and mouse listeners. The action such as selecting the drawing tool and color of drawing is recorded. On the mouse click event the initial coordinates of the location of mouse is saved. The initial coordinates and the new coordinates on mouse move are used to draw the figures. Whenever the mouse is moved the paint function is called and the figures are drawn. The socket used for sending to the other client has already been created and is used by the paint function to send the coordinates to the other client. The receive class receives the ordinates sent by the other class which is then used by the receive panel to draw the images and figures. The server socket created for that particular connection is used to receive the coordinates, type of the figure and the color. In this way drawing has been done in the application.

* Message Texting

The user can type and send text messages to the other user instantly. This has been accomplished by socket connection. The Split Window class gets the sending socket used for connection. The text messages are sent through the socket. The key listener has been implemented on the bottom-most window where a user types the message. On the press of the Enter key, the text message is sent to the other user and displayed over the text area of both the user. The receive function on the other side differentiates the text messages from those of the drawing and text messages and sends it to the text-area of the receiving user. In this way the texting has been implemented in this application.

* Predictive Texting

Predictive testing is another enhancement that has been made in this application. Basically predictive texting means to predict what a user is going to type from the set of letters (substring) that user has already typed. The predictive texting system that has been implemented for this application is a self-learning from the typing of the users using. It does not have a prior database of words but instead adds the database of words as the user types. The words are stored for the particular session of conversation after which the words are not available again. So that the words do not clash with some other user who might later sit on the machine and logins to his account. The Split Window class is where the windows for texting and painting have been implemented. In the class the auto-text complete class is called on the bottom most text field. So on key press events, the functioning of predictive texting begins. If the total length of the word is 3, the hash-chain class is called. A hash table is created for size of 10 words. The substring of 3 letters is passed as on of the parameters to the find function of the hash table. It computes the hashing value for the given substring which can be used to search all the words beginning with the substring or else can be used to insert the word to the specified location of the hash value. The hash function used for this purpose can be explained as follows. The hash value is calculated by summing up the ascii value of the letters of the substring, then a modular division is performed on the sum by the array size of the hash table. Thus a maximum of 10 words can be stored for a particular substring.

A frequency count of the words used has been maintained which provides the recurrence of words in typing. More the frequency of the words higher will be their priority. The lower priority words would be deleted from the hash table when it fills up. In this way the predictive texting has been implemented

# Current System

Currently Dagoo mail application has not a chat feature, it only provides mail service.

# Proposed System

## 4.1 Overview

This section provides a functional overview of the chat system. This will again be properly be divided into two parts

## Functional Requirements:

## The functional requirement of the chat application will be categorized into two the Server and Client.

## Server:

## The server must accept client request

## The server creates socket between clients.

## Maintaining user information

## Parsing message coming from clients

## Upon connection request the server must send friend list and their status

## Client:

## The game must be playable by 2-4 players simultaneously.

## Players must be able to enter their names, and choose specific rules for the variant of the game.

## Players must be able to start a new game when the program is loaded.

## Players must be able to start a new game during the progress of a game with confirmation that they want to cancel the old game.

## The score and the words each player possesses must be displayed at all times on a score board.

## Nonfunctional Requirements

### Usability

* Game must be playable without mouse
* Each player must have unique color representation.
* Tiles on the board should reflect the PLAYER who owns them by being displayed in that PLAYER's color.
* “About” and “Help” information about the game must be able to be retrieved at any point during the game or the run of application.
* Players must be able to choose from a set of several different language dictionaries.
* SYSTEM will use a standard dictionary format so that third party dictionaries that exist in that format may be used.
* SYSTEM will provide status messages to the player to inform the player of the state of the game.
* SYSTEM must play a sound when player has 10 seconds or less during their turn. (Single beep)
* SYSTEM must play a sound when player runs out of time. (Alarm clock)
* SYSTEM must play a sound when player submits word that is not in dictionary. (Two note, low pitch sound)
* SYSTEM must play a sound when player submits a word that is in dictionary. (Ding ding)
* SYSTEM must play a sound when player steals opponent’s tile(s). (Cha ching!)
* SYSTEM must play a sound when player submits a word on a pink square and the board is cleared. (Ding ding ding)
* SYSTEM must play a sound when a player wins the game. (Triumphant horns)

### Reliability

### Components of the project code will be tested alongside the implementation phase to ensure that they are functional.

### Final, integrated project Code will be tested with EclEmma to ensure that greater than or equal to 80% of the integrated code is covered at run-time, and is functioning properly. The remaining 20% will be inspected through manual testing to ensure the highest chance of being quality code.

