REQUIREMENTS REPORT AND PROJECT PLAN

CONTENTS

Jade Harris | **12SDD**

Project Development Report (separate submittable) is attached to this pdf. This was completed during the Implementing phase of the SDLC. Headings are:

AIMS AND OBJECTIVES

OVERVIEW

CONNECT provides users (aged18-60) with an effortless Windows software to organise, centralise and manage real-life events. CONNECT simplifies organsation for users, whenever and wherever, by generating event invitations, facilitating communication between attendees, displaying reminders, and maintaining a 'spending' budget. CONNECT is driven by a MySQL database stored on an Amazon-Elastic-Compute-Cloud virtual server (EC2), allowing the software to connect users through the Internet.

PROBLEM DEFINITION

End User Requirements / Interface Objectives

Organising events is often a tedious task, complicated by personal feelings and misaligning schedules. CONNECT overcomes this problem for its general-public end-user with an interface that is:

- **USER-FRIENDLY** thus interface must be **minimalistic, self-documenting, intuitive**
- **and consistent** so it is not overwhelming, especially for inexperienced users
- **ROBUST** as errors could be fatal for the interface and communication with the
- external database. This involves prevention for inputting harmful data
- **CUSTOMISABLE** to **aid accessibility** (e.g colour-blindness) and enhance **user-**
- **experience** with **settings**
- **SMOOTH/HIGH-SPEED** with minimal response-times for **user-efficiency and convenience**
- **REDUCING SUBJECTIVITY** by **calculating priority** (mostWantedOption=3,
    - secondPreference=2…) and using the randomClass

PROBLEM DEFINITION

Boundaries

CONNECT uses **internet access from a Windows-10-OS machine** to **interface between the database server(EC2) and program.**

SOCIAL, ETHICAL AND LEGAL CONSIDERATIONS

Boundaries

CONNECT greatly benefits society by **connecting users in real-life** to **enhance health, wellbeing, and unity** of the community. It also **reduces the cost and waste** of over-booking events. However, CONNECT raises concerns for **potential real-life interaction with a stranger, stalking, data-privacy/security, and identity theft.**

These are also significant ethical implications. Furthermore, the ethicality of communicating information via the Internet could result in **exploitation of neighbour/public networks**. **Unintentional discrimination** is another consideration. However, the accessibility of CONNECT also promotes **diversity and inclusiveness in the community.**

Legal concerns involve **piracy of CONNECT, manipulation/leak of user data (particularly storing in external database) and copyright of graphics used** in the interface. Legal advantages are that CONNECT's **concept is not copyrighted.**

COMPATIBILITY AND PERFORMANCE REQUIREMENTS

Hardware and Software Compatibility and Performance Requirements

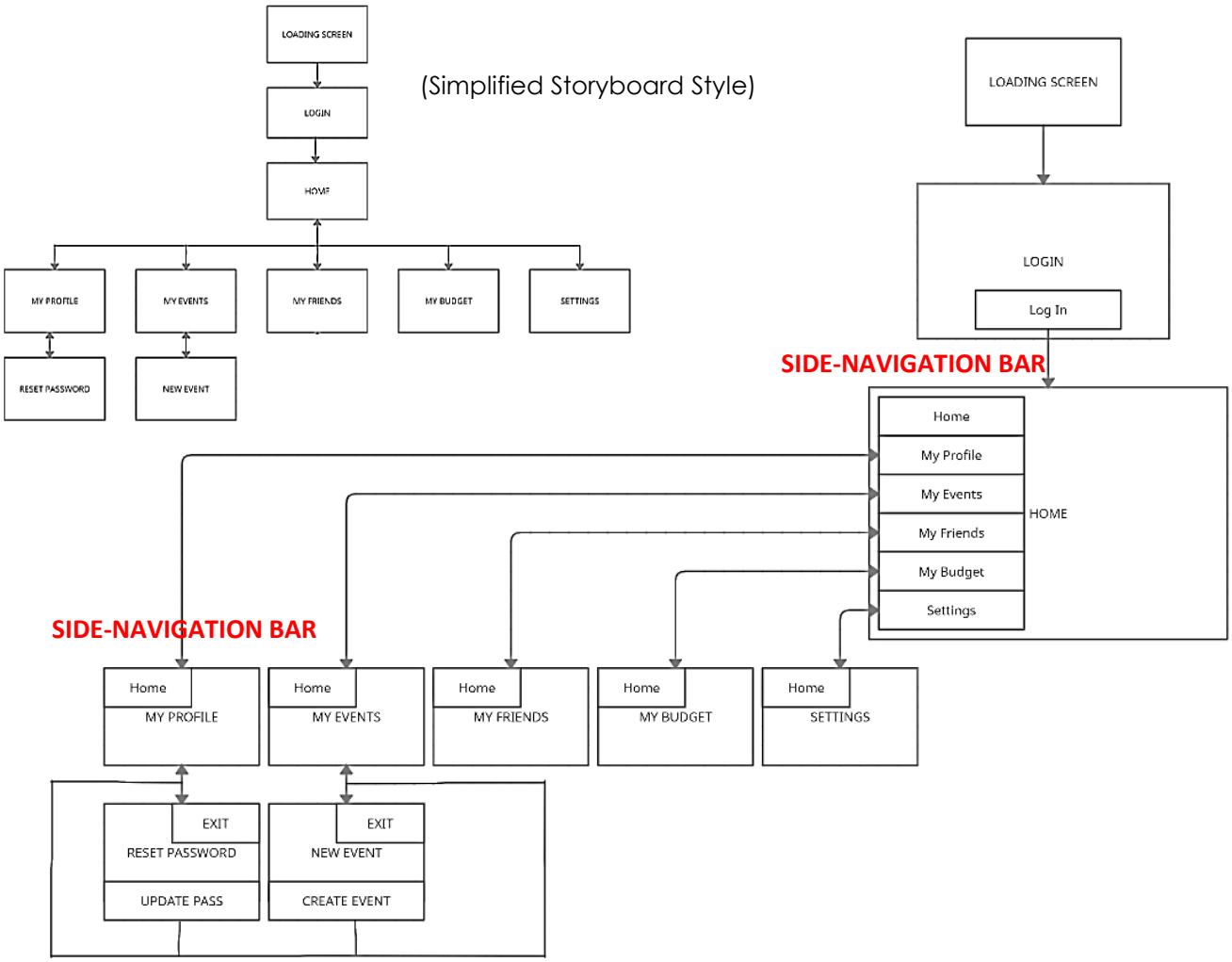
Due to the **small processing power required** to process and send strings to an external server, receive MySQL database table information, and display a minimalistic graphical interface, most systems running **Windows 10 OS** with an **internet connection** are suitable for CONNECT to run **smoothly.**

Peripheral hardware such as **monitor, mouse, and keyboard** (or appropriate substitutes) is required.

DESIGN SPECIFICATIONS / SYSTEM MODELLING

PROTOTYPING TOOLS

Storyboard



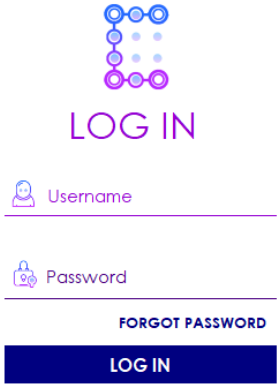
CONNECT first loads in with a brief loading screen. Once the progress bar has complete, a welcome screen automatically loads for the user (where they can either log-in or sign-up then log-in with these new credentials). The user is then taken to the home screen. Here, there is a **side-navigation bar** which can take them to different intefaces/forms of the program. MyProfile provides the option for the user to reset their password, which will open a separate ResetPassword form. MyEvents allows users to create a new event which will load a NewEvent form. The MyFriends, MyBudget and Settings screens can also be loaded from this Home screen. At any time, the user can return to the Home screen by using either the navigation bar or if they have the 'Return to Home' setting on, then they can close the form and return to home.

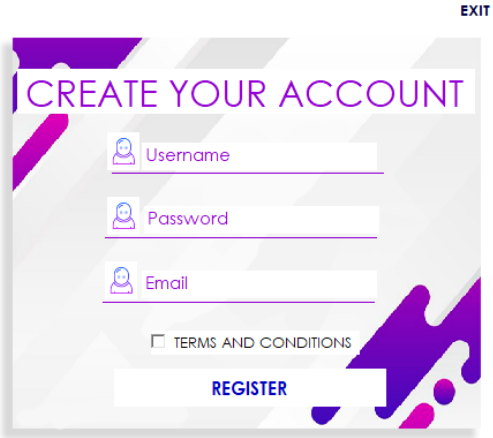
PROTOTYPING TOOLS
(For Interface Mock-ups)

INTERFACE MOCK-UPS *(Made with WinForms in Visual Studio)*

Loading Screen

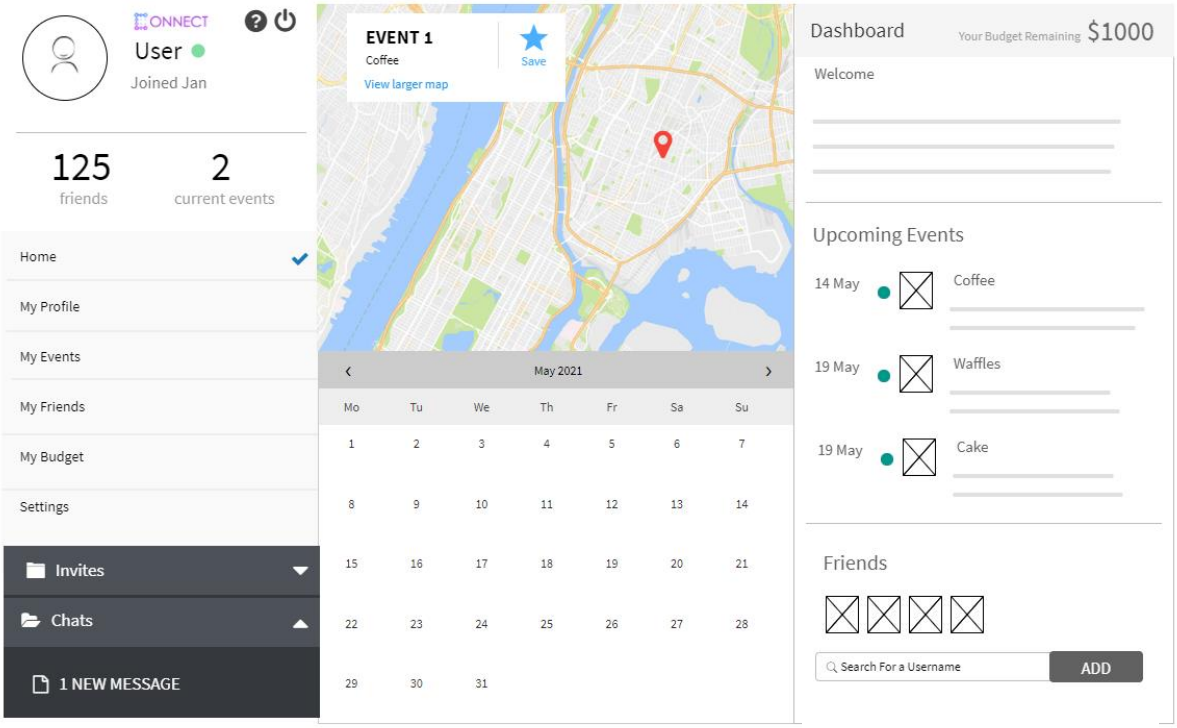
Login-In Screen





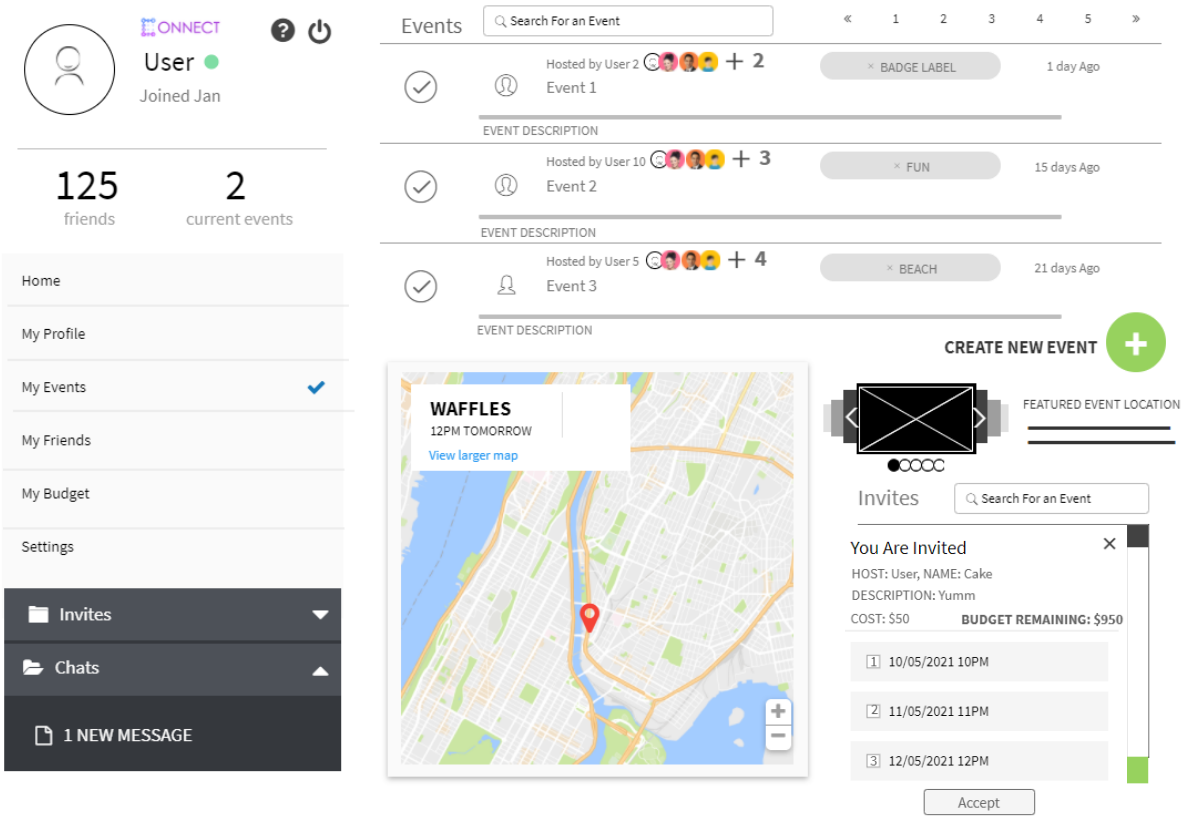
INTERFACE MOCK-UPS *(Made with Interface Designer tool)*

Home Screen



INTERFACE MOCK-UPS
My Profile > Change Password

INTERFACE MOCK-UPS
My Events



INTERFACE MOCK-UPS
New Event

Create New Event

Name: Enter Name...

Description: Enter Short Description

Invitees: Search for Usernames On Friends List...

Location: Search for Locations...

Cost for Location: $

10/05/2021 Enter Times...

11/05/2021 Enter Times...

12/05/2021 Enter Times...

May 2021

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

- [x] Create Event Group-Chat
- [x] Private
- [x] Allow Users to Invite Other Friends
- [x] New Users Can Alter Invite Preferences

Send Invites

INTERFACE MOCK-UPS
My Friends

CONNECT

User

Joined Jan

125 friends

2 current events

Home

My Profile

My Events

My Friends

My Budget

Settings

Invites

Chats

1 NEW MESSAGE

FRIENDS LIST

User 1
User 2
User 3
User 4
User 5
User 6
User 7
User 8
User 9
User 10
User 11
User 12
User 13
User 14

+111

Chats

Search For a Chat

Recent Chats

Upcoming Event Chats

Private Messages

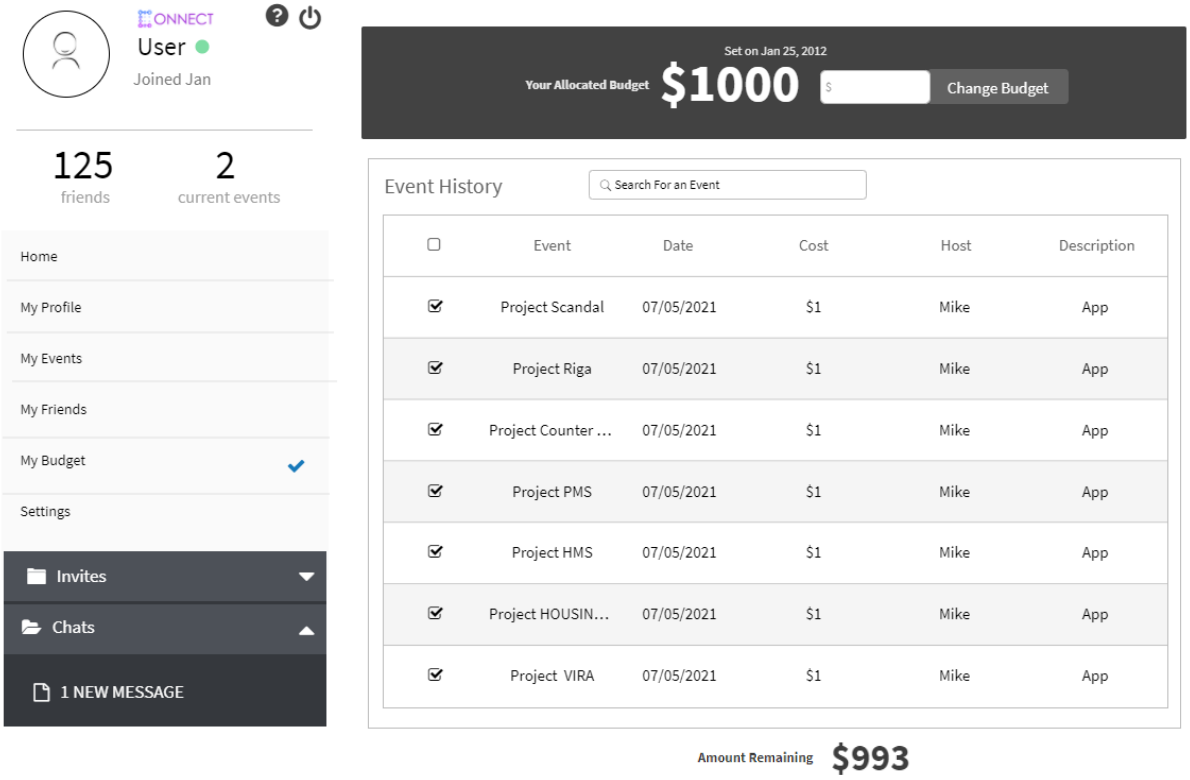
Recents

Open Conversation

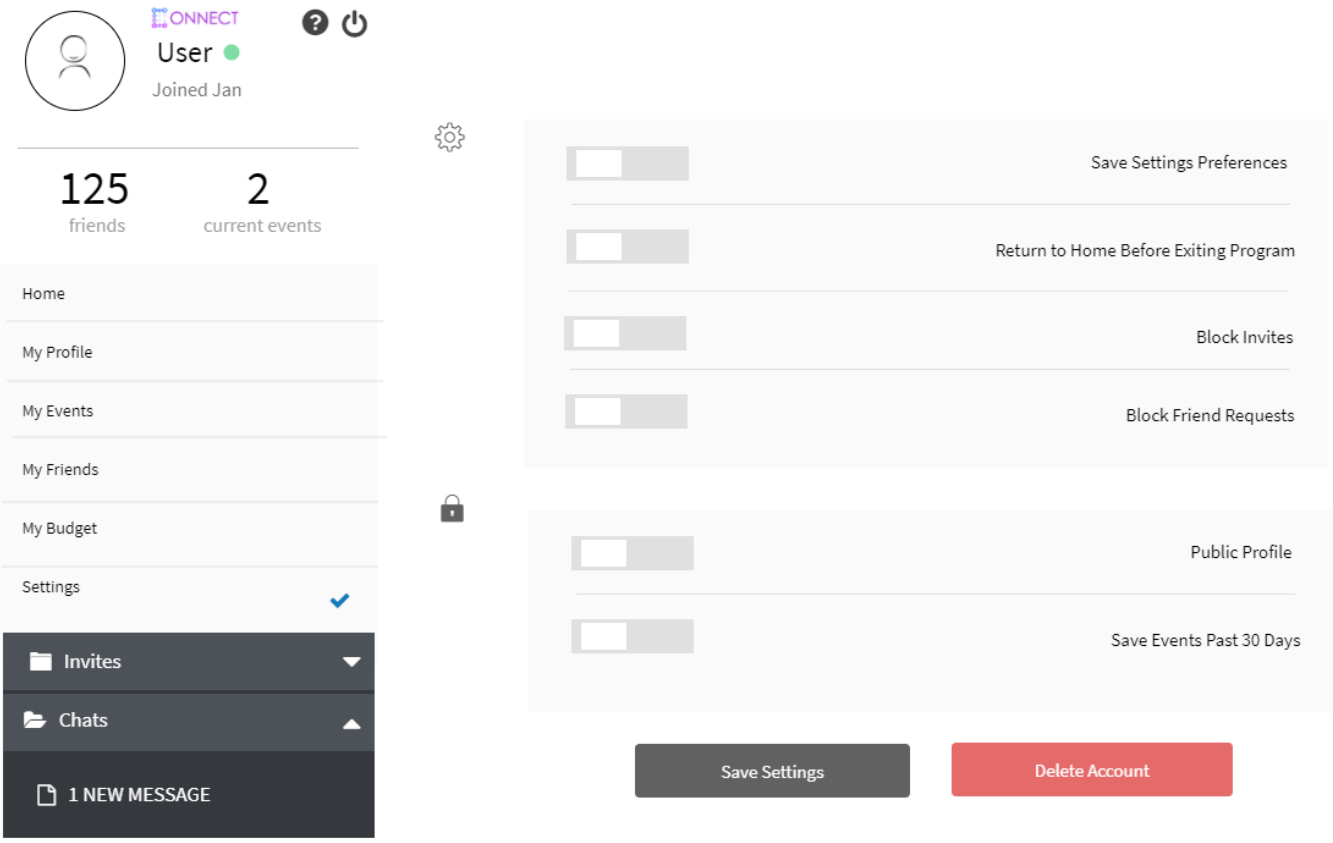
User (you)

send a message...

INTERFACE MOCK-UPS
My Budget



INTERFACE MOCK-UPS
Settings

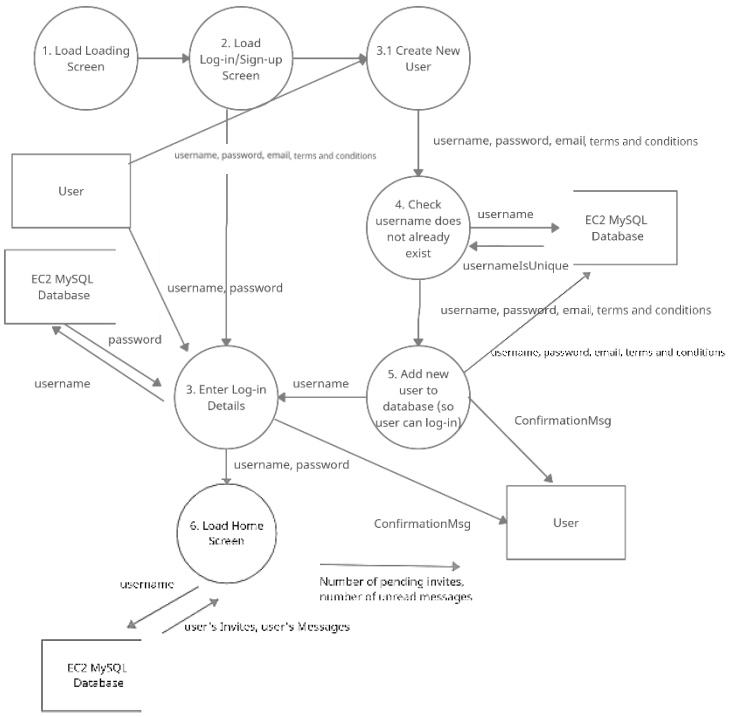


DATAFLOWDIAGRAM 1 | Log-in/Sign up to Home Primary Module

PURPOSE: Allow user to log-in (or optionally sign up then log-in) and then redirecting them to the home screen.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

To begin, the user will be faced with the loading screen. Following this, the log-in or sign-up interface will automatically appear after the timer has completed. Here, the data flow diagram branches into two options: providing the user the ability to either create a new account or log-in. If the user decides to create a new user, then they will have to supply a unique username along with an associated password, email and terms and conditions response (whether they agree or not). This data is sent into the next process where the username is sent to the MySQL database. Here, it is checked if the username is unique (using a MySQL command. Alternatively, this could be achieved by another process where a list of usernames is provided by the MySQL database and the current username is checked against it).



