ANXIETY CONNECTION:

A MOBILE APPLICATION TO AID TEACHERS WORKING WITH STUDENTS WITH ANXIETY

by

VERONICA REEVES-VOELTNER

_____

A Thesis Submitted to the Honors College
In Partial Fulfillment of the Bachelors Degree
With Honors in
Information Science Technology and Arts

The University of Arizona

May 2018

Approved by:

_____

Richard Thompson

School of Information

Abstract:

The goal of this project is to create a mobile application that is a resource for professors and other instructors when working with students with anxiety. The application provides streamline communication between students, professors, and affiliated mental health professionals. It also allows for personalization to individual students' needs, providing resources for instructors when working with a student. These include a manual of interaction techniques that an instructor could access quickly and simply, and a venue for discrete communication between student and faculty. The project is not yet fully implemented. Substantial progress has been made on back-end support for the application including user and document databases, database support, and document retrieval and storage. Due to roadblocks in multi-platform application, front-end development did not progress as far. Code for mobile application and web application views have been started, but connection to the back-end have yet to be completed. This paper explores these topics in greater depth and the existing codebase is provided.

Table of Contents

Introduction:

These days we have apps for social media, for gaming, even for speed dating. Others have taken advantage of the mobile platform to create applications with the intent to help those with mental and emotional disorders cope and organize. But a review of these applications show that there is a hole in the services provided. Specifically, there are not many centralized resources available for the teachers and professors who work with those with anxiety. Brenda Reeves, a licensed social worker in Tucson Unified School District, mentioned in an interview with me that often times the resources a teacher has are in the form of paper print outs or links to websites. This is not an efficient way of interacting with someone with anxiety during everyday communications, and especially not in emergency situations. This leads to both the student and the teacher suffering more than necessary. This has been an unresolved problem for many years. Often the teacher ends up redirecting the student to counselors or school social workers when they could solve the problem themselves with the proper tools.[1] Another issue is that those with anxiety also often have problems asking for help. For instance, someone with social anxiety will sit quietly needing help, but will not ask any questions because they are afraid of the reaction from their peers. If there was a way to remotely, and discretely signal a teacher of a student's need, it would greatly help. My project is mobile app that compiles information, step-by-step instructions, and calming resources on anxiety for teachers or professors to use in their classrooms, as well as allowing the students a discrete method of communication with their teachers in times of need.

[1] Reeves, Brenda LCSW. Personal Interview. 20 Feb. 2017.

I hoped to create a quick reference/resource compilation for teachers to be able to easily search and use. My application was to be a compilation of information and tutorials for professors to use when they need to interact with students with an anxiety disorder and, more specifically, with a student who may be having a meltdown. The platforms the app is made for are iOS, Android, and Windows. It has a browser component as well for organization and customization of the resources. It will include quick reference guides, detailed explanations, and calming techniques as seen in other common anxiety control applications. Another functionality is a connection with the student. It would allow the student to discretely inform the professor of larger issues or request help without including the other students.

This application is coded using HTML, SQL, CSS, JavaScript, and PHP. I chose to use these web development languages because of the Adobe PhoneGap framework. This framework allows the use of web development to be used to create mobile applications that can be used on multiple platforms. The original plan for this application was for it to be written entirely for iOS using objective-C. However the use of HTML as the base for my application allows for wider use as well as a web application, and ease of use when organizing or adding resources.

Usage

Protection of Student Data/User Hierarchy

One of the main considerations while creating this application was the protection of student data while allowing for the sharing of resources between users. Because the application deals with specific data on students' health records, that data should only be accessible by the parties directly involved in that student's education. This is facilitated

through the SQL database. Users are classified by school networks and then further classified by role and classroom group. Documents are kept track of using the database as well; location of origin as well as users accessing a document are kept track of in the SQL database.

