Verification & Validation: Neesh

SF-60: Amir Barkam, Trevor Flanigan, Justin Hua, Yisheng Liu, Arnold Ying

Table of Contents

1. Overview

The project is currently in the early development phase with few features fully implemented. The following verification and validation are planned to be conducted during the full implementation of each of their corresponding requirements.

The majority of verification conducted so far has been manual tests done by the development team.

The majority of validation conducted so far has been informal sprint meetings between the client and the team to receive ongoing feedback for the working prototype.

2. Test Descriptions

2.1 Methodology

For the purpose of verification and validation of the requirements of this project, the team has decided upon certain methodologies of testing which will be referenced and used in the table in section 2.2 and 2.3.

2.1.1 Unit Testing

Unit tests will be adopted and developed for each individual component file[1] in the frontend repository. Unit tests will be implemented using the Jest[2] testing framework. These unit tests will include at least the following test cases, if applicable:

1. Happy path[3]
2. Malformed inputs (i.e. Null, Missing, Invalid type)
3. Storybook[4] UI Test

[1] Component files are stored under the */neesh/src/components* or the */neesh/src/screens* folder of the source code repository. For further detail, see section 4. Access to Data
[2] Jest is a JavaScript Testing Framework: https://jestjs.io/
[3] Happy path means that the component file is passed a known input and produces an expected output
[4] Storybook is a frontend library for building and testing UI components and pages in isolation: https://storybook.js.org

2.2 Verification

2.2.1 Functional Requirements

FR#	Test Descriptions	Approval Criteria
FR1	1. Unit tests will be implemented for the following component files: SignUp.tsx, CreateNewPassword.tsx, ConfirmSignUp.tsx. 2. Manual tests will be conducted where the user creates a new account through the sign-up given a valid phone number and valid password. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted for Sign-up in the same method as manual tests.	1. The user is able to sign up with a phone number and be able to see the feed page. 2. All unit tests pass. 3. Client approval during design review.
FR2	1. Unit tests will be implemented for the individual component files relating to the Onboarding feature (not implemented yet). 2. Manual tests will be conducted where the user goes through the Onboarding process and properly sets their user attributes: pronouns, avatar, name, color, topics of interest. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted for Onboarding in the same method as manual tests.	1. The user is able to finish the onboarding flow and register a user with the selected attributes in our backend. 2. All unit tests pass. 3. Client approval during design review.
FR3	1. Unit tests will be implemented for the following component file: SignIn.tsx.	1. The user is able to sign in with a

	2. Manual tests will be conducted where the user signs in with a pre-existing account through the sign-in page. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	previously registered phone number and password and access their feed page. 2. All unit tests pass. 3. Client approval during design review.
FR4	1. Unit Tests will be implemented for the following component file: CreateConvo.tsx. 2. Manual tests will be conducted where the user goes through the process of creating an Inactive Convo (IC)[5] with their own desired attributes of the IC: title, text, related topics, sensitivity flag. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to create an IC visible to other users with custom attributes defined by the user. 2. All unit tests pass. 3. Client approval during design review.
FR5	1. Unit tests will be implemented for the individual component files relating to the edit IC feature (not implemented yet). 2. Manual tests will be conducted where the user goes through the process of editing a pre-existing Inactive Convo (IC) with their own desired attributes of the IC: title, text, related topics, sensitivity flag. Manual tests will also	1. The user is able to edit a pre-existing IC visible to other users with custom attributes defined by the user. 2. All unit tests pass. 3. Client approval during design review.

