.

AVA An Automated Voice Activated Advisement System

(Test Plan)

By: The Software Gurus

Team Members:

Omar Obidat

Jesse Parron

Allen Asencio

Dan Ferdetta

Sumit Mistry

Software Engineering 515

Fall 2022

Professor Johnson

December 19, 2022

Table of Contents

[1.](#_heading=h.1fob9te) Introduction 2

[2.](#_heading=h.3znysh7) Modules with sample Inputs and outputs 3

[3.](#_heading=h.2et92p0) Modules with objective and testing criteria 5

[4.](#_heading=h.tyjcwt) Summary of the Module-to-test-technique mapping 8

[5.](#_heading=h.3dy6vkm) Dependency graph 10

[6.](#_heading=h.1t3h5sf) Summary of monitoring, reporting, and correcting procedures 11

[7.](#_heading=h.4d34og8) Proposed dates for submission of individual test reports 11

[8.](#_heading=h.2s8eyo1) Defense for integration 12

[8.1.](#_heading=h.17dp8vu) Test cases: 13

[9.](#_heading=h.3rdcrjn) Details of monitoring, reporting, and testing procedures 23

[9.1.](#_heading=h.26in1rg) Details of individual team member assignments 24

# Introduction

The automated voice advisement system (AVA) will allow students in the undergraduate class of the computer science department to sign on and vocally request assistance for academic advice.

The system will also allow students to request their current grade point average (GPA), view a “what-if” GPA which calculates an unofficial grade point average based on different potential grades, inquire about an unofficial transcript, and seek advisement on courses that should be taken at particular junctures in their academic pursuit. In order to achieve success with the development of this system, in-depth tests must be conducted in order to ensure high quality for all students.

Testing is held to a high degree in the Software Gurus team, and follows a strict set of processes, procedures, and standards. The team, Software Gurus, will perform an incremental methodology of testing. Once a module is completed, the team performs a unit test on the module itself. If it performs to specification, the module will be implemented into the system. This allows the team to conduct unit tests on each individual module, as well as test the whole system with implementation in place, while still progressing with the development of the other functionalities in the system.

As soon as the system is completed, the team will perform a full comprehensive functionality test to ensure each function, module, and aspect performs as intended in the design, and specification and requirements documentation.

In the event that bugs or issues are found, the team will follow the testing procedure to mend the problem and provide a solution. Bugs will be reported and documented, as well as the solution that is developed to handle the issue. The team will then conduct another functionality test as well as a final performance evaluation.

# Modules with sample Inputs and outputs

| **Module Name** | **Input** | **Output** |
| --- | --- | --- |
| **Login** | Students will input their Student identification number (ID). | The system will bring them to the home screen, where their student ID and module choices will be displayed. |
| **Language Recognition** | The student will verbally talk to the system inquiring on any requests.  Ex 1. Student verbally says “Hello”.  Ex 2. Student verbally asks “Can you show me my unofficial transcript?” | If the user requested a function. The system will instantiate the functions module.  Ex 1. The user asked for the transcript. The system will activate the transcript module.  If the user said anything that does not involve the modules, then as long as the verbal request is defined in the system's language dictionary, the system will respond accordingly.  Ex 2. The student says “hello”. The system will verbally respond saying “hello, how are you?” |
| **Current Grade Point with Language Recognition API.** | The student will verbally request their current grade point average, utilizing the language API.  Ex 1. “What is my current GPA?”. | The language module will instantiate the GPA calculator module, pull the students past enrollment grades from the database, and output the students current GPA. |
| **What-If Grade Point with Language Recognition API.** | The student will verbally ask what their GPA would be if they got a certain grade in a specific class  Ex 1. “What is my what-if GPA if I get an A in Computer Science 100”. | The system will take in the grade that the user specified for the class they are taking, and add the grade to their current GPA, and then output the new “What-If” GPA for the user.  Ex 1. If the user has a 4.0, and inquiries on receiving a B in their current class. The system will output a “3.75” for their what-if GPA. |
| **Unofficial Transcript with Language Recognition API** | The student will verbally ask for their unofficial transcript.  Ex 1. “Can I see my current unofficial transcript?”. | The student will parse the information and instantiate the unofficial transcript module. The system will output a list of class names, class numbers, grades, and professors that the student has taken in their time in undergrad. |
| **Academic Advisement with Language Recognition API.** | The student will ask the system for course advisement. As well as, specify if they want to include any filters.  Ex 1. “Can you advise me on courses I need to take?”.  The system will give a response and request filter options. It will loop and apply one at a time.  Ex 2. “Please filter for only morning time classes.”  Ex 3. “Please filter for only Computer Science”.  Ex 4. “I want to stay part time, what classes should I take?” | The system will instantiate the advisement module, and then request for filters. Once all filters are received, the system will output a list of classes that fits the students preferences as requested in the students acquisition. |
| **View Course Catalog with Language Recognition API.** | The student will verbally request for the full academic course catalog.  Ex 1. “Can I view the course catalog?”. | The system will produce a full list of all available courses for the semester. This includes the course name, course number, and professor. |
| **View Course Information with Language Recognition API.** | The student can view the course information on any course from the course catalog. The student will verbally request information on the specific class.  Ex 1. “Can I receive more information on the class Computer Science 100”. | The system will parse the sentence and pull information regarding the specified class, it will then output the information to the screen for the student to view. |
| **Logout** | The student will request to logout by verbally stating “logout”. | The system will logout. |

