**FIJI: Entertainment Content Organizer**

**Software Design Specification**

**Version <1.0>**

**Aaron Mai**

**Elizabeth Ortiz**

**Ryan Pham**

**Robert Tang**

**Jaeden Thurston**

**Lead Software Engineers**

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <n/a> | Version 1.0 | <name> | n/a |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Barbara Hecker | Professor/Boss | n/a |
|  | <name> | <title> | n/a |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**Revision History 1**

**Document Approval 1**

**1. Introduction 4**

1.1 Goals and Objectives 4

1.2 Statement of Scope 4

1.3 Software Context 4

1.4 Major Constraints 4

**2. Data Design 4**

2.1 Internal Software Data Structure 4

2.2 Global Data Structure 4

2.3 Temporary Data Structure 5

2.4 Database Description 5

**3. Architectural and Component-Level Design 6**

3.1 Program Structure 6

3.2 Software Interface Description 6

**4. User Interface Design 7**

4.1 Description of the User Interface 7

4.2 Interface Design Rules 8

4.3 Components Available 9

4.4 Development System Description 9

**5. Restrictions, Limitations, and Constraints 10**

**6. Testing Issues 10**

6.1 Classes of Tests 10

6.2 Expected Software Response 10

6.3 Performance Bounds 21

6.4 Identification of Critical Components 22

**7. Appendices 22**

7.1 Requirements Traceability Matrix 22

7.2 Packaging and Installation Issues 24

7.3 Supplementary information 24

**1. Introduction**

**1.1 Goals and objectives**

The purpose of this document is to describe the implementation of Fiji’s software described in the Fiji Software Requirements. Fiji software is designed to provide an online platform to connect users together and a stable interface for indie content creators to share their products with other users on the market.

**1.2 Statement of scope**

Fiji provides a simple user interface that connects content creators to users and open marketplaces. It also organizes and executes content across all platforms while integrating a social media aspect with account profiles with forums, groups, content gifting, and chat systems.

**1.3 Software context**

Fiji will be offered on an official website for free. Development and maintenance would technically be free of cost, but running the server would have some sort of fee put towards it. However, project funding will be open for the taking if the opportunity has been given by sponsors. Future development plans will be based on the features that might not have made it in the initial release of the application. If all of these features are included, there are several experimental features that will potentially be incorporated. These features are not covered in this document.

**1.4 Major constraints**

The most crucial constraint for this project is the hardware aspect for Fiji’s server. Assuming that there will be a vast number of digital products being uploaded to the servers with various file sizes with the possibility that several users are downloading/uploading simultaneously, servers will need to keep up with this load by having plenty of hard drive space and an efficient RAM to process user requests.

**2. Data Design**

**2.1 Internal software data structure**

Fiji’s internal structure is divided into two parts: server-side and client-side. On the client side, data will reside locally in memory. The data on the client will be requested from the server at application initialization and refreshed as necessary based on user actions. The data structure on the server will reflect the structure of the local client in terms of member fields of the classes. The server will be implemented using PHP and storage of user information will be achieved using a MySQL database.

**2.2 Global data structure**

The global data structure of this application depends on the database. The database structure shows the data involved in the application and the client will not directly access the database in the sense that the client will route user requests to the server.

**2.3 Temporary data structure**

The data objects created locally will only exist for the duration of time that the application is running, and will eliminated after the duration has ended. Objects that exist within the exchange between server and client will behave in the same manner in the fact that the transactions will provide object existence and will remove such objects after the transaction has been completed.

**2.4 Database description**

|  |
| --- |
| **CREATE** **TABLE** `users` (  `userID` int(11) **NOT** NULL auto\_increment,  `username` varchar(30) **NOT** NULL,  `displayName` varchar(30) **NOT** NULL,  `password` varchar(30) **NOT** NULL,  `email` varchar(256) **NOT** NULL,  PRIMARY **KEY** (`userID`) ) |

