


Creative Ideation

Software Test Plan

By: Kellen Evoy, Matthew Marini, Vanessa Li

Table of Contents

1. Introduction.....	2
2. Testing Process.....	2
2.1. Overview of Testing Components and Methodologies.....	2
2.2. Test Results and Deficiencies.....	2
3. Testing Environment and Configuration.....	2
3.1. Software Environment.....	2
3.2. Hardware Environment.....	2
3.3. Test Configurations.....	2
4. Component Testing.....	3
Session Item View Model.....	3
Session View Model.....	3
Group View Model.....	3
5. System Testing.....	4
5.1. User and Group Management Test Plan Diagram.....	4
5.2. Manage Profile Test Suite Diagram.....	4
 5.3. Test Splitting Members.....	5
5.4. Session Management Test Plan Diagram.....	5
5.5. Manage Sessions Test Suite Diagram.....	5
5.6. Guide Sessions Test Suite Diagram.....	7
5.7. Artificial Assistance Test Plan Diagram.....	8
5.8. Generate Suggestions Test Suite Diagram.....	8
6. Resources and Schedule.....	9
Allocation of Testing Responsibilities.....	9

1. Introduction

This document will outline the software test plan for Creative ideation. The test plan will define the verification process, testing activities and test-cases used within this project. The test plan aims to help outline the quality of our software solutions and its suitability in tackling a real-world problem,

2. Testing Process

The phases of the testing process include manual testing, unit testing, integration testing and automated UI testing by the XCT framework. Only local components separate from the firebase functionality were tested. Based on time constraints, we were not successful in implementing firebase testing in our app.

2.1. Overview of Testing Components and Methodologies

We organized our testing components based on our view models.

Components Tested

- Session Item View Model
- Session View Model
- User Account View Model
- Idea Generator

The primary methodology used to test the above components was unit testing. Once unit testing was done, we implemented UI testing, which simulated user clicking to ensure that the correct components existed and were functioning.

2.2. Test Results and Deficiencies

Test results were recorded using Jira's Test Plan board and were primarily based on trial and error. If a certain unit test failed, we would look back and reconfigure the unit test till we reached a successful run. Deficiencies were recorded in the Jira backlog and noted down as a test the team needed to revisit in the future.

3. Testing Environment and Configuration

3.1. Software Environment

The requirements needed to successfully execute integration, unit, manual and UI testing:

- iPad simulator using iOS 14+
- XCode IDE
- XCTest framework

3.2. Hardware Environment

The requirements needed to successfully test the application, for instance acceptance testing and system testing:

- Physical iPad Device using iOS 14+
- iPad must have a working camera

3.3. Test Configurations

Possible test configurations for both hardware and software:

- iPad (9th generation)
- iPad Air (4th Generation)
- iPad Pro (4th to 6th Generation)
- iPad mini (6th Generation)

All iPads must be able to run iOS 14 or 15.

4. Component Testing

Session Item View Model

Components:

- Test Updating Stickies
- Test Creating Stickies
- Test Updating Selected Sticky
- Test Sorting Stickies

Test-Case	Methodology	Short Description
Updating with profanity word	Unit Test	Test to check if the function is able to detect words that is within the profanity list array
Creating a new sticky	Unit test	Test to see if the function is able to create a new instance of a sticky
Sticky Note is Selected	Unit Test	Test to see if the function is able to determine which sticky note is selected
Sticky Note Alphabetical Ordering, colour	Unit Test	Test to see if the function is able to sort sticky notes in alphabetical, and colour.

Session View Model

Components:

- Test Best Ids

Test-Case	Methodology	Short Description
Best Sticky Note out of List	Unit Test	Test to make sure that the function is able to select the highest voted sticky note within an array
Test Top sticky	Unit Test	Test to see if function is able to output stickies that share the same amount of votes

Group View Model

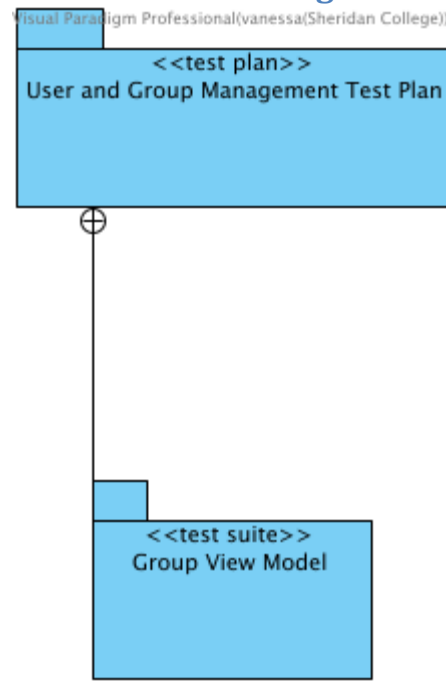
Components:

- Test Splitting Members

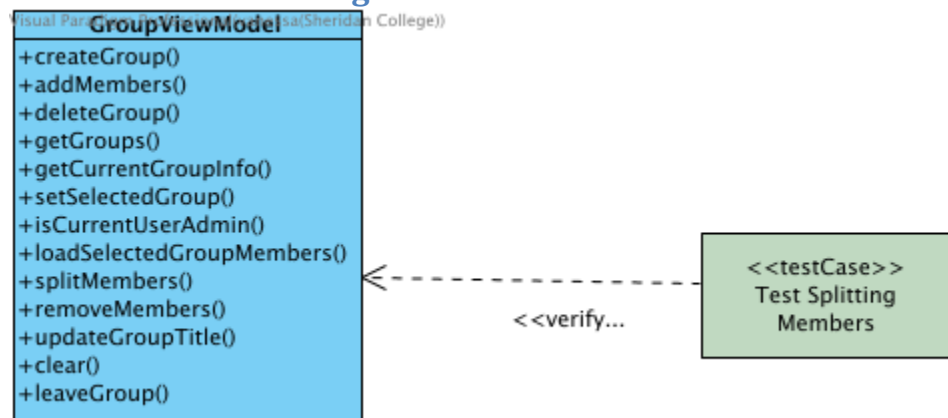
Test-Case	Methodology	Short description
Split Members	Unit	Test to see if the function is able to determine which members within a team are group members and non-members.

5. System Testing

5.1. User and Group Management Test Plan Diagram



5.2. Manage Profile Test Suite Diagram



Steps	Procedure	Expected Result
1. Login to application	Enter credentials and press login	You arrive at home page
2. Create Team with 3 members	Select plus button to create team, share access code with 2 other people.	Team is created with 3 members
3. Create Group with 2 members	Tap create group button, add one member.	Group is now with two members

4. Go to group settings

Tap and hold group icon, tap settings.

Group settings page appears

5. Go to edit member settings

Tap the "edit" button next to "edit members" prompt.