[5] Inactive Convo definitions can be found in the Requirements document

	verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	
FR6	1. Unit tests will be implemented for the individual component files relating to the delete IC feature (not implemented yet). 2. Manual tests will be conducted where the user goes through the process of deleting a pre-existing Inactive Convo (IC). Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to delete a pre-existing IC visible to other users with custom attributes defined by the user. 2. All unit tests pass. 3. Client approval during design review.
FR7	1. Unit tests will be implemented for the individual component files relating to the Profile Page feature (not implemented yet). 2. Manual tests will be conducted where the user accesses the Profile Page and is shown the topics the user follows, the user's posted ICs, and the user's saved ICs. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to access the Profile Page and the topics they follow, their posted ICs, and their saved ICs. 2. All unit tests pass. 3. Client approval during design review.
FR8	1. Unit tests will be implemented for the	1. The user is able to

	individual component files relating to the Profile Page feature (not implemented yet). 2. Manual tests will be conducted where the user accesses another user’s Profile Page and is shown their mutually followed topics and the other user’s posted ICs. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	access another user’s Profile Page and view the common topics they follow and their posted ICs. 2. All unit tests pass. 3. Client approval during design review.
FR9	1. Unit tests will be implemented for the individual component files relating to the Settings feature (not implemented yet). 2. Manual tests will be conducted where the user accesses their Settings Page and is shown two buttons to either delete or log out of their account. Manual tests will also verify the user interface is being displayed as intended and account deletion is reflected in the backend. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to access their Settings Page and have the option to log out of or delete their account. 2. Deleted accounts no longer exist in the database. 3. All unit tests pass. 4. Client approval during design review.
FR10	1. Unit tests will be implemented for the individual component files relating to the Onboarding feature (not implemented yet). 2. Manual tests will be conducted where the user accesses the Onboarding page and has the option of setting their preferred pronouns,	1. The user is able to access the Onboarding Page after account creation and have the option to set their preferred

	topics, username and avatar. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	pronouns, topics, username and avatar. 2. All unit tests pass. 3. Client approval during design review.
FR11	1. Unit tests will be implemented for the individual component files relating to the Report feature (not implemented yet). 2. Manual tests will be conducted where the user accesses any IC and has the option to report an individual IC, comment, or comment reply. Manual tests will also verify that a report is shown on the backend and is displayed in the administrator dashboard and that the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to access an IC and have the option to report an IC, comment, or comment reply. 2. Each report is displayed in the administrator dashboard. 3. All unit tests pass. 4. Client approval during design review.
FR12	1. Unit tests will be implemented for the individual component files relating to the Feed Page (in progress). 2. Manual tests will be conducted where the user accesses the Feed Page and are able to view a list of ICs based on their followed topics. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual	1. The user is able to access the Feed Page and view a list of relevant ICs. 2. All unit tests pass. 3. Client approval during design review.

	tests.	
FR13	1. Unit tests will be implemented for the individual component files relating to the Convo feature (in progress). 2. Manual tests will be conducted where the user accesses a single Convo Page and is shown the content of the IC, including title, text, image, and tagged topics. The user must also be able to see a feed of all the comments and comment replies on the IC as well as the number of likes. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to access an individual IC to view the content of the IC as well as comment, comment replies and like count. 2. All unit tests pass. 3. Client approval during design review.
FR14	1. Unit tests will be implemented for the individual component files relating to the Convo feature (in progress). 2. Manual tests will be conducted where the user accesses a single Convo Page and has the option to like, save, and comment on the IC. Manual tests will also verify the user interface is being displayed as intended. 3. Design reviews with the client will be conducted in the same method as manual tests.	1. The user is able to access an individual IC to like, save, and comment on the IC. 2. All unit tests pass. 3. Client approval during design review.
FR15	1. Unit tests will be implemented with Jest and run automatically in our CI pipeline to test	1. Upon tapping a comment's "like"

