Software Testing Plan

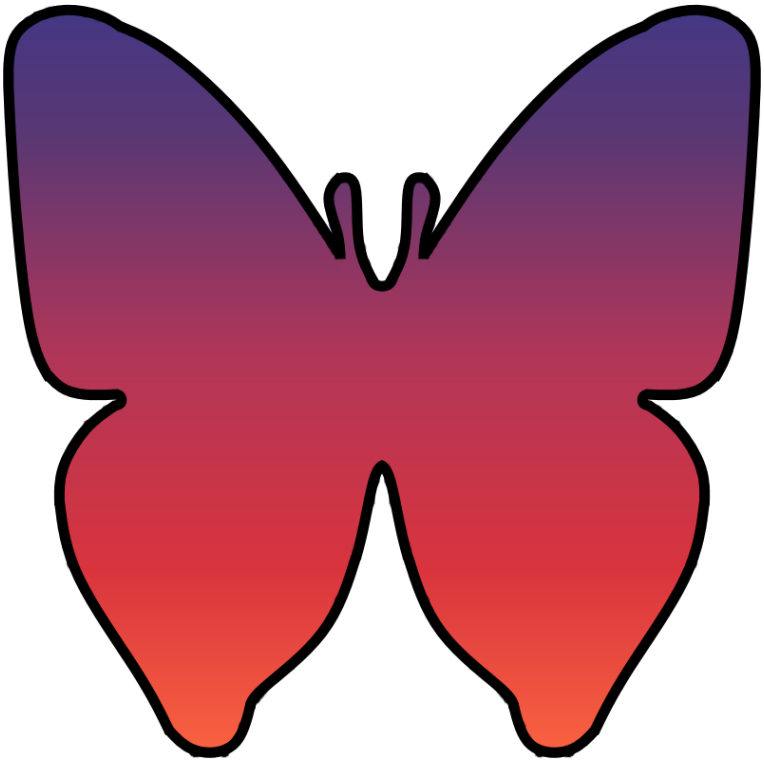
Team Shining Sky

Rosze Voronin, Skyler Hanson, Ashleea Holloway, Logan O'Donnell

March 31st, 2022 - Version 1.0

Sponsored by: Dr. Morgan Vigil-Hayes

Mentored by: Felicity Escarzaga



1. Introduction

Rural Native Americans struggle daily with mental health, with youth being susceptible to rising rates of depression, anxiety, and behavioral issues. The client, Dr. Morgan Vigil-Hayes, aims to address this problem for the Hopi community through development of the ARORA project at her research lab, CANIS Lab. Currently, the ARORA project consists of a youth mental wellness mobile app and accompanying server. The Hopi Tribe is currently creating a Community Mentorship program that will connect community mentors with Hopi youth mentees that they would like to tie into the ARORA project, but the program still needs a software tool to aid in communication and organization that the existing ARORA app does not provide. The developers of team Shining Sky have been working with Dr. Vigil-Hayes to create the start of the ARORA Community Mentor Portal mobile app that provides the software support that is missing for the Hopi mentorship program. The goal of this initial stage of the project is to produce a strong starting point for future development, which Dr. Vigil-Hayes has already displayed as part of her application for a 2.5 million dollar research grant that she hopes to see funding from around July. The Community Mentor Portal app currently consists of a version of the project that is entirely self-contained, providing features like viewing of mentee mood reports and information, appointment scheduling, a chat interface, and a separate view for supervisors. CANIS Lab's future development will focus on integrating it with the rest of the ARORA project, and writing network code that takes advantage of CANIS Lab's specialty in limited online connectivity networks, reflecting the rural area the app is targeted towards use in.