The MySQL database will then output a boolean variable named usernameIsUnique which will then be retrieved by the process. If this is true, then a user does not already exist with this username. Thus, the new user data is added to the external MySQL server and the username. Furthermore, this username flows into the next function so that the username field can be auto-filled (aesthetics). The program will also output a confirmation message to the user.

Whether the user decided to immediately log-in to their account or create a user first, the log-in process will then occur by receiving a username and password from the user. This username will then be sent to the external server to retrieve the password associated with the username. If this password matches, then the user has entered the correct credentials and then the home screen will display.

The last process loads the home screen. This sends the username data to the external database to receives the user's invites and user's messages information used in the interface.

The following dataFlowDiagrams use the username data variable from this diagram.

DATAFLOWDIAGRAM 2 | MyProfile Primary Module

PURPOSE: Provide users with an interface to customise their personal profile that is stored on the MySQL database and for other users to see.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

If the user decides to load the MyProfile screen from the sideNavigation bar, then the loadMyProfile screen process will occur. This loads the appropriate interface and to gather the data necessary for the elements, it will send the username data string to the MySQL database. Here, it will receive the status, aboutMe and

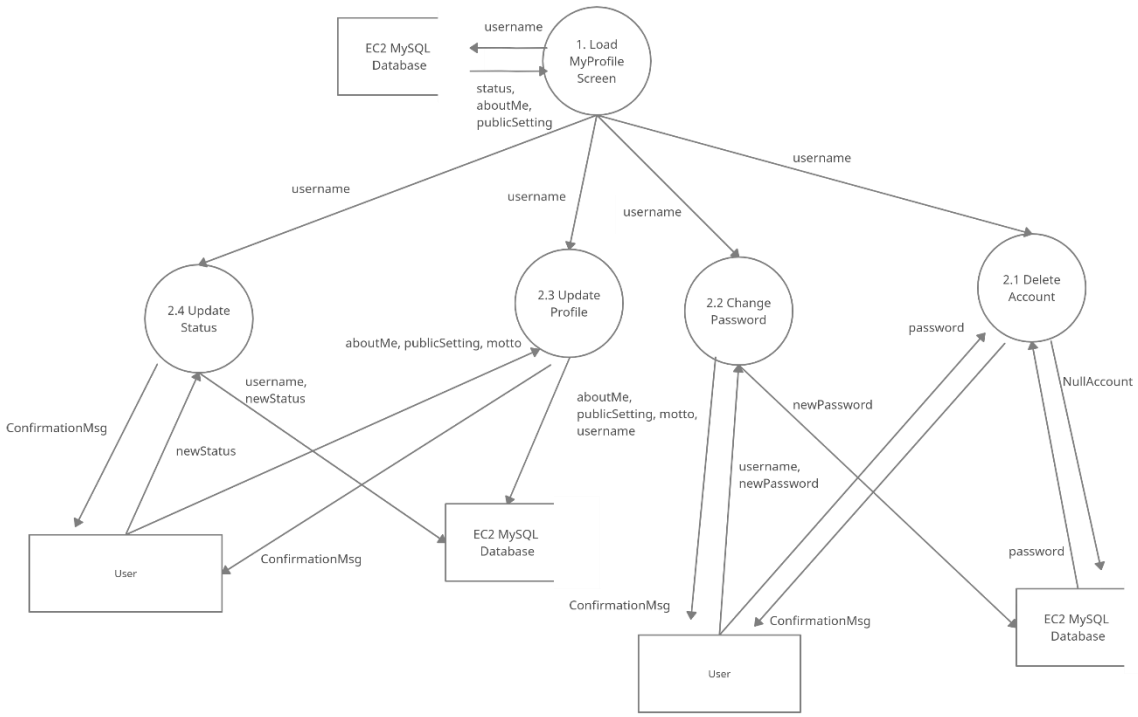
publicSetting variables associated with the user currently logged in. After this (again due to the event-driven nature of winForms) the use has 4 possible processes.

The first is the user's option to update their profile status. This process requires the input of the user's username from the loading interface process which it will send to the MySQL database along with the newStatus which the external entity of the user is required to input into the system. Once this process has been successfully complete a confirmation message will be outputted to the user.

The next process that the user can choose is to update their profile. This process requires the input of the username which will then be sent to the MySQL database along with the user's new aboutMe, publicSetting and motto inputs.

Thirdly, the user can engage in the process to Change Password. Here, the process requires newPassword data input from the user then along with the username, this is sent to update the user's login credentials in the MySQL database. The user receives a confirmation message if this has been successfully reset.

Lastly, the delete account process may occur which requires the user to input their password. This is compared against their old password which is received from the EC2 MySQL database server to reduce the likelihood of accidentally deleting their profile. However, if this is successful and the user's account is deleted then the process will output a confirmation message and nullAccount data (a string replacing the user's name with "-") will be returned to update the mySQL server with the information that the user's profile has been removed. Alternatively, this would involve removing the user's database record.



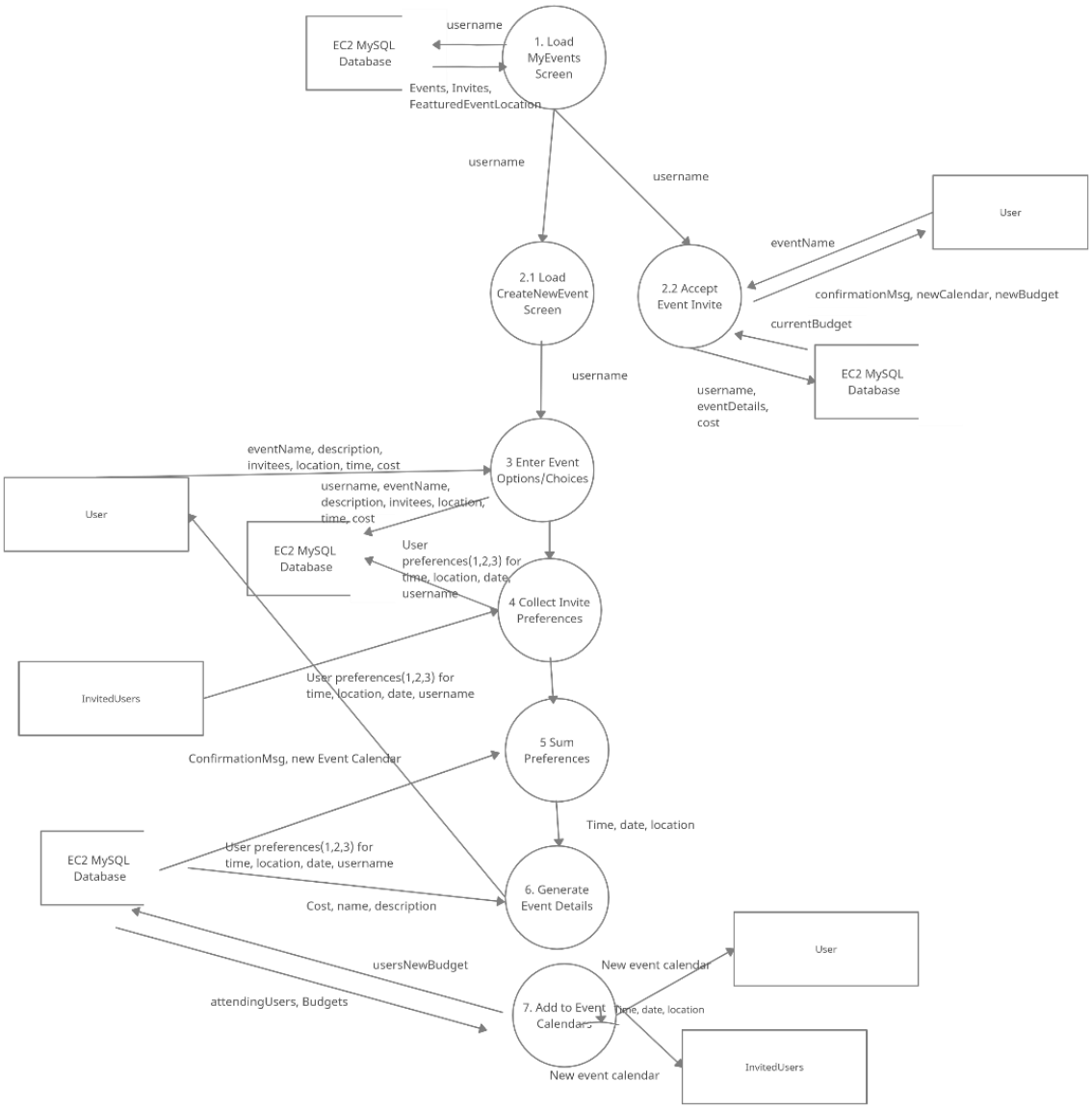
DATAFLOWDIAGRAM 3 | MyEvents Primary Module

PURPOSE: An interface which allows users to centralise, invite other's, accept invitations of and create new events that are stored on the MySQL database.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

This module loads and functions the My events form. Firstly, it collects data using the user's username for the MySQL database to fill the information of the interface - events, invites, and the featured event location.

As the interface is event driven, there are two possible processes branching from here. The user can either accept an event invite which requires the event name from the user, or create a new event which involves loading the createNewEvent screen. If the user chooses to accept an event, the database will receive the username, details of the event (to update the users event list and the cost of the event. This will be used to calculate the user's new budget, which will the. be returned to the function (to showcase if the user can afford the event). Once this process is complete, the user will receive a confirmation message that the event has been accepted, their calendar will be updated with the new event and they will be able to see their new budget. Otherwise, if the user has decided to create a new event they will be required to 8nout the details of the events. These details are stored in the MySQL database. Next, then invited user's external entity receive an invitation retrieved from the MySQL database where they are required to input their preferences. This is summed (where the highest score for each option is selected). Lastly, the event details are generated completing with information from the MySQL server and after updating the user's budgets (and returning this to store in the database, the users event calendars will be updated with the new event.



DATAFLOWDIAGRAM 1 | MyFriends Primary Module

PURPOSE: Create an interface which allows users to view, message and control their friends.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

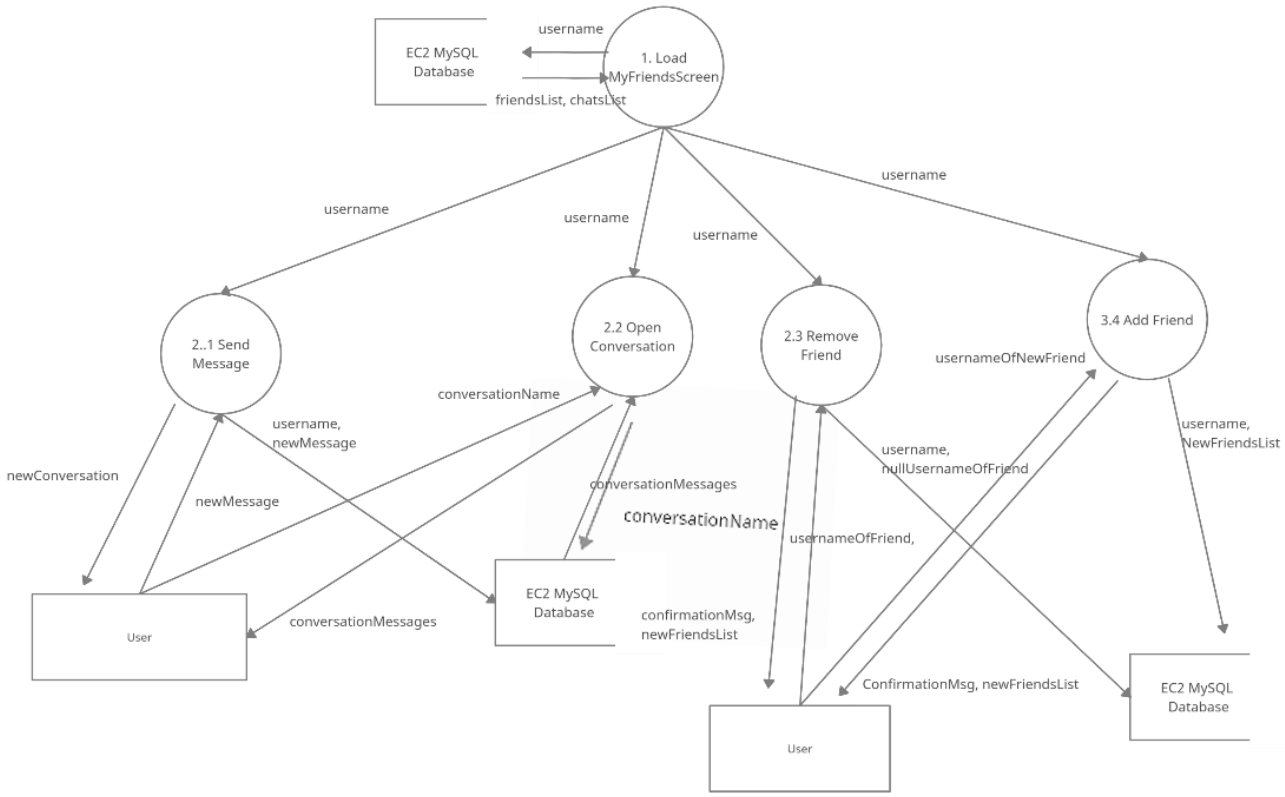
The MyFriends primary module begins with a process to load the MyFriends interface. This screen requires the friends list and chats list data from the EC2 MySQL Database which it retrieves by sending the username of the current user. Once more, because winForms is event-driven, there are 4 different possible branches for the next processes.

The user can send a message by inputting a newMessage and conversationName into the system. Here, the program sends the data of a new message, the username and the conversationName to the MySQL to update the stored conversation with the new message. The external entity user then receives the output of the process as a new conversation.

The next process that the user can use is Open Conversation. This requires the user to input the name of the conversation they wish to open to the process which is sent to the MySQL Database which then outputs the conversationMessages to the process. This then dispays the conversation to the user.

Another process that the module's purpose encompasses is the functionality to remove a friend. Here, the user inputs the username of a friend to the process which is then sent to the MySQL data store and marked to remove (either by actually removing the record or by setting the friend's username to "-". This returns a confirmation message to the user to let them know that their friend has successfully been removed as well as displaying the new friends list.

Finally, this primary module provides the user the ability to add new friends, by using their input of a username to add it to the MySQL data store. Once this is complete, the user receives a confirmation message and the interface is updated with the new friends list.

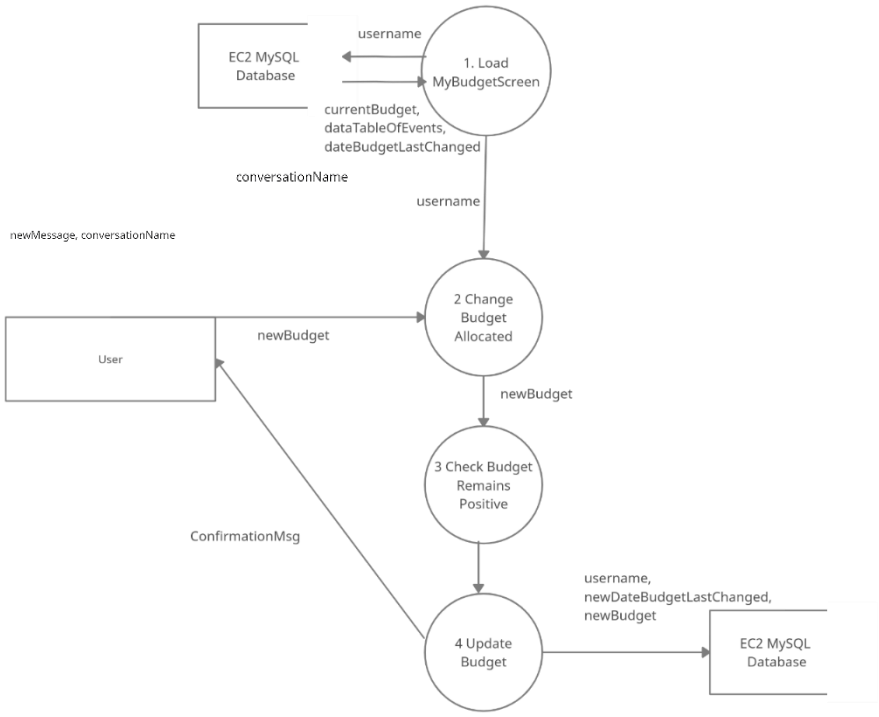


DATAFLOWDIAGRAM 5 | My Budget Primary Module

PURPOSE: Create an interface that allows users to view, manage and reallocate their 'spending' budget.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

The MyBudget module loads the myBudget interface by sending the username to the mySQL database which outputs the according user's data for the interface. This allows the process to receive the users currentBudget. The user can then input the new amount they wish for their budget to be. This is followed by a process that



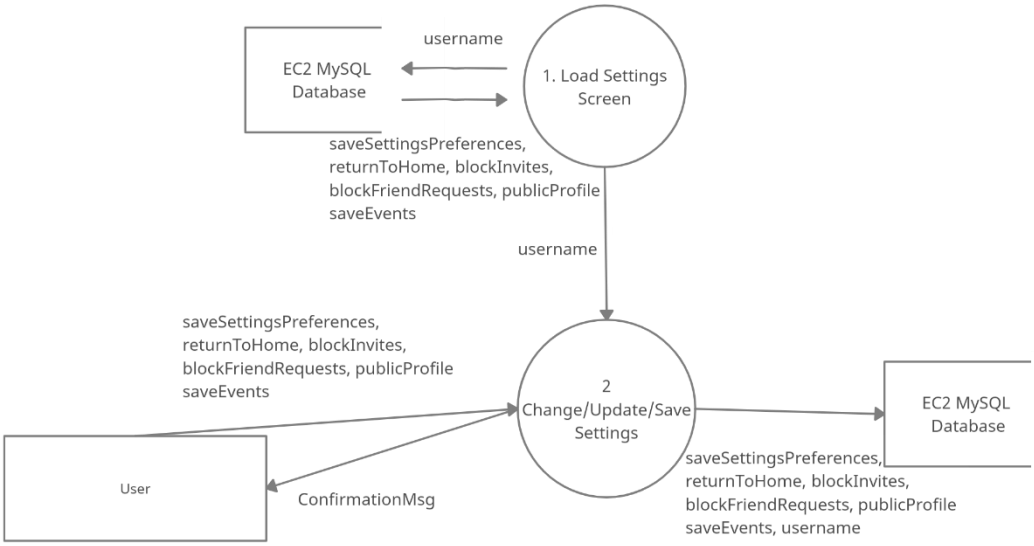
ensure that the budget will remain positive by reading the list of events and costs to ensure that the budget remains positive. If this is successful, then the module outputs and confirmation message which the user receives as a display as well as adjusting the budget to display this new budget.

DATAFLOWDIAGRAM 6 | Settings Primary Module

PURPOSE: Create an interface that allows users manage their account and customise the program.

COMPREHENSIVE DESCRIPTION OF DATA IN/OUT:

The settings module loads the settings screen by retrieving the Boolean data values from the MySQL database that are associated with the current user (by providing it the username data). The only process available for users here is to change their current settings. Once the user has inputted these new settings (the new Boolean data values) are sent to the MySQL database for cloud storage (thus it is customisable on an CONNECT program). The user receives a confirmation once this has been successful.



DATA DICTIONARIES

Log-in Form

FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Password	String	The password the user entered	“Password”
Username	String	The username the user entered	“Username”
TermsAndConditions	Bool	Whether user agrees or disagrees (checkbox)	True
Password2	String	Stored password for the user (password compared against)	"Password”
Email	String	Email for a new user	“Email@email.com”

UsernameIsUnique	Bool	Test if the username is unique for a new user	True
UsersMessages	Int	Counts all of users messages	1
UsersInvites	int	Counts all of users messages	1

DATA DICTIONARIES
MyProfile

FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Password	String	The password the user logged in	"Password"
Username	String	The username of the user logged in	"Username"
AboutMe	String	User's about me section	"Nice to meet you"
publicSetting	Bool	Stores if user has their profile on public or private	True
Motto	String	User's short motto	"Hi there"
ConfirmationMsg	String	Notify user of successful action	"You have successfully updated your status"
NullAccount	String	Remove user from the MySQL field (or set as "-" for null)	-

DATA DICTIONARIES
MyEvents

FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Username	String	The username of the user logged in	"Username"
Events	List (STRUCT)	List of events	
FeaturedEventsLocation	String	Stores the featured event	"Dinner at Bazaar"
EventName	String	Name associated with an event	"Breakfast"
Description	String	Description for an event	"Let's go catch-up"
Invitees	List of Strings (STRUCT)	Contains list of usernames of all users invited	-

Location	List of Strings (STRUCT)	Contains list of all possible locations with their cost	-
Time, Date	Date	Time and date for event	10/10/2021 10:05
User Preferences	Int	Can be either 1,2 or 3 (up to n amount of options) for events. This is added and the option with the highest value is selected.	3
Cost	Float	Cost for the event	123.12
ConfirmationMsg	String	Notify user of successful action	"You have successfully updated your status"

DATA DICTIONARIES
MyFriends

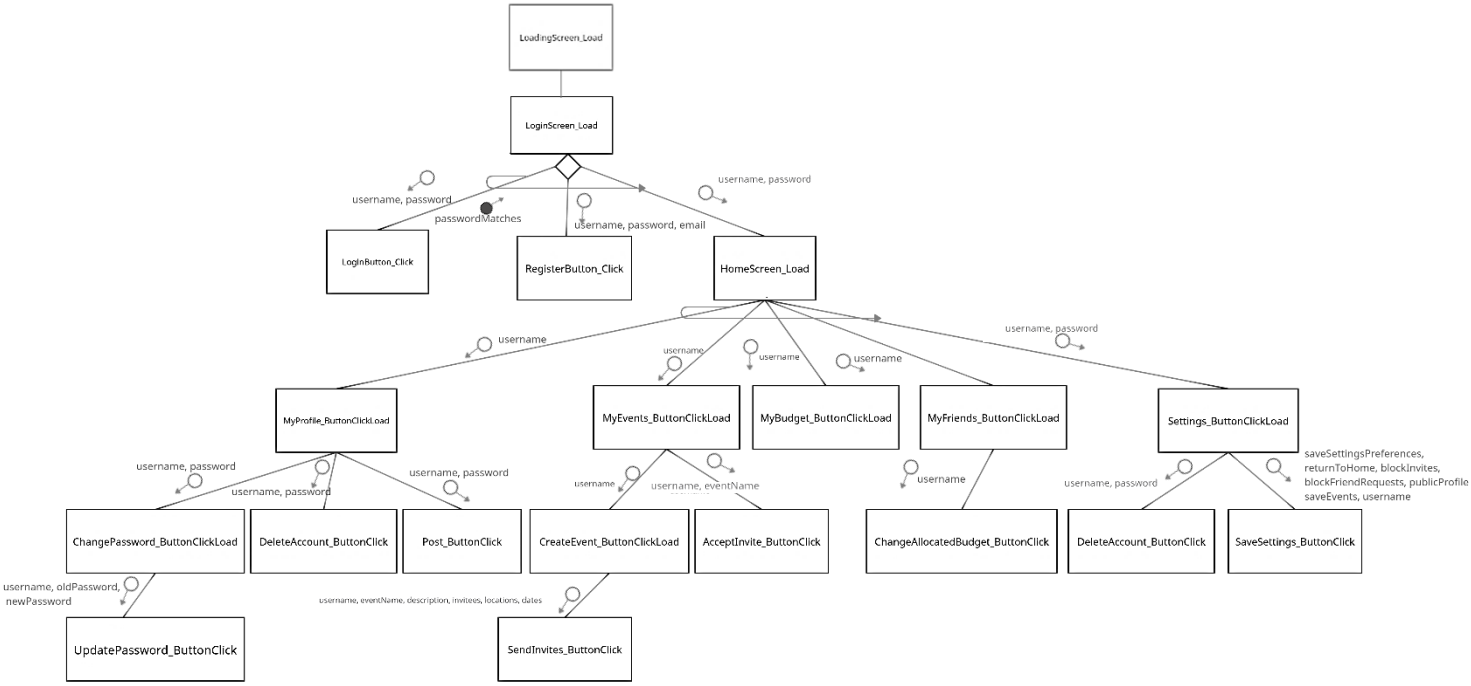
FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Username	String	The username of the user logged in	"Username"
UsernameOfNewFriend / UsernameOfFriend	String	Name of the friend the user wishes to add as a friend or remove; the username of the user which is currently selected for interactions	"User2"
NewFriendsList	String or List of Strings (STRUCT)	Stores the currently logged in user's friend list	"User 2, User 3"
nullUsernameOfFriend	String	Remove the username as a friend from the MySQL field (or set as "-" for null)	"-"
newMessage	String	Message the user is sending to the conversation	"Hey all"
ConfirmationMsg	String	Notify user of successful action	"You have successfully updated your status"

DATA DICTIONARIES
My Budget

FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Username	String	The username of the user logged in	"Username"
NewDateBudgetLastChanged	Date	Last time the user changed their budget	10/21/2021
newBudget	Float	New budget the user wants	123.12
ConfirmationMsg	String	Notify user of successful action	"You have successfully updated your status"

DATA DICTIONARIES
Settings

FIELD NAME	TYPE	DESCRIPTION	EXAMPLE
Username	String	The username of the user logged in	"Username"
saveSettingsPreferences	Bool	If the settings should be saved	False
returnToHome	Bool	If the settings should be saved	False
blockInvites	Bool	If user wants to block invites	False
blockFriendRequests	Bool	If user wants to block friend requests	False
publicProfile	Bool	If user wants to profile to be public (viewable by anyone not a friend)	False
saveEvents	Bool	If the user wants to save events past 30 days	False



To begin, the user experiences a loading screen. This then automatically leads to the LoginScreen_Load module which loads the welcome/login screen for the user. Here, the user must enter a username and password and click the login button (showcasing the pass of username and password to the click function). If the password matches, then a flag **passwordMatches** will be sent back to the LoginScreen and the loop will be left – loading the HomeScreen. However, the user may not have an account yet so instead can also enter a new username, password, email and terms and conditions (to register a new account). These parameters are sent to the RegisterButton_Click module when the user has complete the registration process to create an account. However, this does not return a flag because the user will still need to login with the LoginButon_Click module using these new credentials to get the passwordMatches flag which will redirect them to the homescreen. Thus, these modules are repeated (also allowing users to create many new accounts without having to log in).