### Performance

### Drag and drop of the tiles must be smooth without graphical lagging.

### Validation of the words from the dictionary or keeping track of players’ words will take little or unnoticeable amount of time.

### Supportability

### The application must not be platform dependent, i.e., it should be able to run on any platform supporting JAVA.

### Implementation

* Project will be implemented in JAVA.
* All project graphical user interfaces will be created using a JAVA GUI editor.

## System Models

### Use case model

|  |  |
| --- | --- |
| **Name:** | StartNewGame |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | Application is running.  A Game is not currently in progress. |
| **Flow of**  **Events:** | 1. PLAYER initiates new game function  2. SYSTEM presents PLAYER with NewGameForm  3. PLAYER submits NewGameForm |
| **Exit**  **Conditions:** | Game is now in a new state. |

|  |  |
| --- | --- |
| **Name:** | StartNewWhileGameInProgress |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | A Game is currently in progress |
| **Flow of**  **Events:** | 1. PLAYER initiates new game function  2. SYSTEM confirms that the PLAYER wishes to continue.  3. SYSTEM presents PLAYER with NewGameForm  4. PLAYER submits NewGameForm |
| **Exit**  **Conditions:** | Game is now in a new state. |

|  |  |
| --- | --- |
| **Name:** | PlaceTileFromRack |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  There is at least one tile on the rack  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selects a tile from the rack and places it in a cell on the board.   2. SYSTEM refreshes to show modified rack and board on display. |
| **Exit**  **Conditions:** | The tile is now in the destination cell on the board and no longer in the rack. |

|  |  |
| --- | --- |
| **Name:** | MoveTileOnRack |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  There is at least one tile on the rack  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selects a tile from the rack and places it in another position on the rack.   2. SYSTEM refreshes the rack on display to show the new state. |
| **Exit**  **Conditions:** | The tile is now located at the destination indicated by the PLAYER and not in its previous location, and any other tiles have been shifted accordingly. |

|  |  |
| --- | --- |
| **Name:** | MoveTileOnBoard |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  The PLAYER has placed at least one tile on the board this turn.  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selects a tile from the board that he or she placed this turn and moves it to another value destination cell on the board.   2. SYSTEM updates the rack to show the new state on display. |
| **Exit**  **Conditions:** | The tile chosen is now in the destination cell indicated by the PLAYER and is no longer where it was previously. |

|  |  |
| --- | --- |
| **Name:** | MoveTileFromBoardToRack |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  There is at least one tile on the board that the PLAYER placed there this turn.  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selects a tile he or she previously placed on the board this turn and moves it to the rack.   2. SYSTEM updates the state of both rack and board on display. |
| **Exit**  **Conditions:** | The chosen tile is now in the indicated position on the rack and is no longer on the board. |

|  |  |
| --- | --- |
| **Name:** | PlaceIllegalTile |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | Extends PlaceTileFromRack, MoveTileOnRack, MoveTileOnBoard, and MoveTileFromBoardToRack |
| **Flow of**  **Events:** | 1. After selecting a valid tile, PLAYER attempts to move it to an invalid position (outside the board/rack, on top of another tile on the board).  2. SYSTEM denies the move, and plays the relevant sound. |
| **Exit**  **Conditions:** | The tile is returned to its original position |

|  |  |
| --- | --- |
| **Name:** | SubmitWord |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  PLAYER has placed at least two tiles on the board this turn  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selected the Submit option. |
| **Exit**  **Conditions:** | SYSTEM plays the relevant sound.  PLAYERs’ scores and wordlists are updated, and Timer is reset.  Tiles on the rack have been refilled to 7 if there are enough tiles remaining. If not, fill up the rack with the remaining tiles.  It is now the next PLAYER’s turn. The next Player is defined to be the next player in the list as entered in StartNewGame form. |