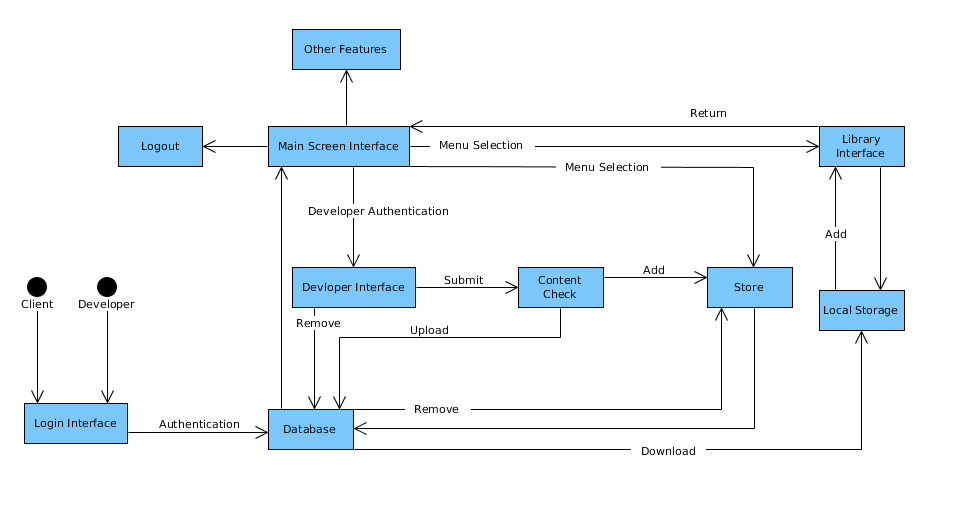
|  |
| --- |
| **CREATE** **TABLE** `products` (  `productID` int(11) **NOT** NULL auto\_increment,  `prodName` varchar(128) **NOT** NULL,  `description` varchar(**MAX**) **NOT** NULL,  `price` int(11) **NOT** NULL,  `uploaded\_by` varchar(30) **NOT** NULL,  `license` varchar(128) **NOT** NULL,  PRIMARY **KEY**(`productID`)  FOREIGN **KEY** (uploaded\_by) **REFERENCES** **users**(userID) ) |

|  |
| --- |
| **CREATE** **TABLE** `reviews` (  `reviewID` int(11) **NOT** NULL auto\_increment,  `reviewedProd` int(11) **NOT** NULL,  `reviewer` int(11) **NOT** NULL,  `review` varchar(**MAX**) **NOT** NULL,  `rating` int(1) **NOT** NULL,  PRIMARY **KEY** (`reviewID`),  FOREIGN **KEY** (reviewedProd) **REFERENCES** **products**(productID),  FOREIGN **KEY** (reviewer) **REFERENCES** **users**(userID) ) |

|  |
| --- |
| **CREATE** **TABLE** `owned\_prods` (  `ownerID` int(11) **NOT** NULL,  `prodOwned` int(11) **NOT** NULL,  FOREIGN **KEY**(ownerID) **REFERENCES** **users**(userID),  FOREIGN **KEY** (prodOwned) **REFERENCES** **products**(productID) ) |

**3. Architectural and Component-Level Design**

**3.1 Program Structure**

**3.1.1 Architecture diagram** 

**3.2 Software Interface Description**

**3.2.1 External machine interfaces**

Internet-enabled computers allow the locally stored FIJI client to access data stored on the FIJI server.

**3.2.2 External system interfaces**

Fiji will interface with MySQL Community Server Database Management System.

**3.2.3 Human interface**

Fiji will interface with humans through the use of I/O devices.