Before moving on to introducing the software testing plan for the Community Mentor Portal app, it is helpful to briefly define software testing. Software testing is the process of systematically exploring a developed software product to look for bugs, usability issues, or any other problems that would merit a change. It is part of the process of polishing a software before release, since software that is hard to navigate or has glitches becomes less useful due to being unpleasant to use. Generally, software testing focuses on two things. First, testing the code; unit testing and integration testing allow developers to ensure that all functionality works as intended behind the scenes through producing some extra output that allows confirmation that all the code is doing what it is intended to do. Second, testing the usage of the app itself; usability and user testing allow developers to identify aspects of the software that may be unpleasant, confusing, or otherwise negatively impact the experience that a typical user would have.

Our proposed testing plan is a mix of unit testing, integration testing, and usability testing. Since our app is self-contained due to the scope of the project, we elected to perform our integration and unit testing manually as it would not be prohibitively complex to do so. To carry out our unit testing, we systematically tested each feature, mostly as we developed them, verifying that all the features performed consistently under repeated use and properly stored all data. Our integration testing was carried out in much the same way, focusing on navigating smoothly between various screens and ensuring that all data accessed in multiple portions of the app would display and update consistently. Our usability testing is what we focused the most on. Since the scope of the project makes our app a starting point for future development, the details of features and functionality may be changed slightly later on, however, the basic design and usability of the app is much harder to alter. Due to this, we decided that we definitely needed to ensure that the basic platform was usable and parsable to users, so that future development could focus on, as intended, making features more robust, rather than working with the basic design principles. Our plan for usability testing is to do in-depth testing with a representative sample of users, expert testing with CANIS Lab, and make an attempt to get the app into the hands of real users from the Hopi community for additional feedback. We believe that our testing plan, within the context of the Community Mentor Portal app project, ensures both a reasonable standard of functionality as well as close checking that the design of the app will provide a usable platform for future development by ensuring that user concerns are taken care of before CANIS Lab progresses further into developing the network code to support the app. The following sections of this document present our testing plan in detail.

2. Unit Testing

Unit testing is a way of verifying that the individual components of a larger project work as intended. It is a necessary part of development because individual components must work correctly before they can all be put together into a larger project. The goal of unit testing is to verify correct functionality before developers move on to subsequent tasks. Finding problems with a component early allows for quick fixing before it is woven into the larger project. If a problem is only discovered at the end of development, significant refactoring may be required, which costs more time and money than catching issues early.

To do this testing with our project, we manually verified that every unit worked before we moved on in development. Since our project has straightforward units and is self-contained, we decided to test them manually as they were completed. This lowered the risk of significant problems down the road and allowed for quicker problem solving if that were to occur. While sections weren't completed from the outer-most facing parts inward, it is easier to conceptualize when describing our testing that way. Furthermore, the user can perform different tasks based on their role. If they are a supervisor they can, for example, make the codes that mentor-level users need to make their accounts. Therefore, we had to ensure that we tested all functionality for both the normal user level and the supervisor user level.

The first step the user must complete is making their account. This includes providing their desired username, email address, and password. When completed, the information gets securely stored in the local database. To test that this unit worked, we entered test information into the text boxes and verified that it was being sent to the database correctly. Immediately after this step, the user then uses these credentials to log in to their account. We needed to again verify that the information they entered was correctly checked against the database.

The next units tested were the various functional pages. These pages serve different purposes and have different actions the user can perform. The first to be tested was the profile page. On this page, the user can change their email, password, and log out. If the user is a supervisor, they can generate the access code that mentors need. Testing each of these actions worked the same as the login page, by verifying that all information was correctly stored and accessed. We repeatedly changed the email and password, verifying that it was stored correctly.

Generating the access code and logging out were also verified by repetition and verification of results.

The next page is for anonymous questions. The questions are displayed and the user is allowed to type a response and submit it. Furthermore, the mentor can flag any question for review which then shows it on the supervisor's view of questions. When the response is submitted, it should be displayed under the question and labeled “Response: …”. We confirmed that this happened every time by submitting several responses for every question. We also flagged several of them and viewed them from a supervisor's account to ensure that flagging was stored properly.