	tapping a comment's "like" button. In addition, tests will be run to attempt to reply to a user's comment, which should then appear as a reply. 2. A walkthrough will be conducted during client design reviews to ensure feature and UI element correctness	button, the comment's state is updated to appear as "liked". 2. After a user replies to a comment, the reply should appear. 3. Client approval during design review. 4. All other unit tests pass.
FR16	1. Unit tests will be implemented with Jest and run automatically in our CI pipeline to simulate a user selecting a topic from the Explore page. 2. A walkthrough will be conducted for the Explore feature during the client design review to ensure feature and UI element correctness.	1. When selecting a topic, feeds representing popular and new ICs containing that topic should be displayed. 2. Client approval during design review. 3. All other unit tests pass.
FR17	3. Unit tests will be implemented with Jest and run automatically in our CI pipeline to simulate creating a new IC, which will then attach a photo to the post and create the post. 4. A walkthrough will be conducted for attaching images to ICs during the client design review to ensure feature and UI element correctness.	1. All unit tests pass. 2. Client approval during design review.
FR18	1. For each kind of action that a user makes that can generate a notification, manual tests will be conducted to ensure that when these	1. Each action generating a notification

	actions are invoked, a notification of the activity is sent to the appropriate user's device. 2. A walkthrough will be conducted for the notifications feature during the client design review to ensure feature and UI element correctness. Because this requirement is still being defined by the client, the actions that can generate a notification are yet to be determined.	successfully sends a notification to the appropriate user's device with the correct text. 2. Client approval during design review. 3. Unit tests pass
FR19	1. Unit tests will be implemented for Activity.tsx 2. Manual tests will be conducted to generate a notification as described in FR18, and then subsequently check to ensure that the notification appears at the top of the Activity page, with all previously sent notifications appearing below. 3. A walkthrough will be conducted for the Activity feature during the client design review to ensure feature and UI element correctness.	1. The notification appears at the top of the Activity page. 2. Previously sent notifications appear below the new notification on the Activity page in chronological order. 3. Client approval during design review. 4. Unit tests pass.
FR20	1. Using an IC flagged for testing, manual tests will be conducted to delete the test conversation from the Admin panel UI. The test should then attempt to view the test IC from within the mobile app. 2. A walkthrough will be conducted for removing	1. The test conversation is successfully removed from the database. 2. The test conversation is no longer visible from the mobile application.

	ICs from the Admin panel during the client design review to ensure feature and UI element correctness.	3. Client approval during design review. 4. Unit tests pass.
FR21	1. Manual tests will be conducted by reporting user generated content including ICs and comments flagged for testing from the mobile app. The Admin panel will then be checked to ensure that the content appears in a list of reported content. 2. A walkthrough will be conducted for viewing reported user content from the Admin panel during the client design review to ensure feature and UI element correctness.	1. The reported user content appears in the list of reported content. 2. Client approval during design review. 3. Unit tests pass.
FR22	1. Manual tests will be conducted by reporting users flagged for testing from the mobile app. Then, from the Admin panel, the test will attempt to ban the reported user. The test should then attempt to use the app as the banned user. Additionally, from a different user logged into the app, the test should attempt to view the banned user's profile. Finally, the test should reactivate the banned user. It will then attempt to use the app as the originally banned user, and outside users can view the user. 2. A walkthrough will be conducted for banning and reactivating users during the client design review to ensure feature and UI element correctness.	1. After banning, the banned user should not be able to post new content from the app. 2. Other users should not be able to view the banned user's profile. 3. After reactivating, the banned user should be able to post new content from the app. 4. After reactivating, other users should be able to view the previously banned user's profile. 5. Client approval during

		design review. 6. Unit tests pass.

2.2.2 Non-Functional Requirements

NF#	Test Description	Approval Criteria
NF1	Unit Tests will be implemented for the following GraphQL models: Comment, CommentReply, Convo, Like, Notification, Report, Topic, User. For each model, the Get*, List* (paginated), Create*, Update*, and Delete* queries / mutations will be performed using Jest and Amplify Mock.	1. For each query, the expected result is retrieved (queries), or represented in the database (mutations). 2. Each query must take less than 2 seconds to complete. 3. The frontend application must still respond to user interactions during any GraphQL query.
NF2	To verify that our data is encrypted at rest, we must ensure that the services we use which store PII encrypt data at rest, via manual review. Currently, the only PII required from users is a phone number, which is stored in Amazon Cognito. Cognito encrypts data at rest by default.	1. All AWS Services which store user data must specify that they encrypt data at rest. 2. Calls to backend services must be done over an HTTPS connection.
NF3	For each UI Component (which is defined by designers in Figma), a Storybook UI test will be	1. Every Figma Component has a