Once the home screen is loaded, due to the event-driven nature of winForms, the user can use the side nav buttons (MyProfile_ButtonClickLoad, MyEvents_ButtonCLickLoad, MyBudget_ButtonClickLoad, MyFriends_ButtonClickLoad and Settings_ButtonClickLoad) to open the according form. These modules are all passed the username parameter (**because the password and most other necessary information is accessible from the EC2 MySQL Database elaborated in the DataFlowDiagrams**).

In the MyProfile module which loads the MyProfile screen, the user can accordingly to the modules load a separate form allowing them to change their password where they will then click a button to confirm this change, delete their account (which uses their username parameter and password parameter to confirm their action) and post a new update.

Next, the My Events screen allows users to create a new event by using the create new event button. They can also accept invites by passing the selected parameter when the Accept button is activated.

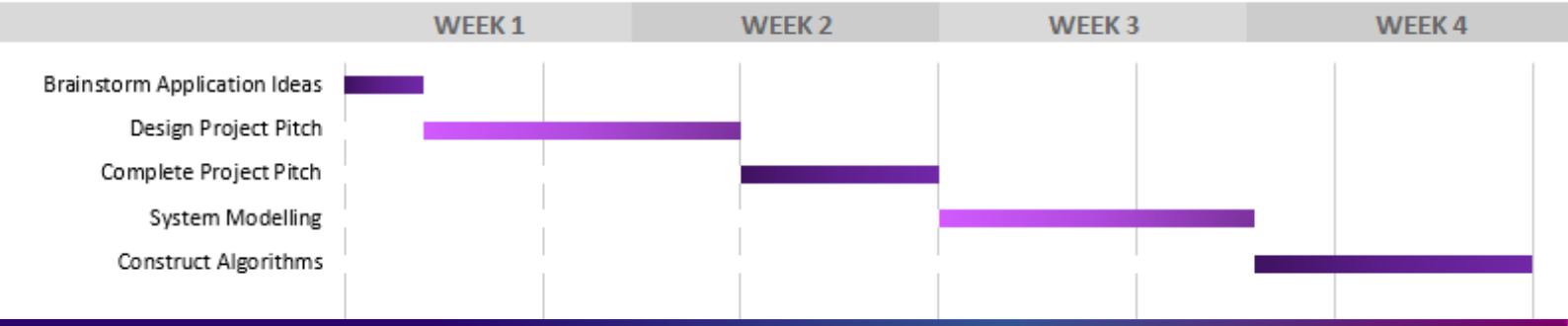
The user can load the MyBudget interface.

Users can also load the MyFriends module and here there is a changeAllocatedBudget subroutine which will allow them to properly change the budget associated with their username (thus passing the username parameter).

Lastly, users can load the Settings form and here they can either delete their account (using the username and password parameters to confirm deletion and prevent accidental removals) or they can use the saveSettings button. This button receives the parameters of the current settings variables.

REQUIREMENTS REPORT AND PROJECT PLAN

RESOURCES ALLOCATION PLAN AND GANTT CHART



RESOURCES ALLOCATION PLAN AND GANTT CHART
Brainstorm Application Ideas

DEPENDENCIES: None

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Paper

TIME RESOURCES: 3 days

DESCRIPTION: Generate ideas for a problem, target audience and interface. Consider scope of assignment and prepare necessary technical resources.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Design / Complete Project Pitch

DEPENDENCIES: Brainstorm Application Ideas

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Powerpoint software

TIME RESOURCES: 11 days (design and complete)

DESCRIPTION: After creating ideas for the program, consolidate information into a brief but concise project proposal to pitch the idea. Involve information such as purpose of software, environment, intended users and concept prototypes.

DEPENDENCIES: Complete Project Pitch

RESOURCES ALLOCATION PLAN AND GANTT CHART
System Modelling

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Modelling software

TIME RESOURCES: 11 days

DESCRIPTION: Once the pitch has been complete and the vision for the program has been realised, begin modelling the system to create and convey the software implementation of the desired structure and functionality. Produce storyboards, interface mock-ups, a dataflow diagram, data dictionaries and structure charts.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Construct Algorithms (DELAYED)

DEPENDENCIES: System Modelling

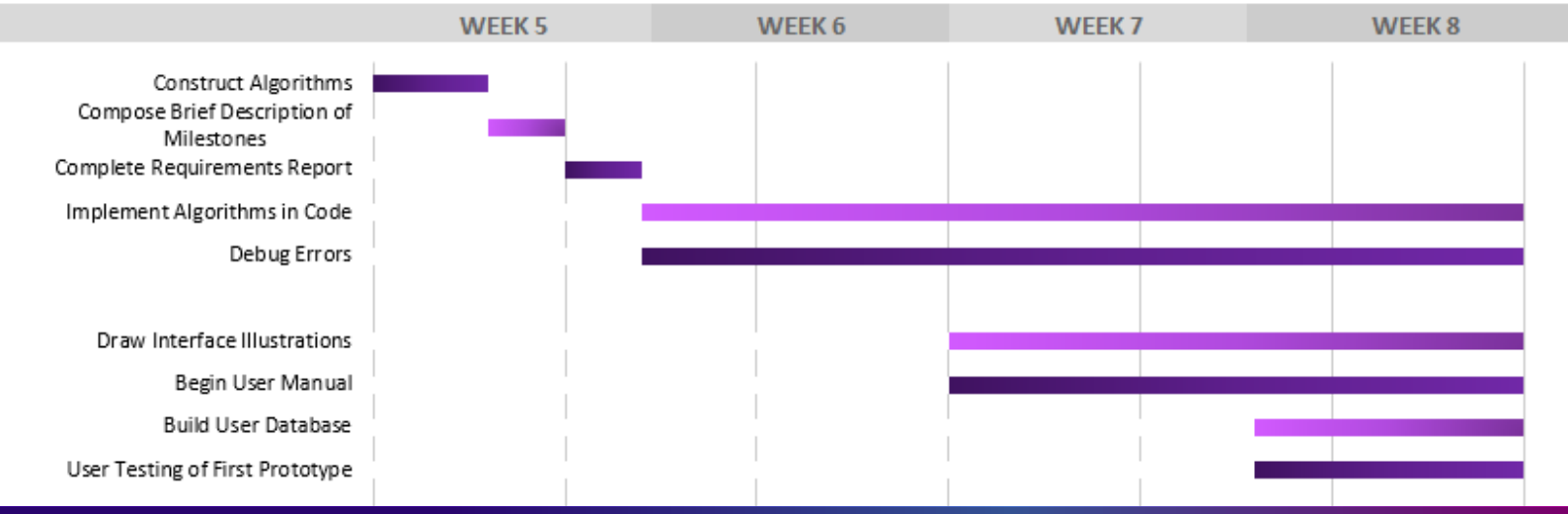
SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Software or paper to compose algorithms

TIME RESOURCES: 14 days

DESCRIPTION: Now the idea has been designed into modules, create algorithms in pseudocode to model and describe each function in preparation for implementation



RESOURCES ALLOCATION PLAN AND GANTT CHART
Construct Algorithms (CONTINUED)

RESOURCES ALLOCATION PLAN AND GANTT CHART
Compose Brief Description of Milestones

SOURCING AND ALLOCATION OF

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Gantt chart software and text processor (for descriptions)

TIME RESOURCES: 3 days

DESCRIPTION: Since the software's modules have been outlined for implementation, resources including Human Resources, TECHNICAL RESOURCES and time can now be planned, allocated, prepared, and sourced in a Gantt chart. Compose a brief description for each milestone while outlining its dependencies.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Complete Requirements Report

DEPENDENCIES: Compose Brief Description of Milestones

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor (for report)

TIME RESOURCES: 3 days

DESCRIPTION: As a developer it is necessary to have a complete understanding of the purpose, environment, and end user of an application. Now that the system's pitch and idea has been realised, system modelling has been complete and a Gantt chart produced, consolidate this information into a report that showcases this comprehension and appreciation. Additionally, compose a series of QA criteria.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Implement Algorithms in Code

DEPENDENCIES: Complete Requirements Report, Construct Algorithms

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: VS Community C# Winforms, Virtual Server, sufficient hardware to program

TIME RESOURCES: 60 days (total)

DESCRIPTION: Once the requirements report has been complete that contains the system modelling tools which allows the creation of algorithms, begin implementing this in code for the software.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Debug Errors

DEPENDENCIES: Implement Algorithms in Code

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: VS Community C# Winforms, Virtual Server, sufficient hardware to program

TIME RESOURCES: 60 days (total)

DESCRIPTION: Whilst implementing the algorithms, ensure simultaneously debugging errors and recording their nature and occurrence for later documentation (Testing and Evaluating report). Additionally, incorporate amendments and improvements on any algorithms.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Draw Interface Illustrations

DEPENDENCIES: Implement Algorithms in Code, Debug Errors

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Adobe Photoshop, Winforms VS Community

TIME RESOURCES: 46 days (total)

DESCRIPTION: Now that implementation in code has begun and the software is being created, design interface graphics accordingly.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Begin/Complete User Manual

DEPENDENCIES: Implement Algorithms in Code, Debug Errors

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor (to create user manual)

TIME RESOURCES: 34 days (total)

DESCRIPTION: While coding and producing interface graphics, the end user should be consistently considered, which will be achieved by simultaneously creating documentation for the user. As a 200 word documentation must be submitted, it is also most time-effective to create this it as the program is being produced.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Build User Database

DEPENDENCIES: Implement Algorithms in Code, Debug Errors

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: EC2 Instance, Winforms VS Community

TIME RESOURCES: 7 days

DESCRIPTION: Configure the EC2 server, create the MySQL database and connect with the program. Populate with test data and users (and setup for later use with the program)

RESOURCES ALLOCATION PLAN AND GANTT CHART
User Testing of First Prototype

DEPENDENCIES: Implement Algorithms in Code, Debug Errors, Complete User Manual

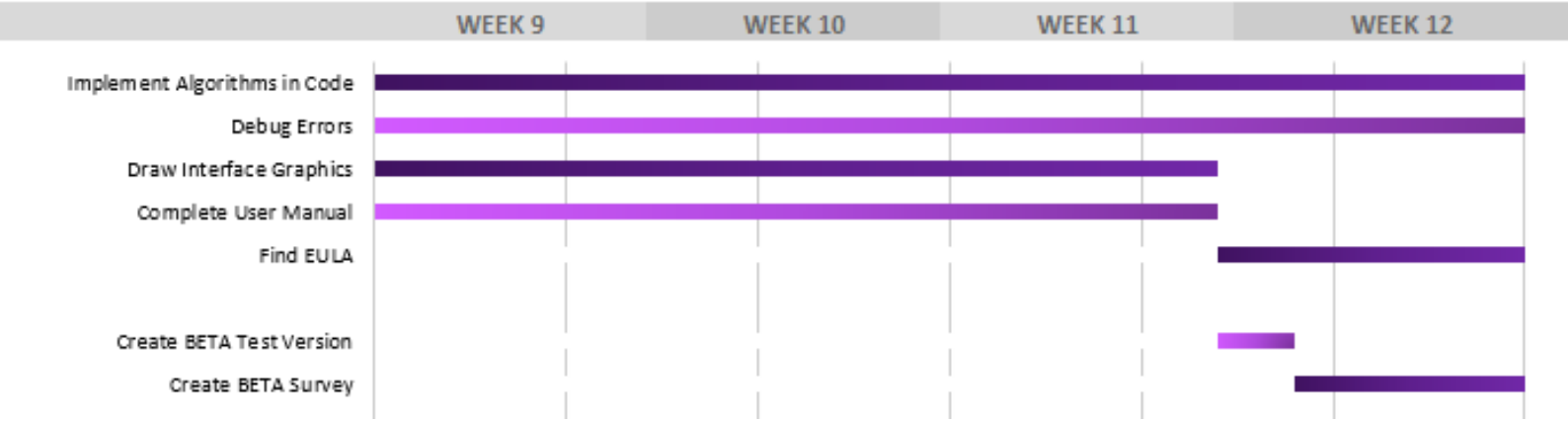
SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: 5+ Human Testers

TECHNICAL RESOURCES: Winforms VS Community, If possible different WindowsOS devices with internet access (preferably that can run simultaneously)

TIME RESOURCES: 7 days

DESCRIPTION: In preparation for the Beta test task and to ensure that the software is user-friendly, reliable, and consistently error-free to avoid any major functionality issues, begin informal user testing of the software. Furthermore, develop ideas for a Beta test survey.



RESOURCES ALLOCATION PLAN AND GANTT CHART
Implement Algorithms in Code (CONTINUED)

RESOURCES ALLOCATION PLAN AND GANTT CHART
RESOURCES ALLOCATION PLAN AND GANTT CHART
Begin/Complete User Manual (CONTINUED)

RESOURCES ALLOCATION PLAN AND GANTT CHART
Find EULA (End User Licence Agreement)

DEPENDENCIES: Implement Algorithms in Code, Debug Errors, Complete User Manual

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Word processor, Internet browser

TIME RESOURCES: 7 days

DESCRIPTION: While arguably no dependencies are required for this milestone, a holistic understanding of the program will inform the best-suited approach. Appropriate this EULA for the EULA task.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Create BETA Test Version

DEPENDENCIES: Implement Algorithms in Code, Debug Errors, Complete User Manual

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: EC2 Instance, Winforms VS Community

TIME RESOURCES: 2 days

DESCRIPTION: Prepare a version of the software to a test-able state for the BETA test task. While development of the program continues, this version will maximise the BETA test.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Create BETA Test Survey

DEPENDENCIES: Create BETA Test Version

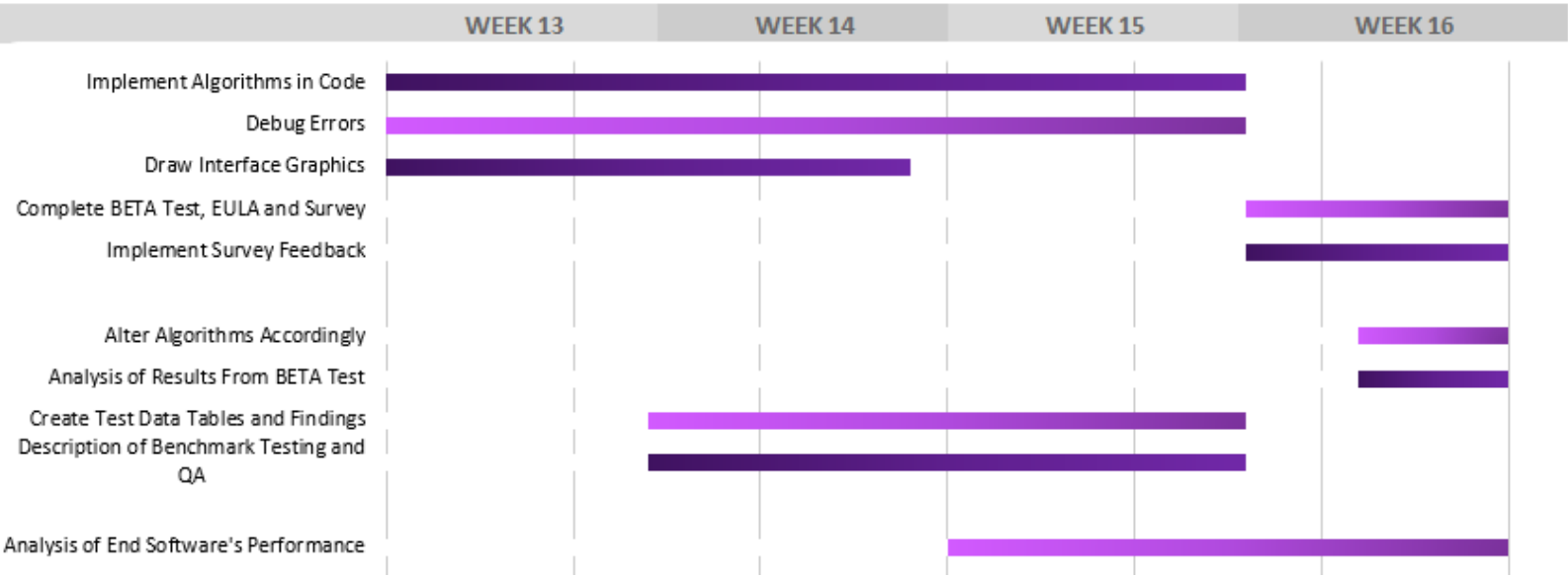
SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Word processor, Survey tool (such as Google Forms)

TIME RESOURCES: 6 days

DESCRIPTION: With the content and scope of the BETA test version of the software in mind, design a survey for testers to complete that will maximise the feedback from the testing. Additionally, organise 10+ 'public' testers (especially with diverse backgrounds due to the broad nature of CONNECT's target audience



RESOURCES ALLOCATION PLAN AND GANTT CHART
Implement Algorithms in Code (CONTINUED)

RESOURCES ALLOCATION PLAN AND GANTT CHART
Debug Errors (CONTINUED)

RESOURCES ALLOCATION PLAN AND GANTT CHART
Draw Interface Graphics (RESUMED/CONTINUED)

Paused for 7 days during BETA testing preparation as BETA versions of the software can use placeholder graphics. Instead, it is more important that the software has majority of its functionality.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Complete BETA Test, EULA and Survey

DEPENDENCIES: Create BETA Test Version, Create BETA Test Survey, Find EULA

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer, 10+ testers

TECHNICAL RESOURCES: Word processor, Survey tool (such as Google Forms), Winforms VS Community, If possible different WindowsOS devices with internet access (preferably that can run simultaneously)

TIME RESOURCES: 7 days

DESCRIPTION: Complete the Beta Test, EULA and Survey task using the material created during week 11 and 12. NOTE: Update or create new versions of the test program, EULA and survey if needed.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Implement Survey Feedback

DEPENDENCIES: Complete BETA Test, EULA and Survey

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Word processor, Survey tool (such as Google Forms)

TIME RESOURCES: 7 days

DESCRIPTION: Begin simultaneously implementing feedback while receiving feedback from the BETA test, EULA and Survey task to reduce workload.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Alter Algorithms Accordingly

DEPENDENCIES: Implement Survey Feedback

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 4 days

DESCRIPTION: Now implementing near-final changes to the code, alter the algorithms that are to be submitted with the Testing and Evaluating report.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Analysis of results from Beta Test

DEPENDENCIES: Complete BETA Test, EULA and Survey

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 4 days

DESCRIPTION: Compose an analysis of the results from the Beta testing for the Testing and Evaluating report while the testing is still recent and testers can still recall majority of the program's functionality in case brief clarification is required.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Create Test Data Tables and Findings

DEPENDENCIES: Implement Algorithms in Code, Debug Errors, Complete User Manual

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 14 days

DESCRIPTION: Prepare for the Testing and Evaluating report by beginning to outline and produce test data tables for the program. Also, begin composing the report on these findings.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Description of Benchmark Testing and Quality Assurance

DEPENDENCIES: Complete Requirements Report

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 14 days

DESCRIPTION: Outline and compose description of benchmark testing and quality assurance for the Testing and Evaluating report.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Analysis of End Software's Performance

DEPENDENCIES: Implement Algorithms in Code, Debug Errors, Complete User Manual

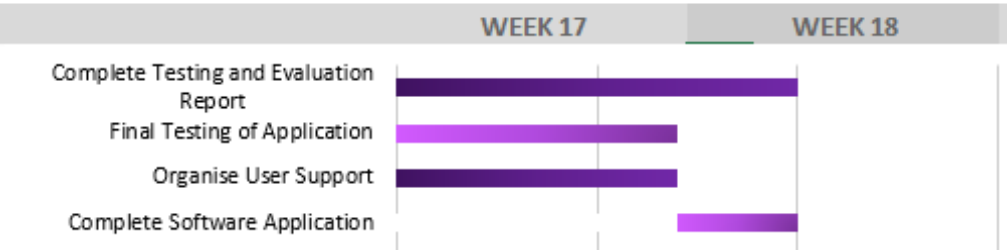
SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 14 days

DESCRIPTION: Compose an analysis of the software's end performance (that can be adjusted accordingly at the completion of the Testing and Evaluating report.



RESOURCES ALLOCATION PLAN AND GANTT CHART
Complete Testing and Evaluation Report

DEPENDENCIES: Alter Algorithms Accordingly, Analysis of results from Beta Test, Create Test Data Tables and Findings, Description of Benchmark Testing and Quality Assurance, Analysis of End Software's Performance

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Text processor

TIME RESOURCES: 10 days

DESCRIPTION: Compile, adjust and conclude the Testing and Evaulating Report. Include formal documentation of how testing and evaluating was executed and include all algorithms for the program.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Final Testing of Application

DEPENDENCIES: Implement Survey Feedback, Debug Errors Complete User Documentation

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer, if possible 2+ 'public' testers

TECHNICAL RESOURCES: EC2 Instance, Winforms VS Community

TIME RESOURCES: 7 days

DESCRIPTION: To ensure that the final state of the program is still error-free and complete, test the application completely and involve two 'public' testers as well.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Organise User Support

DEPENDENCIES: Complete BETA Test, EULA and Survey

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: Internet browser, Text processor

TIME RESOURCES: 7 days

DESCRIPTION: Required as part of the Software Application submission is 'appropriate user support'. Organise and create suitable support whether distributing the documentation or creating a community support form or Youtube demonstration videos.

RESOURCES ALLOCATION PLAN AND GANTT CHART
Complete Software Application

DEPENDENCIES: Implement Survey Feedback

SOURCING AND ALLOCATION OF -

HUMAN RESOURCES: Lead Developer

TECHNICAL RESOURCES: EC2 Instance, Winforms VS Community, Text Processor (Software Application and Testing and Evaluating Report), Display Folder, Printer

TIME RESOURCES: 4 days

DESCRIPTION: Conclude the project and submit the Software Application and PDR google drive folder through email and a PDR printed version in a display folder.

