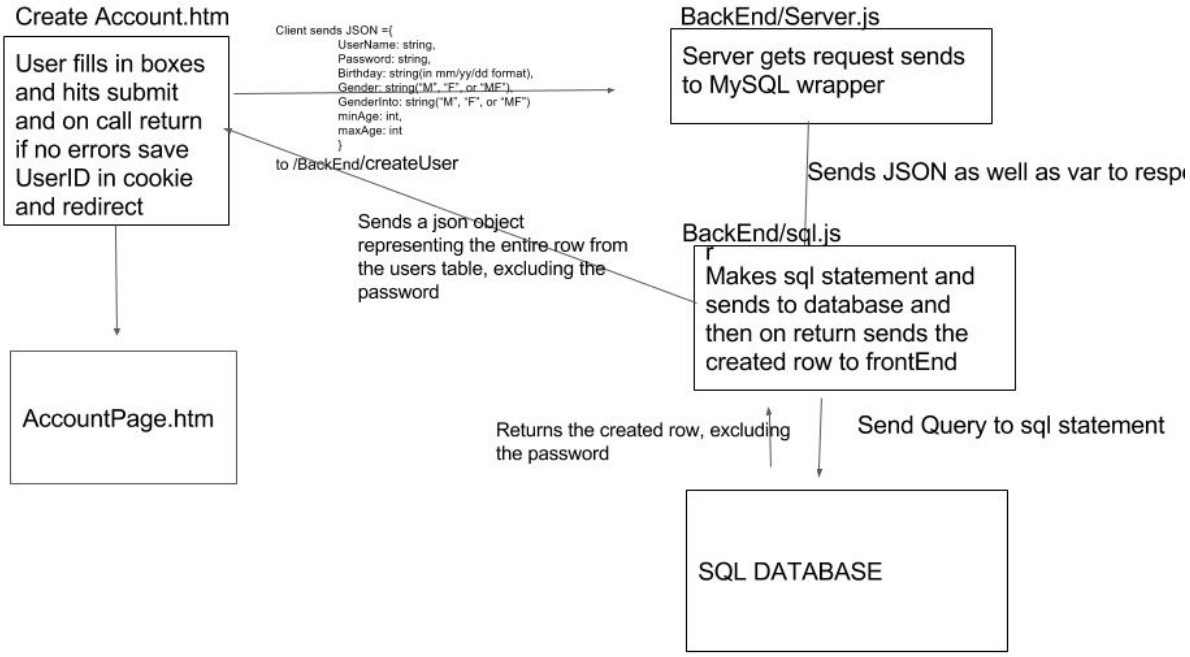
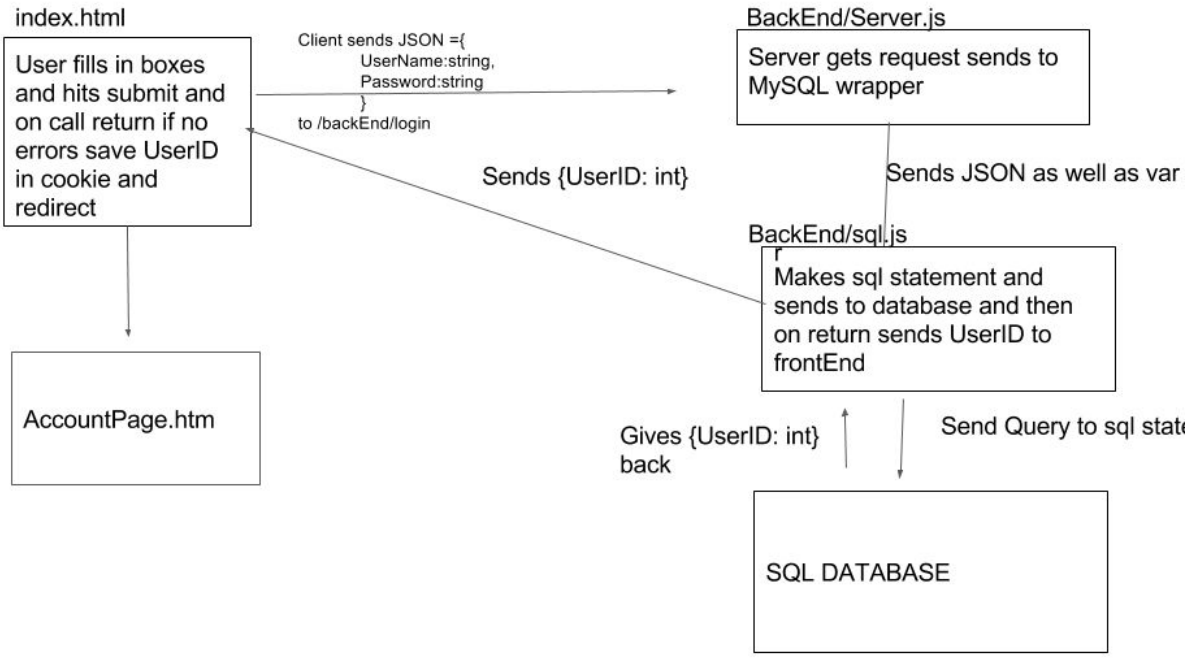
# 1 Classification of Components