	created. The Storybook test will act as a visual and functional verification aid. Storybook will render a sample view of each component, exposing controls for changing each of its input properties. This way, every possible input in the component's property-space can be specified and will be proven to be completely independent of other modules.	React representation. 2. Each react component has a standalone representation in Storybook.
NF4	Manual tests will be conducted by installing the app on iPhone models from iPhone 6S to iPhone 14 using the simulator embedded in the Xcode. The tests will be conducted on different iOS versions from iOS 14 to iOS 16.	1. The compiled app can be installed on the target device running the target iOS version. 2. For different screen resolutions: The GUI elements on the screen are not out of the screen boundary and the elements are in the right place as proposed in the Figma prototype. 3. For different iOS versions: The app response to each action conducted by users (click, input, etc.) are correctly implemented with no

		errors.
NF5	Manual testing will be performed by checking new added dependencies during each code review. Code reviewers will check that the newly added dependencies are open source. The dependency packages are labeled in the file “neesh/package.json”	1. All dependencies are labeled in the file “neesh/package.json”. 2. All dependencies’ source code are open source and available to be checked.
NF6	Manual testing will be performed during code reviews to ensure all newly added code is modular and contains proper documentation in the form of comments in the code and documentation within a README file. A linter will also be run on the codebase during each commit to ensure the code complies with set style guidelines.	1. All new code is modular. 2. Appropriate comments and documentation have been made. 3. Detailed README documents are presented for frontend and backend code, including setup and troubleshooting. 4. The linter passes on all new source code.
NF7	Manual testing will be performed during code reviews to ensure that the app is not asking users for any PII that may be presented within the app to other users.	1. The app does not ask users for PII to show to other users.

2.2 Validation

Upon discussion with the client, it has been determined that validation of the application will be done through design reviews and testing sessions. These sessions will take place at various points throughout the development cycle and will give the designers a chance to use the application for evaluating if the requirements meet their needs and the needs of future users.

The first design review session is tentatively scheduled to be on December 5th, 2022 as a zoom meeting with the client with a walkthrough of Storybook tests and each manual test described in the FR & NFR tables in 2.2.1 and 2.2.2.

3. Presentation & Interpretation of Information

As mentioned in section 1 there is very limited testing right now due to the current phase of implementation. We currently have sample unit tests on parts of the components for **FR16**, specifically for the Topic react component in our frontend. These unit tests are implemented through Storybook, and can be referred to in Appendix C. Additionally, these tests consist of interactions, which are visible on the bottom panel, with each interaction having a pass or fail state when the test is run. This state is indicated by the checkmark to the left of the interaction command. A passed test is indicated by all interactions passing.

Currently, we do not have a large amount of test results available, therefore interpretation of this information will be reserved for future milestones when more data is available.

4. Access to Data

For automated tests (i.e. unit tests with Jest and Storybook), the results of those tests are stored in our GitHub repository: https://github.com/Neesh-App/Neesh-App.

1. For unit testing with Jest, the tests and test results will be stored under the */test/* file path
2. For unit testing with Storybook, the tests and test result will be stored under the */stories/* file path