# Modules with objective and testing criteria

| **Test**  **No.** | **Test Objective** | **Completion criteria** | **Date** | **Test priority** |
| --- | --- | --- | --- | --- |
| 1. | Test the functionality of the Login page. | Allowed the user to input their login information, and verified it with the database. | 12/01/22 | High |
| 2. | Test the connection between the schools database and the Login Page. | After a successful login, the user is redirected to the home page and cannot go back to the login page without logging out. | 12/01/22 | Average |
| 3. | Test if after login each user is confined within a session. | Ensured the user is logged in in order to switch between pages, and redirected the user to the login page upon logging out or being inactive for too long. | 12/02/22 | Average |
| 4. | Test the module that allows the user to view their own GPA. | Calculated the GPA of all of a student’s completed courses with a valid letter grade listed. | 12/02/22 | Low |
| 5. | Test the module that allows the user to calculate possible GPA. | Calculated student’s GPA with the addition of courses not yet completed that can be added in at the student’s discretion with a manually inputted letter grade. | 12/03/22 | Low |
| 6. | Test the module that allows the user to view their own unofficial transcript. | Redirects the user to a page that displays all the courses the user has completed in list form. | 12/03/22 | Low |
| 7. | Test the module that is responsible for advising students | The module provided correct advising for what the student should take based on previous courses taken and any specified filters. | 12/07/22 | High |
| 8. | Test the module that allows the user to view the course catalog. | Allowed the user to view all available courses in the Computer Science/ Information Technology department. | 12/04/22 | Average |
| 9. | Test the module that allows the user to view course information. | Allowed the user to view specific information regarding a course. | 12/04/22 | Average |
| 10. | Test the module that allows the system to recognize what the user says. | allowed the system to recognize the input from the user that allowed the user to interact with all the other functions within the system. | 12/04/22 | High |
| 11. | Test the integration of the language recognition module with GPA calculator module. | Allowed the student to verbally request to view their GPA. | 12/05/22 | Low |
| 12. | Test the integration of the language recognition module language recognition with possible GPA calculator module. | The student is able to verbally ask for the student's possible GPA to be calculated. | 12/05/22 | Low |
| 13. | Test the integration of the language recognition module language recognition with transcript viewer module. | The student is able to verbally request to see their unofficial transcript. | 12/05/22 | Average |
| 14. | Test the integration of the language recognition module with viewing all available courses module. | The student is able to verbally request to view all available courses. | 12/05/22 | Average |
| 15. | Test the integration of the language recognition module with the filtering courses module. | The student is able to verbally filter the courses. | 12/05/22 | Average |
| 16. | Test the integration of the language recognition module with student advising module. | The student is able to apply advising filters such as day time, credit amount, preferred classes, and full time or part time. The system will output what classes the student is able to take based on past enrollments. | 12/6/22 | High |
| 17. | Test the integration of the language recognition module with the viewing course catalog module. | The student is able to verbally display the course catalog. | 12/06/22 | Average |
| 18. | Test the integration of the language recognition module with the viewing course information module. | The student is able to verbally request to view the course information. | 12/06/22 | Average |
| 19. | Test logging out. | Ensure the user is able to logout, and not go back into the system without signing back in. | 12/10/22 | High |
| 20 | Final demonstration/Acceptance test. | Demonstrate and test the system in front of management. | 12/12/22 | High |