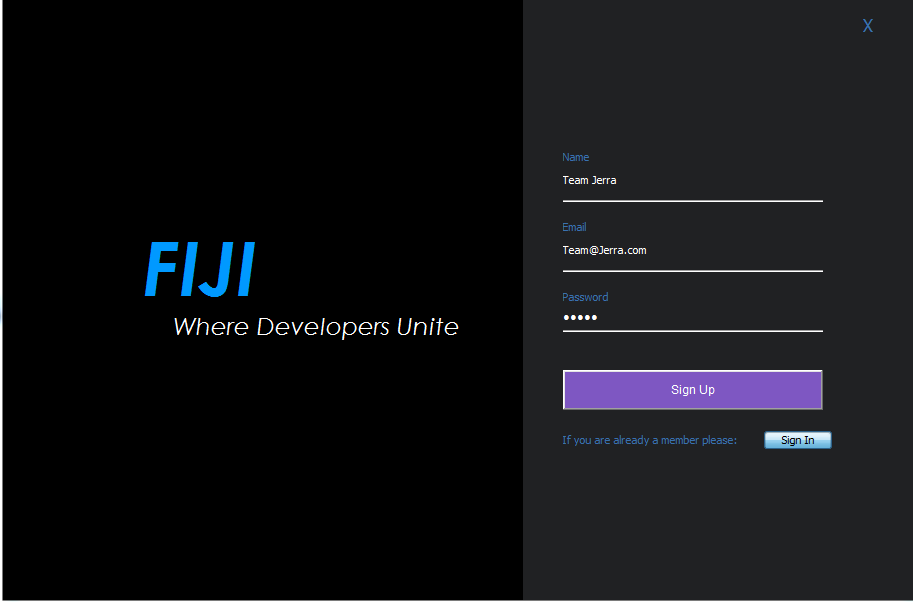
**4. User interface design**

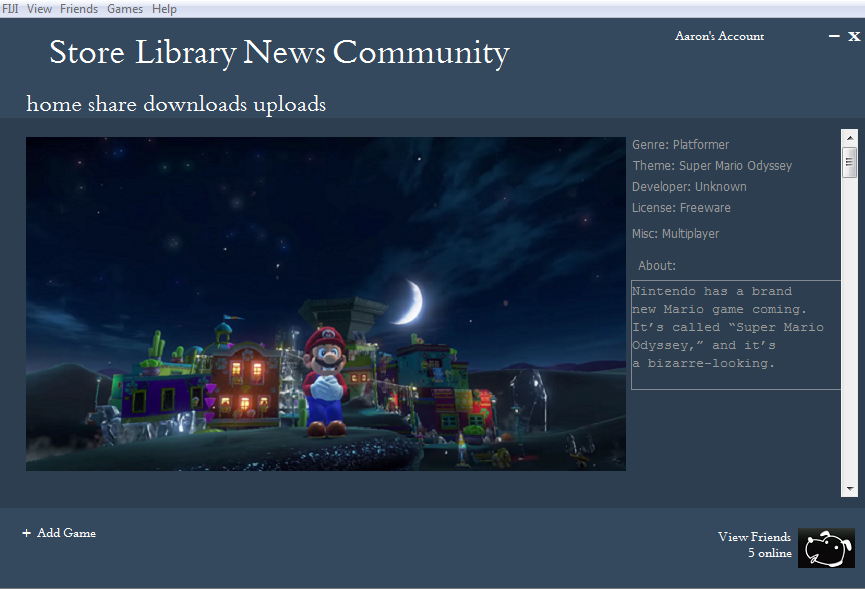
The user interface consists of a set of menu items where the user can interact with data on Fiji’s server. These menus include a “Store”, “Community”, “News”, “Library”, and “Friends” menus. Each menu is simplistic in the fact that user requests will be handled through mouse clicks, unless shown otherwise in occasions where text fields may be provided to enter information.

**4.1 Description of the user interface**

Each menu will consist of various GUI components, such as buttons, labels, text fields, and list objects. These components will be arranged in such a way that the user will be able to quickly understand the purpose of each menu and perform intended tasks efficiently.

**4.1.1 Screen images**





**4.1.2 Objects and actions**

All screen objects and actions are identified.

**4.2 Interface design rules**

4.2.1 Consistency

Consistent sequences of actions should be required in situations. Functionality should be used to access settings, news, and help communities, and consistent help should be accessible throughout. Each menu in FIJI: all of the major options are displayed at the top of the menu (Ex. “Store”, “Library”, “News”, “Community”, “account settings”, etc). Returning to “home” screen, sharing, downloading, and uploading are displayed in the bottom of the major options of the menu. Adding games, or viewing who is online in the message box is displayed on the bottom.

4.2.2 Shortcuts

Users would want to reduce the number of interactions and to increase the pace of interaction. Function keys, macros, and hidden commands are very helpful to an expert user. To make it easier for users, there will be “add game button” or “messenger button” to access the chat function and upload a game to be used at ease.

4.2.3 New Information

For all new information, there will be a menu option that allows a person to check up on information that has just been released. A community option will also allow people to see more information from another person’s point of view.

4.2.4 Action Reversal

This action will allow users to reverse any human-made errors that was possibly made in the system. If a user had made an incorrect purchase, he/she can deny the transaction if the product has not been used or if the item was just bought.

4.2.5 Error Handling

The system will be design so the user cannot make any serious errors. If an error was made, the system will be able to detect the error and offer ways on handling the error.