|  |  |
| --- | --- |
| **Name:** | SubmitInvalidWord |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | Extends SubmitWord |
| **Flow of**  **Events:** | 1. PLAYER places a word on the board that is invalid (not in dictionary, not contiguous, violates variation rule, or not in green square on first turn). |
| **Exit**  **Conditions:** | SYSTEM plays the relevant sound.  SYSTEM updates the status window on the main GUI with the appropriate message.  Tiles placed this turn have been returned to the rack. No points or words have been added to PLAYERs’ scores.  It is now the next PLAYER’s turn. The next Player is defined to be the next player in the list as entered in StartNewGame form. |

|  |  |
| --- | --- |
| **Name:** | SubmitOrangeCellWord |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | Extends SubmitWord  PLAYER has placed a tile on an orange cell |
| **Flow of**  **Events:** |  |
| **Exit**  **Conditions:** | Bonus points have been added to the PLAYER’s information.  Bonus points will be added to the PLAYER’s score if and only if the PLAYER plays a word on a Pink square. |

|  |  |
| --- | --- |
| **Name:** | SubmitPinkCellWord |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | Extends SubmitWord  PLAYER has placed a tile on a pink cell, and the game is not in “noPink” variation. |
| **Flow of**  **Events:** |  |
| **Exit**  **Conditions:** | SYSTEM plays the relevant sound.  SYSTEM updates the status window on the main GUI with the appropriate message.  The board has been cleared of tiles.  The bonus points added to the PLAYER’s information has been added to the PLAYER’s score.  Bonus points of ALL OTHER players are erased. |

|  |  |
| --- | --- |
| **Name:** | SkipTurn |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  Game is not finished |
| **Flow of**  **Events:** | 1. PLAYER selects the Skip Turn option.  2. SYSTEM refreshes the board and rack as necessary. |
| **Exit**  **Conditions:** | All tiles placed on the board this turn have been returned to the rack.  It is now the next PLAYER’s turn. The next Player is defined to be the next player in the list as entered in StartNewGame form. |

|  |  |
| --- | --- |
| **Name:** | ReachedMaxScore |
| **Actor:** | SYSTEM |
| **Entry**  **Conditions:** | A PLAYER’s score has reached or exceeded the maximum score |
| **Flow of**  **Events:** | 1. SYSTEM presents the WinningGameFrame and plays the relevant sound. |
| **Exit**  **Conditions:** | The game is now in the finished state.  SYSTEM plays the relevant sound.  SYSTEM updates the status window on the main GUI with the appropriate message. |

|  |  |
| --- | --- |
| **Name:** | ExpireTime |
| **Actor:** | SYSTEM |
| **Entry**  **Conditions:** | The PLAYER whose turn it is has exceeded the time allotted him or her. |
| **Flow of**  **Events:** | 1. TIMER expires.  2. SYSTEM proceeds with skip turn behavior. |
| **Exit**  **Conditions:** | SYSTEM plays the relevant sound.  SYSTEM updates the status window on the main GUI with the appropriate message.  All tiles placed on the board this turn have been returned to the rack.  It is now the next PLAYER’s turn. The next Player is defined to be the next player in the list as entered in StartNewGame form. |

|  |  |
| --- | --- |
| **Name:** | SkipAllPlayers |
| **Actor:** | SYSTEM |
| **Entry**  **Conditions:** | All PLAYERs have consecutively skipped their turn |
| **Flow of**  **Events:** |  |
| **Exit**  **Conditions:** | The rack is now filled with a new set of 7 tiles if there are enough tiles remaining. If not, fill in with as many tiles as there is left.  SYSTEM updates the status window on the main GUI with the appropriate message. |

|  |  |
| --- | --- |
| **Name:** | ViewHelp |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | The application is running |
| **Flow of**  **Events:** | 1. PLAYER selects the View Help option  2. SYSTEM stops the clock if there is a game in progress.  3. SYSTEM presents HelpFrame. |
| **Exit**  **Conditions:** |  |