# Summary of the Module-to-test-technique mapping for the four required testing techniques

The module-to-test-technique mapping will be used for all modules in the system, below is a summary of how the mapping will be implemented, the example is based on the language recognition module. All modules will undergo each testing method and the four techniques included in each method.

| **Testing** | **Walkthroughs** | **Extensive Logic Testing** | **Input/Output Testing** | **Verification** |
| --- | --- | --- | --- | --- |
| Unit Testing | Walk the team through the code for the language recognition module. And allow feedback on the development, if any errors or problems come up. | Build a decision table to list the conditions and the conditions values, as well as the actions and their outputs. This will allow the user to test the language recognition and ensure that it performs as intended in the decision table. | Test the input of the language recognition code, and see what the output is. Using different words or patterns of speech, ensure accuracy is high with interpreting speech. | Ensure the language recognition was developed as per the requirements and works as per the requirements |
| Integration testing | Have the team review and ensure that the code is compatible and performs correctly with other modules. | Build a decision table for the language recognition module interacting with other modules in the system. List the components and actions, and ensure that it is logically accurate and doesn’t encounter fallacies. | With the language recognition module combined with others, test inputs and outputs by verbally stating commands and seeing what the system does. | Ensure that the language recognition component works as intended when integrated with the other models in the system as specified in the specification and requirements documents. |
| Performance evaluation | Each member of the team tests the language module, rating the performance off of: accuracy, speed, bugs, readability, and requirements. | Based off of the decision tables from before, ensure it meets all actions based off of each condition. | Provide a range of inputs for the language recognition module, and review the outputs. Ensure the outputs are correct. | Fully evaluate the module and ensure that it works as specified in the requirements and specification documents. |
| Functional Testing | Check to see if all the features of the language recognition module are debugged and no issues with the interpretation of voice. | Based off the decision table run all the actions, to make sure all work to full functionality | Test the input of the language recognition code, and see what the output is. Using long phrases, patterns of speech, or quick conversations to ensure accuracy is high with interpreting speech. | Fully evaluate the whole module. |

# Dependency graph

**Diagram

Description automatically generated**

**Figure 1**: Dependency graph

In described Dependency graph, some modules in the system are dependent on others to be complete and functioning properly. As illustrated in the dependency graph above. By following the dependency graph the testing process of the system will be easier to integrate.

# Summary of monitoring, reporting, and correcting procedures

After testing, each team member is expected to submit a full report explaining the error found and possible solutions. Once each member of the team completes the testing phase, the team can work together on resolving the bugs found and reinitiating the testing phase to provide a high quality software that is bug free using the below tools and techniques.

The team will be utilizing a web-based application called Trello. It allows for lists to be made with progression indicators. This will allow the team members to notify each other of the progress they have made. Not only that, but lists of bugs could be made and added to anytime. All work that is completed is updated with a green check mark, and indicates who has completed it with their initials embedded.

Trello will allow us to work efficiently and be organized by allowing us to have clear expectations and understandings between all team members. Lists such as: what is still being coded, what has a bug, what can be tested, the type of tests being performed, and what modules are fully implemented and correct as per the specifications requirements documentation. These are all examples of the type of lists that will be implemented in our Trello workspace. A separate list will be utilized to specify which modules need to be corrected and can be picked up by other members who are finished.