**4.3 Components available**

Netbeans provides plenty of GUI components. The components that FIJI uses

include the following:

4.3.1 Hierarchy view

View allows FIJI to display all of its GUI components and all of FIJI’s menus

components will be easily accessible through this function.

4.3.2 Free Layout

The Free Layout component allows the placements of components

anywhere as long as it is within its bounds.

4.3.3 Buttons

The button component allows users to interact with each menu by clicking it. When the button is clicked it will be handled by an event handler inside its

functionality.

4.3.4 Viewing text information

The viewing text component will allow information to be in a text box that will

display in a linear manner. FIJI will use text boxes to display messages and information.

4.3.5 Viewing List

Listing component will have a scroll bar to display new content which

allows the user to have easier access to looking at any new information.

**4.4 Development system description**

The user interface development system is provided by the ADT plugin for Netbeans. Our UIDS generate XML code from a graphical drag and drop menu designer. The graphical components are limited because the buttons would have to be done by modifying code directly. Each menu will be implemented in Java and consist of Event Handlers and any menu class members.

**5. Restrictions, limitations, and constraints**

Due to time limitations, optional features previously mentioned in the SRS document have been omitted from discussion in this document. This decision was made because it is highly unlikely that these features will be implemented within the allotted completion time for this project. With FIJI’s highly modular design and organization of data – as well as the unlimited server side expansion potential – it would be arguably easier to implement these optional features at a later date than incorporate them into the first design. Another limitation of the software is the lack of a web interface. One major constraint that is mentioned numerous times in this document as well as the SRS is the requirement for the user to have internet access on their computer. This access is essential because all data altering actions make a call to the server in order to complete that action. The user of this client is required to have an email address, which is another constraint. This is used to communicate with user about changes to accounts, security related information, and purchases.

**6. Testing Issues**

To ensure our production software as a whole is fully functional, we will do a series of test and functionality checks on each class. After the program has been assembled, it will be tested overall to ensure optimal synergy between components.

**6.1 Classes of tests**

**6.1.1 White Box Testing**

Since FIJI is programmed in Java, the first series of tests will be to verify that our classes work. Making sure that attributes can be properly modified with methods. And methods will properly change attributes if needed. And if certain classes - such as the utility classes in FIJI , interact properly with other classes.

**6.1.2 Black Box Testing**

Afterwards, once all FIJI components are checked and verified, Black Box testing will be used to test our software as a whole(fully assembled). Running through many different situations. This includes checking for proper error checking when inputs are incorrect.

**6.2 Expected Software Responses**

**6.2.1 White Box Expected Results**

**User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Username** | **String** | | **A unique String that suffices as the User’s login credentials and reference in Database.** |
| **Password** | **Char** | | **Combination of unique characters, numbers, symbols.** |
| **Email** | **String** | | **String variable of the User’s email. Also, can suffice as a proper login credential.** |
| **Birthday** | **Integer** | | **Integer value of User’s birthday in MMDDYYY format.** |
| **Rank** | **Integer** | | **A simple Integer Value that indicates if User has permissions to certain Administration commands.** |
| **DisplayName** | **String** | | **A non-unique String that serves as a display name between users. Does not have to be unique and can be altered at any time.** |
| **Boolean Login (String Username, Char Password)** | |  | |
| **Input:** | | **The User’s username and password in string format.** | |
| **Output:** | | **True or False** | |
| **Description:** | | **Method returns True if User and Password match data found in database.** | |
| **Void Logout(User A)** | |  | |
| **Input:** | | **A User Object that has been logged in.** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method disconnects the User from the server.** | |
| **Void EditAccount(User A)** | |  | |
| **Input:** | | **A User Object.** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method prompts a menu that allows the User to edit common settings.** | |
|  |  |  |  |

