Test Cases

Project: Vox Realtime Groupchat App

1. Test Title: Create Account

Executed by:

SRS Section Id: 3.1

- Description: To login into the application, a user must be logged in. To do that, they must have an existing account by creating one.
- Preconditions: None
- Dependencies: If valid, account information gets saved in to the database.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1.1	Click on register button		Redirected to register form page			
1.2	Enter your account information	Any alphabets or digits				
1.3	Click on register button		Redirected to the homepage	Account is registered		
2.1	Click the register button		Redirected to register form page			
2.2	Enter invalid account information	Symbols such as . or !				

1

Step #	Test Step	Data	Expected	Actual	Status	Notes
2.3	Click on register button		An error message pops up indicating the user to enter valid information			

2. Test Title: User Login

Executed by:

SRS Section Id: 3.2

- Description: To access the application features, a must be log in.
- Preconditions: An account must exist
- Dependencies: Verify the log in info with the one in database. On success, user gets access to the application features. Otherwise, a user must create an account to use the application.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1.1	Click on login button		Redirected to the login form page			
1.2	Enter your account information	Information that was used to register the account				

Step #	Test Step	Data	Expected	Actual	Status	Notes
1.3	Click on login button		Redirected to the home-page and account is regis-tered			
2.1	Click the login button		Redirected to login form page			
2.2	Enter invalid account infor-mation	Account that has not been regis-tered yet				
2.3	Click on login button		An error mes-sage pops up saying that the user-name and pass-word do not match			

3. Test Title: Create Vox Server

Executed by:

SRS Section Id: 3.3

- Description: To interact with another user, they both need to be in the same server. On way to do that is to create a vox server.

- Preconditions: User must be logged in.
- Dependencies: Database for holding server entries.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1.1	In the text box above the create button, enter your server name	Any character				
1.2	Click on create server		A Popup appears indicating that server is created and server created			

4. Test Title: Chat on Servers

Executed by:

SRS Section Id: 3.4

- Description: If you need to interact with other users, you can do that by messaging them in the chat area.
- Preconditions: Must be/have an server
- Dependencies: Database for holding server entries.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Login in with a valid account		Moved to the home-page			
2	Click on an existing server		Moved to that server page			
3	Click on the textbox	Enter any text	Message is sent and appears in the box above			
4	Ask your friend in the same server to send you a message		Message appears on the box above			

5. Test Title: View/Edit Notebook

Executed by:

SRS Section Id: 3.5.1

- Description: A user may create notebooks for each server they are in. The contents of the notebook are saved by the user manually by clicking the save notebook button.
- Preconditions: The user must have an account and be part of a server.
- Dependencies: Database for holding notebook entries.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Open note-book in server	User creden-tials and DB	The user should recieve their notes from the db if any			
2	Edit note-book and make changes if needed					This step is op-tional since they may not have changes to make
3	Save note-book	Notebook entry will be sent to DB				POST on intitial save, UP-DATE on saves after the intial

6. Test Title: View Shared Notebook

Executed by:

SRS Section Id: 3.5.1

- Description: A user of a server may share their notebook with other users of the same server. When a notebook is shared it's title is visible to all users in the server, and users can open the notebook in a readonly mode.
- Preconditions: A user must be part of a server and another user must have shared their notebook

- Dependencies: Database for holding notebook entries

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Open shared notebook in server	User credentials and DB	The user should recieve a copy of another person's notes			This may be outdated if the other user is updating their notes, the shared notebook won't be updated until the person who shared the notebook saves it again.
2	View notes					READ ONLY on other user's notes

Step #	Test Step	Data	Expected	Actual	Status	Notes
3	Close notes/switch to other notes					A user can switch to another notebook, or click clear to start creating their own

7. Test Title: Share Notebook with Server

Executed by:

SRS Section Id: 3.5.2

- Description: A user of a server may share their notebook with other users of the same server.
- Preconditions: A user must be part of a server
- Dependencies: Database for holding notebook entries

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Open notebook in server	User credentials and DB	To be able to see their notebook			
2	Enable sharing		User clicks option to enable sharing on notebook			

Step #	Test Step	Data	Expected	Actual	Status	Notes
3	Update the contents and save		The shared notebook should be updated for other users.			