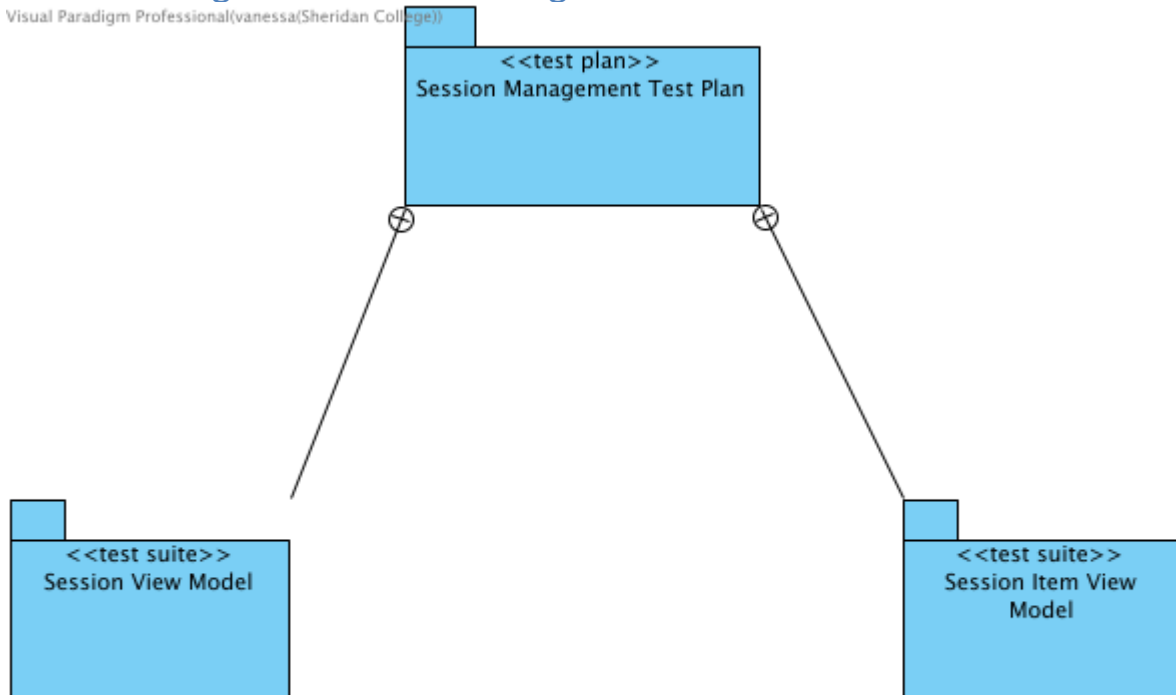
List of members should appear

5.3. Test Splitting Members

Testing to split the team members into group members and non group members. This is to show who has yet to be added to a group in the "edit members" settings

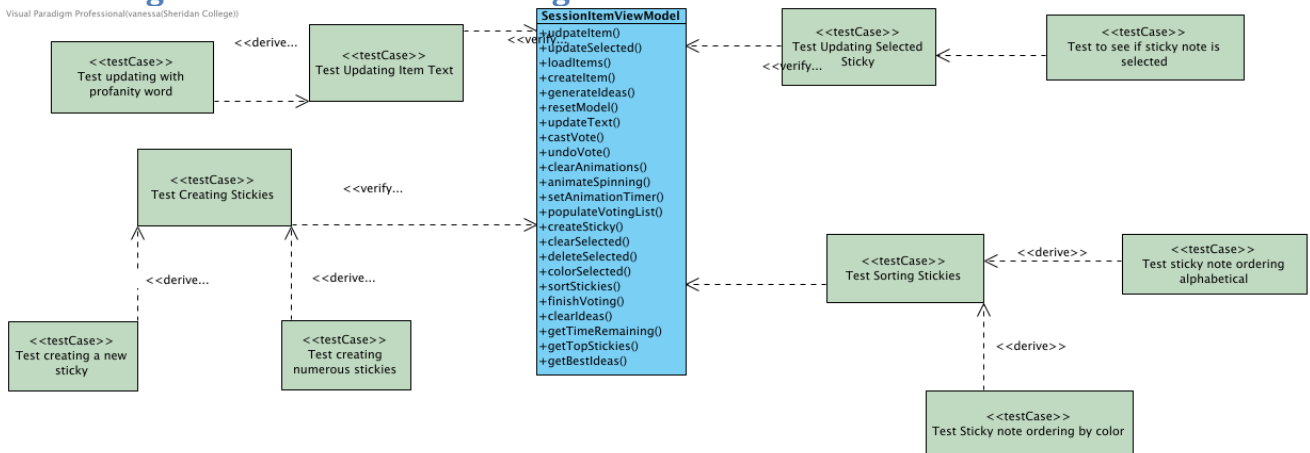
5.4. Session Management Test Plan Diagram

Visual Paradigm Professional(vanessa(Sheridan College))



5.5. Manage Sessions Test Suite Diagram

Visual Paradigm Professional(vanessa(Sheridan College))



Test creating stickies

Test Case	Procedure	Expected Result
1. Navigate to a Session	Log in, create or join a Session	User is now in a Sticky Note Session
2. Create a sticky note	Press the "+" button on the top right of the screen, hit the check mark to submit sticky to the Session	sticky note now appears on the board

Test Updating Item Text

Test Case	Procedure	Expected Result
1. Navigate to a Session	Log in, create or join a Session	User is now in a Sticky Note Session
2. Select a sticky note	Tap the text area of a sticky note to edit	Text field should appear on sticky note
3. Change sticky note text	Type in something else and click the checkmark to submit.	Sticky note text changed. Stars will appear if profanity involved.

