Cloudgile Final Report

Prepared by
Akshant Jain, Ayna Jain, Shaan Shekhar and Tirth Patel
***for use in* CS 440**
at the
University of Illinois Chicago

Spring 2021

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

Collaborating on software development is hard. That is the reason cloudgile exists. Cloudgile is a cloud-based platform that allows developers and managers to collaborate in an agile manner. It provides essential tools that already existing applications lack such as in-built communication supporting in-app messaging, voice chat and video chat along with integration of git that allows automating user's progress along the development of their project.

2 Project Domain

The project is intended to target developers with varied experiences ranging from new developers with very little to no experience, to highly experienced developers. The minimalist design and ease of use of the project is intended to get new devs on board with agile development. And the integration of all necessary features with modern UI elements, a stable platform which is a better alternative to switch to, for the experienced developers.

3 Relationship to Other Documents

The Project Description Documentation, Project Requirements Documentation and the Project Design Documentation identifies the project's need, and gives the project manager permission to request resources and carry out project tasks. All the other reports including this one allows the stakeholders to be informed about the progress of the project on whether it is meeting the performance standards or the tasks are on schedule and most importantly meeting the project's planned targets and deliverables.

- All the requirements that are mentioned in Description, requirements gathering and design that are fulfilled by the project are mentioned in this report including the testing and inspection for different releases. Also, describing some of the features that were mentioned in the Description report and needs to be implemented in the future releases.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Below are few naming conventions and definitions to properly understand the testing and inspection process. For other conventions related to the project, please refer to the project description report of cloudgile project.

Sprint:

Sprint is used to define a short period of time (e.g. 1 week) in which a “Done” product of highest possible value is created. It is not used to refer to a full speed run over a short distance.

Scrum Master:

Scrum master is a project lead responsible for tracking the overall progress of the development process. They are responsible to check if the team is following proper scrum processes. They can also be thought of as product owners.

Backlog:

Backlog is a collection of issues or functionality that needs to be implemented in the product in future. There can be a sprint backlog or product backlog. Sprint backlog means that the issues need to be finished by the end of the sprint whereas product backlog means that the issue/ functiliny should be fixed or added by when the product is finished.

Tasks:

Tasks are story/issue assigned to a particular team member during a sprint. A task maps more precisely to the product’s features and is easier to understand technically.

Timeline:

Timeline is a collection of issues that have been closed or completed. It can be reopened by only the scrum master.

4b UML and Other Notation Used in This Document

This document does not contain any UML diagrams instead we used Screenshots of the project from different releases for the better demonstration of the development process.

4c Data Dictionary for Any Included Models

Account :

Account is a user instance that contains information unique to the user.

Reminders:

Reminders are tasks that are due or near deadlines.

Website:

A website is a page on the internet that handles user interactions with buttons and input and displays the content of the server.

Issues:

Issues are functionality that needs to be implemented in the product being developed.

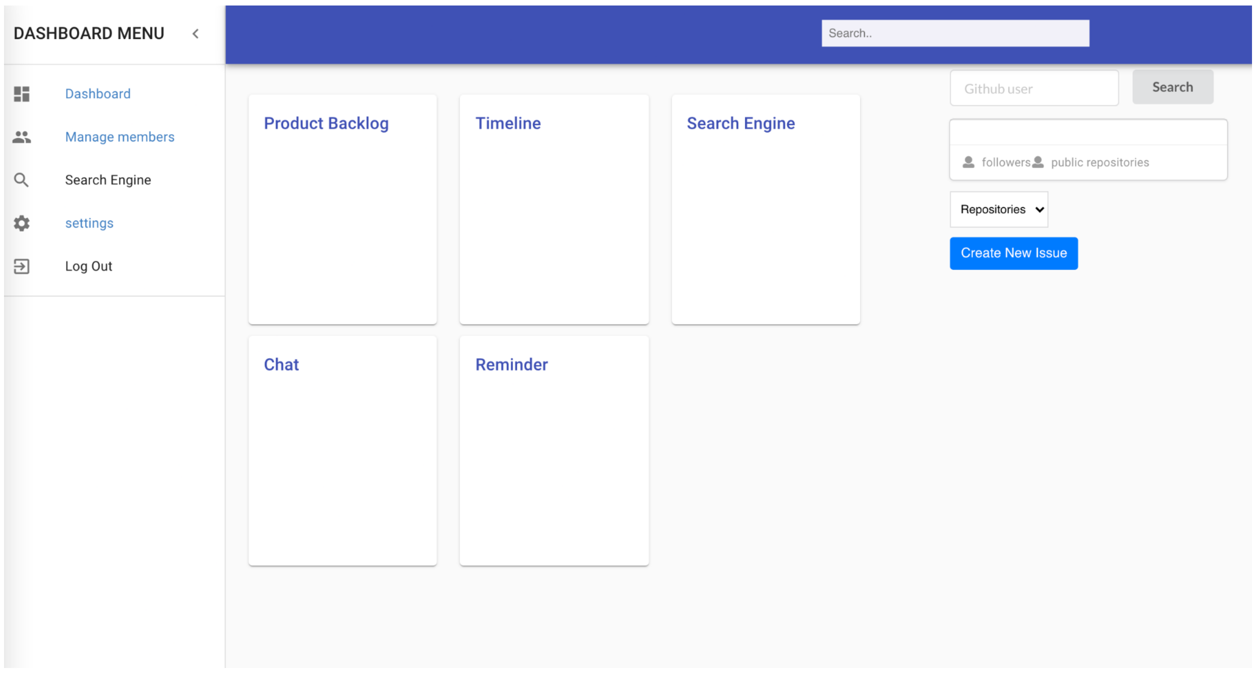
Session:

A session is a term used to describe when a user is using and making changes to the website. It temporarily stores user information and expires after a period of time.

II Project Deliverables

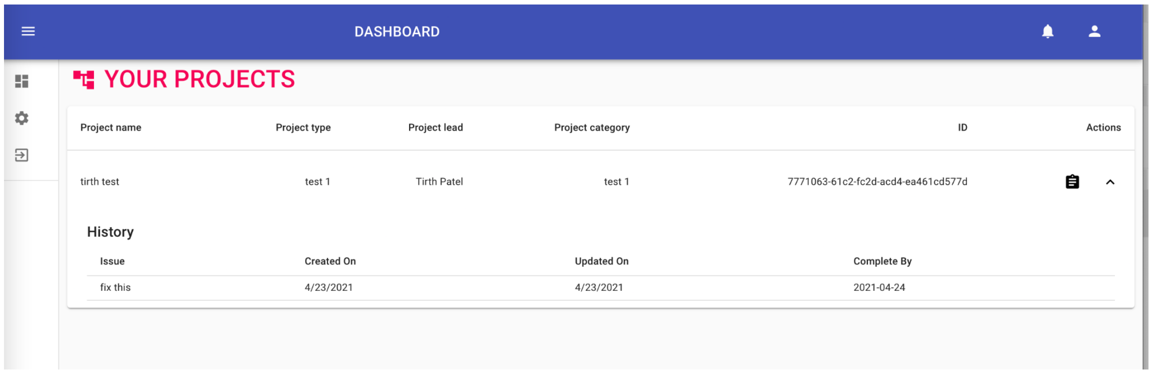
Our group has made a working prototype of the project cloudgile. It allows a team to work together and collaborate on projects in an agile manner. The application provides a method to create a project and add or remove members to and from the projects. It also has the functionality to create backlog issues and assign it to different group members. Moreover, the application also provides a seamless way to communicate with other members through a chat feature. It also includes features such as only the project lead can add or remove members, only the person assigned an issue or the project lead have the ability to to edit the issues and reopen it from timeline. Moreover, issues from the product backlog can be added to a sprint.

1 First Release



The product's first release was on February 26, 2021. By the end of the first release, our product looked something like this. We were mainly able to do some work on the front-end of the project. The product also had github integration which allowed to fetch information from a person's github account. But, it turned out that it is not useful as we were just able to display a user's information from github and not actually integrate any version control. The product also had a separate chat feature ready that needed to be integrated with the project. It includes other features such as only the project lead can remove or add members, only a person assigned to an issue and the project lead have the ability to edit issues and reopen it. Moreover, the application also provides features to add issues to a sprint.