## 1.1 Define all components

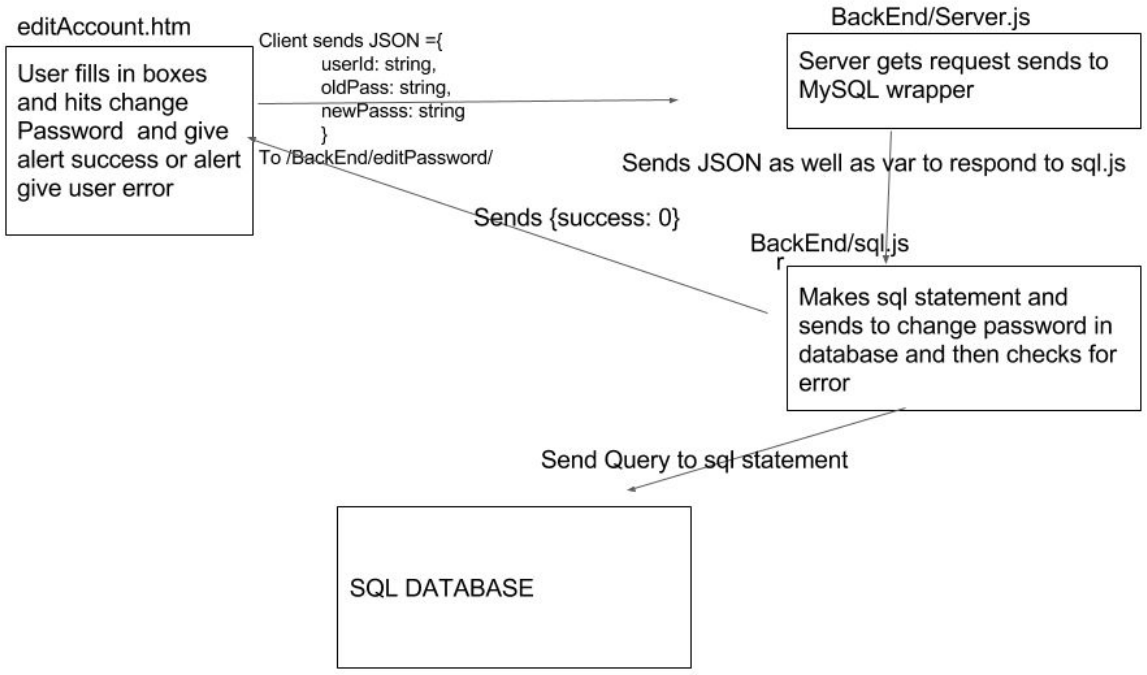
### A. - Create Account



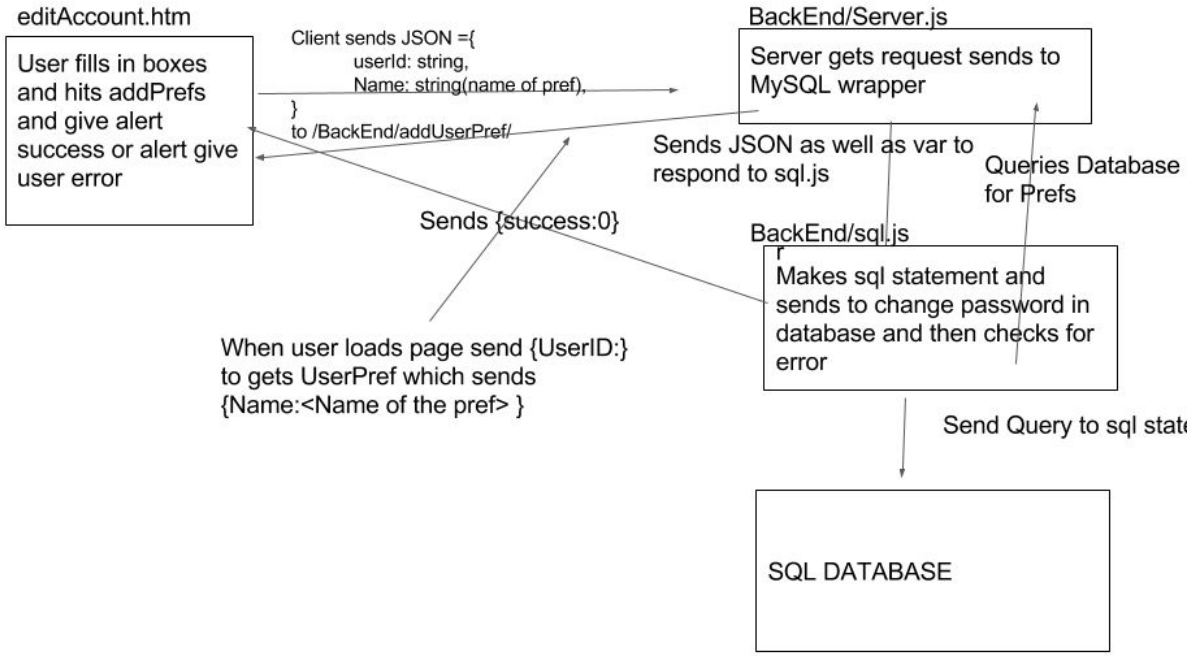

B.