# Proposed dates for submission of individual test reports

Below is a table of individual test report due dates in reference to the test numbers stated earlier.

| Team Member | Assignments Due Dates |
| --- | --- |
| Jesse | Test Number 12: 12-4-22  Test Number 17: 12-5-22  Test Number 18: 12-6-22 |
| Omar | Test Number 11: 12-4-22  Test Number 18: 12-6-22  Test Number 19: 12-6-22  Test Number 20: 12-10-22 |
| Allen | Test Number 6: 12-3-22  Test Number 7: 12-3-22  Test Number 10: 12-4-22  Test Number 15: 12-5-22  Test Number 16: 12-5-22 |
| Dan | Test Number 4: 12-2-22  Test Number 5: 12-3-22  Test Number 13: 12-5-22  Test Number 14: 12-5-22 |
| Sumit | Test Number 1: 12-1-22  Test Number 2: 12-1-22  Test Number 3: 12-2-22 |

# Defense for integration

The team's integration plan utilizes a bottom up approach, the team decided to utilize this method because it allows for lower level modules to be produced and slowly implemented.

Modules, such as the login, are low level and allow for the establishment with the database creating a good foundation for the system. As the development of other modules progresses, the team progresses towards higher level functionalities of the system. As the systems architecture progresses towards high level modules, implementation could be difficult. But since the integration of the system started at the bottom, all modules were implemented incrementally. If a high level module creates an error, the team will know that the other modules are not the ones causing the error, instead it's the current module being implemented. This makes it easy for the team to spot where the problem is occurring from.

Once all the individual modules are integrated, then the team will integrate the language recognition module into each of them, as that encompasses the entire system. This will allow for the language module to incrementally be established with each of the other modules, and allow for rigorous testing while being implemented.

## Test cases:

| Test Number 1 | Login |
| --- | --- |
| Test Objective: | To test the login module and make sure it works properly |
| Test Input: | |
| Valid input case | Enter a username and password that match a username and password from the database |
| Invalid input case | Enter a username that is in the database but with an incorrect password |
| Test procedures: | The input is compared with the information in the database |
| Test controls: | The test is run multiple times to cover every test case. |
| Test Expected Output: | |
| Valid input case | The user is redirected to the home page |
| Invalid input case | The user has a message popup that explains that the credentials entered are invalid. |
| Test Result: | |
| Valid input case | The login screen was tested with multiple users, to ensure that each time it logged in as the appropriate user.  The login screen was tested with a user who was not apart of computer science or information technology, denied access   * Pass |
| Invalid input case | The login screen was tested with an incorrect ID and password and the system refused entry.  The login screen was tested with a user who was not apart of computer science or information technology, denied access   * Pass |

| Test Number 2 | User sessions |
| --- | --- |
| Test Objective: | Test the functionality of each logged in user creates a session that allows for their information to be processed until logging off |
| Test Input: | |
| Input case 1 | A successful login |
| Test procedures: | A session is created |
| Test controls: | For each test we create a different session and check if the old session ended properly |
| Test Expected Output: | |
| Input case 1 | the user is redirected to the home page and cannot go back to the login page without logging out, and the user can only view their information and is unable to create another session until logged out. |
| Test Result: | |
| Input case 1 | * Pass |

| Test Number 3 | View users current GPA |
| --- | --- |
| Test Objective: | Test the GPA calculator module |
| Test Input: | |
| Input case 1 | The students grades from the database |
| Test procedures: | Pulls the grades from the table that includes the grades of the previous courses taken by the student |
| Test controls: | The test is ran twice completely after changing the grades in the database to double check the accuracy |
| Test Expected Output: | |
| Input case 1 | For the system to display the GPA of the student accurately |
| Test Result: | |
| Input case 1 | User requests their current GPA from the home screen, system switches to the GPA module and outputs the current GPA of the user.   * Pass |