Test Updating Selected Sticky

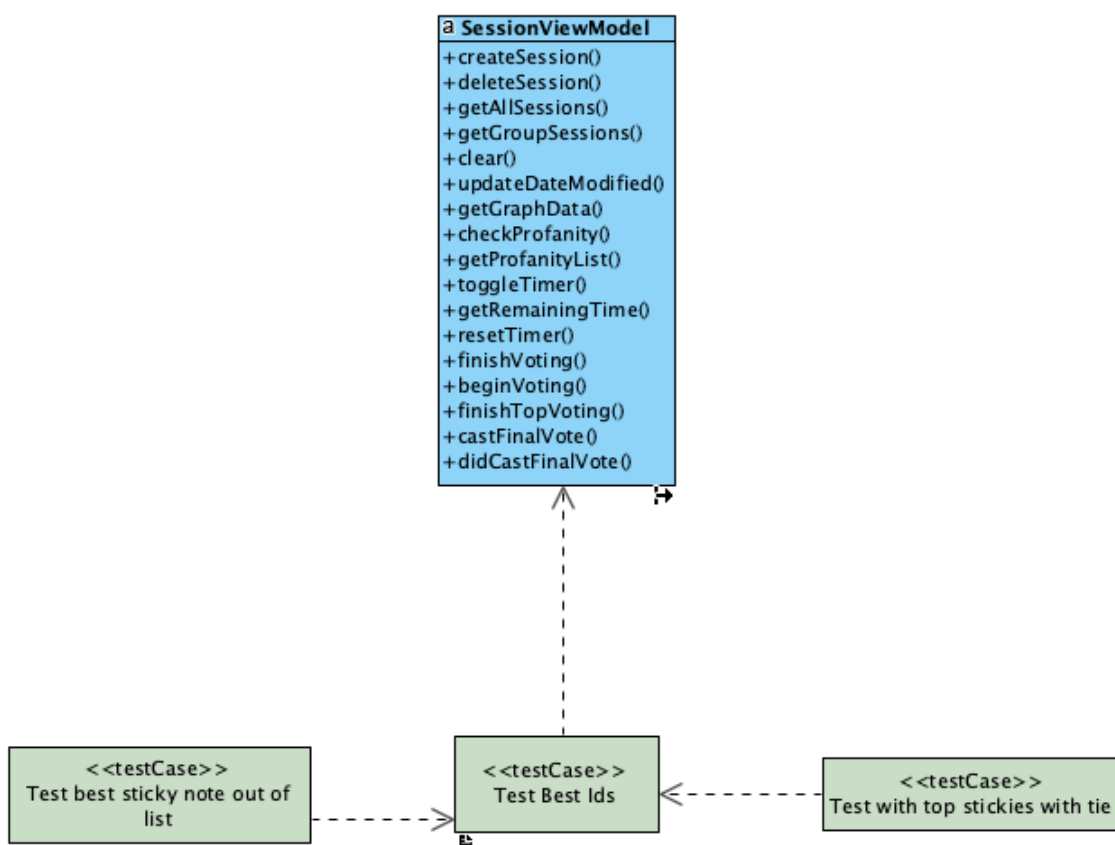
Test Case	Procedure	Expected Result
1. Navigate to a Session	Log in, create or join a Session	User is now in a Sticky Note Session
2. Select a Sticky Note	Hold and tap of the selected sticky	Sticky note should have a grey shadow.

Test Sorting Stickies

Test Case	Procedure	Expected Result
-----------	-----------	-----------------

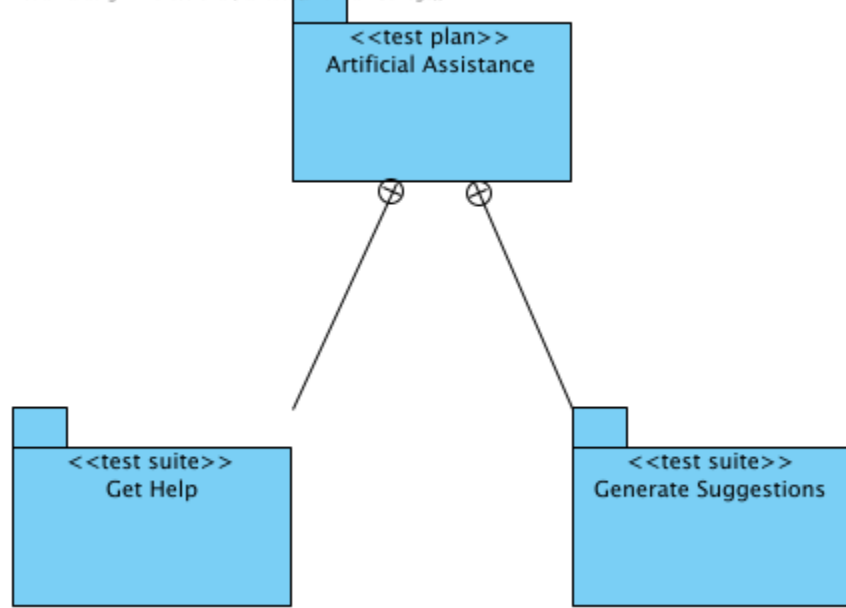
1. Navigate to a Session	Log in, create or join a Session	User is now in a Sticky Note Session
2. Sort Stickies by color (assume stickies are on screen)	Click on sorting button and press "Sort by Color"	Stickies will be ordered by roygbiv
3. Reverse color order	Repeat step 2.	Stickies will be ordered by vibgyor.
4. Sort Stickies by alphabet (assume 3 stickies with text "A", "B", "C")	Click on sorting button and press "Sort Alphabetically"	Sticky notes should be ordered "A", "B", "C"
5. Reverse alphabetic order	Repeat Step 2.	Stickies notes should be ordered by "C", "B" ,"A"