2 Second Release

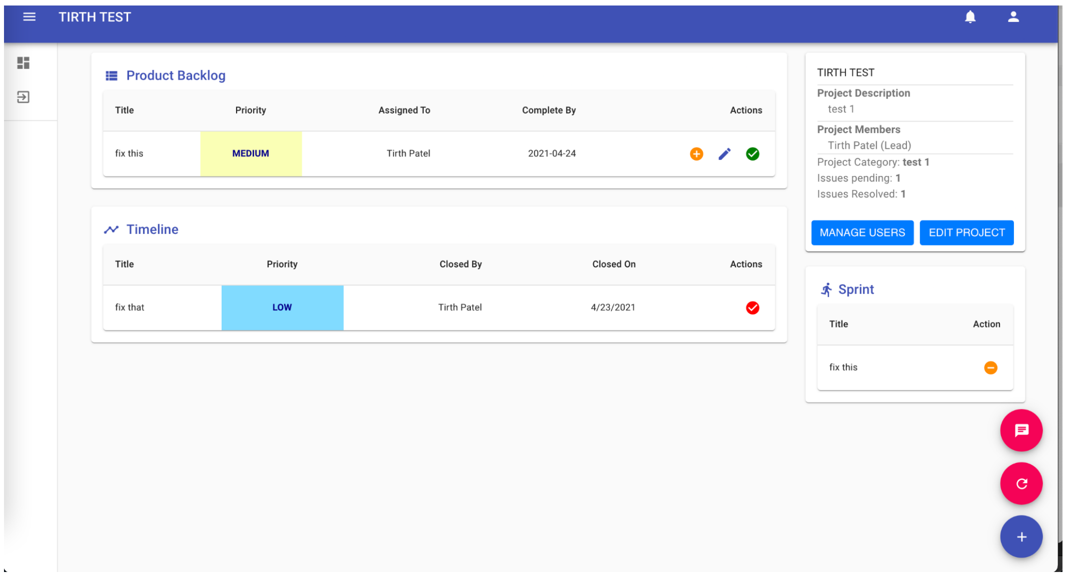

DASHBOARD

YOUR PROJECTS

Project name	Project type	Project lead	Project category	ID	Actions
tirth test	test 1	Tirth Patel	test 1	7771063-61c2-fc2d-acd4-ea461cd577d	

History

Issue	Created On	Updated On	Complete By
fix this	4/23/2021	4/23/2021	2021-04-24


TIRTH TEST

Product Backlog

Title	Priority	Assigned To	Complete By	Actions
fix this	MEDIUM	Tirth Patel	2021-04-24	

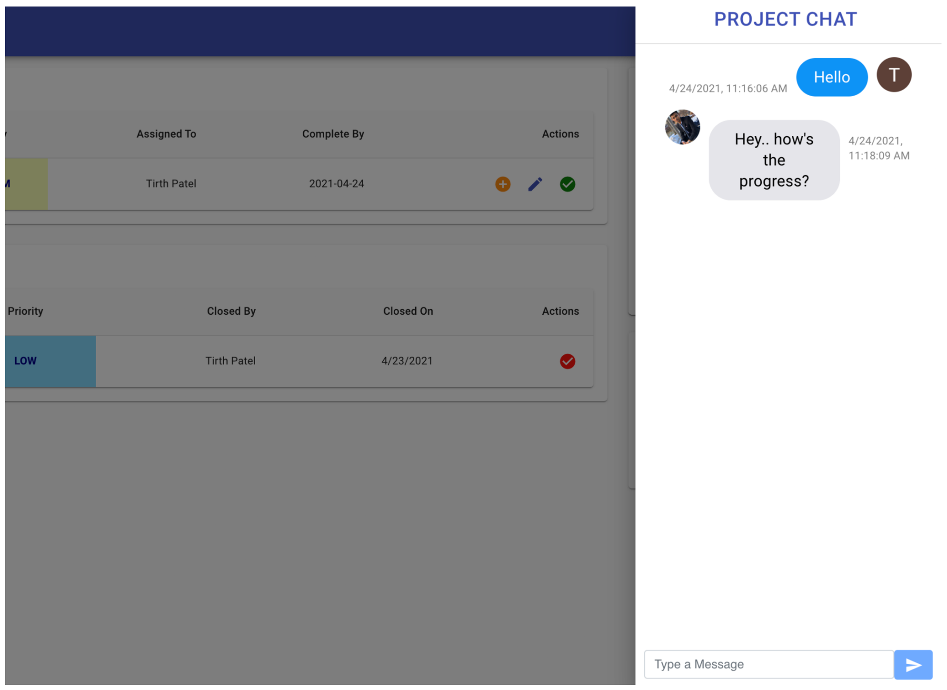
Timeline

Title	Priority	Closed By	Closed On	Actions
fix that	LOW	Tirth Patel	4/23/2021	

TIRTH TEST
Project Description
test 1
Project Members
Tirth Patel (Lead)
Project Category: **test 1**
Issues pending: **1**
Issues Resolved: **1**

Sprint

Title	Action
fix this	



Hello
4/24/2021, 11:16:06 AM

Hey.. how's the progress?
4/24/2021, 11:18:09 AM

The products' second release was on April 2, 2021. By the end of our second release, we had included all possible functionalities. The second release had a completely redesigned front-end and also a new back-end. Second release had integrated a chat feature that was unique to each project and its members allowing them to easily communicate from the application. The second release also had the ability to add and remove users to and from the project along with a feature to send notifications to group members whenever there is a major change to the product. It also included a tutorial for first time users that showed users their way around the application.

