Team Project

Test Plan

"Student Assist app"

Course ID: CS487 Software Engineering I

Professor: Dennis Hood

Team Name: Team G8

Team Members:

- Shivdeep Bisurkar (A20525712)
- Prajakta Kumbhar(A20523710)
- Colin Brennan (A20418083)
- Hao-Chih Weng (A20480011)

Date: 2/25/2023

Section 1 Summary of Proposed Testing Strategy

I. Summary of Application Functionality

According to the requirement analysis provided by the G7 group, we can make some conclusions. The application is designed to provide students with various features such as User Management, Collaboration Tools, Job Search, Assignment Management, Usability, Robustness, Portability, Scalability, Evolvability, Security, and Accessibility. It allows students to manage their personal information, collaborate with their peers and instructors, search for job opportunities, manage their assignments, and view their grades and feedback from instructors. The app is designed to be user-friendly and intuitive, accessible from any device with an internet connection, and able to handle large amounts of data and traffic. It should be designed with the ability to be updated and improved over time to meet changing user needs and expectations, maintain the privacy and security of student information, and accessible to users with disabilities.

II. Strategy for testing

A. In the proposed strategy for testing the provided application, our group has decided to go forward with a total of four testers. This helps to ensure the repeatability and reliability of the tests. The more individuals testing an application the higher the chance there is to catch and mitigate potential bugs.

B. Testing tools

1.Selenium: A popular open-source tool for functional testing that allows automated testing of web applications across different browsers and platforms.

2.JMeter: An open-source tool used for performance and load testing to simulate high user traffic to a web application to test its scalability and reliability.

C. Test data:

I. Five user should be created with below data:

1. Username
2. Password
3. Email id
4. Mobile number
5. Address

II. Five “.doc” files ,Five”.pdf” files,Five”,”.xlsx” files should be available for testing.

D. System environments:

1. Test the app on a mobile phone:

a) Verify that all features of the application, such as result inquiry, course selection, and school activity inquiry, are functioning as intended on the mobile phone.

b) Verify that the application is compatible with the mobile phone's operating system and version.

c) Test the application's responsiveness on a mobile phone, ensuring that it loads quickly and without errors.

2. Test the app on a personal computer:

a) Operating system: The software may need to be tested on multiple operating systems, such as Windows, Mac OS, and Linux.

b) Browser compatibility: If the software is web-based, it should be tested on different browsers such as Chrome, Firefox, Safari, and Edge to ensure that it functions properly on each.

c) System resources: The software should be tested on different hardware configurations to ensure that it runs smoothly on a range of devices.

Section 2 Test Case

User Management Test Case

Test case title	User Management- Create and Manage Accounts
Test case description	To ensure that the app is able to create and manage new accounts a new account will be created then edited.
Initial state	The device being used for testing, has a stable internet connection, and is open to the Create a New User Page.
Inputs	A new account will be created then the personal information such as email address or home address will be edited. Upon new account creation the accounts privacy preferences will also be set.
Actions	1.Launch the app on a desktop computer using a popular web browser (e.g., Google Chrome, Mozilla Firefox, Safari). 2.Open application to the “Create New User” page. 3.Enter all relevant user information details such as Name, email, and school. 4.Finalize the user creation. 5.Verify that a new account has been created. 6.Sign into the new account and begin to change personal information and privacy preferences. 7.Save changes then logout of the account and verify that the changes remain in place.
Expected end state	If the new account creation test works then the application should result in a successful account creation as well as a successful account update.

Collaboration Tool Test Case

Test case title	Collaboration Tool - Create/Join and participate in groups
Test case description	This test case is to ensure that the users on the application are able to create or join groups, participate in discussions, share files and documents, and communicate in real-time
Initial state	The device being used for testing, has a stable internet connection, and is open to the Create a New Group Page
Inputs	A new group will be created in which the tester will also send files and messages in.

Actions	1.Launch the app on a desktop computer using a popular web browser (e.g., Google Chrome, Mozilla Firefox, Safari). 2.Login with valid credentials. 3.Open application to the “Create New Group” page. 4.Enter all relevant new group information details such as group name, description, and purpose. 5.Finalize the group creation. 6.Verify that a new group has been created. 7.Attempt to send both messages as well as files/documents in the new group. 8.Verify that files and messages are being sent and delivered in real-time.
Expected end state	If the new group creation test works then the application should result in a successful new group creation as well as a successfully sent message in said group.