The third was the home page, which is the most complicated page. On the top level of this page, the mentor’s mentees are shown as well as if they have been flagged for supervisor attention or not. Then when a mentee is selected, the user is brought down a layer to the page for that mentee. On this page, the mentor can see the mentee’s previous mood reports, chat with the mentee, jump to the calendar, or flag the mentee. The messaging and calendar buttons need to bring the user out of the home page and to the appropriate places. Testing of this page consisted of verifying that when each of these actions were performed, the results were appropriate. We repeatedly flagged several mentees and observed the updated flag status on the outer home page. Then we, again and again, moved to the different pages from inside a mentee’s page.

The fourth page is for messaging. On this page, the user selects which mentee they would like to message and can then send their message. This page needed to bring the user to the correct mentee selected every time to prevent messages from being sent to the wrong person. Furthermore, the messages needed to be shown as being sent the same as when sending a text message, and needed to be persistently stored. Every time we selected a mentee to message, we were brought to the right recipient, time and time again. Then we sent various messages to every mentee and made sure they were updated and displayed correctly.

The last page tested was the calendar. This page is meant to help the user keep track of events or meetings they may have scheduled. The user enters a date, time, and description; they then press submit which updates the calendar and shows their event. To test this unit we made an assortment of different events, each with different dates, times, and descriptions.

Every time we did, we verified the information and ensured that it was persistent. The user can delete events which we tested as well.

The last unit tested was the database. The previous units all rely on the database to store information, meaning testing this unit was very important. Since all of the different units had not been stitched together yet, all of the information stored had to be created manually. We would create fake mentees, access codes, users, etc. We then made sure that the type of data to be stored was correct and that the unit sending it only allowed that type. For example, when entering a date in the calendar, it should be in a typical date format. So we verified that the calendar page only allowed for that type of information to be sent and that the database was capable of receiving it and storing it persistently.

Once we knew that every unit on its own worked properly, we needed to put all of the pieces together and make sure they communicated correctly between themselves through integration testing.

3. Integration Testing

While unit testing is checking that the different parts work by themselves, they also need to work correctly together too. Integration testing is a way of checking and proving that the different parts of a whole software project are working together as intended. In this case, we are testing that the different modules of our software all work cohesively and transfer data properly amongst themselves. The goal of integration testing is to ensure that the project works as a cohesive whole. Everyone tackles problems differently, and with several developers working on a single project, sometimes different solutions don't work together as easily as might be hoped. We need to check that the individual modules communicate with each other correctly. By proving that everything is working together as intended, we lower the chances of the project crashing in unforeseen ways or losing data.

To do so, we are systematically working through every feature of our product and checking that information is being correctly passed between different modules that need it. For example, when someone makes an account, the information they enter needs to be given to the database and stored correctly so that it persists upon reopening the app. Then, when they try to log in, the information they enter needs to be checked against what is in the database. If our product does either of those parts incorrectly, the user won't be able to log in.

We started testing from the outermost layer and worked our way inward. We tested login functionalities, then moved to the first pages the user sees. The user can immediately navigate to different pages, so we tested that they are brought to the right pages and information is shown accordingly and in line with what should be stored in the database. We continued this process until every function of the app had been tested. Table 3.1 below shows the various main components we tested and the current status of testing for each.

Table 3.1 Integration Testing Plans

Test	Summary	Status
Account Creation	Creating and logging into an account.	Complete
Navigation	Navigating between pages via the navigation bar and buttons.	Complete
Profile	Actions found on the profile page.	Complete
Questions	Viewing and answering questions.	Complete
Home Page	Selecting mentees and viewing their pages.	Complete
Messaging	Messaging a selected mentee.	In-Progress
Calendar	Creating and modifying events.	In-Progress

The first steps every user must complete are creating and logging into their account. Supervisor accounts are made by an administrator so no access code is necessary. The supervisors can then create the codes necessary for mentors to create their accounts. We tested that everything needed to log in was appropriately sent to and from the database. If the credentials were correct, the user would be brought to the home page. Below is a layout of the specific functionality we tested.