3 Comparison with Original Project Design Document

Below is a table comparing the prototype produced with the original project design document.

Original Project Design Document	**Protype produced by our group**
Login Validation	✅
Create new project	✅
Maintain already existing project	✅
Tutorial on how to use the system	✅

Send alerts to the team members regarding the work to be done	✅
Chat with other online team members	✅
Search Engine	❌
Switch role	✅
Manage Git from Cloudgile	❌
Set Reminder	❌

III Testing

1 Items to be Tested

The following items are needed to be tested:

- Account creation
- New user tutorial slides
- New project creation
- New issue creation
- Modifications in the issues or projects
- Chatting Feature

2 Test Specifications

ID# - Account Creation

Description: Creation of a new Cloudigle account with google signup.

Items covered by this test: Google signup/login API

Requirements addressed by this test: 6, 9

Environmental needs: working internet connection, web browser installed, a Gmail account

Intercase Dependencies: N/A

Test Procedures: The user is required to open the cloudigle website, click on the login/signup button, and follow the instructions on screen to login via a valid working google account.

Input Specification: The input for the test is a valid google email address which is utilized for logging into the application.

Output Specifications: If the test is successful, the user will be registered in the application database, and will be redirected to the dashboard of the application.

Pass/Fail Criteria: The test is passed if the user is able to login in the application with google, and see the dashboard of the application.

ID# - New User Tutorial

Description: User is shown a tutorial of the application when they login for the first time.

Items covered by this test: tutorial screen, swipeable views

Requirements addressed by this test: 1

Environmental needs: no special needs required.

Intercase Dependencies: test **ID# Account Creation** must be passed before this test .

Test Procedures: as a part of an automated script, the swipeable views are presented automatically when a user signs in the application for the first time.

Input Specification: the input is successful account creation of a user.

Output Specifications: a flag is flipped in the code which prevents the swipeable views appearing next time the user signs in the application. Secondly, as an output, the swipeable views must be displayed.

Pass/Fail Criteria: if the swipeable views are displayed correctly for the first time, and not displayed subsequent times, the test is considered passed.

ID# - Project Creation

Description: Test checks if the form to create projects is working correctly and corresponding backend is handled correctly.

Items covered by this test: forms functionality, backend database, project screen

Requirements addressed by this test: 3

Environmental needs: no special environmental needs required.

Intercase Dependencies: tests **ID# Account Creation**, **ID# - New User Tutorial**

must be passed before this test.

Test Procedures: user clicks create new project button on the bottom right page, fills in the form in complete, and clicks submit button, an automated script runs and performs the test.

Input Specification: the user must input valid information in all input fields in the form, The form must be completed in full to be able to activate the submit button.

Output Specifications: A new project screen is created and displayed with the specifics of the project which was taken by the user in the initial form.

Pass/Fail Criteria: If the form is displayed correctly, the submit button activates only when all the fields are filled, project is registered in the backend database, the new project screen is created with all the correct specifics, the test is considered as passed.

ID# - Issues Creation

Description: Test checks if the form to create issues is working correctly, handling issues by specific, and corresponding backend is handled correctly.

Items covered by this test: forms functionality, backend database, issue components, issue roles, miscellaneous functionalities

Requirements addressed by this test: 3, 4, 5, 8

Environmental needs: no special environmental needs required.

Intercase Dependencies: tests **ID# Account Creation**, **ID# - New User Tutorial, ID# - Project Creation** must be passed before this test.

Test Procedures: user creates an issue with a form, toggles with all issue settings including whom to assign the issue, priority of the issue, date of completion, etc.

Input Specification: the user is required to input all fields given in the form for creation of the issue and clicking the submit button.

Output Specifications: The issue must be displayed as a list item on the main project screen.

Pass/Fail Criteria: If the form is displayed correctly, the submit button activates only when all the fields are filled, issue is registered in the backend database, the new issue component is created with all the correct specifics, the test is considered as passed.