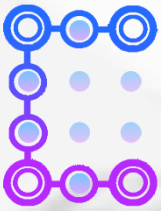
REQUIREMENTS REPORT AND PROJECT PLAN

QUALITY ASSURANCE

STATEMENT	TESTING CRITERIA
Smooth/Quick-Responding Interface	Manually test responds to buttons within 1 second
Communicates smoothly with server	Manually test server-related interactions complete within 1 second.
Organise events in objective manner	Software calculates priority and uses randomNumbers to create events.
User-friendly	Distribute 10+ program prototypes with a survey and receive positive/satisfied feedback. Interface uses consistent buttons and messageboxes.
Robust	Test each input with a variety of illegal data. Use checkboxes/comboboxes where possible Distribute 10+ program prototypes and ensure no errors occur.
Customisable	Distribute 10 program prototypes with a survey and receive positive/satisfied feedback. 4+ different settings.
Functions on Windows OS platforms	Test program works on range of 5+ windowsOS systems (with internet).
Connects users anywhere	Information is stored and accessed in a Cloud-server using the Internet. Test program works on range of 5+ systems (with internet).

REPORT

PROJECT DEVELOPMENT



CONNECT

Jade Harris | **12SDD**

LAST UPDATED: 2021

CONTENTS

ALGORITHMS

PSEUDOCODE

Jade Harris | **12SDD**

CONTENTS

FRM_LOADINGSCREEN

BEGIN timer_Tick

COUNTDOWN a timer THEN

DISPLAY Frm_font

CLOSE this form

END timer_Tick

FRM_FONT

BEGIN btn_closeWelcome_Click

DISPLAY loginForm

CLOSE this form

END btn_closeWelcome_Click

BEGIN btn_send_Click

DISPLAY userManual.pdf

END btn_send_Click

FRM_WELCOMENEWUSER

SET cs TO connection string to cloud database

SET user TO username for frm_nav

BEGIN btn_closeWelcome_Click

DISPLAY hub form

SetUserNotNewAnymore

CLOSE this form

END btn_closeWelcome_Click

BEGIN SetUserNotNewAnymore

OPEN connection to cs

SET new TO false for user

CLOSE connection to cs

END SetUserNotNewAnymore

BEGIN btn_send_Click

DISPLAY userManual.pdf

END btn_send_Click

FRM_LOGIN

SET cs to connection string TO cloud database

BEGIN frm_login

For each textbox, once the user clicks into the textbox SET TO “” and when they leave, if the text is still empty return to the original text

When the user hovers over SHOW button, unconceal password and change the button text to HIDE. When their cursor leaves, hide password and change button text to SHOW.

“ ‘ “ characters or spaces cannot be entered into any textbox

END frm_login

BEGIN cb_EULA_Click

```
IF cb_EULA is checked THEN
    DISPLAY the EULA pdf
    DISPLAY “Are you sure you have completely read and agree to EULA?”
    GET DialogResult
    IF DialogResult == OK THEN
        Leave checkbox checked
    ELSE
        Uncheck checkbox
    ENDIF
ENDIF
```

END cb_EULA_Click

```
BEGIN GeneratePassword

        SET passwordGenerated;

        SET allowedCharacters TO
        "abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ0123456789!@$?_-"

        SET passwordCharacters TO array of 12 characters

        FOR i = 0 TO 11 STEP 1

                SET passwordCharacters[i] TO random character in allowedCharacters

        NEXT i

        FOREACH character IN passwordCharacters

                passwordGenerated += character

        NEXT character

        SET signup_password TO passwordGenerated

END GeneratePassword


START LogIn

        IF any field is empty THEN

                DISPLAY "Please complete all fields"

        ELSE

                LogInProcess

        ENDIF

END Login


BEGIN LogInProcess

        GetLoginDetails

        CheckLoginDetails

        SetLoggedIn

END LogInProcess
```

```
BEGIN GetLoginDetails
        OPEN connection to cs
        POPULATE dgv_user WITH user record
        CLOSE connection to cs
END GetLoginDetails


BEGIN CheckLoginDetails
        IF a record was found for the user THEN
                IF the entered password matches the stored password THEN
                        IF it is stored that the user is not currently logged in  THEN
                                RedirectUser
                        ELSE
                                DISPLAY "Account already logged in on another machine"
                        ENDIF
                ELSE
                        DISPLAY "Incorrect password"
                ENDIF
        ELSE
                DISPLAY "No user exists"
        ENDIF


END CheckLoginDetails


BEGIN RedirectUser
        SetHubFormsUserVariables
        CheckIfNew
END RedirectUser


BEGIN SetHubFormsUserVariables
        SET frm_nav.username TO username entered
```

END SetHubFormsUserVariables

BEGIN CheckIfNew

SET newUser TO the value of the 'new' column

IF newUser == true THEN

SET frm_welcomeNewUser.user TO username entered

DISPLAY frm_welcomeNew

HIDE this form

ELSE

REM "user must not be new"

DISPLAY frm_hub

HIDE this form

ENDIF

END CheckIfNew

BEGIN Register

IF any register field is empty THEN

DISPLAY "Please complete all fields"

ELSE

IF cb_EULA is not checked THEN

DISPLAY "Please agree to EULA to create an account"

ELSE

userExists = CheckIfUserExists

emailExists = CheckIfEmailExists

IF email is < 7 characters OR username is < 4 characters OR password is < 9 characters

DISPLAY "Error, email is < 7 characters OR username is < 4 characters OR password is < 9 characters"

ELSE

CreateNewUser

```
                    ENDIF

            ENDIF
        ENDIF
END Register

BEGIN CheckIfEmailExists
    OPEN connection to cs
    POPULATE table with records where email matches entered email
    CLOSE connection to cs
    IF a record exists THEN
        RETURN true
    ELSE
        RETURN false
    ENDIF
END CheckIfEmailExists

BEGIN CheckIfUserlExists
    OPEN connection to cs
    POPULATE table with records where username matches entered username
    CLOSE connection to cs
    IF a record exists THEN
        RETURN true
    ELSE
        RETURN false
    ENDIF
END CheckIfUserlExists
```

```
BEGIN CreateNewUser
        OPEN connection to cs
        INSERT entered username, password and email as a new record
        CLOSE connection to cs
        ResetFieldsNowNewUser
END CreateNewUser


BEGIN ResetFieldsNowNewUser
        DISPLAY "User successfully created"
        Reset username, password and email to default
END ResetFieldsNowNewUser
```

FRM_EVENTS

```
SET cs to connection string TO cloud database
SET username to user (public variable set from frm_nav)


BEGIN frm_events
        SET row height of datagridview to 40px
        Refresh
        FOR each row in dgv_events STEP 1
                IF the status column == "Pending" for that row THEN
                        SET second button TO "Wait until hosted"
                ELSE
                        SET second button TO "View Details"
                ENDIF
        NEXT row
END frm_events
```

BEGIN Refresh

PopulateUser

PopulateWithMyEvents

END Refresh

BEGIN PopulateUser

OPEN connection to cs

POPULATE dgv_currentUserInfo with user information

CLOSE connection to cs

END PopulateUser

BEGIN PopulateWithMyEvents

REM "to track last time updated"

UPDATE lbl_lastUpdate TO current time

GET all of user's events

OPEN connection to cs

POPULATE dgv_events with the information for these events from the Event table

CLOSE connection to cs

END PopulateUser

BEGIN PopulateWithMyEvents

REM "to track last time updated"

UPDATE lbl_lastUpdate TO current time

GET all of user's events

OPEN connection to cs

POPULATE dgv_events with the information for these events from the Event table

CLOSE connection to cs

END PopulateUser

```
BEGIN btn_newEvent_click
        DISPLAY frm_myEvents
END btn_newEvent_click


SET selectedRowIndex TO 0


BEGIN dgv_events_Click
        IF user clicks on an event row THEN
                SET selectedRowIndex TO index of event
                SET selectedEvent TO name of the selected event
                SET mostRecent = CheckLastUpdated_Event(selectedEvent)
                IF mostRecent = false THEN
                        DISPLAY "Event has been recently updated"
                        PopulateWithMyEvents
                ENDIF
        ENDIF
END dgv_events_Click


BEGIN CheckLastUpdated_Event(eventName)
        OPEN connection to cs
        POPULATE dgv_events with the information for this event
        CLOSE connection to cs
        SET lastUpdated TO lastUpdated column
        SET clientsLastUpdated TO lbl_lastUpdate
        IF lastUpdated is later than the clientsLastUpdated THEN
                RETURN false
        ELSE
                RETURN true
        ENDIF
```

END CheckLastUpdated_Event

BEGIN ReUpdate_Event(eventName)

OPEN connection to cs

UPDATE lastUpdated for eventName with the current time'

CLOSE connection to cs

END ReUpdate_Event

REM "CellContentClick is triggered when a button is called associated for the event"

BEGIN dgv_events_CellContentClick

IF user is using the most recent version of the table THEN

IF user clicks LEAVE button THEN

DISPLAY "Are you sure you want to leave?"

GET DialogResult

IF DialogResult == OK THEN

LeaveEvent

ReupdateEvent(selectedEvent)

Refresh

ENDIF

ELSE IF user clicks VIEW button THEN

IF the event has been hosted THEN

SET frm_viewingInformation.eventName = selectedEvent

DISPLAY frm_viewingInformation

ENDIF

ELSE

REM "User must be selecting the VIEW button for an event that hasn't been hosted yet"

DISPLAY "Please wait until the event is hosted"

```
            ENDIF
        ENDIF
END dgv_events_CellContentClick


BEGIN LeaveEvent
        PopulateSelectedEventDetails
        REM "If event has not yet been hosted then remove the user's votes when they leave (from attending)"
        IF event status is pending THEN
            RemoveUsersVotesRecord
        ENDIF
        RemoveUserFromEvent
        RemoveEventFromUser
END LeaveEvent


BEGIN RemoveUsersVotesRecord
        SET Votes as Array of dgv_currentEventSelection.recordVotes split at the '|' character
        SET currentUsersRecord TO ""
        SET currentUser TO ""
        SET idForUsersVote TO 0
        FOR i = 0 TO elements in Votes STEP 1
            IF Votes[i] != "" THEN
                SET currentUsersRecord TO the username part of Votes[i]
                IF currentUsersRecord == username THEN
                    idForUsersVote = i
                ENDIF
            ENDIF
        NEXT i
```

SET currentUser TO Votes[idForUsersVote]

GET the current value of the options priority stored in the [ ] for each location and time/date option

SET the new value for each option to the current priority – the priority the user voted

SET updatedVoteRecord TO name of option + [new priority]

OPEN connection to cs

UPDATE with updatedVoteRecord, location_newPriorities, dayTime_newPriorities

CLOSE connection to cs

END RemoveUsersVotesRecord

BEGIN PopulateSelectedEventDetails

OPEN connection to cs

POPULATE dgv_currentEventSelection with data for selectedEvent

CLOSE connection to cs

END PopulateSelectedEventDetails

BEGIN RemoveUserFromEvent_Hosted

SET attendees to attendees column from dgv_currentEventSelection

REMOVE username from attendees

OPEN connection to cs

UPDATE attendees

CLOSE connection to cs

END RemoveUserFromEvent_Hosted

BEGIN RemoveEventFromUser_Hosted

SET events to events column from dgv_currentEventSelection

REMOVE event from events

OPEN connection to cs

UPDATE events

CLOSE connection to cs

END RemoveEventFromUser_Hosted

FRM_MYEVENTS

SET cs to connection string TO cloud database

SET username to user (public variable set from frm_nav)

BEGIN frm_myEvents

    SetupForm

    FOREACH row in dgv_events

        IF the status column of row == “Pending” THEN

            SET that row button TO “Host”

        ELSE

            SET row button TO “Finish”

        ENDIF

END frm_myEvents

BEGIN SetupForm

    SET dgv_events row height to 40pc

    Refresh

END SetupForm

BEGIN Refresh

    PopulateUser

    PopuldateWithMyEvents

END Refresh

BEGIN PopulateUser

    OPEN connection to cs

    POPULATE dgv_userData with information for the user

```
        CLOSE connection to cs
END PopulateUser

BEGIN PopulateWithMyEvents
        OPEN connection to cs
        POPULATE dgv_events where owner=username
        CLOSE connection to cs
END PopulateWithMyEvents

BEGIN btn_newEvent_Click
        Open frm_newEvent
END btn_newEvent_Click

INITIALISE selectedEvent
BEGIN dgv_events_CellContentClick
        IF mostRecent == true THEN
                IF cancel button is clicked THEN
                        DISPLAY "Are you sure you want to cancel" + selectedEvent
                        GET DialogResult
                        IF DialogResult == OK THEN
                                CancelEvent
                                Refresh
                                DISPLAY selectedEvent + "successfully cancelled"
                        ENDIF
                ELSE IF host/finish button is clicked THEN
                        IF status column of the event == "Pending" THEN
                                DISPLAY "Are you sure you want to host" + selectedEvent
                                GET DialogResult
                                IF DialogResult == OK THEN
```

```
                    PopulateSelectedEvent
                    HostEvent
                    Refresh
                    ReUpdate_Event(selectedEvent)
                    DISPLAY "Successfully hosted."
            ENDIF
        ELSE
            REM "Otherwise column must already be hosted in which case user
            would be wanting to finish the event"

            DISPLAY "Are you sure event is complete?"
            GET DialogResult
            IF DialogResult == OK THEN
                    CancelEvent
                    Refresh
                    ReUpdate_Event(selectedEvent)
                    DISPLAY "Successfully complete."
            ENDIF
        ENDIF
    ELSE IF manage button is clicked THEN
        Open the managing form
        SET eventname in frm_managing to selectedEvent
        SET username in frm_managing to selectedEvent
END dgv_events_CellContentClick


BEGIN PopulateSelectedEvent
    OPEN connection to cs
    POPULATE dgv_currentEventSelection where eventName = selectedEvent
    CLOSE connection to cs
END PopulateSelectedEvent
```

```
BEGIN HostEvent
        SetAccordingToVotes
        SetHostStatusAndClearInvitees
        AddToHostsEvents
END HostEvent
BEGIN AddToHostsEvents
        SET eventIDs to eventIDs column from dgv_currentUserInfo
        ADD selectedEvent to eventIDs
        OPEN connection to cs
        UPDATE eventIDs
        CLOSE connection to cs
END AddToHostsEvents


BEGIN SetAccordingToVotes
        SET equalVotes TO false
        SET dayTimeCurrentVotes TO array of
        SET lowestIndexes_dayTime TO list of 3 integers
        SET lowestValue_dayTime TO dayTimeCurrentVotes[0]
        ADD 0 TO lowestIndexes_dayTime
        FOR i = 1 TO dayTimeCurrentVotes
                SET vote TO dayTimeCurrentVotes[i]
                IF vote <> “”
                        SET value TO the value contained between the [ ] of vote
                        IF lowestValue_dayTime > value THEN
                                lowestValue_dayTime = value
                                CLEAR all elements from lowestIndexes_dayTime
                                ADD i TO lowestIndexes_dayTime
                        ENDIF
```

```
                IF lowestValue_dayTime == value THEN
                        lowestValue_dayTime = value
                        ADD i TO lowestIndexes_dayTime
                ENDIF
        ENDIF
NEXT i
REPEAT for lowestIndexes_location
```

REM "Now just get the name of the highest votes options"

```
SET dayTimeNames TO each option in the eventTime column of dgv_currentEventSelection split at ','
FOR i = 1 TO dayTimeNames.Length – 1 STEP 1
        dayTimeNames[i] = dayTimeNames[i].Substring(0, index of the first [)
NEXT i
SET dayTime_newPriorities TO ""
FOREACH index in lowestIndexes_dayTime
        dayTime_highestVoted += dayTimeNames[index] + ","
NEXT index

REPEAT for lowestIndexes_location to get location_ highestVoted

IF lowestIndexes_dayTime.Count > 1 THEN
        SET equalVotes TO true
ENDIF
IF lowestIndexes_location.Count > 1 THEN
        SET equalVotes TO true
ENDIF

OPEN connection to cs
UPDATE eventTime=eventTime_highestVoted eventLocation = location_highestVoted for
selectedEvent
```

```
        CLOSE connection to cs


        REM "User must choose between the options that got even votes and the event location and time is set to this"

        IF equalVotes == true THEN

                Open frm_equalVotes

        ENDIF

END SetAccordingToVotes


INITIALISE invitedUser

BEGIN SetHostStatusAndClearInvitees

        OPEN connection to cs

        UPDATE status="Hosted" where eventName = selectedEvent

        TRY

                SET invited TO invitees column of dgv_currentEventSelecton split at ,

                FOREACH user IN invited STEP 1

                        SET invitedUser TO user

                        UpdateInvitees

                NEXT user

        END TRY

        CATCH

        ENDCATCH

        UPDATE invitees=' ' where eventName = selectedEvent

        CLOSE connection to cs

END SetHostStatusAndClearInvitees


INITIALISE attendingUser

BEGIN CancelEvent

        TRY
```

```
            SET invited TO invitees column of dgv_currentEventSelecton split at ','
            FOREACH user IN invited STEP 1
                SET invitedUser TO user
                UpdateInvitees
            NEXT user
        END TRY
        CATCH
        ENDCATCH
        TRY
            SET attendees TO attendees column of dgv_currentEventSelecton split at ','
            FOREACH user IN attendees STEP 1
                SET attendingUser TO user
                UpdateAttendees
            NEXT user
        END TRY
        CATCH
        ENDCATCH
        RemoveEventFromHost
        OPEN connection to cs
        DELETE RECORD where eventName = selectedEvent
        CLOSE connection to cs
END CancelEvent


BEGIN UpdateInvitees
        PopulateFriend(invitedUser)
        SET invitationIDs to invitationIDs column from dgv_friend
        REMOVE event from invitationIDs
        OPEN connection to cs
        UPDATE invitationIDs for invitedUser
```

```
        CLOSE connection to cs
END UpdateInvitees

BEGIN UpdateAttendees
        PopulateFriend(invitedUser)
        SET eventIDs to eventIDs column from dgv_friend
        REMOVE event from eventIDs
        OPEN connection to cs
        UPDATE eventIDs for attendingUser
        CLOSE connection to cs
END UpdateAttendees

BEGIN PopulateFriend(friend)
        OPEN connection to cs
        POPULATE dgv_friend where username=friend
        CLOSE connection to cs
END PopulateFriend

BEGIN RemoveEventFromHost
        SET eventIDs to eventIDs column from dgv_currentUserInfo
        REMOVE event from eventIDs
        OPEN connection to cs
        UPDATE eventIDs
        CLOSE connection to cs
END RemoveEventFromHost

BEGIN dgv_events_Click
        mostRecent = checkLastUpdated_Event(username)
        IF mostRecent == false THEN
```

```
            DISPLAY "Your events have been recently changed. Updating"
            Refresh
        ENDIF
END dgv_events_Click


BEGIN CheckLastUpdated_Event(eventName)
        OPEN connection to cs
        POPULATE table2 with the information where eventName=eventName
        CLOSE connection to cs
        SET lastUpdated TO lastUpdated column of table2
        SET clientsLastUpdated TO lbl_lastUpdate
        IF lastUpdated is later than the clientsLastUpdated THEN
            RETURN false
        ELSE
            RETURN true
        ENDIF
END CheckLastUpdated_Event


BEGIN ReUpdate_Event(eventName)
        OPEN connection to cs
        UPDATE lastUpdated for the event where eventName = eventName
        CLOSE connection to cs
END ReUpdate_User
```

FRM_INVITATIONVOTE

```
SET cs to connection string TO cloud database
SET username to user (public variable set from frm_nav)


BEGIN frm_invitationVote
```

SetupForm

Set up buttons so that

END frm_invitationVote

BEGIN SetupForm

PopulateEvent

PopulateUser

SetEndTimes

DISPLAY the duration

DISPLAY each location and time/date options by creating a substring from 0 to the [character. Place in a textbox beside a button which which the user will toggle to set the priority for the option.

IF the option does not exist THEN

Textbox = “NO OPTION SET”

ENDIF

END SetupForm

BEGIN PopulateEvent

OPEN connection to cs

POPULATE dgv_eventData with information for the event

CLOSE connection to cs

END PopulateEvent

BEGIN PopulateUser

OPEN connection to cs

POPULATE dgv_userData with information for the user

CLOSE connection to cs

END PopulateUser

```
BEGIN SetEndTimes
        Add duration to the options for each time/date
END SetEndTimes


BEGIN btn_createEvent_Click
        IF the user has not set any options for location or time/date at the same priority THEN
                ReUpdate
                NewDateTimeVotes
                NewLocationVotes
                InsertVotes
                AddUserToAttendees
                MoveEventToAttending
                RecordUsersVote
                CALL ReturnToEvents FROM frm_invitation
                ReUpdate_Event(eventName)
                Close this form
        ELSE
                DISPLAY "Cannot have two or more buttons at the same priority"
        ENDIF
END btn_createEvent_Click


BEGIN ReUpdate_Event(eventName)
        OPEN connection to cs
        SET lastUpdated TO the time now
        CLOSE connection to cs
END ReUpdate_Event


BEGIN ReUpdate
        PopulateEvent
```

```
        PopulateUser
END ReUpdate_Event


INITIALISE dayTime_newPriorities;
BEGIN NewDayTimeVotes
        SET dayTime TO list of dayTimes split at ','
        SET dayTimeNames TO list of dayTimes split at ','
        FOR i = 0 TO dayTimeNames.Length STEP 1
                DayTimeName[i] = dayTimeName[i].Substring(0, '[')
        NEXT i
        SET dayTime_idOfOption1 to the index of lbl_dayTime_option1.Text
        SET dayTime_currentPriority_option1 TO dayTime[dayTime_idOfOption1].Substring(indexes between
        the [ and ])
        SET dayTime_newPriority_option1 TO dayTime_currentPriority_option1 + number on the button
        beside the option
        SET dayTime_redone_option1 TO lbl_dayTime_option1.Text + dayTime_newPriority_option1


        TRY
                SET dayTime_redone_option2 TO process above for lbl_dayTime_option2
        ENDTRY
        CATCH
        ENDCATCH
        TRY
                SET dayTime_redone_option3 TO process above for lbl_dayTime_option3
        ENDTRY
        CATCH
        ENDCATCH
        SET dayTime_newPriorities = dayTime_redone_option1 + dayTime_redone_option2 +
        dayTime_redone_option3
```

END NewDayTimeVotes

INITIALISE location_newPriorities;

BEGIN NewLocationVotes

SET location TO list of locations split at ‘,’

SET locationNames TO list of locations split at ‘,’

FOR i = 0 TO locationNames.Length STEP 1

LocationName[i] = locationName[i].Substring(0, ‘[‘)

NEXT i

SET location_idOfOption1 to the index of lbl_location_option1.Text

SET location_currentPriority_option1 TO location[location_idOfOption1].Substring(indexes between the [and])

SET location_newPriority_option1 TO location_currentPriority_option1 + number on the button beside the option

SET location_redone_option1 TO lbl_location_option1.Text + location_newPriority_option1

TRY

SET location_redone_option2 TO process above for lbl_location_option2

ENDTRY

CATCH

ENDCATCH

TRY

SET location_redone_option3 TO process above for lbl_location_option3

ENDTRY

CATCH

ENDCATCH

SET location_newPriorities = location_redone_option1 + location_redone_option2 + location_redone_option3

END NewLocationVotes

BEGIN AddUserToAttendees

SET invitees to invitees column from dgv_userData

REMOVE user from invitees

SET attendees to attendees column from dgv_userData

ADD user to attendees

OPEN connection to cs

UPDATE invitees

UPDATE attendees

CLOSE connection to cs

END RemoveUserFromEvent_Hosted

BEGIN AddUserToAttendees

SET invitees to invitees column from dgv_userData

REMOVE user from invitees

SET attendees to attendees column from dgv_userData

ADD user to attendees

OPEN connection to cs

UPDATE invitees

UPDATE attendees

CLOSE connection to cs

END RemoveUserFromEvent_Hosted

BEGIN MoveEventToAttending

SET invitationIDs to invitationIDs column from dgv_eventData

REMOVE event from invitationIDs

SET eventIDs to eventIDs column from dgv_eventData

ADD event to eventIDs

OPEN connection to cs

UPDATE invitationIDs

UPDATE eventIDs

CLOSE connection to cs

END MoveEventToAttending

BEGIN RecordUsersVotes

SET votes to recordVotes column from dgv_eventData

ADD username + "-[" + record the user's vote for each time/date option separated by , + "]-[" + record the user's vote for each location option separated by , + "]|"

OPEN connection to cs

UPDATE invitationIDs

UPDATE recordVotes

CLOSE connection to cs

END MoveEventToAttending

BEGIN InsertVotes

OPEN connection to cs

UPDATE with location_newPriorities and eventTime_newPriorities

CLOSE connection to cs

END InsertVotes

FRM_FRIENDS

SET cs to connection string TO cloud database

SET username to user (public variable set from frm_nav)