Job Search Test Case

Test case title	Job Search - Students should be able to search for jobs based on various criteria such as location, field, and job type.
Test case description	This test case is to ensure that the users on the application are able to search for various jobs based on a variety of filters such as location and field.
Initial state	The device being used for testing, has a stable internet connection, and is open to the Job Search Page.
Inputs	A job search will be inputted with the following parameters: Location = Chicago, Field = Technology, and Job Type = Help Desk
Actions	1.Launch the app on a desktop computer using a popular web browser (e.g., Google Chrome, Mozilla Firefox, Safari). 2.Login with valid credentials. 3.Open application to the “Job Search” page. 4.Enter all relevant job search information with specific parameters regarding the search. 5.Load the job results from the search. 6.Verify that the resulting jobs are a match with the provided search parameters. 7.Select a job and attempt to apply as well as save the job posting. 8.Verify a successful application as well as that the job was saved successfully.
Expected end state	If the job search test works then the application should result in a list of jobs that match the user’s search parameters. From that list of jobs the

	user should have the option to save job postings, apply for jobs, and track the status of their applications.

I.

Test case title	Portability - App Accessibility on Various Devices
Test case description	To ensure that the app can be accessed from any device with an internet connection, including desktops, laptops, smartphones, and tablets.
Initial state	The device being used for testing has a stable internet connection.
Inputs	Create an account, edit personal information, etc on major devices like mobile phones, desktops and laptops.
Actions	1.Launch the app on a desktop computer using a popular web browser (e.g., Google Chrome). 2.Attempt to log in to the app using valid credentials. 3.Verify that the app successfully logs in and is accessible on the desktop computer. 4.Repeat steps 1-3 on a different desktop computer using a different web browser. 5.Repeat steps 1-3 on a laptop computer using a popular web browser. 6.Repeat steps 1-3 on a tablet using a popular web browser. 7.Repeat steps 1-3 on a smartphone using a popular web browser. 8.Verify that the app is accessible and functions properly on all devices tested by attempting to perform common tasks like edit info, share files, etc.
Expected end state	If the app is accessible and functions properly on all devices tested, and all common tasks can be performed without issue, then the test case passes.

Test case title	App Evolvability
Test case description	To verify that the app is designed with the ability to be updated and improved over time to meet changing user needs and expectations.
Initial state	The app has been used for some time by users
Inputs	User feedback or suggestion for a new feature or improvement and Communication with the developer through the feedback mechanism
Actions	1.Verify that the app has a feedback mechanism, such as a feedback form or a way to contact the developer.

	2.Send a feedback message to the developer requesting a new feature or improvement to an existing feature. 3.Verify that the developer responds to the feedback message within a reasonable timeframe, such as within a week. 4.Verify that the developer takes the feedback into consideration and implements the requested feature or improvement in a future app update. 5.Verify that the app notifies the user of the availability of the new update and provides an easy way to install it. 6.Install the new update and verify that the requested feature or improvement has been implemented. 7.Verify that the app remains stable and functional after the update.
Expected end state	The updated app is stable and functional.

Test case title	App Scalability
Test case description	To ensure that the app can accommodate an increasing number of users without any degradation in performance.
Initial state	The app is up and running in the developer portal.
Inputs	1.Load testing software or a similar tool to simulate a higher number of concurrent users. 2.Performance metrics monitoring tool to measure response time and CPU and memory usage.
Actions	1.Identify the maximum number of concurrent users that the app can currently support while maintaining good performance. 2.Using load testing software or a similar tool, simulate a higher number of concurrent users than the maximum number identified in step 1. 3.Monitor the app's performance metrics, such as response time and CPU and memory usage, during the load test. 4.Verify that the app can handle the increased load without any significant degradation in performance, such as slow response times or crashes. 5.Repeat the load test with increasing numbers of concurrent users until the app's performance begins to degrade. 6.Identify the maximum number of concurrent users that the app can handle while still maintaining good performance. 7.Document the results of the load test and the maximum number of concurrent users that the app can support.
Expected end state	The app meets the requirement for scalability, as it can accommodate an increasing number of users without any degradation in performance.

Test case title	Security
Test case description	Verify that the app should maintain the privacy and security of student information and data.
Initial state	The user is on sign-up or login page
Inputs	User should be on login/ sign up page

Below are the action steps for above test case

Step No	Actions	Expected Results
1	Click on the sign up button and verify that the user should be able to navigate to create an account page.	User is able to see create account page with below mandatory fields: 1. Username 2. Password 3. Confirm Password. 4. Security question. 5. Birth date 6. Country 7. Address. 8. University name 9. Course name 10. Email id 11. Contact number 12. Multi-factor authentication using phone and email (with radio button as yes/no)
2	Click on Create Account	Account created successfully and the user navigates to the Home page.
3	Click on Logout button	User should be able to log out from application and navigate back to login page
4	Add Login credential (Username and password) which was created in step 1.	user is able to login into application and navigate to two factor authentication page
5	Add answer to security question.	User is able to login successfully and navigate to home page
6	Click on Logout button	User should navigate back on login page
7	Add Username which is create in step1 and add password other than created in step1	Verify that User should not be able to login into the app.