Account Creation:

- Supervisor:
    - Logging in to the account
    - Creating access code for mentor
- Mentor:
    - Using access code to create account
    - Logging into the account

Once arriving at the home page, the user has a navigation bar at the bottom of their screen which allows them to move to the various pages of the app. Furthermore, there are buttons nested within the mentee's pages for convenient navigation to their messaging page. For example, if a mentor wants to view a mentee's mood report they select the mentee by name, which brings them to their page. From there if they wanted to message the mentee, instead of moving to messages, then the mentee's messages, they could move straight there via a button on the mentee page. Each of these functions were tested repeatedly to ensure proper navigation through the app, outlined in the list below.

Navigation:

- Navigating to pages via the navigation bar
- Navigating to pages from within other pages

The profile page varies slightly depending on if the user is a supervisor or mentor. The actions they have in common are changing their associated email address and password. Testing these meant confirming that the information entered was what was being sent to the database to be updated and stored persistently. In the case of the password, we then verified that it would allow the user to log in. If the user is a supervisor, they can additionally create access codes for mentor account creation. We simply verified that the access codes generated were being stored on the database and could be used to create an account. An outline of what functions were tested during this process is listed below.

Profile:

- Changing email address
- Changing password
- Creating access code (Supervisor)

The question page also varies depending on the role of the user. The mentors view all of the questions and respond to them. If they decide the question should be reviewed, they flag it for a supervisor. A supervisor user only sees flagged questions on their questions page. Testing to make sure the question page was integrated properly consisted of creating many sample questions. We then answered them several times and flagged multiple of them. We then verified that the flagged questions were seen from the supervisor's perspective and that the supervisor could add in an answer. The tested functionalities are listed below.

Questions:

- Answering a question
- Flagging a question
- Viewing and answering flagged questions (Supervisor)

The home page allows mentors to view their mentees and whether or not they have been flagged. They can then select a given mentee and view their page. A mentee page shows their previous mood reports, current risk, and has buttons to jump to chat, calendar, and flag the mentee. If the user is a supervisor, they can view their assigned mentors, then each mentor's mentees. To test this page we made several sample mentees, each with different mood reports. We viewed them from a mentor account and went to each of their pages, traveling between their messages and the calendar repeatedly. We would flag and unflag the mentee multiple times, checking for the updated marker on the home page. With the mood reports we created, we also verified that the correct calculated risk was being displayed. We also made sure that as a supervisor, the user could select a mentor and then their mentees. The functionalities covered by this section of testing are listed below.

Home Page:

- Selecting a mentor (Supervisor)
- Selecting a mentee
- Flagging a mentee
- Viewing their mood reports

Messaging is another page that varies between users. We needed to make sure that when a mentor selects a mentee and messages them, the message is sent and stored appropriately. If messages weren't sent, or worse, sent to the wrong mentee, there could be

considerable problems both in terms of functionality and privacy. We checked that every time a mentor messages a mentee, it is sent and delivered correctly. Supervisors are allowed to view all messages sent between mentors and mentees for transparency and safety. We also verified that after a message was sent from a mentor account, it was observable from a supervisor account. The functionalities tested during this process are listed below.

Messaging:

- Selecting a mentee
- Messaging the selected mentee
- Viewing messages of mentors (Supervisor)

The calendar page is the same for both mentors and supervisors. They can create, edit, and delete events. We tested this section by performing those actions repeatedly and checking for their existence in the database. The functionalities tested by that process are listed below.

Calendar:

- Adding an event
- Editing an event
- Deleting an event

4. Usability Testing

While the previous two sections focus on the code, usability testing looks at the user experience. Usability testing is software testing that focuses on the interaction between users (any human operator of software) and the software itself. Generally, the goal of this type of testing is to examine whether or not users can understand and operate the software in question effectively. It also looks at factors such as how a less tech-savvy user (ie, not a developer) experiences the software, and whether or not someone who is picking it up sight-unseen with little to no instructions can successfully use the software. Usability testing is needed to ensure that software is not just functional, but usable; it does not matter if the software works or not on a technical level if users are unable to make it work for its intended purpose. On an abstract level, usability testing is usually conducted by simply giving the software to a sample set of users and allowing them to test it out with limited guidance and then give feedback on their experience. This section will outline the plan for usability testing of the ARORA Community Mentor Portal app in its current stage of development.