**Settings**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Account** | **Object** | | **Object that contains general settings and/or preferences.** |
| **Friends** | **Array[ Object ]** | | **Object array contain Friends of User.** |
| **Overlay** | **Sub-Object** | | **Object containing Overlay Settings** |
| **Chat** | **Sub-Object** | | **Object containing Chat Settings** |
| **Downloads** | **Sub-Object** | | **Object containing Download settings.** |
| **Account get Account()** | |  | |
| **Input:** | | **Void** | |
| **Output:** | | **Currently Logged In User’s Account Object.** | |
| **Description:** | | **Method returns an Account Object of currently Logged In User.** | |
| **Void setAccount(Account A)** | |  | |
| **Input:** | | **An Account Object** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method changes Account Object being modified.** | |
| **Friends[ ] getFriends()** | |  | |
| **Input:** | | **Void** | |
| **Output:** | | **Returns an Array of Friends.** | |
| **Description:** | | **Method returns a dynamic array of Objects containing “Friends” of current User.** | |
| **Void setFriends(Friends [ ])** | |  | |
| **Input:** | | **An array of Friends** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method updates array with new Friends[ ]. Updated every time a User remove or adds.** | |
| **Overlay getOverlay()** | |  | |
| **Input:** | | **Void** | |
| **Output:** | | **Returns Overlay Object** | |
| **Description:** | | **Method retrieves Overlay Object containing setting of User’s Overlay** | |
| **Void setOverlay(Overlay A)** | |  | |
| **Input:** | | **An Overlay Object** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method updates with new Overlay settings. Possibly change Theme or Disable/Enable Overlay feature.** | |
| **Chat getChat()** | |  | |
| **Input:** | | **Void** | |
| **Output:** | | **Returns Chat Object** | |
| **Description:** | | **Method returns settings of Chat. This also contain logs of chats between Users.** | |
| **Void setChat(Chat A)** | |  | |
| **Input:** | | **A Chat Object** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method updates with new Chat settings** | |
| **Download getDownload()** | |  | |
| **Input:** | | **Void** | |
| **Output:** | | **A Download Object.** | |
| **Description:** | | **Method returns the Download settings of currently associated Account Object. Useful in altering Download server locations.** | |
| **Void setDownload(Download)­­** | |  | |
| **Input:** | | **A Download Object.** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method changes Download Object of Account.** | |
|  |  |  |  |

**Advertisement**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **User\_Preference** | **Boolean** | | **Contains value of User’s preference when it comes to information gathering.** |
| **Information\_Gatherer** | **String** | | **A large string of condensed information gathered. Used in EnableInfoGatherer() function. Third party application will take advantage of this information.** |
| **Void EnableInfoGatherer(Boolean)** | |  | |
| **Input:** | | **True Boolean Value(CheckBox)** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method indicates that the User approves of information gathering.** | |
| **Void DisableInfoGather(Boolean)** | |  | |
| **Input:** | | **False Boolean Value(Checkbox)** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method Indicates User does not approve of Information gathering, therefore it will be excluded from Information Pools.** | |
|  |  |  |  |

**Overlay**

|  |  |
| --- | --- |
| **Void Display()** |  |
| **Input:** | **Void** |
| **Output:** | **Void** |
| **Description:** | **Method displays transparent version(s) of main user interface and all other graphical user interfaces.** |
| **Void RunSelection(Boolean Val)** |  |
| **Input:** | **Boolean Value.** |
| **Output:** | **Void** |
| **Description:** | **Method Selects between Fullscreen and Windowed mode for GUI.** |

**User Authentication**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Login ID** | **Int** | | **Unique ID associated with User’s Account** |
| **Password** | **Char** | | **Same value as Password variable from User class.** |
| **Temp Password** | **Char** | | **In the case of Account recovery, a temporary password will be given.** |
| **Email** | **String** | | **Same as User Email variable. Used for sending temporary password and PIN too.** |
| **Int RandPin(Login ID)** | |  | |
| **Input:** | | **Login ID** | |
| **Output:** | | **6 Digit Int.** | |
| **Description:** | | **Method used as a recovery mechanism or email verification method. This function randomly generates a 6 Digit Pin that is sent to the User via Email.** | |
| **Char RandPassword(Login ID)** | |  | |
| **Input:** | | **Login ID** | |
| **Output:** | | **8-14 Characters** | |
| **Description:** | | **Method used to create random password for password recovery. In the case of a compromised account, this method will send a randomly generated password to the email associated with the Login ID.** | |
|  |  |  |  |

**Verify Permissions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Username** | **String** | | **A unique String that suffices as the User’s login credentials and reference in Database.** |
| **Rank** | **Integer** | | **A simple Integer Value that indicates if User has permissions to certain Administration commands.** |
| **Boolean isAdmin(Username A)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **Boolean Value** | |
| **Description:** | | **Method will call getRank() and return True if Rank corresponds with the Admin Rank.** | |
|  |  |  |  |

|  |  |
| --- | --- |
| **Int getRank(Username)** |  |
| **Input:** | **String Username** |
| **Output:** | **Integer (0 - 2).** |
| **Description:** | **Method will return a number between 0-2. 0 is an Admin. 1 is a Developer. 2 is a Client (i.e. Default User).** |