5.6. Guide Sessions Test Suite Diagram



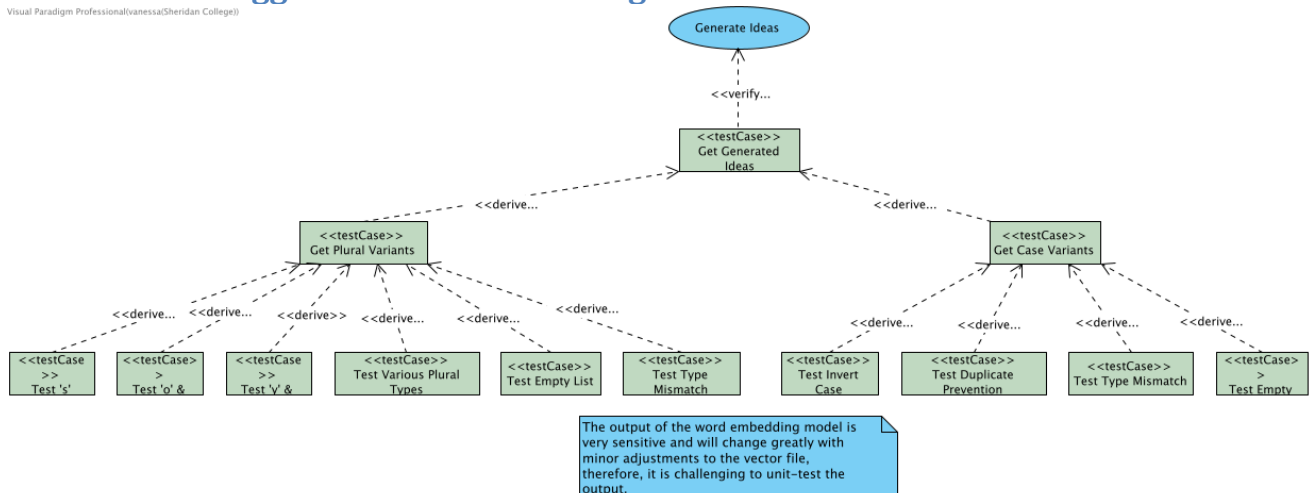
5.7. Artificial Assistance Test Plan Diagram

Visual Paradigm Professional(vanessa(Sheridan College))



5.8. Generate Suggestions Test Suite Diagram

Visual Paradigm Professional(vanessa(Sheridan College))



Get Generated Ideas

Test case

1. Within a Sticky Notes Session, create a few ideas on sticky notes related to a single theme.

Procedure

Tap the "Add Sticky Note" button and enter single words on each sticky note like "car", or "truck".

Expected Result

Numerous sticky notes are on screen with ideas within them relating to a single theme.

2. Get Generated Ideas.	Tap the light bulb icon in the right-hand sidebar.	A pop-out menu will appear with a carousel of generated words related to the theme (vehicles, in this example).
3. Cycle through ideas.	Tap the left or right chevron icon to cycle through the pop-out menu of ideas.	The menu should cycle through the generated list of ideas endlessly.
4. Submit a generated idea onto the screen.	Tap one of the generated ideas in the list.	A new sticky note should appear containing the generated idea.

6. Resources and Schedule

Allocation of Testing Responsibilities

Testing was allocated based on each team members functional area. Kellen was responsible for Artificial Assistant unit testing, Vanessa and Matthew split the testing for the view models as there were only so many local unit tests that were available to test. Vanessa completed the automated swift UI testing. Test were recorded and monitored on the Jira Test Plan Board.