BEGIN frm_friends

ReUpdate

When user clicks the friend name textbox, set the text to empty and if they leave the textbox and the text is still empty return to default text

Friend name textbox cannot have ‘ or spaces entered

END frm_friends

BEGIN ReUpdate

PopulateFriends

PopulateRequests

PopulateSentRequests

END ReUpdate

BEGIN PopulateFriends

PopulateUser

SET lbl_lastUpdate TO the time now

FOREACH user IN the friends column of dgv_user STEP 1

ADD user to dgv_friends

NEXT user

END PopulateFriends

BEGIN PopulateSentRequests

PopulateUser

SET lbl_lastUpdate TO the time now

FOREACH user IN the sentRequests column of dgv_user STEP 1

ADD user to dgv_sentRequests

NEXT user

END PopulateSentRequests

BEGIN PopulateRequests

PopulateUser

```
        SET lbl_lastUpdate TO the time now
        FOREACH user IN the friendInvitations column of dgv_user STEP 1
                ADD user to dgv_friendRequests
        NEXT user
END PopulateSentRequests


BEGIN PopulateUser
        OPEN connection to cs
        POPULATE dgv_user with user's data
        CLOSE connection to cs
END PopulateUser
```

INITIALISE selectedFriendName

```
BEGIN dgv_friends_CellContentClick
        SET selectedFriendName to the name column of dgv_friends
        IF the remove button is clicked for user in any row THEN
                DISPLAY "Are you sure you want to remove" + selectedFriendName
                GET DialogResult
                IF DialogResult == OK THEN
                        RemoveFriend
                        ReUpdate_User(selectedFriendName)
                        ReUpdate_User(username)
                        PopulateFriends
                ENDIF
        ENDIF
END dgv_friends_CellContentClick


BEGIN RemoveFriend
        PopulateWithSelectedFriend
```

```
        RemoveFromUser
        RemoveFromFriend
        DISPLAY selectedFriendName + "successfully removed."
END RemoveFriend

BEGIN PopulateWithSelectedFriend
        OPEN connection to cs
        POPULATE dgv_friend with user data where username=selectedFriendName
        CLOSE connection to cs
END PopulateWithSelectedFriend

BEGIN RemoveFriendFormUser
        SET friends TO friends column from dgv_userData
        REMOVE selectedFriendName from friends
        OPEN connection to cs
        UPDATE friends for user
        CLOSE connection to cs
END RemoveFormUser

BEGIN RemoveUserFormFriend
        SET friends TO friends column from dgv_friend
        REMOVE user from friends
        OPEN connection to cs
        UPDATE friends for selectedFriendName
        CLOSE connection to cs
END RemoveFormUser

BEGIN btn_requestFriend_Click
        Exists = checkUserExists
```

```
        IF txt_friendName.Text == user logged in THEN
                DISPLAY "You cannot add yourself as a friend"
        ELSE
                IF exists == true THEN
                        Message = CheckIfFriendExists
                        IF message == "" THEN
                                PopulateWithFriendRequest
                                AddFriends
                                ReUpdate_User(txt_friendName.Text)
                                ReUpdate
                                DISPLAY "Request successfully sent to " + txt_friendName.Text"
                        ELSE
                                DISPLAY message
                        ENDIF
                ELSE
                        DISPLAY "User does not exist."
                ENDIF
        ENDIF
END btn_requestFriend_Click


BEGIN PopulateWIthFriendRequest
        OPEN connection to cs
        POPULATE dgv_friend where username = txt_friendName.Text
        CLOSE connection to cs
END PopulateWIthFriendRequest


BEGIN CheckIfFriendExists
        SET message TO ""
        SET frendName = txt_friendName.Text
```

```
        SET friendExists TO if the friends column of dgv_user contains friendname
        SET inviteExists TO if the sentReqeusts column of dgv_user contains friendname
        SET recieved TO if the friendInvitations column of dgv_user contains friendname

        IF friendsExists == true THEN
                SET message TO "You are already friends with this user."
        ELSE IF sentExists == true THEN
                SET message TO "You have already requested this user."
        ELSE IF received == true THEN
                SET message TO "You already have a request from this user."
        ENDIF
        RETURN message
END CheckIfFriendExists

BEGIN AddFriends
        UserToFriendsList
        FriendToSentList
END AddFriends

BEGIN UserToFriends
        SET friendName TO txt_friendName.Text
        SET sentRequests TO sentRequests column from dgv_friend
        ADD user TO sentRequests
        OPEN connection to cs
        UPDATE sentRequests for friendName
        CLOSE connection to cs
END UserToFriends
```

```
BEGIN FriendToSentList
        SET friendName TO txt_friendName.Text
        SET sentRequests TO sentRequests column from dgv_userData
        ADD friendName TO sentRequests
        OPEN connection to cs
        UPDATE sentRequests
        CLOSE connection to cs
END FriendToSentList


BEGIN CheckIfUserExists
        OPEN connection to cs
        POPULATE table2 where username = txt_friendName.Text
        CLOSE connection to cs
        IF number of rows in table2 > 0 THEN
                RETURN true
        ELSE
                RETURN false
        ENDIF
END CheckIfUserExists


INITALISE selectedInviteName
BEGIN dgv_friendInvites_CellContentClick
        REM "public variable set"
        REM "mostRecent is called on the click event"
        IF mostRecent == true
                SET selectedInviteName to username column of selected row
                IF accept friend request button is clicked THEN
                        DISPLAY "Are you sure you want to add" + selectedInviteName
                        GET DialogResult
```

```
                IF DialogResult == OK THEN
                        MoveUsers
                        ReUpdate_User(selectedInviteName)
                        ReUpdate_User(username)
                        REM "update the datagridview"
                        ReUpdate
                ENDIF
        ELSE IF ignore button is clicked THEN
                DISPLAY "Are you sure you want to ignore request from" + selectedInviteName
                GET DialogResult
                IF DialogResult == OK THEN
                        RemoveRequest
                        ReUpdate_User(selectedInviteName)
                        ReUpdate_User(username)
                        REM "update the datagridview"
                        ReUpdate
                ENDIF
        ENDIF
    ENDIF
END dgv_friendInvites_CellContentClick


BEGIN RemoveRequest
    PopulateUser
    PopulateWithFriend
    RemoveInviteFormUser
    RemoveFromFriendsSent
END RemoveRequest


BEGIN PopulateWithFriend
```

```
        OPEN connection to cs
        POPULATE dgv_friend with information where username = selectedInviteName
        CLOSE connection to cs
END PopulateWithFriend

BEGIN RemoveInviteFormUser
        SET friendInvitations TO friendInvitations column from dgv_user
        REMOVE selectedInviteName from friendInvitations
        OPEN connection to cs
        UPDATE friendInvitations
        CLOSE connection to cs
END RemoveInviteFromUser

BEGIN RemoveFromFriendsSent
        SET sentRequests TO sentRequests column from dgv_friend
        REMOVE username from sentRequests
        OPEN connection to cs
        UPDATE sentRequests for selectedInviteName
        CLOSE connection to cs
END RemoveFromFriendsSent

BEGIN MoveUsers
        PopulateUser
        PopulateWithFriend
        MoveFriend
        MoveUser
END MoveUsers
BEGIN MoveFriend
        SET friendInvitations TO friendInvitations column from dgv_user
```

```
        REMOVE selectedInviteName from friendInvitations
        SET friends TO friends column from dgv_user
        ADD selectedInviteName TO friends
        OPEN connection to cs
        UPDATE friendInvitations
        UPDATE friends
        CLOSE connection to cs
BEGIN MoveFriend

BEGIN MoveUser
        SET friendInvitations TO friendInvitations column from dgv_friend
        REMOVE user from friendInvitations
        SET friends TO friends column from dgv_friend
        ADD user TO friends
        OPEN connection to cs
        UPDATE friendInvitations for selectedInviteName
        UPDATE friends for selectedInviteName
        CLOSE connection to cs
BEGIN MoveFriend

INITALISE mostRecent
```

REM "Do not confuse dgv_friends which displays all of the user's friends and dgv_friend which displays information for the selected friend"

```
BEGIN dgv_friends_Click
        mostRecent = checkLastUpdated_User(username)
        IF mostRecent == false THEN
                DISPLAY "Your friends have been recently changed. Updating"
                ReUpdate
        ENDIF
```

```
END dgv_friends_Click


BEGIN CheckLastUpdated_User(user)
        OPEN connection to cs
        POPULATE table2 with the information for this event
        CLOSE connection to cs
        SET lastUpdated TO lastUpdated column of table2
        SET clientsLastUpdated TO lbl_lastUpdate
        IF lastUpdated is later than the clientsLastUpdated THEN
                RETURN false
        ELSE
                RETURN true
        ENDIF
END CheckLastUpdated_User


BEGIN ReUpdate_User
        OPEN connection to cs
        UPDATE lastUpdated for the user with the current time
        CLOSE connection to cs
END ReUpdate_User


BEGIN dgv_friendInvites_Click
        mostRecent = checkLastUpdated_User(username)
        IF mostRecent == false THEN
                DISPLAY "Your friends have been recently changed. Updating"
                ReUpdate
        ENDIF
END dgv_friendInvites_Click
```

```
INITIALISE selectedRequestName

BEGIN  dgv_sentRequests_CellContentCLick

        IF mostrecent == true THEN

                IF unrequest button is pressed THEN

                        SET selectedRequestName TO username of row selected

                        DISPLAY "Are you sure you want to remove your friend request to" +
                        selectedRequestName

                        GET DialogResult

                        IF DialogResult == OK THEN

                                PopulateWithSelectedRequestFriend

                                ReUpdate_User(selectedRequestName)

                                ReUpdate_User(username)

                                RemoveSentRequestOfFriend

                                RemoveRequestFromFriend

                                ReUpdate

                                DISPLAY "Removed friend request"

                        ENDIF

                ENDIF
        ENDIF

END dgv_sentRequests_CellContentClick


BEGIN PopulateWithSelectedRequestFriend

        OPEN connection to cs

        POPULATE dgv_requestData with information where username= selectedRequestName

        CLOSE connection to cs

END PopulateWithSelectedRequestFriend


BEGIN dgv_sentRequests_Click

        mostRecent = checkLastUpdated_User(username)

        IF mostRecent == false THEN
```

```
            DISPLAY "Your friends have been recently changed. Updating"
            ReUpdate
        ENDIF
END dgv_sentRequests_Click


BEGIN RemoveSentRequestOfFriend
        SET sentRequests TO sentRequests column from dgv_user
        REMOVE selectedRequestName from sentRequests
        OPEN connection to cs
        UPDATE sentRequests
        CLOSE connection to cs
END RemoveSentRequestOfFriend


BEGIN RemoveRequestFromFriend
        SET friendInvitations TO friendInvitations column from dgv_requestData
        REMOVE user from friendInvitations
        OPEN connection to cs
        UPDATE friendInvitations for selectedRequestName
        CLOSE connection to cs
END RemoveRequestFromFriend
```

PRE-USE

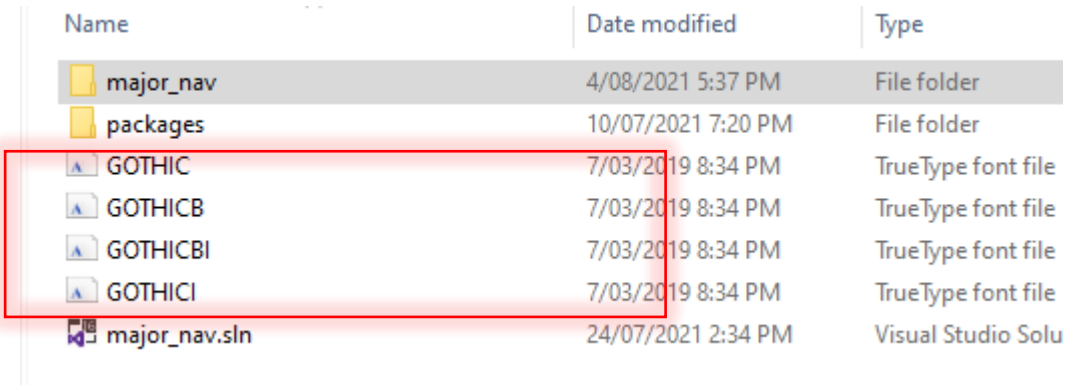
CENTURY GOTHIC

CONNECT

Jade Harris | **12SDD**

TO INSTALL FONT

I. Locate the CENTURY GOTHIC fonts included in the software package
   a. Else, download off: https://freefontsfamily.com/century-gothic-font-family/
II. Drag into C:\Windows\Fonts
   a. Else, follow these instructions: https://faqs.skillcrush.com/article/275-downloading-installing-a-font-on-your-computer



Name	Date modified	Type
major_nav	4/08/2021 5:37 PM	File folder
packages	10/07/2021 7:20 PM	File folder
GOTHIC	7/03/2019 8:34 PM	TrueType font file
GOTHICB	7/03/2019 8:34 PM	TrueType font file
GOTHICBI	7/03/2019 8:34 PM	TrueType font file
GOTHICI	7/03/2019 8:34 PM	TrueType font file
major_nav.sln	24/07/2021 2:34 PM	Visual Studio Solu

QUICK-START

USER DOCUMENTATION

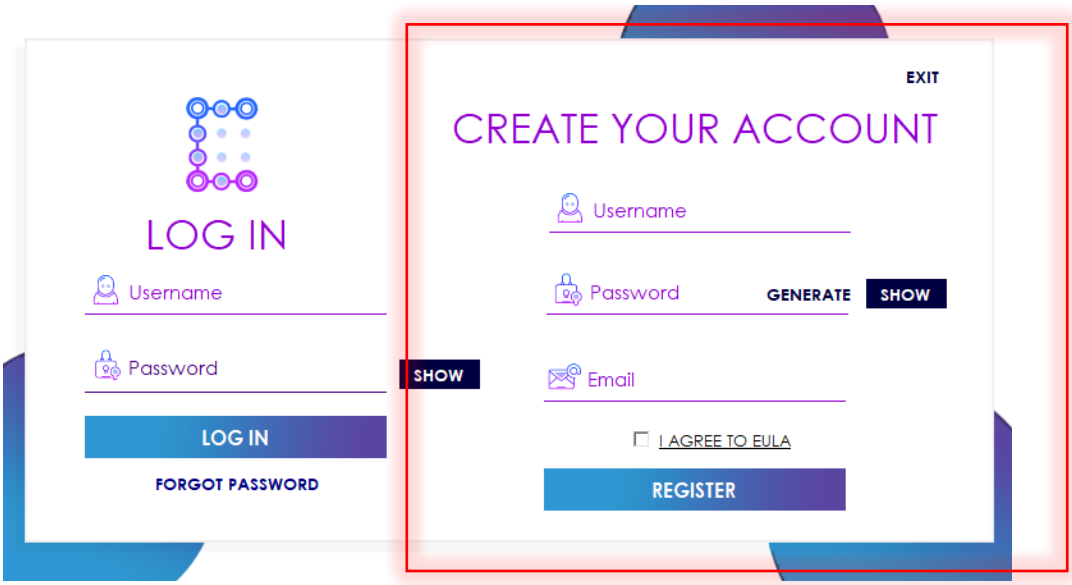
Jade Harris | **12SDD**

LAST UPDATED: 2021

CONTENTS (Click to navigate to)

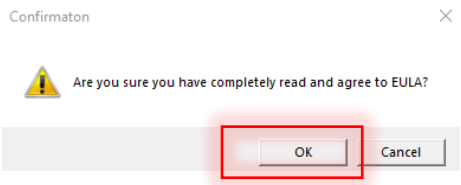
NEW USER

CREATING A USER



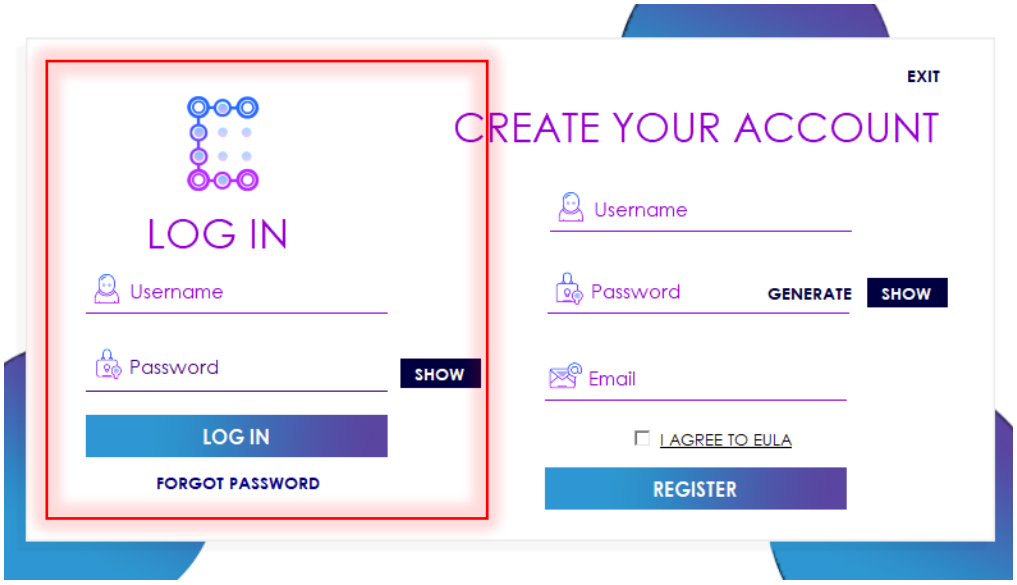
NOTE: No textboxes can have ' or spaces entered (including create event)

I. Create a username
   a. Max 9 characters
II. Create a password
   a. Max 12 characters
   b. GENERATE: 12 random alpha-numeric characters
   c. SHOW: When HOVERING, reveal your password
III. Enter your email
   a. Max 25 characters
   b. Ensure you have access to email
IV. Read and agree to EULA



NEW USER

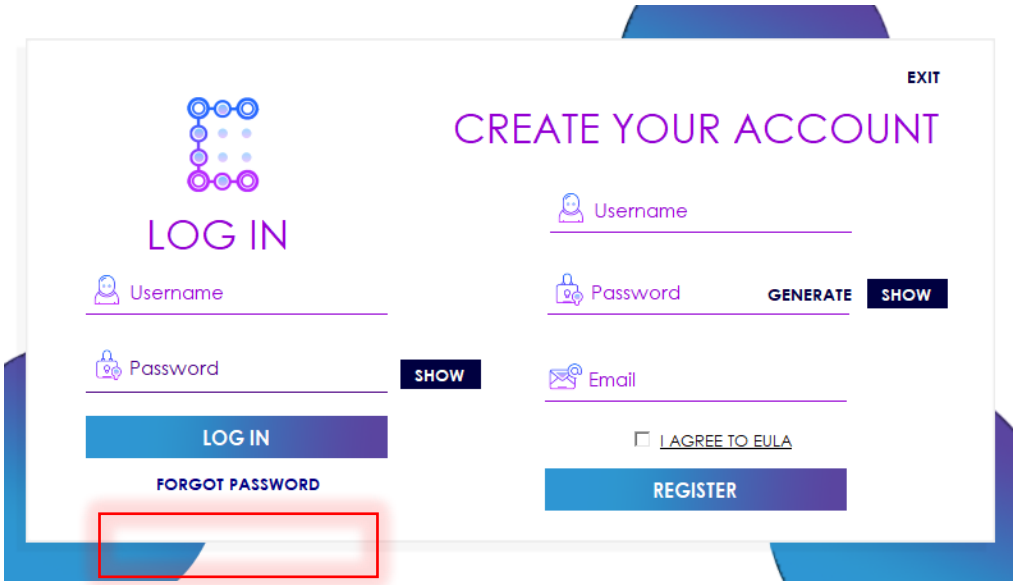
LOG IN



I. Enter username
II. Enter password
    a. **SHOW:** When HOVERING, reveal your password

NEW USER

FORGET PASSWORD

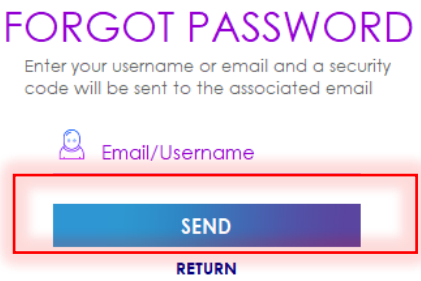


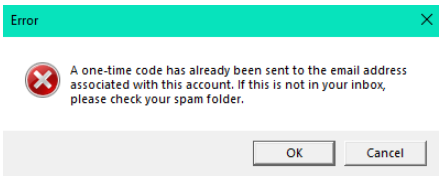
I. Select FORGOT PASSWORD

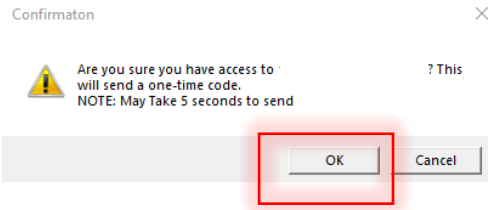
II. Enter remembered email/username

   a. Ensure you have access to email

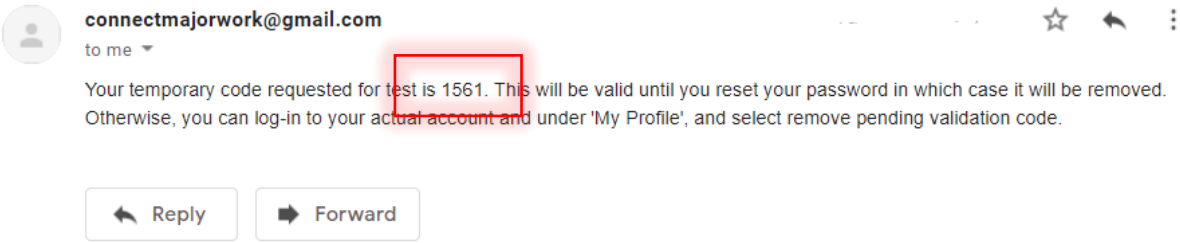
   b. If you already have requested a code before, check your emails.