Several factors are important to consider when designing usability testing for the Mentor Portal app. As the Mentor Portal app is a mobile app, the assumption has to be made that a user can use a smartphone and has some general experience with using mobile apps. This does not impact the design of the testing plan much, aside from the clear need to test the app with people who are smartphone users. Since most or all users will be from a rural area, it is important to keep in mind that smartphone users might not be as savvy as the urban average, so the testing plan takes this potentiality into account by not assuming more than basic mobile app using skills. The client advised that the majority of mentors will be in the 18-25 age range, however, some supervisors could be as old as 60, which provides another reason to err on the side of caution and test for lower smartphone skill levels. The app will be used as part of a community mentorship program with defined procedures, so it is also important to note that the target user for the app will likely have some idea of its functionality and what they are supposed to use it for, as opposed to an app downloaded from the app store which may be used with much less prior knowledge. This has a stronger impact on the testing plan, as users will be part of a structure in which they will receive at least some instruction, so this needs to be simulated during testing. Table 4.1 provides a summarization of our plan for usability testing, and the following three subsections describe each item in greater detail.

Table 4.1 Usability Testing Plans

Test	Summary	Status
Sample User Studies	Test the app with a representative user sample to get close feedback on usability via simulated tasks and open-ended questions.	In-Progress
Expert Reviews	Test the app with CANIS Lab in order to give space for tweaks/suggestions before handoff and ensure consistency with ARORA aesthetics.	In-Progress
Real User Testing	Test the app with real users from the Hopi community to get feedback from people who will be involved in the mentorship program.	Planned

4.1 Sample User Studies

These studies will involve testing the app with at least six sample users. Sample users will be selected based on criteria that should make them roughly similar to the real users of the app. Namely, they should be familiar with the operation of typical mobile applications, and pre-prepared with some general instruction on the purpose of the app. A developer will provide the user with a brief explanation of what the app is for, and a brief list of vague tasks (for example: “Send a mentee a chat message.” or “Check a mentee’s mood report.”) in order to simulate goals that would exist during typical usage of the app in the community mentorship program. Users will then be asked to complete the tasks as they see fit, and talk through their thought process as they do so. At the end of the test, users will be asked about what they liked and disliked about the app, as well as anything that they might want to suggest, add, or change. This open-ended portion of the feedback aims to discover any parts of the application that users see as unpleasant or problematic, but might not want to directly denounce in front of the person who made it. Developers will then discuss the feedback of all the users amongst themselves to see if any problems were repeatedly mentioned.

The advantage of this portion of usability testing is that developers have a lot of control over it. Meeting in person with the users and talking to them directly about their feedback can provide more concrete insight. Logistically, picking sample users in this way is also the least complex, and since the criteria are generally applicable, it can be assumed that the chosen users are a fair sample of what a typical user might experience. Dr. Vigil-Hayes has assured us that she knows a few people on the upper end of our age range that she can likely bring into our testing, so we should be able to get a sample group that mostly represents our target userbase. The main challenge to this portion of the testing plan is that the users are not a pure representation of the people who will be using the app, only an approximate simulation. However, feedback on usability and navigability should be universally applicable, so this portion of usability testing will focus on those two things and try to glean as much as possible from closer interaction with this group of test users. The timeline for this part of the testing should see it completed mostly or fully by the time of the UGrads presentation so that usability concerns can be addressed in time to help the demo. Currently, we are working on gathering our test users so that we can get this portion of the testing started.

4.2 Expert Reviews

These studies will involve helping CANIS Lab set up the app on a few test devices and allowing them to experiment with it as they see fit. Since the researchers at CANIS Lab will be continuing the development of the Mentor Portal app after it is handed over to them, and since they are more familiar with the typical users of the ARORA project, gathering some of their feedback will likely provide helpful insight. While users in this case will not be given a guided test, they will be asked to provide feedback on anything they notice, as well as compare the aesthetics and functionality to the rest of the ARORA project. This information will be taken into account as part of the final adjustment process before the handoff.