| Test Number 4 | Calculate What-If GPA |
| --- | --- |
| Test Objective: | Test the What-If GPA module |
| Test Input: | |
| Input case | User give possible grades for each course they are taking |
| Invalid input case | User gives an invalid grade for example (S) |
| Test procedures: | The user speaks the input and its recognized by the language recognition module and is translated to a letter grade |
| Test controls: | The test has to be ran twice completely from start to finish to test both the accuracy of the calculation and the accuracy of the input identification |
| Test Expected Output: | |
| Valid input case | The system should display the possible GPA the user will have if they got the grades spoken |
| Invalid input case | The system should ask the user to repeat the grade until a valid input is given |
| Test Result: | |
| Valid input case | User requests from the home module to calculate what if GPA, system instantiates the what-if module, the user specified how many grades they want to enter, and enters appropriate grade points. The system outputs updated GPA.   * Pass |
| Invalid input case 1 | User requests from the home module to calculate what if GPA, system instantiates the what-if module, the user specified how many grades they want to enter, and enters not-appropriate grade points. The system states grade not recognized and to try again.   * Fail |
| Invalid input case 2 | User tried to request to go back home, system encountered a bug.   * Fail * Team went and modified the code to fix this bug |

| Test Number 5 | View Unofficial Transcript |
| --- | --- |
| Test Objective: | Test the View Unofficial Transcript module |
| Test Input: | |
| Input case 1 | A list of courses the student has completed before from the database |
| Test procedures: | Pulls the previous courses taken by the student from the database. |
| Test controls: | The test is ran twice completely after changing the courses taken by the student from the database to double check the accuracy |
| Test Expected Output: | |
| Input case 1 | For the system to display the Transcript of the student accurately |
| Test Result: | |
| Valid input case | User requests to view their unofficial transcript by saying transcript, the system presented the students unofficial transcript was presented.   * Pass |
| Invalid Input case | User did not say transcript when trying to request for their transcript, the system presented a message saying we could not hear that please try again.   * Pass |

| Test Number 6 | Advising |
| --- | --- |
| Test Objective: | Test the advisement module, should advise students on what courses to take based on past enrollments. |
| Test Input: | |
| Valid input case | Verbally requesting advisement from the system. |
| Invalid input case | The user does not state anything regarding advisement. |
| Test procedures: | The user gives a verbal request for class advisement. |
| Test controls: | The test will be run 2 times to ensure the advisement output is correct each time. |
| Test Expected Output: | |
| Valid input case | Displays a list of courses the student can take based on past enrollments. |
| Invalid input case | Will display an error message stating that the input was invalid. |
| Test Result: | |
| Valid input case | User verbally stated course advisement, the system instantiated the advisement module and the output contains the courses they are able to take based on previous completed enrollments.   * Pass |
| Invalid input case | User does not state anything or states the wrong function. The system does not instantiate the advisement system   * Pass |

| Test Number 7 | View course catalog |
| --- | --- |
| Test Objective: | Test the View course catalog module |
| Test Input: | |
| Input case 1 | All the courses in the computer science and Information technology department from the database |
| Test procedures: | Pulls all the courses in the computer science and IT department from the database and displays them |
| Test controls: | The test is ran twice completely after changing the courses in the database to double check the accuracy |
| Test Expected Output: | |
| Input case 1 | The system displays a list of all courses in the computer science and IT department |
| Test Result: | |
| Valid Input case | The system outputs the full course catalog of courses for the user to view.   * Pass |
| Invalid Input case | The system does not hear the user correctly and asks to try again.   * Pass |

| Test Number 8 | View course information |
| --- | --- |
| Test Objective: | Test the View course information module |
| Test Input: | |
| Input case 1 | The course name that is chosen by the user through speech |
| Test procedures: | The user gives input in the form of speech the language recognition module translates that input and the view course information module pulls the information from the database |
| Test controls: | The test is ran twice completely after changing a courses information on the database side to double check the accuracy |
| Test Expected Output: | |
| Input case 1 | The system displays the information of the chosen course |
| Test Result: | |
| Valid Input case | User request to see the course information for the course they have said to the system, the system shows them a new screen with the information of the course.   * Pass |
| Invalid Input case | User does not say the course name correctly when trying to say the name of the course, system responds with Please try again we could not understand   * Pass |