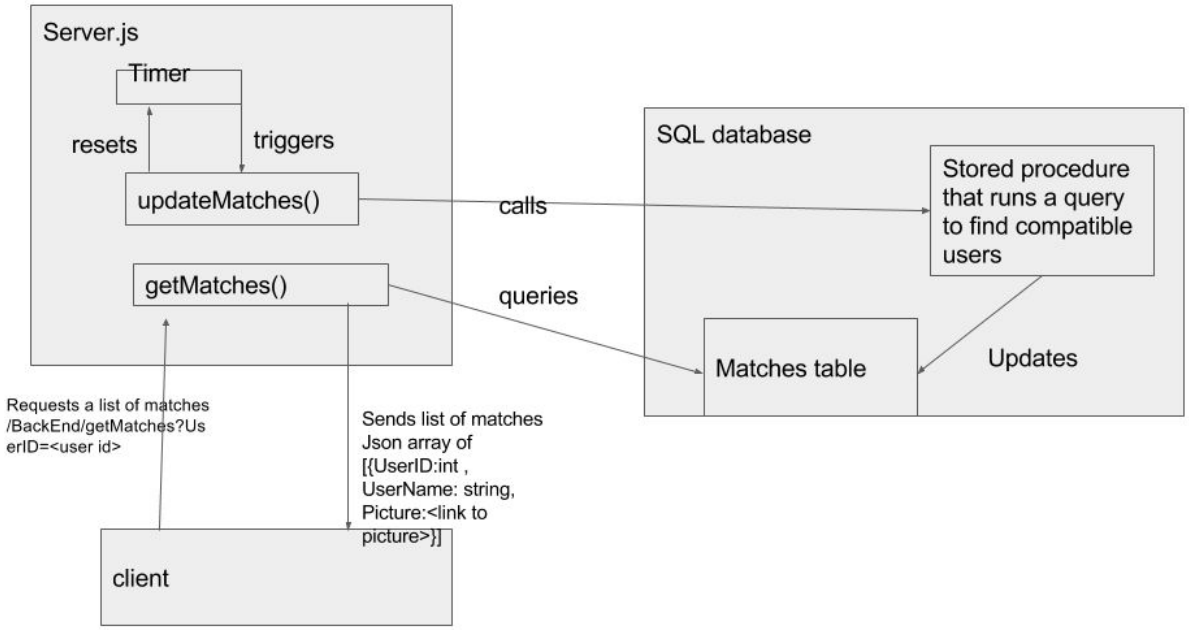

C.


editAccount.htm
User fills in boxes and hits change Password and give alert success or alert give user error
Client sends JSON ={
userId: string,
oldPass: string,
newPasss: string
}
To /BackEnd/editPassword/
BackEnd/Server.js