**Manage**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Username** | **String** | | **A unique String that suffices as the User’s login credentials and reference in Database.** |
| **Rank** | **Integer** | | **A simple Integer Value that indicates if User has permissions to certain Administration commands.** |
| **Password** | **Char** | | **The password associated with the User’s account.** |
| **Email** | **String** | | **The email associated with the User’s account.** |
| **Content** | **Object [ ]** | | **A Content Object array that contains all pertinent information about the content (Genre, Price, Title, etc.) and all the content owned by the User.** |
| **String getUsername(String Email)** | |  | |
| **Input:** | | **String containing Email.** | |
| **Output:** | | **String of User’s Username** | |
| **Description:** | | **Method will retrieve the User’s Username from the Database. The Username that was just used to specify changed, will no longer be valid.** | |
| **Void setUsername(String name)** | |  | |
| **Input:** | | **String** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method redefines Username from String Argument. Username will still be checked for properness.** | |
| **Int getRank(String Username)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **Integer value of User’s Rank (0-2)** | |
| **Description:** | | **Method retrieves User’s Rank. This will be used to check User’s permission access.** | |
| **Void setRank(Int Rank)** | |  | |
| **Input:** | | **Integer value** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method sets the User’s Rank to a new value. It will be checked so the value cannot exceed 2 or go below 0.** | |
| **String getPassword(String Username)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **String of Password** | |
| **Description:** | | **Method returns the password of the User specified. The password will be completely visible.** | |
| **Void setPassword(String Username)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **Void** | |
| **Description:** | | **This changes the password of the specified User. Notice using the function alone is not the same as retrieve a forgotten password. This is a Admin level command and can change password without User’s consent.** | |
| **String getEmail(String Username)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **String containing the User’s email.** | |
| **Description:** | | **Method retrieves the email of the User.** | |
| **Void setEmail(String Username)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method changes the email of the User.** | |
| **Object [ ] getContent(String Uername)** | |  | |
| **Input:** | | **String Username** | |
| **Output:** | | **Object Array of all content owned by User** | |
| **Description:** | | **This Method returns an array of Objects contain all the Objects owned by the User.** | |
| **Void setContent(Object [ ])** | |  | |
| **Input:** | | **An array of Objects** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method updates the Object array with new Object Array. Will be used when new Content is purchased or removed. Or at the Admin’s discretion.** | |

**Message**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | | **Description** |
| **Recipient** | **String** | | **Recipient is a String data type equivalent to Username.** |
| **Sender** | **String** | | **Sender is a String data type equivalent to Username** |
| **Message** | **Object** | | **Message is a Object data type that contains information about the message. Including the Sender, Recipient, Time, and most importantly the message body.** |
| **Time** | **String** | | **Time is a string data type containing the toString(Date()). Will be used as a timestamp.** |
| **String getRecipient(Message info)** | |  | |
| **Input:** | | **An Object containing the information of the message.** | |
| **Output:** | | **A String containing the name of the Recipient.** | |
| **Description:** | | **Method takes an Object message and returns the Recipient in String format.** | |
| **String getSender(Message info)** | |  | |
| **Input:** | | **An Object containing the information of the message.** | |
| **Output:** | | **A String containing the name of the Sender.** | |
| **Description:** | | **The method takes an Object message and returns the Sender in String format.** | |
| **Void sendMessage(String Sender, String Recipient)** | |  | |
| **Input:** | | **Two strings, containing Usernames of the Sender and Recipient.** | |
| **Output:** | | **Void** | |
| **Description:** | | **Method proceeds to take the Message Object and sends it from the Sender’s memory to the Recipient inbox.** | |