| Test Number 9 | Language recognition |
| --- | --- |
| Test Objective: | Test the language recognition module and its functionality to redirect the user to other modules |
| Test Input: | |
| Valid input case | User gives a valid input on the main screen for example GPA calculator |
| Invalid input case | User gives invalid input such as a module that does not exist |
| Test procedures: | The user gives input in the form of speech and the language recognition module translates it to input that then redirects the user to the appropriate module |
| Test controls: | The test will be run on from the main home screen to test the accuracy of the module. |
| Test Expected Output: | |
| Valid input case | The user is redirected to the module specified in their input |
| Invalid input case | The user is asked to repeat what they said due to the option not being available |
| Test Result: | |
| Valid input case | The system recognizes the users voice and translates it to a string in the system to parse with all functions.   * Pass |
| Invalid input case | The system does not recognize the users voice and asks to try again.   * Pass |

| Test Number 10 | Language recognition integrated with GPA modules |
| --- | --- |
| Test Objective: | Test the language recognition module integrated with GPA modules, Both viewing the GPA and calculating possible GPA |
| Test Input: | |
| Valid input case 1 | User gives a valid input on the main screen for viewing GPA |
| Valid input case 2 | User gives a valid input on the main screen for possible GPA calculator |
| Invalid input case | User gives invalid input such as a module that does not exist |
| Test procedures: | The user gives input in the form of speech and the language recognition module translates it to input that then redirects the user to the viewing the GPA module or the calculating possible GPA module |
| Test controls: | The test will be run on from the main home screen three times to test the accuracy of the module for each unique case |
| Test Expected Output: | |
| Valid input case 1 | The user is redirected to the viewing GPA page |
| Valid input case 2 | The user is redirected to the calculate possible GPA page |
| Invalid input case | The user is asked to repeat what they said due to the option not being available |
| Test Objective: | Test the language recognition module and its functionality to redirect the user to other modules |
| Test Result: | |
| Valid input case | User verbally requests their current GPA from the home screen, system switches to the GPA module and outputs the current GPA of the user.   * Pass |
| Invalid Input case | User requests their current GPA from the home screen, system does not hear the user correctly and asks to try again.   * Pass |

| Test Number 11 | Language recognition Integrated with Transcript module |
| --- | --- |
| Test Objective: | Test that the language recognition module instantiates the transcript module and outputs the students unofficial transcript. |
| Test Input: | |
| Valid input | Verbally requesting to view the unofficial transcript. |
| Invalid input | Verbally stating nothing to do with the transcript. |
| Test procedures: | The user gives a verbal input to the language recognition module which will redirect the user to the transcript screen and instantiate the module to produce the unofficial transcript from the database. |
| Test controls: | The test will be run from the home screen and redirected to the transcript screen. Where the unofficial transcript will be displayed. |
| Test Expected Output: | |
| Valid input | The transcript is displayed based off information from the database. |
| Invalid input | The transcript is not display, and an error message stating that the command given was not recognized. |
| Test Result: | |
| Valid input | User verbally requests to view their unofficial transcript by saying transcript, the system presented the students unofficial transcript was presented.   * Pass |
| Invalid input | User did not say transcript when trying to request for their transcript, the system presented a message saying we could not hear that please try again.   * Pass |

| Test Number 12 | Language recognition integrated with view course modules |
| --- | --- |
| Test Objective: | The language recognition module will instantiate the view courses modules. These modules include viewing all courses, and also viewing information on individual courses. |
| Test Input: | |
| Valid input | Verbally requesting to view all courses. |
| Valid input | Verbally requesting specific information on a specific course. |
| Invalid input | Not stating anything about the viewing courses. |
| Test procedures: | The user will give a verbal input asking to view all available courses or to view information on a specific course. The language recognition API will instantiate one of the two modules, and then redirect the user to the appropriate screen and display either all courses or specific information on a course depending on what the user requested. |
| Test controls: | This module will be tested twice to cover both modules. |
| Test Expected Output: | |
| Valid input | A list of all courses. |
| Valid input | Information on a specific course requested. |
| Invalid input | An error message stating that the input given was not appropriate and to try again. |
| Test Result: | |
| Valid input | User verbally requests to view the course catalog by saying catalog then tries viewing a specific course by saying the course name, the system presents the course catalog and the specific course information.   * Pass |
| Invalid input | User did not say transcript when trying to request for the course information, the system presented a message saying we could not hear that please try again.   * Pass |