The advantage of this portion of usability testing is that CANIS Lab researchers are extra familiar with the ARORA project, and can provide feedback on making the Mentor Portal app more in line with the rest of the system. This process is of medium logistical complexity, as it does require coordinating some busy schedules. To get around that, a looser methodology that relies mostly on feedback in general with attention paid to a few particular asking points seems appropriate. As members of CANIS Lab are considered experts as far as the app is concerned, it can be reasonably assumed that they can be allowed to explore it as they see fit. Feedback

from this section is not as critical in terms of usability, but will be very useful in making small aesthetic or functional tweaks, and can help discover any small changes that were not thought of but would improve the app noticeably. In addition, it allows for space to make a few suggested changes before the final handoff in order to increase client satisfaction. The timeline for this test group expects testing to be finished around UGrads or shortly after so that there is time to make any desired edits. Currently, we are working with Dr. Vigil-Hayes to get our app running on test phones in CANIS Lab so that this testing can begin.

4.3 Real User Testing

These studies will involve getting the Mentor Portal app into the hands of some of the people who would be using it as part of the Hopi community mentor program. This has a clear benefit to the future of the project, however, distance and difficulty of communication prevent it from being easy to undertake. As CANIS Lab and Dr. Vigil-Hayes have better connections to the project, the best course of action is to go through them, and Dr. Vigil-Hayes has communicated that once CANIS Lab is able to get test versions of the app running, she will take it to her Hopi community contact and set up some usability testing.

Logistically, this type of testing is very challenging, so it will not be relied on. Based on conversations with Dr. Vigil-Hayes, it will take several weeks to get the app into the hands of users on the Hopi reservation. The decision was made to not expect to receive any feedback for this part of the testing in time for final handoff, but to get it started anyways because CANIS Lab will likely need the feedback one way or another if they are to take over the future development of the project and can benefit from developer support on how to get it running and what kinds of things could benefit from more testing. Therefore, the timeline of this part of the usability testing will be essentially down to external factors, and so will be worked on as much as possible before the final handoff, with CANIS Lab handling the rest and doing what they will with the feedback on the app in its handoff state. As it stands right now, we have discussed a plan with Dr. Vigil-Hayes, targeted some potential dates in mid-April, and are looking at ways we could provide the test remotely (via a combination of Zoom and TeamViewer is the current plan) in order to make participation as easy as possible to increase our chances of being able to utilize this feedback before the handoff.

5. Conclusion

The ARORA Community Mentor Portal app in its current state is a self-contained version of CANIS Lab's idea for providing software support to the Hopi community mentor program as part of their larger scale work with them. CANIS Lab plans to use what we at Team Shining Sky developed as a platform for their continued work on the project; we focused on the app's design and functionality, while they will focus on building the network code to support it in the low-connectivity environment of the Hopi reservation. Our testing plan consists of manual unit and integration testing, focusing on verifying that our functionality in its self-contained state will work correctly and smoothly moving into future development, as well as extensive usability testing on three user groups (sample users, expert users, and real users) that ensures that the design portion of the app will not require much more work than a few tweaks and maybe some nicer art assets.

Team Shining Sky is currently working to carry out our testing plan. We performed some of our unit and integration testing during development, making sure to verify the functionality of major features as we created them. We are currently performing another round of code testing for good measure, focusing on items we did not test as extensively in the past. Our usability testing is still in its early stages, however, we are progressing with testing a representative sample of users and testing with CANIS Lab, while constructing plans for testing with real users from the Hopi mentorship program with Dr. Vigil-Hayes' help. We are confident that our testing will verify that the Community Mentor Portal app represents a solid platform for CANIS Lab to integrate into the greater ARORA project upon successfully receiving funding, and that we will be able to hand off a great product at the end of April.