ID# - Modification Handling

Description: Test checks if any modification to a project or an issue is handled correctly, the modifications are reflected in the UI as well as the backend database.

Items covered by this test: forms functionality, backend database, UI components, project edits, issue edits, manage members, etc.

Requirements addressed by this test: 3, 4, 5, 8

Environmental needs: no special environmental needs required.

Intercase Dependencies: tests **ID# Account Creation**, **ID# - New User Tutorial, ID# - Project Creation** must be passed before this test.

Test Procedures: users may change the project details, issue details, project members, etc. through the UI of the application.

Input Specification: the user must input any modification that they want to do with the project, or an issue.

Output Specifications: The changes must be reflected in the backend and the frontend of the application on click of the update button.

Pass/Fail Criteria: if all changes are being reflected correctly in the backend and the frontend, the test is passed.

ID# - Chatting Feature

Description: test checks if the chatting feature of the project is functional.

Items covered by this test: forms functionality, backend database, UI components, chatting feature.

Requirements addressed by this test: 7

Environmental needs: no special environmental needs required.

Intercase Dependencies: tests **ID# Account Creation**, **ID# - New User Tutorial, ID# - Project Creation** must be passed before this test.

Test Procedures: the chat icon on the bottom right of the screen is needed to be clicked which opens up a sidebar which enables messaging among all the group members in a project.

Input Specification: the chat message is the input of the test.

Output Specifications: when the user clicks the send button, the message is sent to all the group members in the group. On the receiving end, the message pops up on the screen.

Pass/Fail Criteria: if the message is going through and being received on the other end, the messages are not mixed with messages of the other groups, and the database is updated with the messages in real time, the test is passed.

3 Test Results

ID# - Account Creation

Date(s) of Execution: February 5-7

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: The account should get created with the database updated.

Actual Results: The account was created and the database was updated.

Test Status: Pass

ID# - Project Forms

Date(s) of Execution: February 14-16

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: A form should open up with relevant fields.

Actual Results: A form was created with relevant fields.

Test Status: Pass

ID# - Project Screen

Date(s) of Execution: February 16-19

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Clicking submit button should create a project screen with user input information.

Actual Results: The screen was created but the database was not being correctly updated. After some modifications, the test was successful.

Test Status: Pass

<u>ID# - Creating Issues</u>

Date(s) of Execution: February 17-19

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Clicking submit button should create an issue item with user input information.

Actual Results: The component with issue information was created.

Test Status: Pass

<u>ID# - Modification of Issues and Project</u>

Date(s) of Execution: March 23-30

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Making any changes to the information of the project or an issue will change the database records and change the information on the project's main page.

Actual Results: Changes were reflected as expected with the database being updated as expected.

Test Status: Pass

<u>ID# - UI Testing</u>

Date(s) of Execution: Feb 20-25, March 7-18, April 10-19

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Making any changes to UI of the application will update correctly in the UI and database.

Actual Results: Changes were reflected as expected with the database being updated as expected.

Test Status: Pass

<u>ID# - Chat Features</u>

Date(s) of Execution: April 1-19

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Chatting features like ability to send text messages, receive text messages, store chat data in order in the database, display old text messages, integration of the chat UI in the main project page, and all UI tweaks should work.

Actual Results: Started with a full dedicated screen given to the chat. Later removed the full screen implementation and implemented it in a sidebar. The backend to store chat history of one chat group separate from another group eventually worked after rigorous testing.

Test Status: Pass

<u>ID# - Git Integration</u>

Date(s) of Execution: March 21-30

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: The user can add members to the group via Github API. Load projects, and all kinds of data from Github directly to the project page.

Actual Results: Although we were able to add users from github, the anticipated result didn't cover the entirety of our anticipation as the results were not correct and required users to directly put github profile links and couldn't utilize the search feature.

Test Status: Fail

<u>ID# - FireChat Integration</u>

Date(s) of Execution: March 21-30

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: The user will directly be able to use the firechat feature directly integrated with the application. Additionally, it will become very easy to manage the backend and implement the UI in the application.

Actual Results: Firechat turned out to be incompatible with the type of the project that we are working on.

Test Status: Fail

<u>ID# - Search Integration</u>

Date(s) of Execution: March 11-30

Staff conducting tests: Akshant Jain, Ayna Jain, Shaan Shekhar, Tirth Patel

Expected Results: Utilizing Google's search API, we anticipated to implement the search feature where users can do a google search through our website.

Actual Results: The integration required to use some features out of our scope of knowledge, and the implementation idea was dropped after multiple failed attempts.