Server gets request sends to MySQL wrapper
Sends JSON as well as var to respond to sql.js
Sends {success: 0}
BackEnd/sql.jsr
Makes sql statement and sends to change password in database and then checks for error
Send Query to sql statement
SQL DATABASE

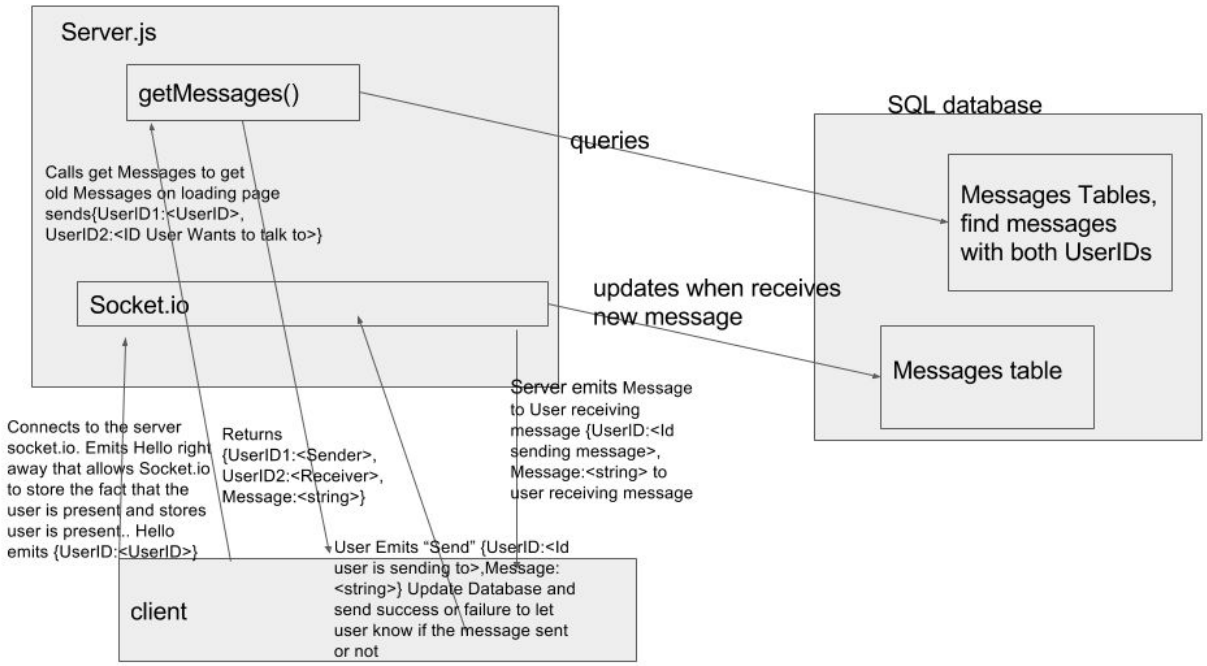
D.