7. Test Title: Create/save Notebook

Executed by:

SRS Section Id: 3.5.2

- Description: A user of a server may create a notebook and save it.
- Preconditions: A user must be part of a server
- Dependencies: Database for holding notebook entries

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Open notepad		Notepad opens and the user can start typing notes			
2	Save notepad	User, Server, Name, Contents	The notepad is saved into the database and pops up in your saved notebooks tab			

9

Step #	Test Step	Data	Expected	Actual	Status	Notes
3	Update the contents and save		The notebook is updated with the newly saved contents			

8. Test Title: Create Event

Executed by:

SRS Section Id: 3.6.1

- Description: If you instructor gives you an important due date for a project. You would probably want to mark that in a calendar.
- Preconditions: Must have an existing account
- Dependencies: Database for holding calendar events

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Login in with a valid account		Moved to the homepage			
2	Hover over the day when your assignment is due		Cursor is on that day in the Calendar			
3	Click on add event button	Enter a description	Event gets saved in the calendar			

9. Test Title: Remove Event

Executed by:

SRS Section Id: 3.6.2

- Description: If your assignments due date has passed and you don't want that event to exist in your calendar. It can be removed from the calendar.
- Preconditions: Must have an existing account and if the event is for a server then they must be the owner
- Dependencies: Database for holding the calendar events

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Login in with a valid account		Moved to the home-page			
2	Click `Remove Personal Event` on home-page		Can see the existing per-sonal events			
3	Click on event you want to delete		Event is deleted from the cal-endar, changes must be re-freshed			

10. Test Title: Remove Users

Executed by:

SRS Section Id: 3.7

- Description: There should be the ability to remove a user from a server
- Preconditions: Have users in a server other than the owner and be the owner of the server
- Dependencies: Database containing the server information

11

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Go to server	User credentials and DB	To see the users of a server			
2	Click Remove Users in hamburger menu		See existing members of the server except for the owner			
3	Click user to remove from server		The user will no longer be able to chat in the server			

11. Test Title: Delete Server

Executed by:

SRS Section Id: 3.8

- Description: If the server you created is no longer needed then you can delete it.
- Preconditions: Server must exist already and user must be the owner
- Dependencies: Database for holding the server information and it's users information

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Login into your account		Moved to the homepage			

Step #	Test Step	Data	Expected	Actual	Status	Notes
2	Click on a server that you created		Moved to that server page			
3	Click on the delete button in hamburger menu	Confirmation for deletion	Server is deleted			

12. Test Title: Invite User to server

Executed by:

SRS Section Id: 3.9

- Description: If you want to invite your friend to your gaming server, and talk about new games. This can simple be done by sharing your invite code.
- Preconditions: A user can only invite if they are the server owner
- Dependencies: Database for holding the server information and your friends information.

Step #	Test Step	Data	Expected	Actual	Status	Notes
1	Login into your account		Moved to the homepage			
2	Click on a server that you created		Moved to that server page			

Step #	Test Step	Data	Expected	Actual	Status	Notes
3	Click on invite button	The invite code	User has code copied to clipboard			
4	Send Invite Code to friend		Friend can enter code and join your server			

Non functional Requirements ### 13. SSL Certificate - This will be tested by opening the webpage using HTTPS, if this doesn't work then we know we do not have a SSL certificate for the webserver.

14. Response Time of < 500ms for Chat Messages

- This will be tested by running the webapp on two computers and checking the time it is sent to the time it is shown on the other computer. This is to check we are efficently dealing with database inserts and that websockets are functioning.