Test Status: Fail

4 Regression Testing

Below are the tests that required regressive testing:

- **Modification of Issues and Project**
- **UI Testing**
- **Git Integration**
- **Account Creation**
- **Search Integration**
- **FireChat Integration**
- **Chat Features**

IV Inspection

1 Items to be Inspected

Below are the items listed that were needed to be inspected:

- Forms
- Login API and Firebase Authentication
- UI Components
- Frontend-Backend integration
- Database Inspection
- Notifications
- Chat

2 Inspection Procedures

ID# - Forms

Submitted by: Tirth Patel

Checked by: Akshant Jain, Ayna Jain, Shaan Shekhar

Items covered by this item: forms are used during the creation of new projects, creation of new issues, updating project details, and updating issues.

Intercase Dependencies: tests **ID# Account Creation**, **ID# - New User Tutorial** must be passed before performing the inspection.

Test Procedures: ID# - Project Creation, ID# - Issues Creation, ID# - Modification Handling must be performed to test out the item.

Pass/Fail Criteria: If the form is displayed correctly, the submit button activates only when all the fields are filled, issue is registered in the backend database, the new issue component is created with all the correct specifics, the test is considered as passed.

ID# - Login API and Firebase Authentication

Submitted by: Akshant Jain

Checked by: Ayna Jain, Shaan Shekhar, Tirth Patel

Items covered by this item: Login API and the firebase authentication is utilized when the user tries to login or create an account on the application.

Intercase Dependencies: none.

Test Procedures: ID# - Account Creation must be performed to test out the item.

Pass/Fail Criteria: The test is passed if the user is able to login in the application with google, and see the dashboard of the application.

ID# - UI Components

Submitted by: Shaan Shekhar

Checked by: Ayna Jain, Akshant Jain, Tirth Patel

Items covered by this item: All the UI components must function as intended by the developers.

Intercase Dependencies: none.

Test Procedures: ID# - Project Creation, ID# - Issues Creation, ID# - Modification Handling, ID# - Chatting Feature, ID# - New User Tutorial, ID# - Account Creation must be performed to test out the item.

Pass/Fail Criteria: The test is passed if the application does not break when some or all of the UI components are triggered.

<u>ID# - Database Inspection</u>

Submitted by: Ayna Jain

Checked by: Akshant Jain, Tirth Patel, Shaan Shekhar

Items covered by this item: Data of one project should now overflow to the other project, data of one account should now overflow to any other account.

Intercase Dependencies: none.

Test Procedures: <u>ID# - Project Creation</u>, <u>ID# - Issues Creation</u>, <u>ID# - Modification Handling</u>, <u>ID# - Chatting Feature</u>, <u>ID# - New User Tutorial</u>, <u>ID# - Account Creation</u> must be performed to test out the item.

Pass/Fail Criteria: if one user can see only his data and chat of a specific project in that project only, database inspection is successful.

<u>ID# - Frontend-Backend integration</u>

Submitted by: Akshant Jain

Checked by: Ayna Jain, Tirth Patel, Shaan Shekhar

Items covered by this item: The UI changes and modification in any information by the user in the frontend of things must be updated correctly in the backend.

Intercase Dependencies: none.

Test Procedures: <u>ID# - Project Creation</u>, <u>ID# - Issues Creation</u>, <u>ID# - Modification Handling</u>, <u>ID# - Chatting Feature</u>, <u>ID# - New User Tutorial</u>, <u>ID# - Account Creation</u> must be performed to test out the item.

Pass/Fail Criteria: The test is passed if the application does not break when some or all of the UI components are triggered, and all the triggers are handled correctly by the backend and it produces intended results.

<u>ID# - Notifications</u>

Submitted by: Akshant Jain

Checked by: Ayna Jain, Tirth Patel, Shaan Shekhar

Items covered by this item: The notifications are created when a new project is created, a new issue is created, the project is updated, and issues are updated.

Intercase Dependencies: **ID# - New User Tutorial, ID# - Account Creation** must be performed prior to inspecting this item.

Test Procedures: **ID# - Project Creation**, **ID# - Issues Creation, ID# - Modification Handling**, **ID# - Chatting Feature,** must be performed to test out the item.

Pass/Fail Criteria: Notifications must be generated when any of the **ID# - Project Creation**, **ID# - Issues Creation**, **ID# - Modification Handling**, **ID# - Chatting Feature** is executed.