Appendices

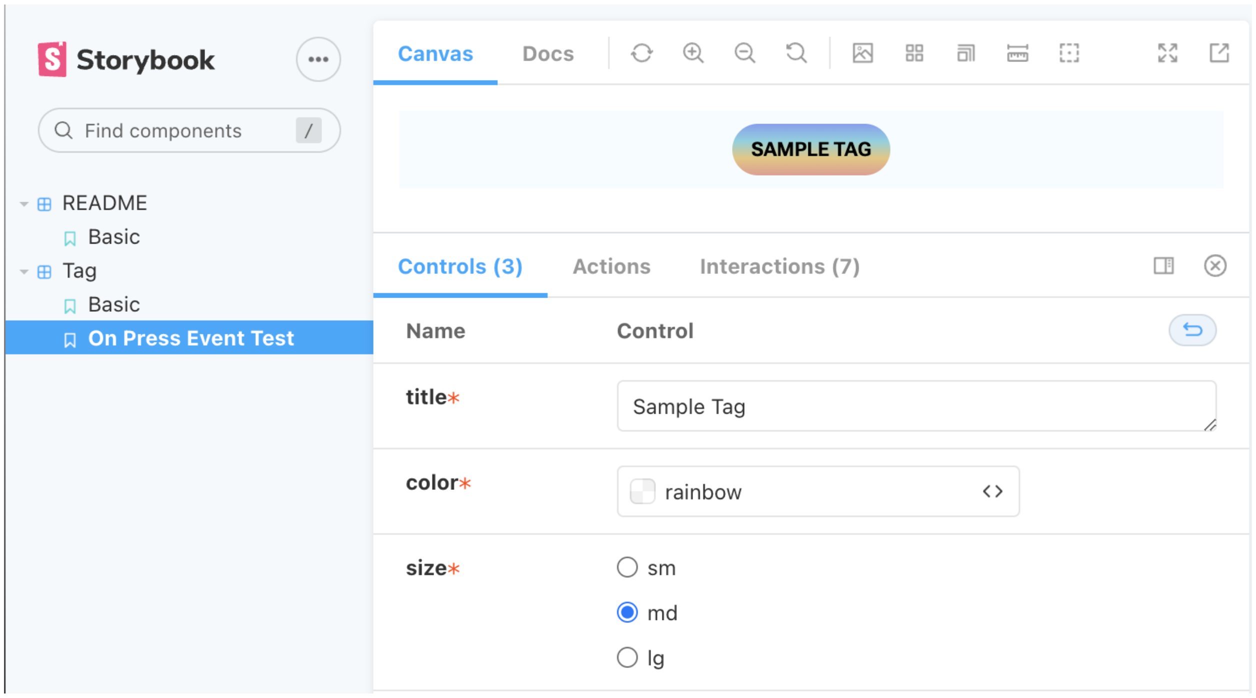
Appendix A - Team Information

Name	**Initials**	**Tech Lead**	**Management Lead**
Amir Barkam	AB	Frontend	
Trevor Flanigan	TF		Scrum Master
Yisheng Liu	YL	AWS / Cloud	
Arnold Ying	AY		Product Manager
Justin Hua	JH	Testing	