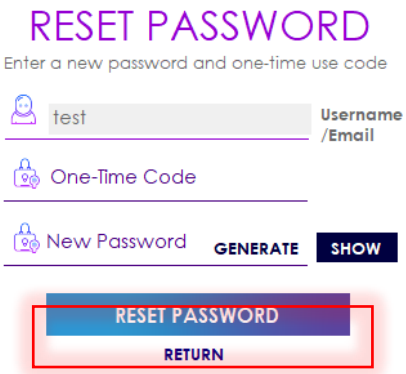




III. Get security code



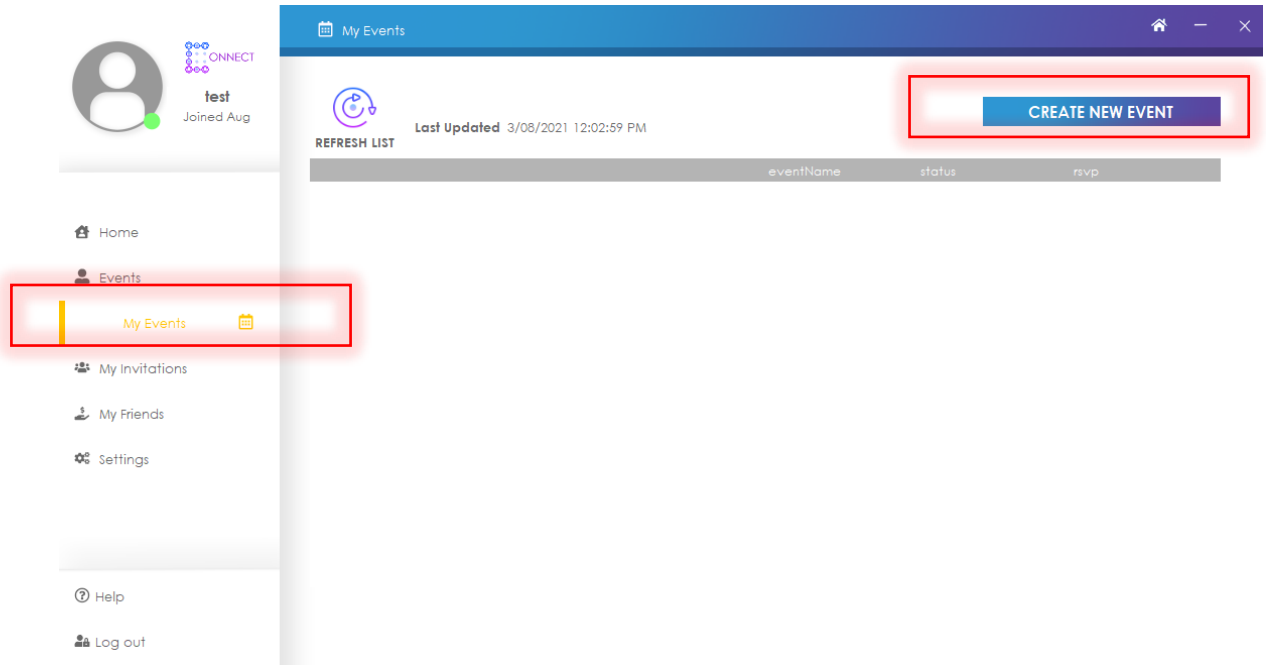
IV. Enter this code and your new password

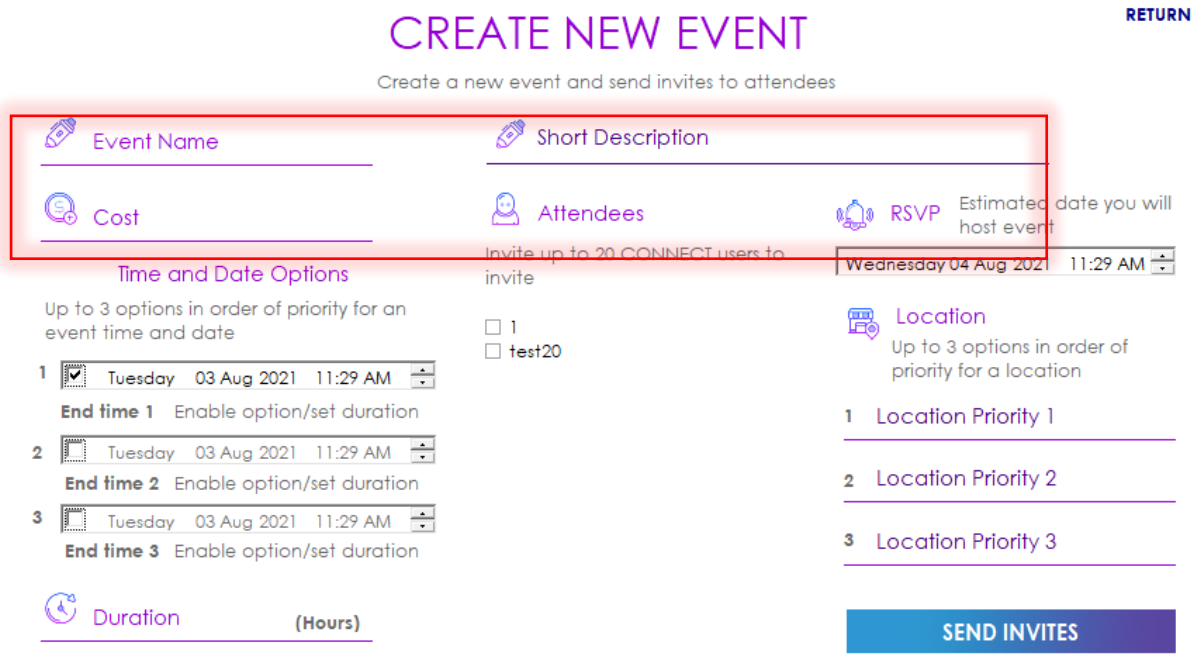


MY EVENTS

CREATE NEW EVENT

I. Go to MY EVENTS
II. Select CREATE NEW EVENT

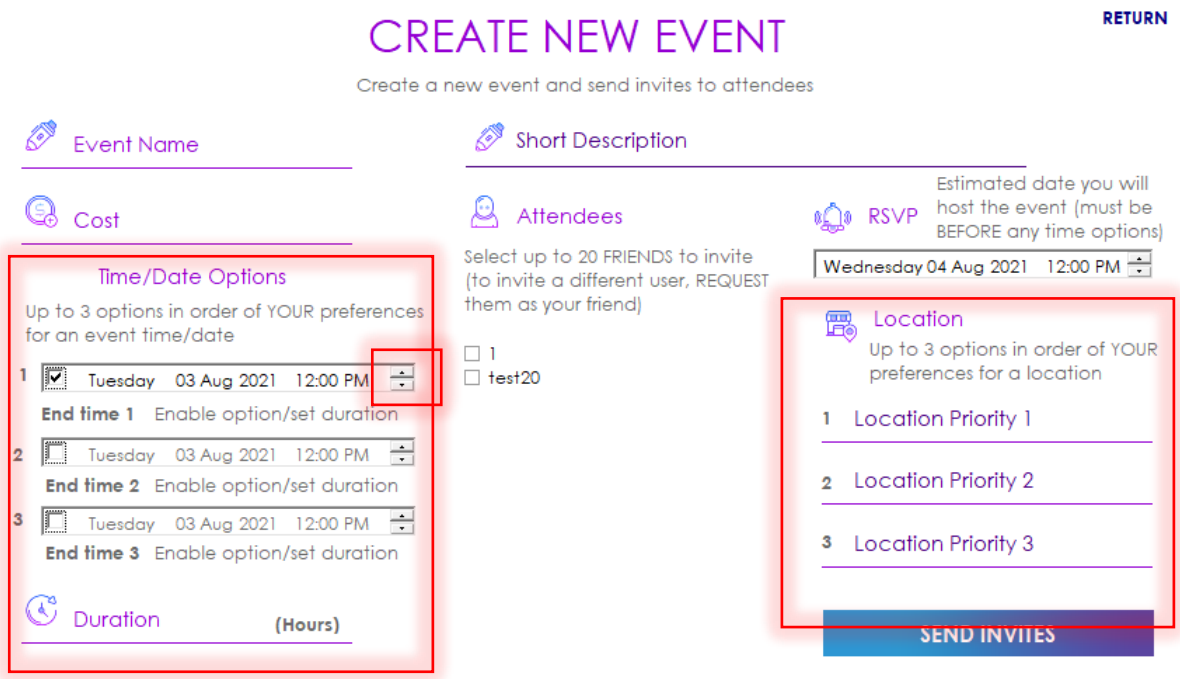




I. Enter event name
   a. Max 20 characters
   b. NOTE: Your event cannot have the same event as another regardless of who is the host
II. Enter short description
   a. Max 40 characters
III. Enter cost
   a. To 2 decimal places
IV. Enter short description
   a. Max 40 characters

SETTING TIME AND LOCATION PREFERENCES

I. Add YOUR preferences for location/time(date) for the event
II. Use ARROWS for time/date
   a. Selecting only ONE time date option will mean USERS CANNOT VOTE and these will automatically be set when you host your event
   b. No time/location can be identical
   c. Once hosted, NO OPTIONS CAN BE ADDED (only removed)
   d. Time and date options are influenced by DURATION

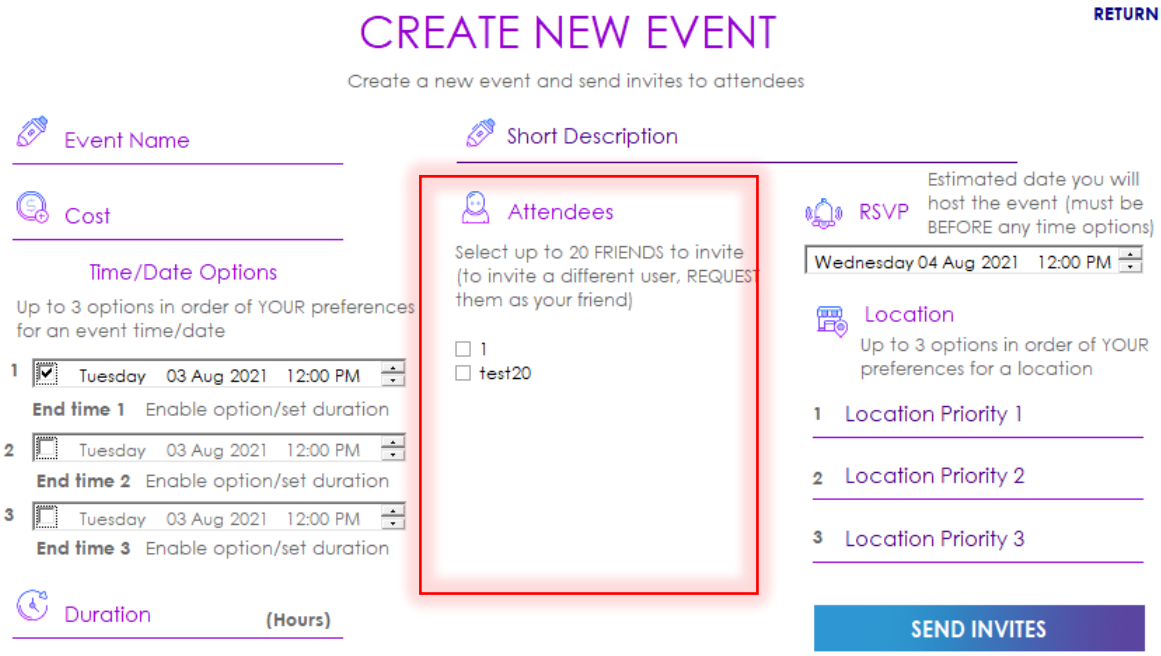


INVITING FRIENDS (ATTENDEES)

=

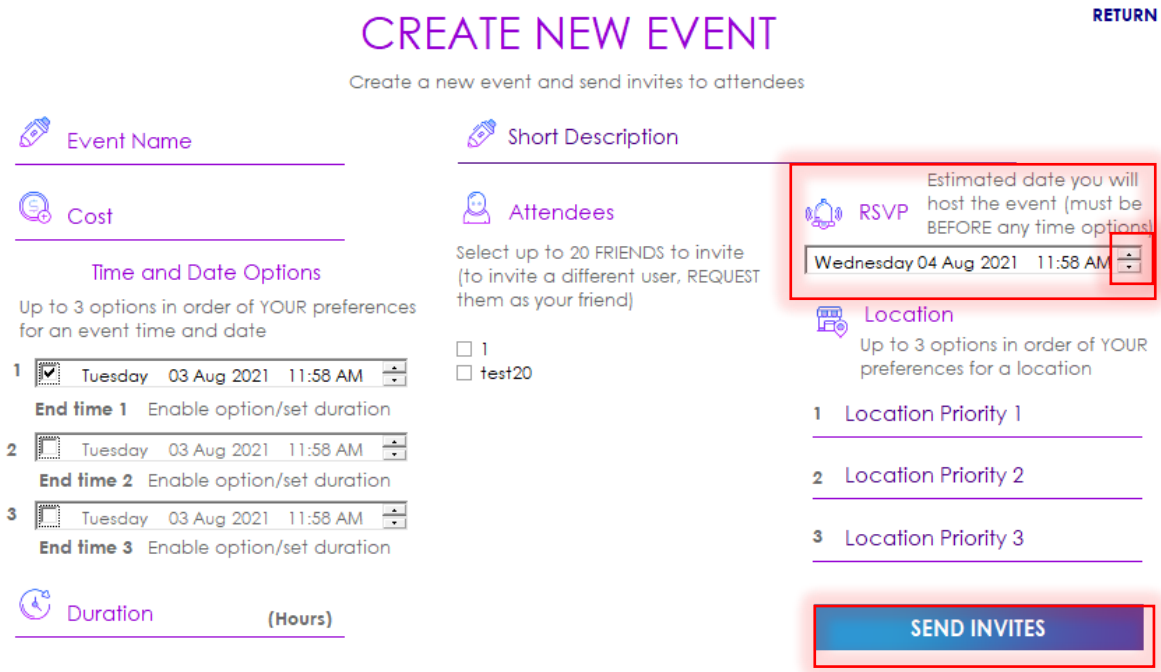
NOTE: To invite friends, YOU MUST ADD THEM FIRST (ADD A FRIEND)

I. Use checkboxes to select friends



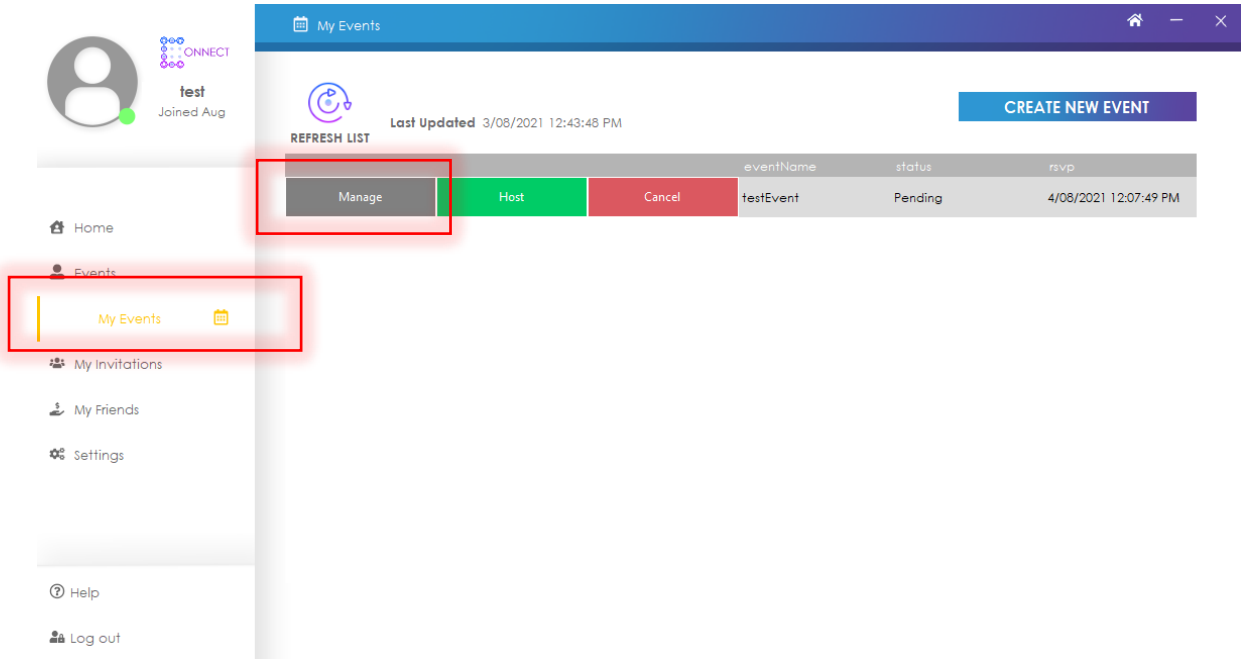
SETTING RSVP

I. Must be BEFORE any time/date option

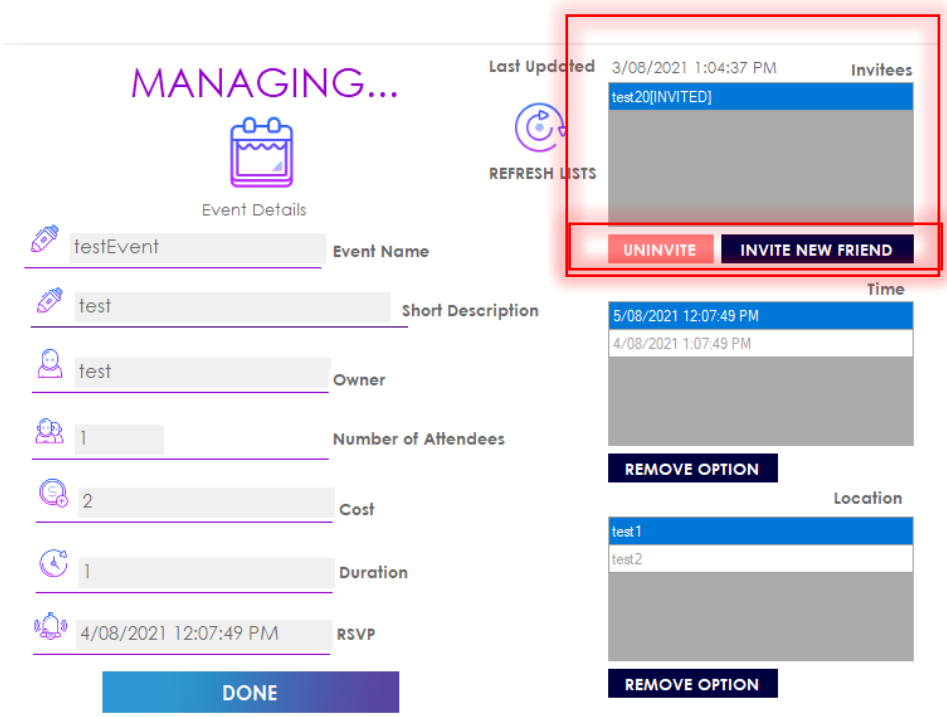


MANAGING MY EVENTS

I. Go to MY EVENTS
II. Click MANAGE



MANAGE(ADD/REMOVE) ATTENDEE/INVITEES



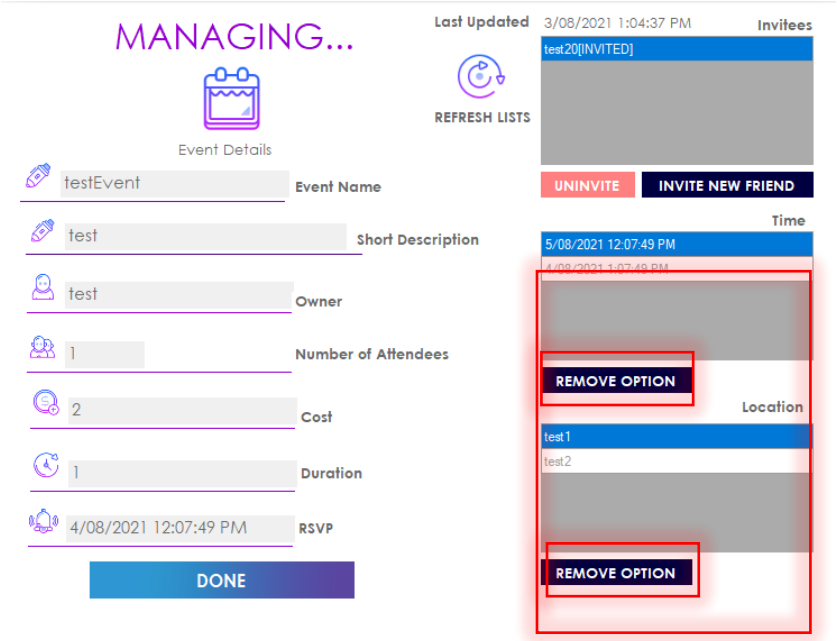
REMOVE ATTENDEE/INVITEE

I. Select USER
   a. Users suffixed with whether they have already accepted invite
II. Click UNINVITE

ADD NEW ATTENDEE/INVITEE

III. Click INVITE NEW FRIEND
   a. If EVENT
      i. ALREADY HOSTED: user automatically added (they can leave)
      ii. PENDING: invite sent (for their vote)

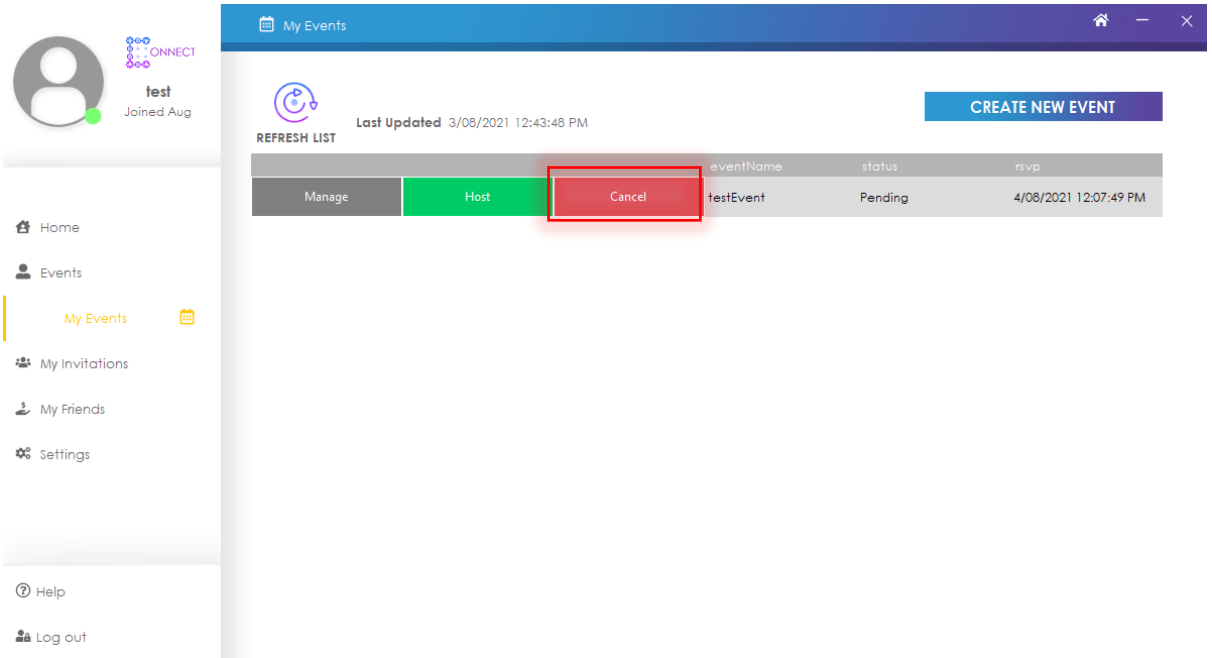
REMOVE TIME/LOCATION OPTION



I. Select appropriate option
II. Click REMOVE OPTION
   a. Once removed, you CANNOT ADD another option

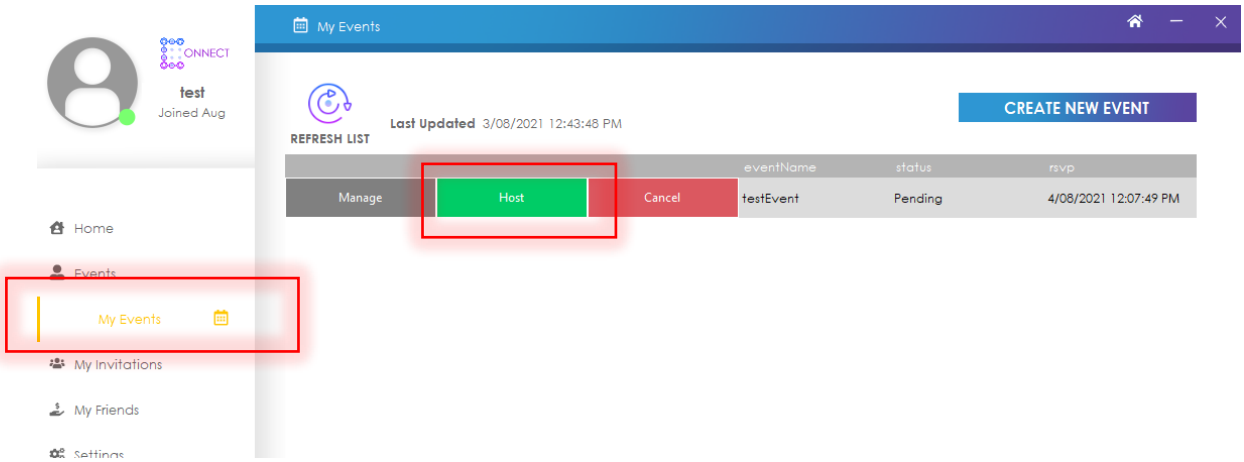
MY EVENTS

CANCEL AN EVENT



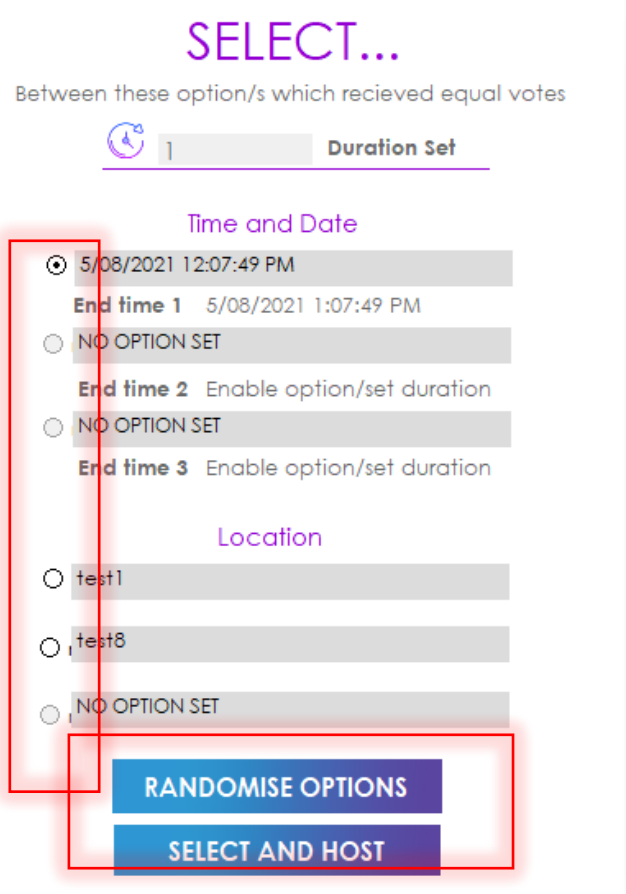
MY EVENTS

HOST AN EVENT



I. Go to MY EVENTS
II. Click HOST
III. If no options received the same number of votes, then HOSTING the event will automatically set the location and time to the option that received the highest number of votes. The invitees will then become attendees, and once the event is done you can complete it.