ID# - Chat

Submitted by: Ayna Jain

Checked by: Akshant Jain, Tirth Patel, Shaan Shekhar

Items covered by this item: The notifications are created when a new project is created, a new issue is created, project is updated, and issues are updated..

Intercase Dependencies: **ID# - New User Tutorial, ID# - Account Creation, ID# - Project Creation** must be performed prior to inspecting this item.

Test Procedures: **ID# - Chatting Feature,** must be performed to test out the item.

Pass/Fail Criteria: **ID# - Chatting Feature,** **ID# Database Inspection** must be functional to pass this test.

3 Inspection Results

ID# - Forms

Submitted by: Tirth Patel

Checked by: Akshant Jain, Ayna Jain, Shaan Shekhar

Date(s): Feb 14-18

Result: The forms eventually performed as expected. Initially, the forms were taking blank inputs (which were invalid as the user is required to fill everything), but the issue was resolved.

Meetings: Only 1 meeting was held to discuss the functionality of the forms. The issue of fixing the issue of blank inputs was resolved in the meeting itself. The results were discussed in the meeting and through a group chat.

ID# - Login API and Firebase Authentication

Submitted by: Akshant Jain

Checked by: Ayna Jain, Shaan Shekhar, Tirth Patel

Date: Feb 17-24

Result: The firebase authentication worked fine in the first few attempts but as the load increased (multiple new accounts), the firebase crashed due to some minor error in the code. The error was later fixed and the authentication worked smoothly.

Meetings: 2 meetings were held to discuss the functionality of the API and authentication. The issue was found in the meeting and fixed at a later date. The results were discussed in the meeting and through a group chat.

ID# - UI Components

Submitted by: Ayna Jain, Akshant Jain, Tirth Patel, Shaan Shekhar

Checked by: Ayna Jain, Akshant Jain, Tirth Patel, Shaan Shekhar

Date: Feb 1 - April 18

Result: Since the inspection included all the UI components of the application, and the application took over two months to complete, the UI inspection was in progress almost the entire duration of the project. The results included UI components of the application from not working properly, to partially functioning, to flawless functioning components. Although some components started bad, the errors were fixed over time and achieved symbiosis in the end.

Meetings: About 25 meetings were held over the course of three months to discuss UI components. 50+ issues were found in the meetings and 100+ issues were discovered outside of the meeting. The issues range from very small to very large issues and were solved in both the meetings and outside of meetings. The results were discussed in the meeting and through a group chat.

ID# - Database Inspection

Submitted by: Ayna Jain

Checked by: Akshant Jain, Tirth Patel, Shaan Shekhar

Date: Feb1 - April 18

Result: From the beginning of the project, the directories were properly set up, and the code was written with the database structure in mind. Hence, we faced no difficulties in managing the backend database portion where mixing of two similar types of data could occur.

Meetings: No meetings were held to discuss the database inspection. Very few to no issues were found and were resolved independently. The results were discussed in the meeting and through a group chat.

ID# - Frontend-Backend integration

Submitted by: Akshant Jain

Checked by: Ayna Jain, Tirth Patel, Shaan Shekhar

Date: Feb 20 - April 18

Result: The integration between the frontend and the backend was a step by step process. As soon as a component is functional, it will be integrated with the backend database to make it fully functional. Apart from some instances where data was being stored in the wrong directory, we faced very few problems in managing the frontend with the backend.

Meetings: About 10 meetings were held to discuss and debug frontend-backend compatibility. Variety of issues were found and debugged in the meeting and fixed at a later date or in the meeting. The results were discussed in the meeting and through a group chat.

ID# - Notifications

Submitted by: Akshant Jain

Checked by: Ayna Jain, Tirth Patel, Shaan Shekhar

Date: April 14-15

Result: The code included a functionality which can enable users to get notifications of various activities that are happening with any of his projects. The code was flawless and worked perfectly fine from the beginning.

Meetings: No meetings were held to discuss the database inspection. Very few to no issues were found and were resolved independently. The results were discussed in the meeting and through a group chat.

ID# - Chat

Submitted by: Ayna Jain

Checked by: Akshant Jain, Tirth Patel, Shaan Shekhar

Date: March 10-April 18

Result: Since the chat is an important part of the project, it took slightly longer to fully test the chat in its entirety. The very first initial chat implementation was based on an independent model that produced various errors while integrating with the main project screen. The independent model was ditched and a new model with Firestore backend was utilized instead.

In the UI, the initially presented code covered full screen and there was no way to go to the previous page of the project. This was resolved when a sidebar was introduced in the UI design and the chat was pushed to the side.