|  |  |
| --- | --- |
| **Name:** | ViewAbout |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | The application is running |
| **Flow of**  **Events:** | 1. PLAYER selects the View About option  2. SYSTEM stops the clock if the game is in progress.  3. SYSTEM presents AboutFrame. |
| **Exit**  **Conditions:** |  |

|  |  |
| --- | --- |
| **Name:** | ExitApplication |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | The application is running |
| **Flow of**  **Events:** | 1. PLAYER selects the Exit option  2. SYSTEM asks for confirmation with confirmation form.  3. PLAYER submits exit confirmation. |
| **Exit**  **Conditions:** | The application has been terminated. |

|  |  |
| --- | --- |
| **Name:** | SaveGameLog |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | The game is in a finished state |
| **Flow of**  **Events:** | 1. PLAYER selects the Save Log… option  2. SYSTEM presents standard OS save file dialog  3. PLAYER submits form |
| **Exit**  **Conditions:** | Log of game now saved |

|  |  |
| --- | --- |
| **Name:** | Undo |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  PLAYER has placed at least one tile this turn |
| **Flow of**  **Events:** | 1. PLAYER selects the undo option  2. SYSTEM refreshes the rack and board to match the state of the game. |
| **Exit**  **Conditions:** | The last tile moved is now in its original position. |

|  |  |
| --- | --- |
| **Name:** | Redo |
| **Actor:** | PLAYER |
| **Entry**  **Conditions:** | It is the initiating PLAYER’s turn  The PLAYER’s last action was an Undo |
| **Flow of**  **Events:** | 1. PLAYER selects the redo option.  2. SYSTEM refreshes the rack and board to match the state of the game. |
| **Exit**  **Conditions:** | The tile moved by the previous Undo action is returns to its originally intended destination. |

4.2 Object Model

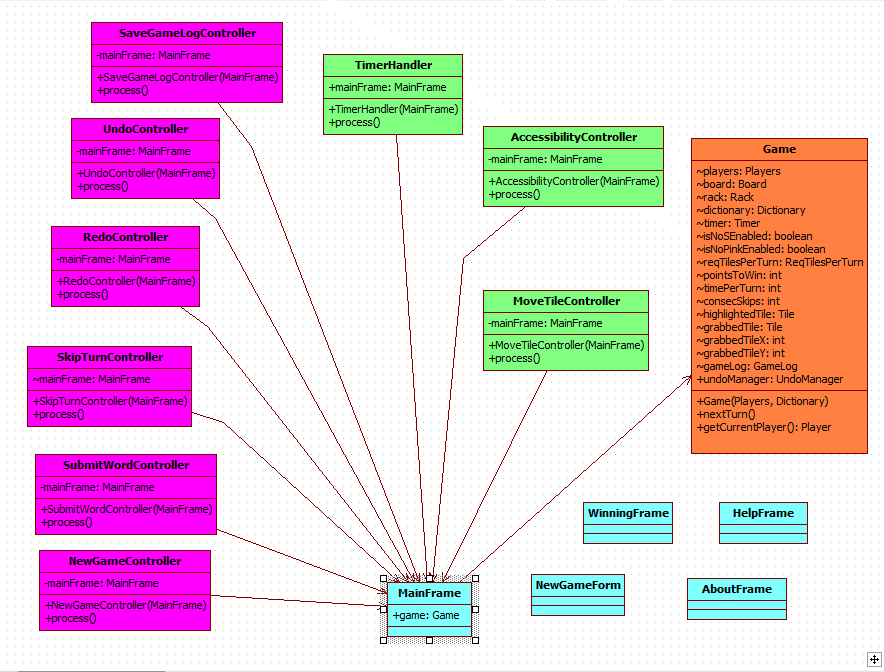


Fig 2. Object Model Boundary controllers

### 4.2 User Interface – Navigational Paths and Screen Mock-ups



# Change History

As this document changes, it will be made available with Microsoft Track Changes turned “on” so you will be able to see all the updates that have occurred.

|  |  |
| --- | --- |
| **Version** | **Description** |
| 20100128.1421 | Initial Version released to the class |
|  |  |

# References

Java Printing Resources [http://articles.techrepublic.com.com/5100-10878\_11-6150617.html]