editAccount.htm
User fills in boxes and hits addPrefs and give alert success or alert give user error
Client sends JSON ={
userId: string,
Name: string(name of pref),
}
to /BackEnd/addUserPref/
BackEnd/Server.js
Server gets request sends to MySQL wrapper
Sends JSON as well as var to respond to sql.js
Queries Database for Prefs
Sends {success:0}
BackEnd/sql.jsr
Makes sql statement and sends to change password in database and then checks for error
When user loads page send {UserID:} to gets UserPref which sends {Name:<Name of the pref> }
Send Query to sql state
SQL DATABASE

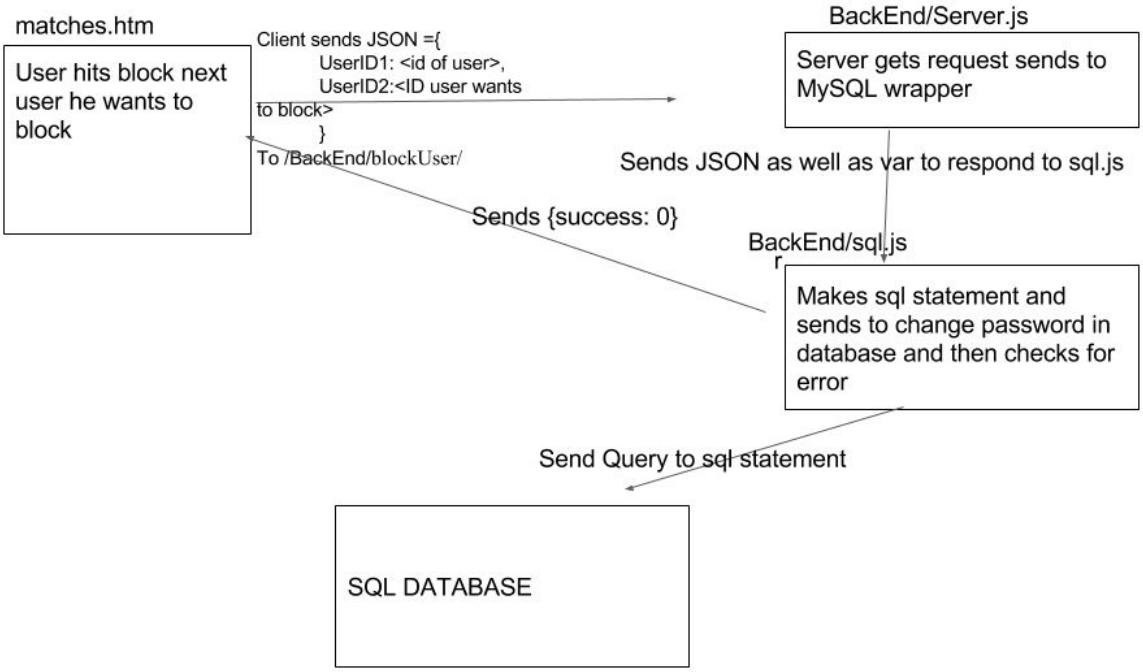
E.