| Test Number 13 | Language recognition integrated with advising modules |
| --- | --- |
| Test Objective: | Test the language recognition with the advising module and filtering. The user should be able to ask for course advisement based off previous classes. |
| Test Input: | |
| Valid input | Verbally requesting advisement. |
| Invalid input | Verbally requesting nothing to do with advisement. |
| Test procedures: | The user will request advisement. |
| Test controls: | This will be run 2 times. |
| Test Expected Output: | |
| Valid input | All classes that can be taken with no filters applied. |
| Valid input | All classes that can be taken with filters applied. |
| Invalid input | An error message stating that what was requested was not understood and to try again. |
| Test Result: | |
| Valid input | The user verbally requests advisement and the system checks past enrollments and outputs advisement based on future courses and removes any that the user has already taken.   * Pass |
| Invalid input | User does not state anything or states the wrong function. The system does not instantiate the advisement system   * Pass |

| Test Number 14 | Language recognition integrated with logging off |
| --- | --- |
| Test Objective: | The user tests that the log off module works with the language recognition. |
| Test Input: | |
| Valid input | The user verbally requests to log off. |
| Invalid input | The user does not verbally request to log off. |
| Test procedures: | The user will verbally request to end their session or log off from the system. The system will redirect back to the login screen ending the current session. |
| Test controls: | This will be tested 5 times with different users, guaranteeing that the system logs off and ends the current session. |
| Test Expected Output: | |
| Valid input | The system redirects to the login screen and ends the session. |
| Invalid input | The system displays an error message stating that the verbal request was not understood and to try again. |
| Test Result: | |
| Valid input | User requests to end session.   * Pass |
| Invalid input | User does not state anything to end session.   * Pass |

# Details of monitoring, reporting, and testing procedures

In order to monitor the progression of all the modules completion, the team is incorporating the usage of a web application named Trello. With the help of this web app, members will be able to monitor other members progress of their assigned modules. Trello allows for task tracking by creating lists and tasks. Members will have tasks assigned, the task will show an indicator that the objective has not been completed.

Once the task has been completed, members can then update the task to complete and store the task within the completed list. This will notify members that the task has been completed. If bugs, or problems occur members can insert comments, and store the tasks within the specified lists. Once tasks are inserted into the completed list, they can then be moved into the testing list for the specified member to test. If bugs occur during testing, the bug should be reported, and then store the task in the bug list until it is updated and fixed. We can also specify the tests that need to be accomplished by inserting the task in multiple lists. If we decide to do three different tests, then we can store the same task in three different lists, and update upon completion.

This will allow for the team to stay organized and up-to-date monitoring of everyone's progression as the project moves forward. If a member is not completing their assigned duties in a timely manner, they will be notified to complete their tasks before the due date.

## Details of individual team member assignments

| **Team Member** | **Assignment** |
| --- | --- |
| Sumit | Test the login page. |
| Sumit | Test the connection between the schools database and the login page. |
| Sumit | Test if after login each user is confined within a session. |
| Dan | Test the module that allows the user to view their own GPA. |
| Dan | Test the module that allows the user to calculate a possible GPA. |
| Dan | Test the integration of the language recognition module with GPA. |
| Dan | Test the integration of the language recognition module language recognition with the “what-if” GPA calculator module. |
| Allen | Test the module that allows the user to view their own unofficial transcript. |
| Allen | Test the module that allows the user to view all available courses. |
| Allen | Test the module that allows the user to view the course catalog. |
| Allen | Test the integration of the language recognition module language recognition with transcript viewer module. |
| Allen | Test the integration of the language recognition module with viewing all available courses module. |
| Jesse | Test the function of filtering courses on the advisement page. |
| Jesse | Test the module that is responsible for advising students. |
| Jesse | Test the module that allows the system to recognize what the user says. |
| Jesse | Test the integration of the language recognition module with the filtering courses module. |
| Jesse | Test the integration of the language recognition module with the student advising module. |
| Omar | Test the module that allows the user to view course information. |
| Omar | Test the integration of the language recognition module with the viewing course catalog module. |
| Omar | Test the integration of the language recognition module with the viewing course information module. |
| Omar | Ensure the user is able to logout, and not go back into the system without signing back in. |