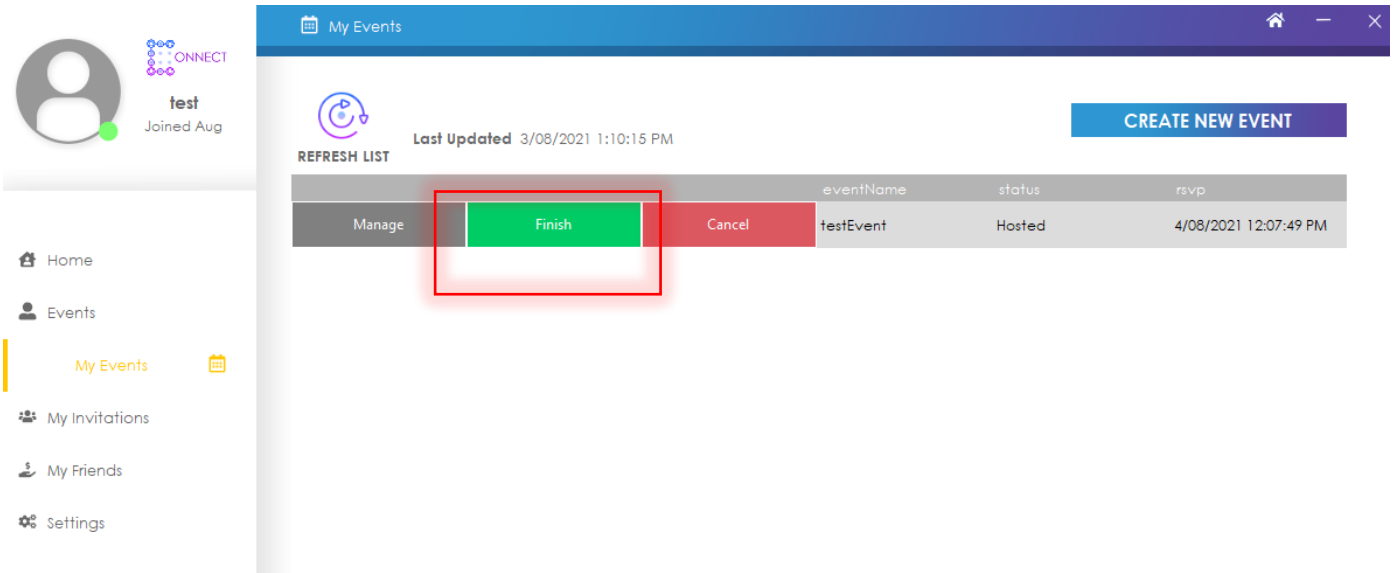
EQUAL VOTES



I. Select RADIO BUTTONS of option that the event will be set to
   a. **RANDOMISE OPTIONS** will randomly select the radio buttons

EVENTS

COMPLETING EVENT

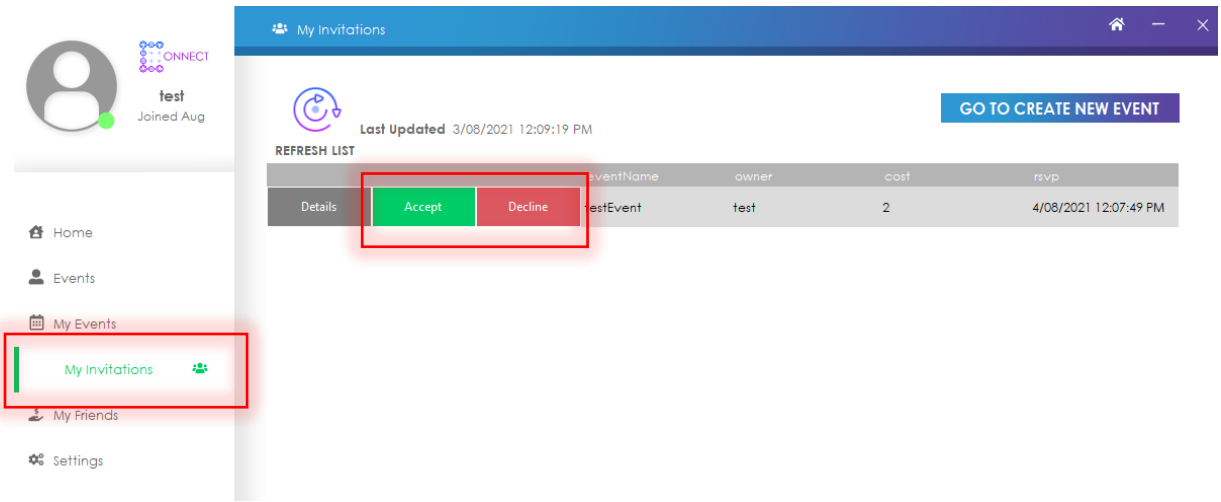


I. If event is hosted, **HOST button becomes FINISH**
II. Once you are ready to COMPLETE event, CLICK FINISH
   a. This is irreversible. Event will be removed.

MY INVITATIONS

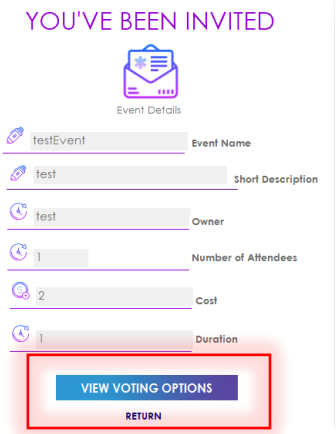
ACCEPT/DECLINE INVITATION

I. Go to MY INVITATIONS
II. Use buttons to ACCEPT/DENY selected invitation

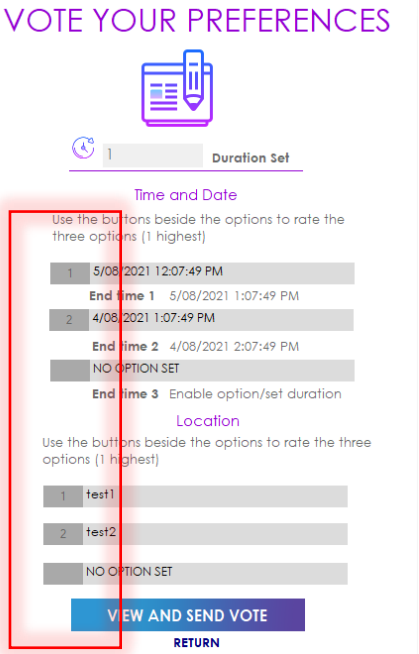


SETTING VOTES/PREFERENCES

III. Click ACCEPT button



IV. Use the buttons beside each option to toggle in order of preference

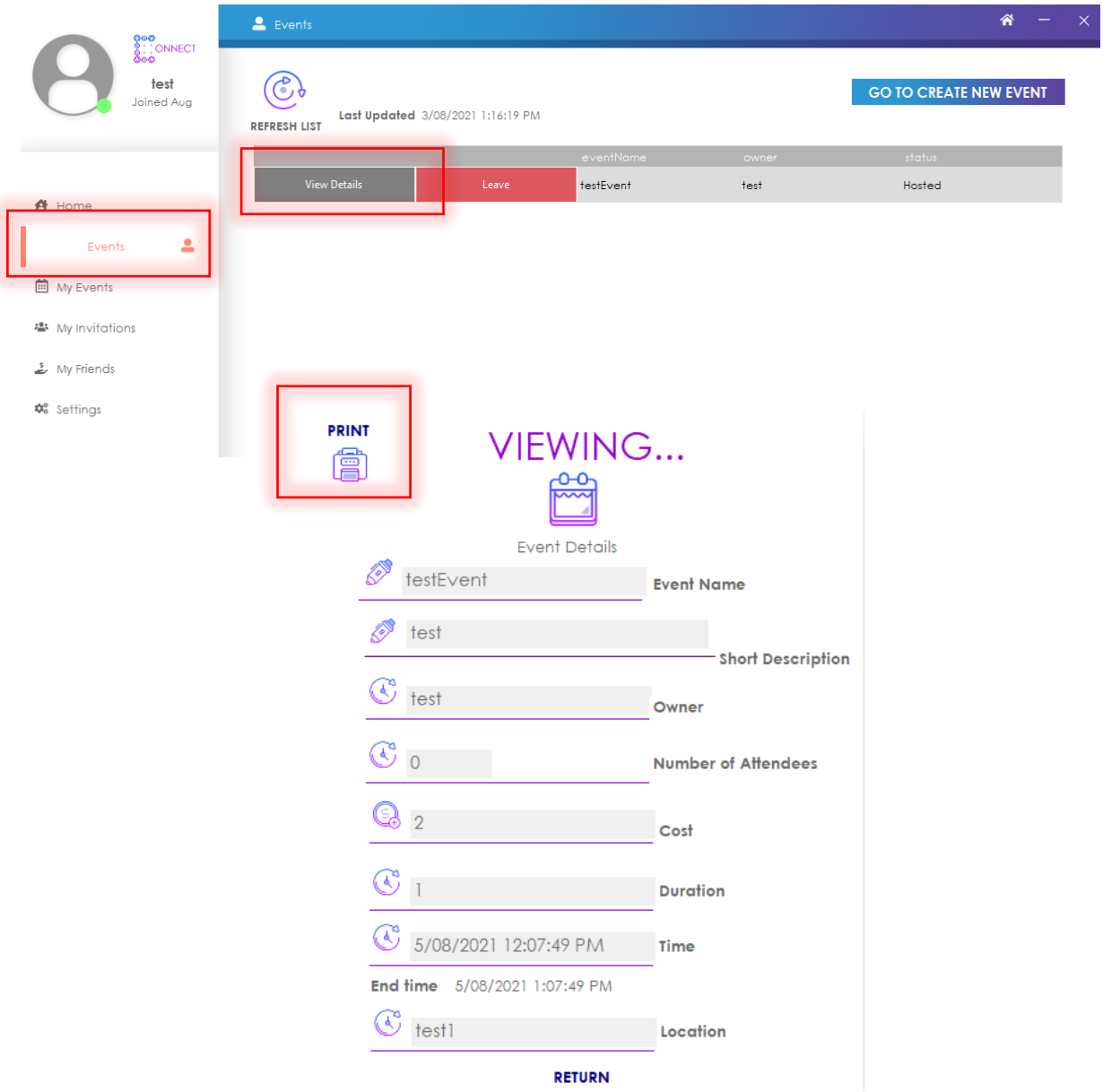


EVENTS

VIEW DETAILS

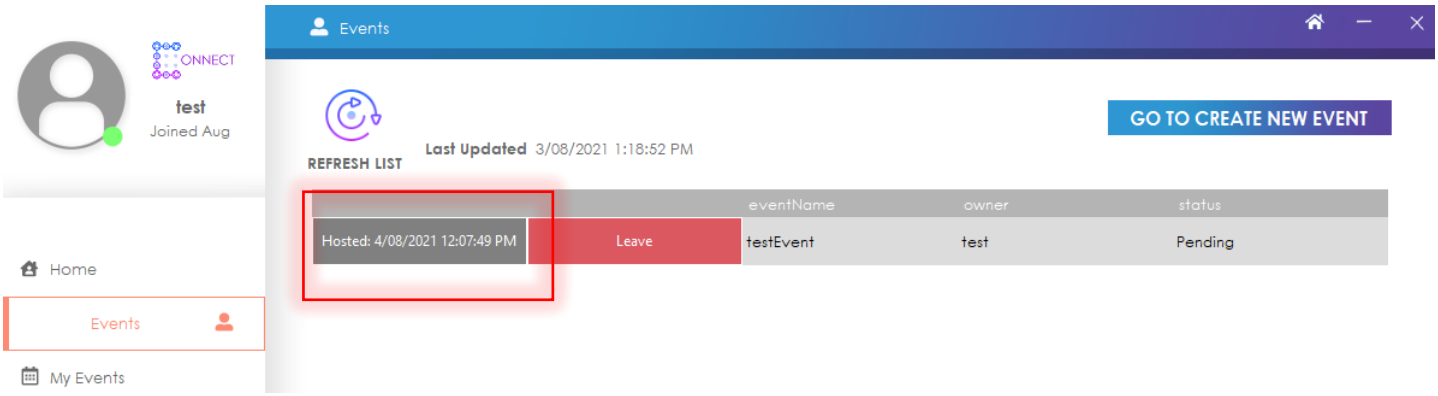
IF EVENT IS HOSTED

I. Go to MY INVITATIONS
II. Click VIEW DETAILS
III. Option to PRINT event details



IF EVENT IS NOT HOSTED:

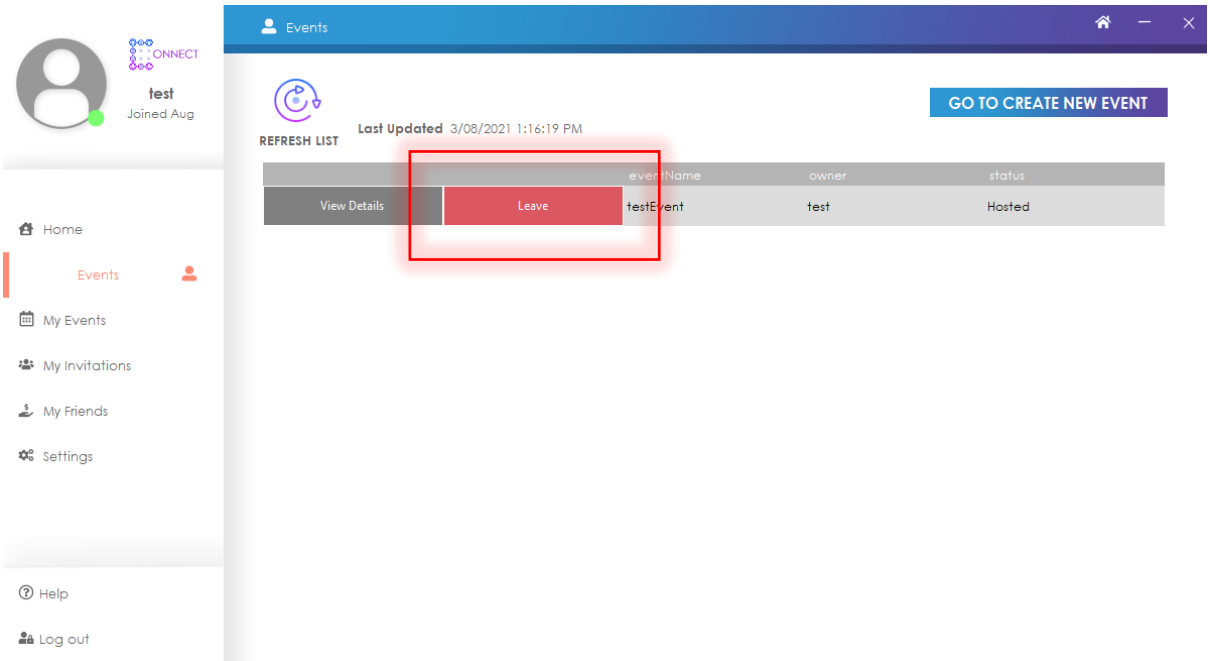
I. Cannot see details only RSVP date



EVENTS

LEAVING AN EVENT

I. NOTE: Leaving an event not yet hosted will **remove your votes**



MY FRIENDS

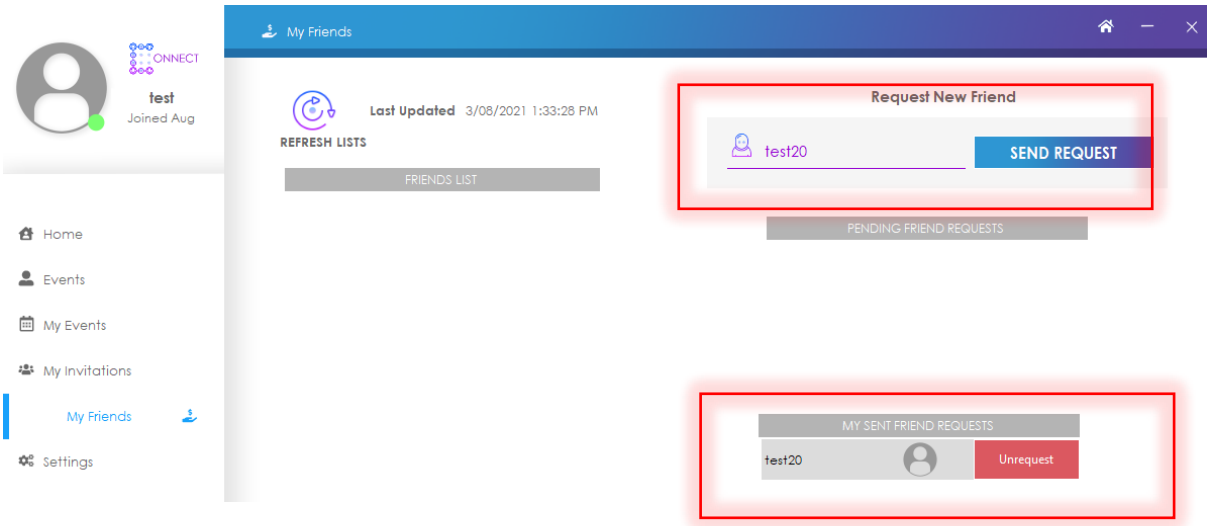
SENT/CANCEL FRIEND REQUEST

SEND

I. Enter username of other account
II. Wait for user to accept

CANCEL REQUEST

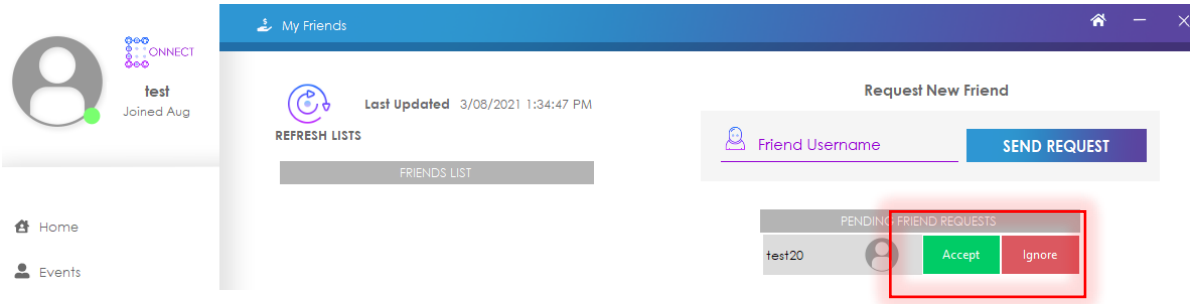
I. (Assuming user has accepted request)
II. Click UNREQUEST



MY FRIENDS

ACCEPT/IGNORE RECEIVED REQUEST

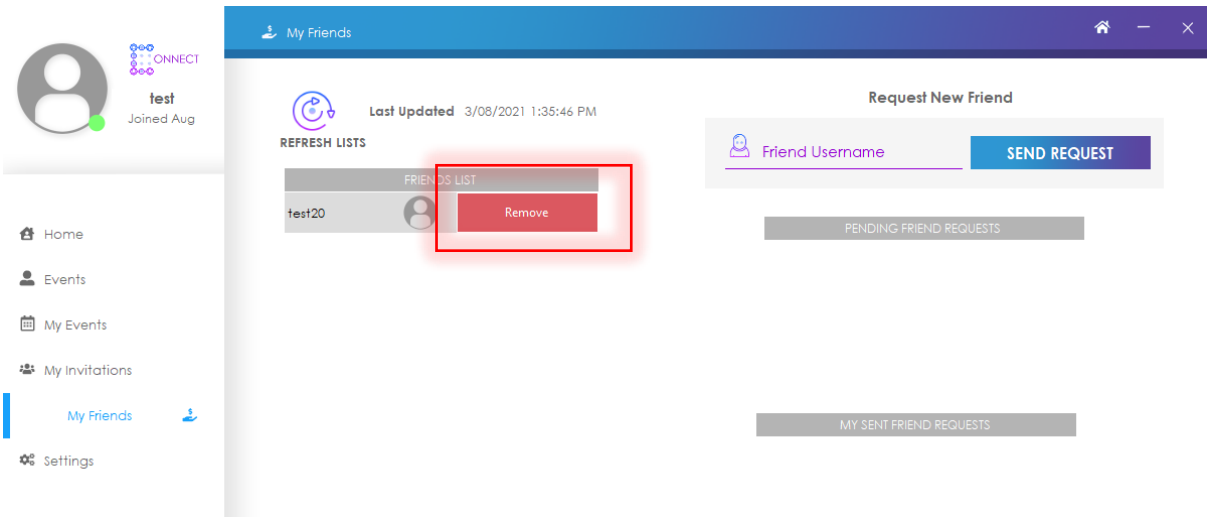
I. Use buttons to ACCEPT or IGNORE request



MY FRIENDS

REMOVE FRIEND

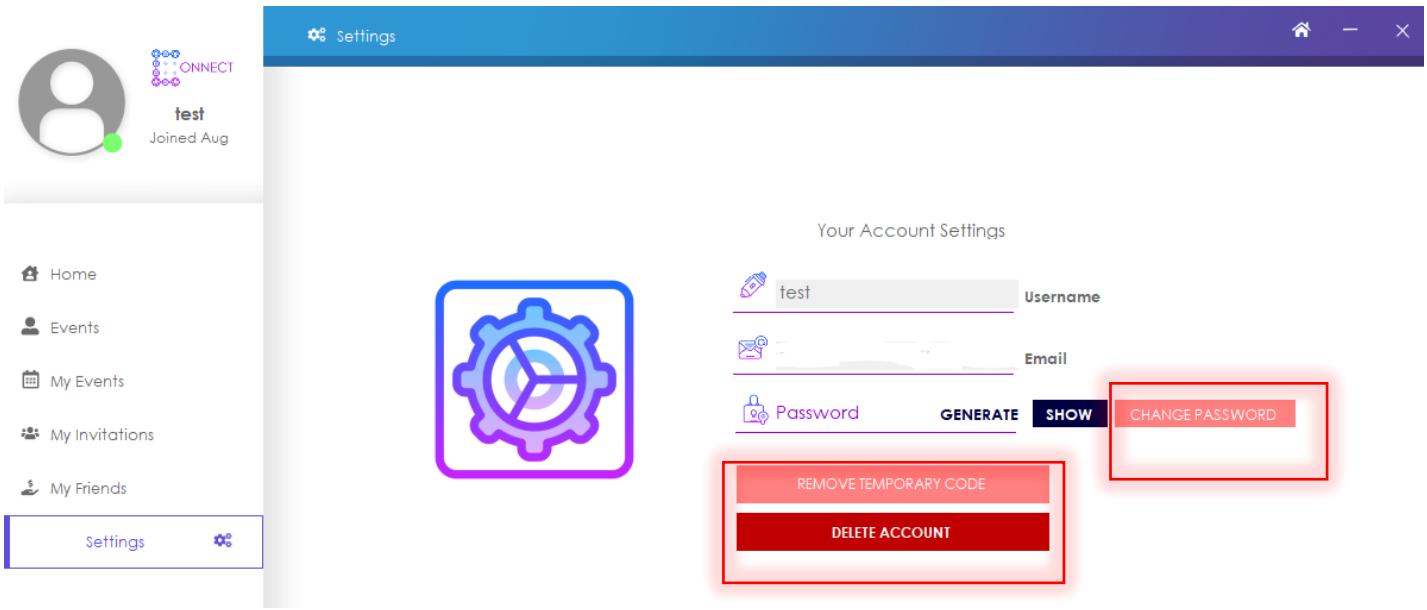
I. Click REMOVE beside friend's name
   a. You must re-request them
   b. Any events of yours they are in will remain



SETTINGS

RESET YOUR PASSWORD

I. Resets password from next log in



SETTINGS

REMOVE TEMPORARY CODE

II. Remove the code sent to your email allowing you to reset password (if any exist)
   a. Next time forgot password, new passcode will be sent

SETTINGS

DELETE ACCOUNT

I. Removes account along with any events invited to and friends

REPORT

TESTING AND EVALUATING

AUTHOR: Jade Harris | **12SDD**

MENTOR: Adam Leserve

C O N T E N T S

TESTING AND EVALUATING REPORT

TESTING AND EVALUATING EXEUCTION

Testing and evaluating of CONNECT was conducted by combining a variety of effective methods throughout the development process.