**6.2.2 Black Box Expected Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Input** | **Expected Result** | **Actual Result** |
| **1** | **Enter user credentials** | **Display Fiji’s dashboard** |  |
| **2** | **Enter invalid user credentials** | **Throw exception and allow user to retry** |  |
| **3** | **Select Friends List** | **Display window with all friends user has added** |  |
| **4** | **Select friend from Friends List** | **Display chat box** |  |
| **5** | **Select Add Item to Cart in store** | **Item will be saved and added to “cart” to be paid for later** |  |
| **6** | **Adding non-existent item to Cart** | **Throw 404 error and give user option to return to store page** |  |
| **7** | **Select Add to Wishlist** | **Item will be saved and added to “wishlist”** |  |
| **8** | **Adding non-existent item to Wishlist** | **Throw 404 error and give user option to return to store page** |  |
| **9** | **Selecting a gift recipient** | **Item will be selected as “Purchase as a gift” with a pop up display window of friends list** |  |
| **10** | **Selecting Uploading Content** | **Window display will show which file you want to upload** |  |
| **11** | **Selecting Downloading Content** | **A download display will pop up with the time it will take to finish** |  |
| **12** | **Selecting Store Option** | **Displays featured, new, and available content** |  |
| **13** | **Selecting Library Content** | **Displays a list of content that is owned** |  |
| **14** | **Selecting Account Recovery** | **Email will be sent with a random password/Pin generator** |  |
| **15** | **Selecting Account Setting** | **Shows User Information** |  |

**6.3 Performance bounds**

Since the content in our database and market will vary greatly, the User must ensure that their machine and bandwidth meet specifications. We will not be responsible if you User’s purchases content that they are ill-equipped to execute. However, the application itselfs is very minimal in resource usage. We do not expect our program standalone will take up much memory and its processing to be virtually negligible. One of the great bounds we will encounter are downloads and uploads to our system. The exchange of data between our servers and clients should have a very large bandwidth. If we expect to operate on a national scale or even global scale we should have servers across the global to ease the burdens on our limited resources. However, this can be a remedy for a more successful FIJI. In the meantime, our software will be limited to our area.

**6.4 Identification of critical components**

The payment system and servers will be the two most critical components of our FIJI software. Since the purpose of our application is to serve as a median between consumers and developers, we must ensure that monetary transactions between everybody is secure and valid. We operate in a time where these transactions and pertinent information can be very susceptible to interceptions by malicious software and hackers. Furthermore, the functionality of our software would be useless without proper online capabilities. FIJI depends on the exchange of information between the server and clients. Without connectivity you have a unprofitable content explorer. And the social features included would be useless.

**7. Appendices**

**7.1 Requirements Traceability matrix**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Functional Requirement** | **Design** |
| 1 | Log-In | User |
| 2 | Log-In | User |
| 3 | Messaging | Settings |
| 4 | Messaging | Settings |
| 5 | Store | Store |
| 6 | Store | Store |
| 7 | Wishlist | Store |
| 8 | Wishlist | Store |
| 9 | Gifting | Store |
| 10 | Uploading/Downloading | Store |
| 11 | Uploading/Downloading | Store |
| 12 | Store | Store |
| 13 | User Accounts | Manage |
| 14 | User Authentication | User Authentication |
| 15 | Settings | Settings |
| Login | Log-In | User |
| Logout | Log-In | User |
| Edit Account | User Accounts | User |
| Get Account | Settings | Settings |
| Set Account | Settings | Settings |
| Get Friends | Settings | Settings |
| Set Friends | Settings | Settings |
| Get Overlay | Settings | Settings |
| Set Overlay | Settings | Settings |
| Get Chat | Settings | Settings |
| Set Chat | Settings | Settings |
| Get Download | Settings | Settings |
| Set Download | Settings | Settings |
| Enable Info Gatherer | Advertisements | Advertisements |
| Disable info Gatherer | Advertisements | Advertisements |
| Display | Overlay | Overlay |
| Run Selection | Overlay | Overlay |
| Rand Pin | User Authentication | User Authentication |
| Rand Password | User Authentication | User Authentication |
| Is Admin | Log-In | Verify Permissions |
| Get Rank | Administration | Verify Permissions |
| Get Username | Administration | Manage |
| Set Username | Administration | Manage |
| Get Rank | Administration | Manage |
| Set Rank | Administration | Manage |
| Get Password | Administration | Manage |
| Set Password | Administration | Manage |
| Get Email | Administration | Manage |
| Set email | Administration | Manage |
| Get Content | Administration | Manage |
| Set Content | Administration | Manage |

**7.2 Packaging and installation issues**

Nothing special should be considered during installation except for the fact that FIJI will be only available for Windows operating system for the moment until all features are established and stabilized. The user must also have the latest version of Java JDK and JRE to have the client function properly.

**7.3 Supplementary Information**

**7.3.1 UML Diagram**