Server.js
Timer
resets
triggers
updateMatches()
getMatches()
calls
queries
SQL database
Stored procedure that runs a query to find compatible users
Matches table
Updates
Requests a list of matches /BackEnd/getMatches?UserID=<user id>
Sends list of matches Json array of [{UserID:int , UserName: string, Picture:<link to picture>}]
client

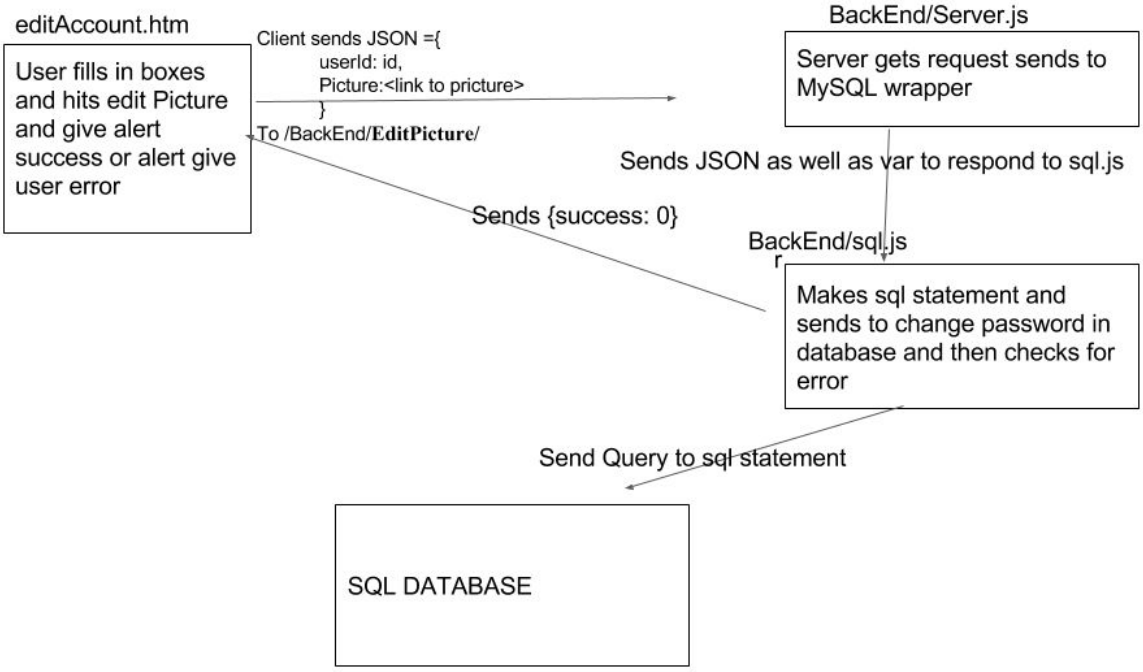
F.


Server.js
getMessages()
Calls get Messages to get old Messages on loading page sends{UserID1:<UserID>, UserID2:<ID User Wants to talk to>}
Socket.io
queries
SQL database
Messages Tables, find messages with both UserIDs
updates when receives new message
Messages table
Server emits Message to User receiving message {UserID:<Id sending message>, Message:<string> to user receiving message
Connects to the server socket.io. Emits Hello right away that allows Socket.io to store the fact that the user is present and stores user is present.. Hello emits {UserID:<UserID>}
Returns {UserID1:<Sender>, UserID2:<Receiver>, Message:<string>}
User Emits "Send" {UserID:<Id user is sending to>,Message: <string>} Update Database and send success or failure to let user know if the message sent or not
client

G.




H.