During development, testing commenced continuously by constructing and UNIT TESTING each isolated function in a separate program. Through the use of drivers and stubs where necessary, once the envisioned functionality was achieved, the function was implemented into the other functions and INTEGRATION TESTED. Further, I gained results from VOLUME TESTING with 5 simultaneous computers accessing the database. These were evaluated against the QUALITY ASSURANCE and BENCHMARK standards. If runtime or logic errors occurred, I employed breakpoints and debugging output statements to identify the issue. Otherwise, the module was sufficiently tested and evaluated.

Once the functionality of the program was complete and able to be used completely, I utilised BETA TESTING to evaluate the end user's experience with a diverse range of hardware and software, important for the 'general public' target audience of CONNECT. Accompanied by a BETA test survey, this allowed me to evaluate their feedback against the application's requirements, particularly ergonomic and subjective specifications.

TESTING AND EVALUATING REPORT

ANALYSIS OF BETA TESTING RESULTS

Overall, the BETA test results revealed significant interface and reliability issues that occurred due to using a different system configuration, and testing from a fresh perspective. These were resolved immediately.

SECTION 1: FUNCTIONALITY

Overall, the demographic of testers was deliberately diverse with ages ranging from 18-50 and best reflected the 'general' target audience. All responses strongly agreed that CONNECT achieved its purpose successfully.

SECTION 2: INTERFACE

Most responses rated the interface 3, 4 and 5. This rating fulfilled the application's requirement of receiving positive/overall satisfied feedback.

For the response that rated the interface a 3, the main concern was the clarity of the error and success messages. This was overlooked during development, but is extremely impactful in UX, thus it was resolved easily and quickly.

Both the 3 rating and 4 rating noted the poor interface due to the font not transferring over different software configurations, This significantly impacted the information available for the user on the interface. To rectify this, I included the font required for the interface in the software package as well as instructions to install it.

SECTION 3: RELIABILITY

Responses for the level of reliability of the software were between 4 and 5, achieving the programs desired specification of positive/satisfied feedback.

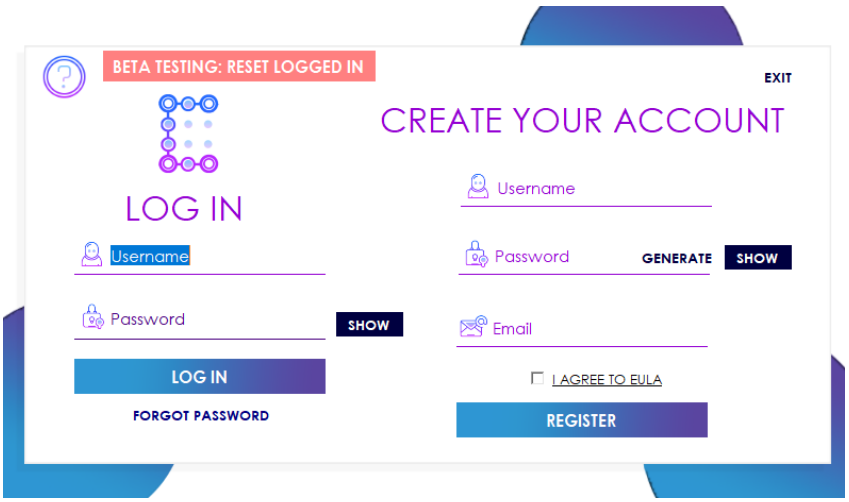
The two ratings of 4 identified bugs where features of the application had not yet been implemented: when both the VIEW button on the frm_invitedEvents and the HOSTED button in frm_myEvents did not cause any action. The other rating of 4 raised the bug of being able to copy-and-paste a longer password into the entry form. This was resolved by disabling copy-and-paste by turning the ShortcutsEnabled property to false.

These intricate responses ensured that the reliability of the database remained intact.

TESTING AND EVALUATING REPORT

TEST DATA TABLES

FRM_LOGIN



This form allows user to both create and log in to program. Use of alphanumeric, and special characters is required for testing. Only valid username, password and email can be entered.

PASSWORD/USERNAME

- 4-9 characters
- Cannot enter ‘ character
- No spaces

Further, EMAIL

- Must contain @ symbol

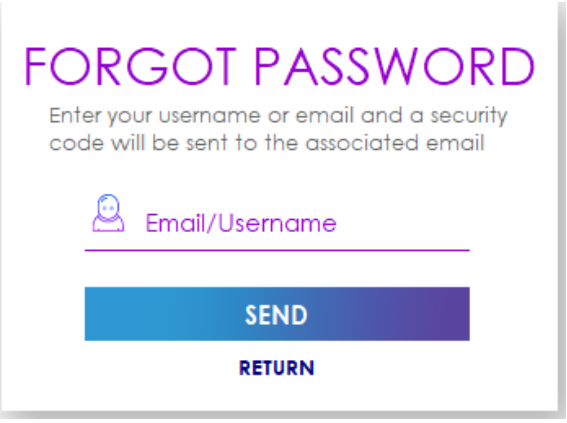
SIGNUP USERNAME (differs from normal username

- 4-9 characters
- Cannot enter ‘ character
- No spaces

INPUT	EXPECTED OUTPUT	OUTPUT
Test user	Testuser	Testuser
Test’user	Testuser	Testuser
Testuserrrrrrr	Testuserr	Testuserr
$Testuser	$Testuser	$Testuser

“” or Username	Message Box Error: Please complete all fields.	Message Box Error: Please complete all fields
T	Message Box Error: Must be at least 4 characters	
Testuser (User doesn’t exist)	Message Box Error: No user exists.	Message Box Error: No user exists.
Testuser (User does exist and password correct)	Open hub form	Open hub form

FRM_FORGOTPASSWORD



This form allows users to enter a remembered email or username (if it exists) to get a temporary reset code. Password can then be reset. Only valid username/email can be entered and searched. NOTE: Email not required to have @ symbol (because it is checked)

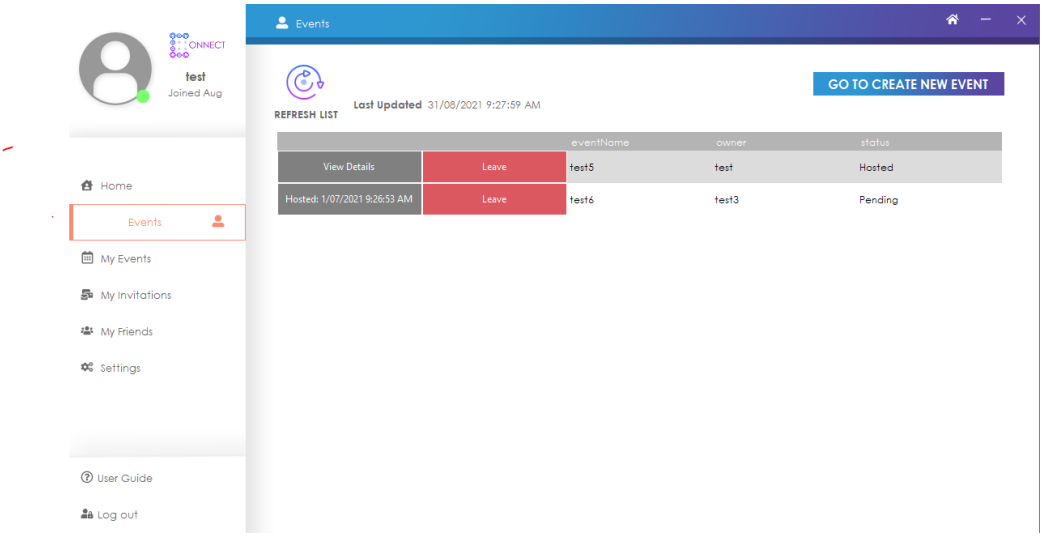
- Max 25 characters
- Cannot enter ‘ character

EMAIL/USERNAME

INPUT	EXPECTED OUTPUT	OUTPUT
Test user	Testuser	Testuser
Test’user	Testuser	Testuser
Testuserrrrrrrrrrrrrrrrrr	Testuserrrrrrrrrrrrrrrrrr	Testuserrrrrrrrrrrrrrrrrr
$Testuser	$Testuser	$Testuser
“” or Email/Username	Message Box Error: Please enter a username or email (cannot be default or empty)	Message Box Error: Please enter a username or email (cannot be default or empty)
T (User doesn’t exist)	Message Box Error: No user exists.	Message Box Error: No user exists.
T (User does exist and error code doesn’t exist)	Message Box: This will send a one-time code to your email Open newPassword form	Message Box: This will send a one-time code to your email Open newPassword form

T (user does exist but error code already exists)	Message Box Error: One time code has already been sent. Open newPassword form	Message Box Error: One time code has already been sent. Open newPassword form

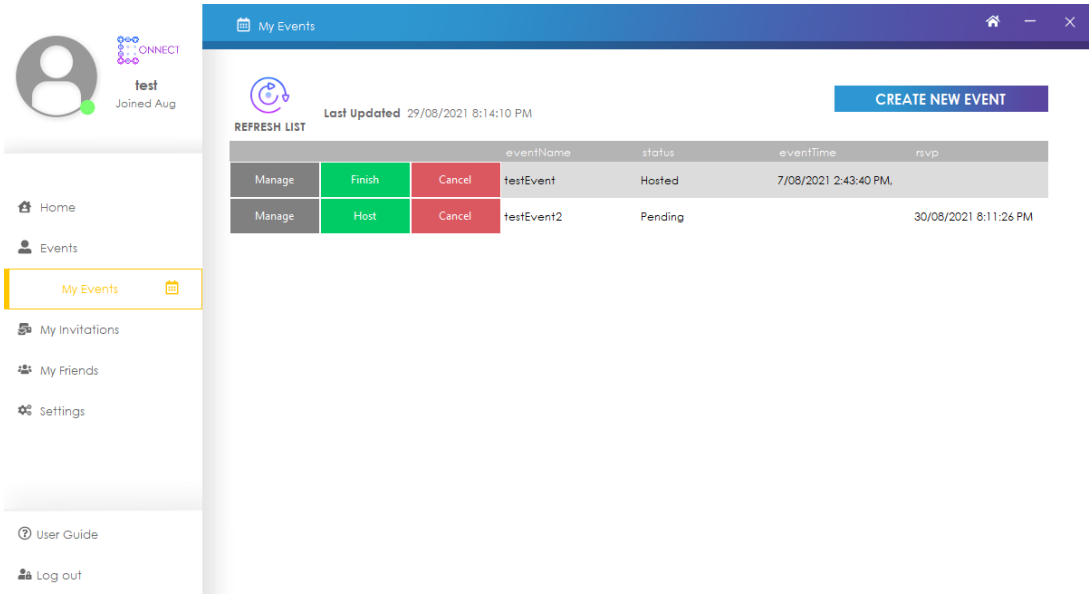
FRM_EVENTS



The 'conditional' buttons embedded in the second column should allow the user to VIEW DETAILS or wait until the event is hosted depending on whether the event is hosted or pending.

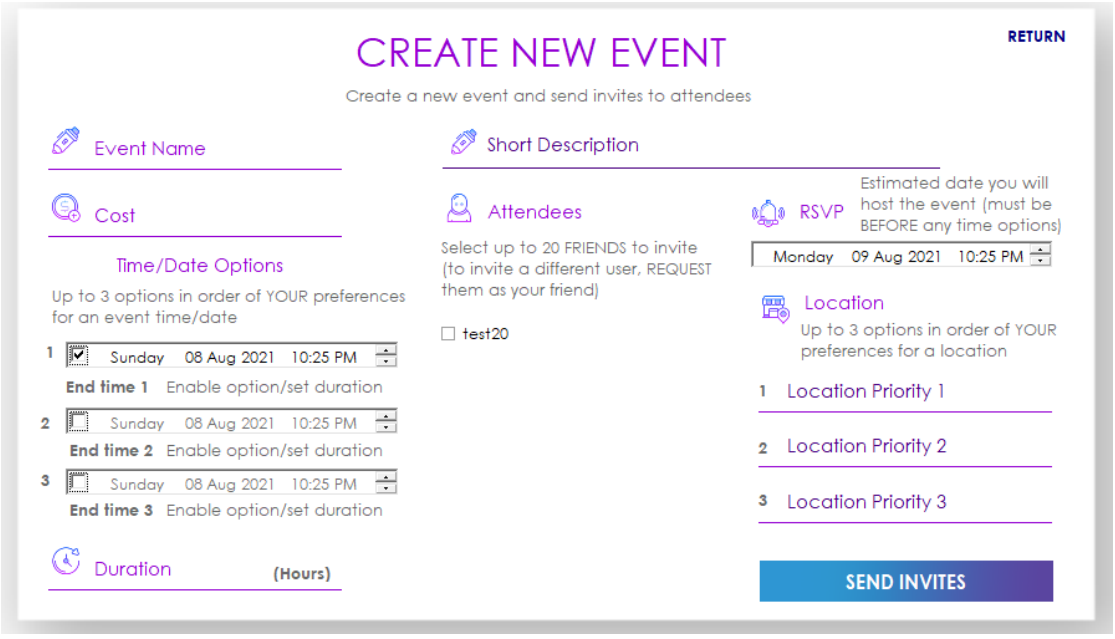
INPUT	EXPECTED OUTPUT	OUTPUT
Status: Hosted	Button displays View Details Open viewEvent form	Button displays View Details Open viewEvent form
Status: Pending	Button displays Hosted: {date hosted} Message Box Error: Please wait until the event is hosted by {user}	Button displays Hosted: {date hosted} Message Box Error: Please wait until the event is hosted by {user}

FRM_MYEVENTS



The 'conditional' buttons embedded in the second column should say HOST or FINISH depending on whether the event is hosted or pending.

INPUT	EXPECTED OUTPUT	OUTPUT
Status: Hosted	Button displays Finish Message Box Confirmation: Are you sure you want to finish the event?	Button displays Finish Message Box Confirmation: Are you sure you want to finish the event
Status: Pending	Button displays Host Message Box Confirmation: Are you sure you want to host the event?	Button displays Host Message Box Confirmation: Are you sure you want to host the event?



This form allows users to create a new event. Primarily, data-validating elements have already been used. For the other text inputs:

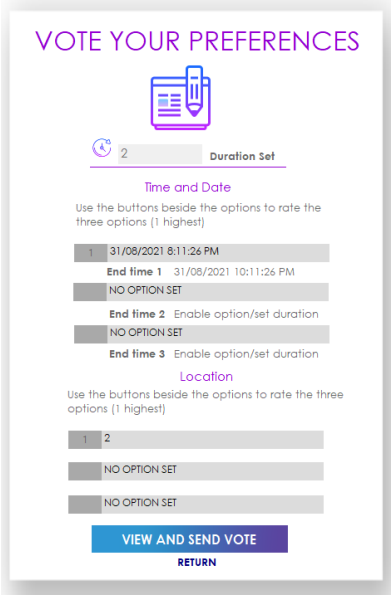
- Cannot enter ‘ character
- DESCRIPTION: Max 40 characters
- EVENT NAME: Max 20 characters
- LOCATION OPTIONS: Max 10 characters

Most prominently, the cost and duration information can only contain numbers and decimals to two places.

COST/DURATION

- COST: Max 20 characters
- DURATION: Max 6 characters
- No spaces
- Numbers only
- Two decimal places

INPUT	EXPECTED OUTPUT	OUTPUT
abcdefg	“”	“”
Abc123	123	123
$123	123	123
1 2 3	123	123
12.ab123	12.12	12.12
12335123123232122222222	12335123123232122222	12335123123232122222
123124124.122222	123124124.12	123124124.12



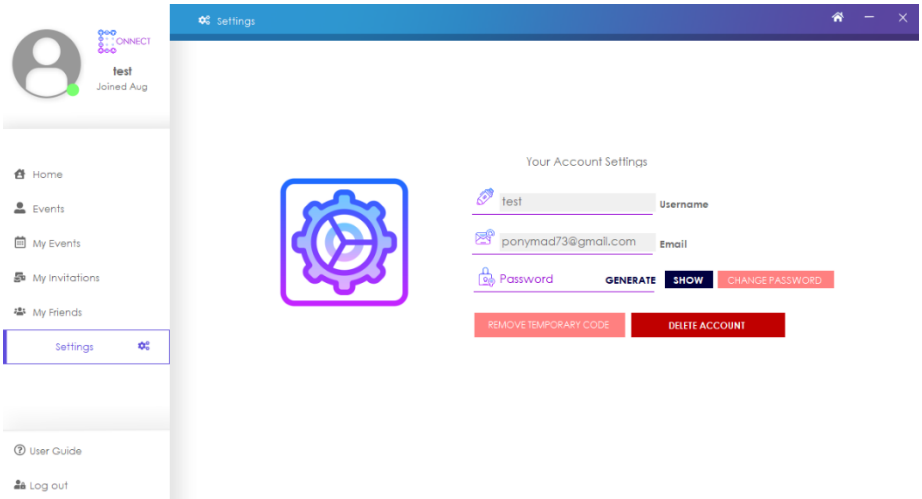
This form allows users to cast their preference on the options.

- No buttons can be the same priority
- Can have toggle options up to number of options

TIME AND DATE OPTIONS

INPUT	EXPECTED OUTPUT	OUTPUT
(3 options) Button 1: 3 Button 2: 3 Button 3: 3	Message Box Error: Cannot have two or more options at the same priority.	Message Box Error: Cannot have two or more options at the same priority.
Button 1: 3 Button 2: 2 Button 3: 3	Message Box Error: Cannot have two or more options at the same priority.	Message Box Error: Cannot have two or more options at the same priority.
Button 1: 3 Button 2: 2 Button 3: 1	Message Box Confirmation: Successfully accepted invitation and sent votes.	Message Box Confirmation: Successfully accepted invitation and sent votes.
(2 options) Button 1: 1 Button 2: 2 Button 3: 3	Impossible (button 3 should be blank/disabled)	Impossible (button 3 is blank/disabled)

FRM_SETTINGS



Allow users to adjust their account settings. Users can input a new password (this testing applies to the textbox used in login as well).

- 9-12 characters
- No spaces
- No ‘ character

INPUT	EXPECTED OUTPUT	OUTPUT
TestPassword	TestPassword Message Box Confirmation: Do you want to reset your password?	TestPassword Message Box Confirmation: Do you want to reset your password?
Abc123	Abc123 Message Box Error: Must be greater than 8 characters	Abc123 Message Box Error: Must be greater than 8 characters
$123 12!12’2	$12312!122 Message Box Confirmation: Do you want to reset your password?	$12312!122 Message Box Confirmation: Do you want to reset your password?
Abcesfsda’	Abcesfsda Message Box Confirmation: Do you want to reset your password?	Message Box Confirmation: Do you want to reset your password?
TestPass11111111	TestPass1111 Message Box Confirmation: Do you want to reset your password?	TestPass1111 Message Box Confirmation: Do you want to reset your password?

REPORT ON FINDINGS

Overall, my program successfully handles majority of the testing cases (especially assuming the user is following the user guide) through the self-validating elements of buttons, date-time pickers, and radio buttons. The main issues were ‘ character inputs which caused an error with the MySQL string as it is recognised as the end of a parameter. However, these illegal inputs were prevented from input.

One relevant finding, though, is that scaling up would be inefficient with the current use of strings as IDs for events and users. This is because searching (for a distinct username, for example) checks against other strings, which would take an immense amount of time with a lot of records. Due to this search occurring at a database level within the mySQL queries (SELECT * FROM x WHERE condition), changing programming languages and implementing string-matching algorithms would not increase its speed.

Although tedious to alter the current code, this could be overcome by using an autoincrementing ID to identify each record (thus search with numerical values). However, a search of strings would still be required to check if usernames and event names are unique. Thus, mySQL optimisation techniques could be combined with these numerical IDs, such as specifying the exact columns to search rather than *. Alternatively, the database could incorporate in-memory caching using Redis or Memcached, which is compatible with Amazon services and C#. The working principle for these efficient database structures is that if 3000 people searched for the profile each month, the first person would retrieve the information from the database (disk) and the remaining 2999 would retrieve from cache (memory).

BENCHMARK TESTING AND QUALITY ASSURANCE

CRITERIA	ANALYSIS OF END SOFTWARE
Smooth/Quick-Responding Interface Manually test responds to buttons within 1 second	The software meets this requirement aside from an approximately 4 second delay when sending an email to the user (with their forgotten password). This is because the SMTP client must communicate with external services. To mitigate this, the wait cursor and a warning in the confirmation messageBox visually indicates the interface is processing. Another consideration is if the number of users and events increase, search functions may take longer. However, considering the scope of this project, the program accurately fulfills this criteria.
Communicates smoothly with server Manually test server-related interactions complete within 1 second.	As addressed before, within the scope of this project, load testing with numerous events and users was not necessary. Thus, CONNECT successfully communicates with the cloud server within 1 second to meet this specification. However, as more records and users accumulate, this criteria may not be met because there are more records to search through. Instead, it may be more effective to identify events and users by their ID's (as opposed to string of their names).
Organise events in objective matter Software calculates priority and uses randomNumbers to create events.	The end application successfully calculates the priority of time and event options with frm_invitationVote. Further, it incorporates the random number class if numerous options receive the same number of votes, allowing the user to randomise which radio button is selected.
User-friendly Distribute 10+ program prototypes with a survey and receive positive/satisfied feedback. Interface uses consistent buttons and messageboxes.	Due to the pandemic and the security risk of the database connection string, testing with 10+ program prototypes was unachievable. However, from the 5 versions which were distributed, positive feedback was received, suggesting minor adjustments which were immediately incorporated. This is especially achieved through incorporating the user manual.

	The interface successfully uses a consistent colour scheme, the same Flat button style, and the Windows Message Box to fulfill this criteria.
Robust Test each input with a variety of illegal data. Use checkboxes/comboboxes where possible Distribute 10+ program prototypes and ensure no errors occur.	Overall, the end program was sufficiently robust and fulfilled this criteria. The above test data tables illuminate how the application employs message boxes and handling events to prevent the input of most illegal data. Further, SQL attacks are significantly deterred by the inability to copy and paste, and limit of 40 characters in any input box. Throughout, the program successfully uses data-validating checkboxes, combo boxes and radio buttons to enhance robustness. While the program prototype could not be distributed to 10 users, the minor errors identified by the feedback was immediately resolved, and overall, the development process of the program meant that there were minor opportunities to enter dangerous information. However, one reduction of the end product's robustness is the MySQL connection string which could potentially result in a security breach with malicious intent.
Customisable Distribute 10 program prototypes with a survey and receive positive/satisfied feedback. 4+ different settings.	As mentioned, regarding the user interface requirement, distributing 10+ program prototypes was unachievable. However, from the 5 prototypes that were tested, the feedback was satisfied with the UI irrespective of customisability. Thus, although the program does not incorporate 4+ different settings to achieve this quality assurance criteria, it was found as unnecessary.
Functions on Windows OS devices Test program works on range of 5+ windowsOS systems (with internet).	The end application achieves this specification through successful distribution on 5 different hardware and software configurations running WindowsOS and with internet connection.
Connects users anywhere Information is stored and accessed in a Cloud-server using the Internet. Test program works on range of 5+ systems (with internet).	Through the cloud MySQL database and CRUD manipulation, this feature was successfully achieved. The varying setup configuration and internet of the 5 BETA testers did not affect the performance of CONNECT.