Meetings: About 7 meetings were held to discuss the chat feature with some other issues. Almost half of those meetings were discussions about compatibility of the chat with the main project and trying to fix issues regarding the same. The other half of the meetings were utilized to debug issues of the new chat implementation. The results were discussed in the meeting and through a group chat.

V Recommendations and Conclusions

Yes, by the end of our demonstration, all of the tests conducted passed and the application was deployed live. For future testing, git integration, search engine functionality, voice and video chat will have to be tested.

VI Project Issues

1 Open Issues

- The application is being developed on web based platforms on the modern operating systems for the users to use it with ease.
- With the increasing use of mobile devices and tablets for being handy more than ever, the application should be developed separately for Android/ iOS devices. Although we used React for the front-end development therefore, the UI components can be changed with less difficulty for different kinds of user experience.
- Documentation for future releases for upcoming discoveries based on latest technology and software practices should be made available during and after the documentation process and the developers should be able to fix the related issues in future.

2 Waiting Room

Being a collaborative application the product should allow the functionality to voice and video call. Also, it should allow the users to screen share for pair programming.

Although the Chat feature has been implemented in our application, the product can also have integration with Social Networking sites such as LinkedIn so that the user can not only interact with the group members but also to the people who have the same motivation for building Softwares. The product is a fully-flexed cloud-based application therefore, it should allow the users to collaborate in any way possible. The functionalities that I mentioned above are the must-have features for any agile environment application allowing the user to implement the development process while performing various agile methodologies.

- Integration with Social Networking sites so that the users can interact on Social media.
- For screen sharing, the device should turn on screen sharing as it is not turned on by default.
- Should be able to use and test WebVR on their devices as it gives the web page control over displaying the content through mirroring on the user's selected display.

3 Ideas for Solutions

- Git Integration should be implemented in the future release with the project that allows version control directly from the application.
- Reminders feature should be implemented allowing the users to set and receive reminders regarding the deadlines of the project.
- A search engine will be integrated in the application that will provide developers to refer to other websites to fix their code quickly. It will search all the team's reports and code in the database.

4 Project Retrospective

We implemented the Agile methodologies for the Software development which made it easier for us to add and update new functionalities in the product while testing all the updated features for every release. We implemented a pair programming approach which decreased the development time by 20%. Before the product is made, an analysis should be conducted. This research would consist of a sample of potential users' preferences for a website similar to ours. If the input from potential buyers differs significantly from the design proposal, a meeting between the customer and the production team may be scheduled, and the product will need to be replanned. Any new feature implemented in the project should be done following the documentation for the same.

VII Glossary

Users that are new to agile environments such as startups that shifted their focus to developing softwares needs to know these key terms before using the product or before building it for the market.

Cloudgile - Developers, administrators, and other partners can easily collaborate and communicate on projects using this platform.

cloud-based - A cloud-based software is an Internet-based application with web elements and several cloud-based operations. Any app that is made up of data and processing logic that is encoded as code and requires a physical location/space to run.

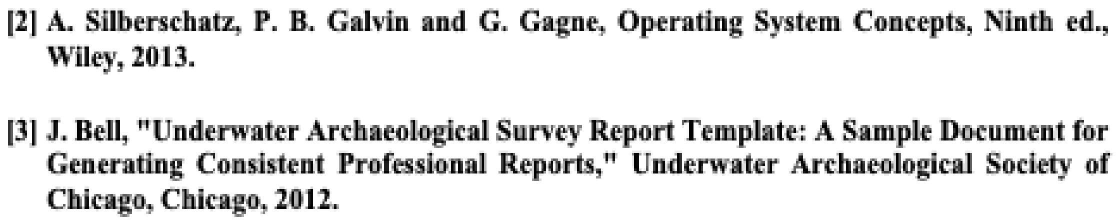
Agile environment - an agile environment is a company philosophy that promotes agile project management. As a result, in an agile environment, transition, innovation, versatility, and imagination are accepted and encouraged in the software development process.

Realtime Database - A real-time database is a database system that processes data in real time to manage workloads that are continuously evolving. Real-time processing refers to the processing of a transaction quickly enough for the outcome to be returned and operated on immediately.

VIII References / Bibliography

[1] Tianniu Lei, Kevin Nguyen, Naga Jyothi Sai Pavani Velagapudi ,Erik Reyes

"Cloudgile Project report", Spring 2016


[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[2] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

IX Index