1. 2 We used Bottom up for part one of this sprint we tested each base module of our program, We then for this part tested the base with the node.js server running as well, making this bottom-up testing

2.1

We decided to create our scripts to run tests instead of using a given framework. We did research into a few different types,such as nemo, and decided for our simple app it would be just as easy to write our own test cases and just run them with bash scripts. These test scripts can be found in the BackEnd/tests/ file, but needs to be ran in the BackEnd file. Our strategy was to go through the test log and test what was needed. If you run node server.js you can then run Backend/tests/testClient to run through all our integration tests for the different parts of server calls. On top of this we used test node server.js as the backend while we ran testMessage2.js, testMessage3.js, testMessage1.js to test the messaging part of the system.

| Module | Component A - Create Account |
|---|---|

## Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Issue with min age where if it is empty it cause issues on the server | 2 | Check if value is not a int and if it is not change min age to 18 and max age to 120 |

## Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | When adding pictures we broke the form to add picture | 1 | We gave the form an id to make we were addressing the right form |
| 2 | Test0 failed, test case will not allow a creation of an account | 3 | Updated test case to fix change in create User |
| 3 | Would not allow you to register if you are older than 18 | 1 | Changed if statement to handle condtion |
| 4 | Some test cases are failing because the password does not meet the newly-introduced minimum length | 5 | Increased the length of the passwords in those test cases. |

| Module | Component B - Create Login in |
|---|---|

Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Loging out does not remove cookie, completly still causes errors | 3 | TODO, have not fixed yet |

Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Did not find any errors in incremental testing | | |

| Module | Component C - Edit Password |
|---|---|

Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Did not find any errors in incremental testing | | |

Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Did not find any errors in incremental testing | | |

| Module | Component D - User Prefs |
|---|---|

## Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Get User prefs was set to a post not a get. | 2 | Changed it to a get |
| 2 | All addUserPref tests are failing. Most are receiving a body of {Error: -2}, the others are receiving a body of {success: 0} | 1 | For the test cases that were receiving {success: 0}, body was a string but we were treating it like a JSON object inside the test script. It was fixed by using JSON.parse.<br><br>For {Error:-2}, changed the test cases to expect -2 like they're supposed to |
| 3 | Fix getUserPref return Success instead of rows | 2 | Changed callback to send rows instead of success |
| 4 | addUserPref allows users add null, numbers, and empty strings as preferences. | 3 | Added input checks at the beginning of addUserPref.<br><br>Changed test cases to allow pref names to be numbers, because there's no real reason to disallow numbers. |

## Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Get User Prefs changed to sending success instead of prefs for some reason. | 2 | Changed callback function on success to send rows instead of success. |

| Module | Component E - Matches |
|---|---|

## Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | GetMatches test cases are all failing | 3 | Updated the test cases so that they use a query string in the URL, instead of sending a form. |

## Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 2 | No users are being matched anymore | 1 | The mySQL stored procedure “update_matches_procedure()” had an error, causing my “INSERT IGNORE” statement to ignore every insertion instead of just duplicates. Adjusted the WHERE clause to make the SQL error go away. |
| 3 | Wrong length of rows when getting matches; expected two, but got zero. | 5 | Updated the test cases so that they use a query string in the URL, instead of sending a form. |

| Module | Component F - Messaging |
|---|---|

## Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Socket versus client in socket.io. The message was sending a message to the sender of the message instead of the receiver. | 2 | Changed socket to client- which is the socket of the user the message is sending to. |

## Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | The chat box is no longer updating when a user sends a message | 2 | Success got changed to lower case |
| 2 | Check to make sure UserID is not blocked before returning messages | 2 | Change getting messages to check if the user is blocked or not. |

| | | | |
|---|---|---|---|
| 3 | Errors were not being checked | 3 | Put a way to check if server is returning an error |

| Module | Component G - Blocking |
|---|---|

Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| | | | |

Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| | | | |

| Module | Component H - Profile Pictures |
|---|---|

Incremental Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| 1 | Not testing if picture is null | 3 | Test if picture link is null to correct |

Regression Testing

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|
| | | | |

//TODO get rid of this

| Defect # | Description | Severity | How Corrected |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 3 | | 3 | |
| 4 | Test1 failed; “allowed creating two users” | 3 | Fixed Test Case which fixed this |
| | Tests 32 and 33 fail; “wrong number of rows when getting language” | 3 | Front-end did not add language so we will look into a latter time |
| | | | |