Test case title	Accessibility
Test case description	Verify that physically challenged persons should be able to login into application
Initial state	The user is on sign-up or login page
Inputs	User should be on login/ sign up page /voice assistance should on

Step No	Actions	Expected Results
1	Verify that user should be able to login into application via voice assistance by dictating username and password only by his/her voice	User logged into application successfully
2	Verify that user should be able to see high contrast option under setting	High contrast option available in application and user is able to change display to high contrast mode.
3	Verify that user should be able to log out using voice commands	User is able to log out

Robustness:

Test case title	Test app's ability to handle high traffic during peak hours
Test case description	To verify that the software application can handle large amounts of data and traffic, and function effectively during periods of high usage.
Initial state	The initial state of the test case could be the application running generally with an average amount of traffic. This would serve as the baseline for comparing high traffic during peak hours.
Inputs	The input part of the test case for the app's ability to handle high traffic during peak hours, you would specify the type and amount of input data that will be used to simulate a high-traffic scenario. This could include simulating multiple users accessing the app simultaneously or generating a large amount of data to be processed by the app. The input data should be designed to test the app's ability to handle the expected peak load and identify any potential bottlenecks or performance issues.
Actions	1. Create test data with a large number of records, such as 10,000 or more. 2. Simulate high traffic by running multiple instances of the application simultaneously. 3. Submit requests to the application at a high frequency, for example, once per second. 4. Observe the behavior of the application during the test, and record any issues or errors encountered. 5. Monitor system resource usage (such as CPU and memory) during the test, to ensure the application is not overloading the system. 6. Repeat the test with increasing levels of data and traffic until a limit is reached, or until the application is unable to function effectively.
Expected end state	1. The application should be able to handle a large amount of data without experiencing any issues or errors. 2. The application should be able to handle high traffic without slowing down or becoming unresponsive. 3. The application should be able to function effectively even during periods of high usage, without encountering any issues or errors. 4. The system resource usage should not exceed safe limits during the test.

Usability:

Test case title	User Interface Usability Test
Test case description	This test case aims to verify the usability of the software application's user interface, ensuring that it is intuitive and user-friendly.
Initial state	The software application is open and running on the device.
Inputs	The user is presented with the application's main screen, which contains various options for different functions.
Actions	1. Navigate through the application's menu options, ensuring that each function is easy to find and select. 2. Attempt to complete a specific task, such as checking grades or creating an assignment, while observing the application's interface and usability. 3. Check that the application's interface is consistent throughout all functions. 4. Attempt to return to the previous screen or exit the application, ensuring that the process is intuitive and easy to complete. 5. Check that the application's interface complies with standard usability guidelines, such as color contrast, font size, and spacing.
Expected end state	1. The user should be able to easily navigate through the application's menu options. 2. The user should be able to complete the task with minimal effort and without experiencing any confusion or frustration. 3. The application's interface should be consistent throughout all functions. 4. The process of returning to the previous screen or exiting the application should be intuitive and easy to complete. 5. The application's interface should comply with standard usability guidelines.

Assignment Management:

Test case title	Ability to Manage Assignments Effectively
Test case description	To ensure that the app provides students with a robust platform to manage their assignments, set deadlines, and receive

	reminders. The app should also allow students to view their grades and feedback from instructors.
Initial state	The student has logged into the app and is viewing the Assignment Management page.
Inputs	The student creates a new assignment, sets a deadline, and adds relevant details (such as a description and any attached files). The student then saves the assignment.
Actions	1. Log in to the app and navigate to the Assignment Management page. 2. Click on the "New Assignment" button. 3. Fill in all required fields (e.g. assignment title, description, due date, attached files). 4. Click on the "Save" button. 5. Verify that the new assignment is displayed on the assignment. 6. Management page, with all relevant details visible. 7. Verify that the student receives a notification or reminder before the assignment deadline.
Expected end state	The app should successfully save the new assignment, and display it on the Assignment Management page with all relevant details. The student should also receive a notification or reminder before the deadline.

Section 3 Recommendations

1. One recommendation I have for the other team is to consider implementing Google OAuth to help manage users and provide access control to the application. Also please provide what type of users you intend on having; teachers, students, admins etc.

2. The non-functional requirements can be explained in more detail, the current one’s feelings to be generic.

3. Security: If unauthorized users try to login into the system, how the application will behave. (i.e. it show message at login page or it will also send notification to user whos credential that person is using to login into application)

4. Accessibility : Specify the user with specific what kind of disability and how should they access the application