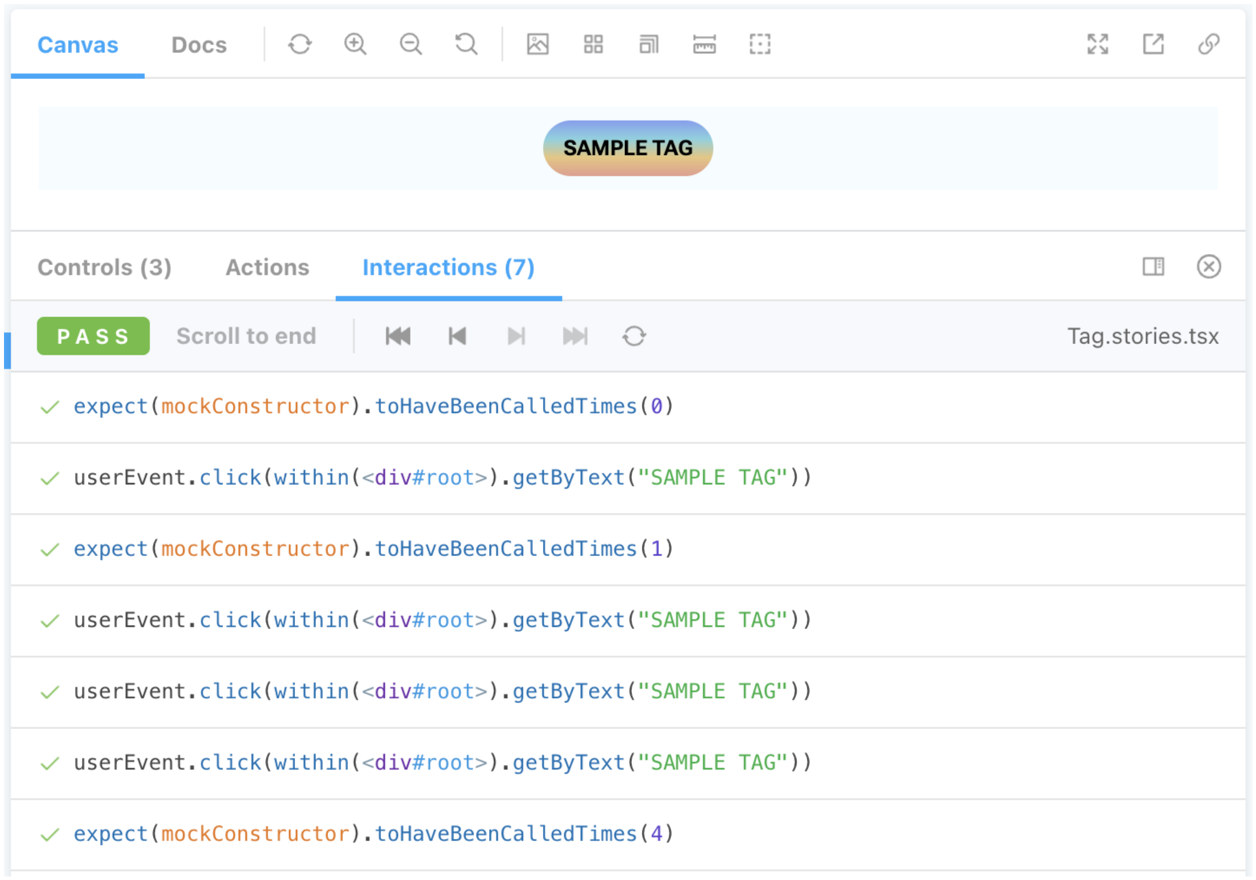
Appendix B - Project Report Contributions

Section	Major Content	Minor Content	Author	Reviewer
1. Overview	AY		AY	JH, TF
2.1 Verification	AY	TF, AB, YL, JH	TF	JH, TF, YL
2.2 Validation	AB	AY	AB	JH, TF
3. Presentation & Interpretation of Information	AY	JH	AY	JH, TF
4. Access to Data		AY		JH, TF

Appendix C - Storybook UI Test Implementation



Name	Control
title*	Sample Tag
color*	rainbow
size*	sm [✓ md] lg



```
✓ expect(mockConstructor).toHaveBeenCalledTimes(0)
✓ userEvent.click(within(<div#root>).getByText("SAMPLE TAG"))
✓ expect(mockConstructor).toHaveBeenCalledTimes(1)
✓ userEvent.click(within(<div#root>).getByText("SAMPLE TAG"))
✓ userEvent.click(within(<div#root>).getByText("SAMPLE TAG"))
✓ userEvent.click(within(<div#root>).getByText("SAMPLE TAG"))
✓ expect(mockConstructor).toHaveBeenCalledTimes(4)
```

Changelog

Date	Version	Description	Author(s)
Nov. 27th, 2022	1.2	Fill out sections 2-4	Arnold Ying, Trevor Flanigan, Justin Hua, Amir Barkam, Yisheng Liu
Oct. 16th, 2022	1.1	Fill out document skeleton	Arnold Ying, Trevor Flanigan, Justin Hua, Amir Barkam, Yisheng Liu
Oct. 6th, 2022	1.0	Created document	Arnold Ying