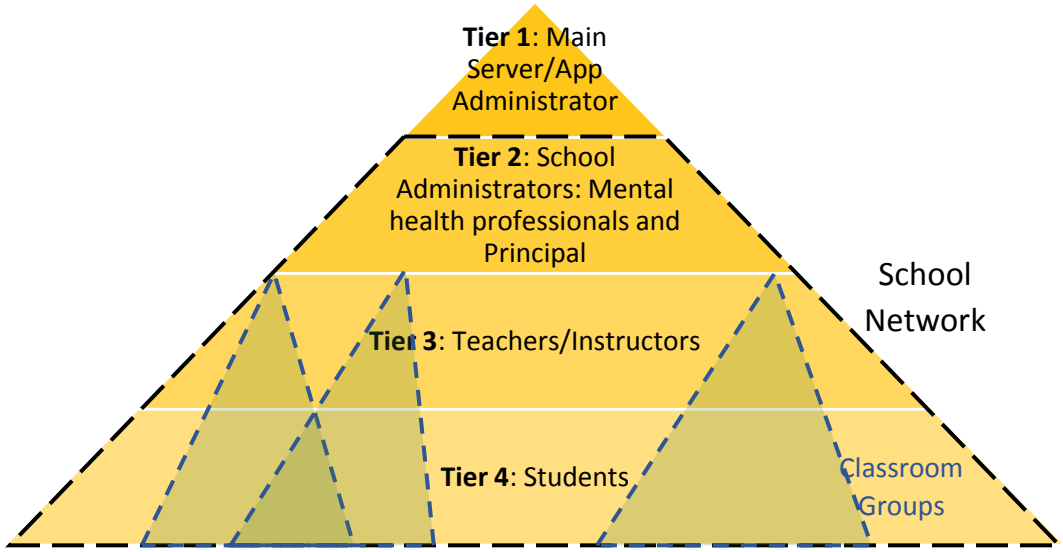

Figure 1: Layout of user hierarchy in one school network.

The tier 1 is the application administrator. This individual or group of individuals has control over the adding and removing of school networks and oversees the main database of resources and sharing of resources. In order to address the concern of student privacy, this tier does not have direct access to any student information, only school administrator information. There is direct communication between users in this tier and users in other tiers, but only when initiated by users in the other tier. This allows other users to report problems and ask questions while remaining disconnected from the main administrator. Main administrators have the ability to communicate with other users indirectly through notifications sent to all users or all users in a given tier.

The next tier is the school administrator. These are individuals such as school social workers, psychologists, counselors, and principals. In general, individuals in this

tier have add and remove privileges in all three tiers of the school network, as well as control over which classroom groups students are a part of. This way school administrators are in control of who is a part of their school network on the application. This is also the tier that directly interfaces with the main server to upload and download resources. This is done through the web application version of Anxiety Connection. This tier has direct access to all student information in their network so they can add resources to specific student accounts or class groups. This tier also has the ability to enroll in multiple school networks since mental health professionals often work multiple schools.

The next tier is the instructors. This tier is broken down into class groups which consist of a single instructor and any students in their classes that are enrolled in the school network. This keeps student data compartmentalized to only the instructors that work with them directly. This tier does not have add and remove privileges over any other tier, but they can communicate with other instructors and school administrators as well as students in their classroom group through the application. They can access resources provided by tier 2 users but cannot directly add resources or access the web application.

The final tier is the students. This tier is the most restricted. Users in this tier only have access to their own information and resources assigned to them by tier 2 users. They can communicate with instructors in their classroom groups and all system administrators. They do not have access to the web application and as such cannot upload new resources. Students can be a part of multiple classroom groups as students often have multiple instructors.

Registration

When registering a new account, a user provides their name, email (school affiliated for tier 2 and 3 users), the tier they are enrolling in, and school network they wish to join. If the school network does not exist yet, a request will have to be sent to have the school network created. This request can only be sent by a tier 2 user. The request includes the school name, address, school district, primary phone number for the front office, and principal email. There is then verification of the identity of the individual requesting the new network by contacting the school or school district directly. When the school network is verified and created, the person who sent the request is notified via email. When a new user is requesting access to a school network, all tier 2 users are notified and one of those users has to verify the new user in order for them to be added to the network.

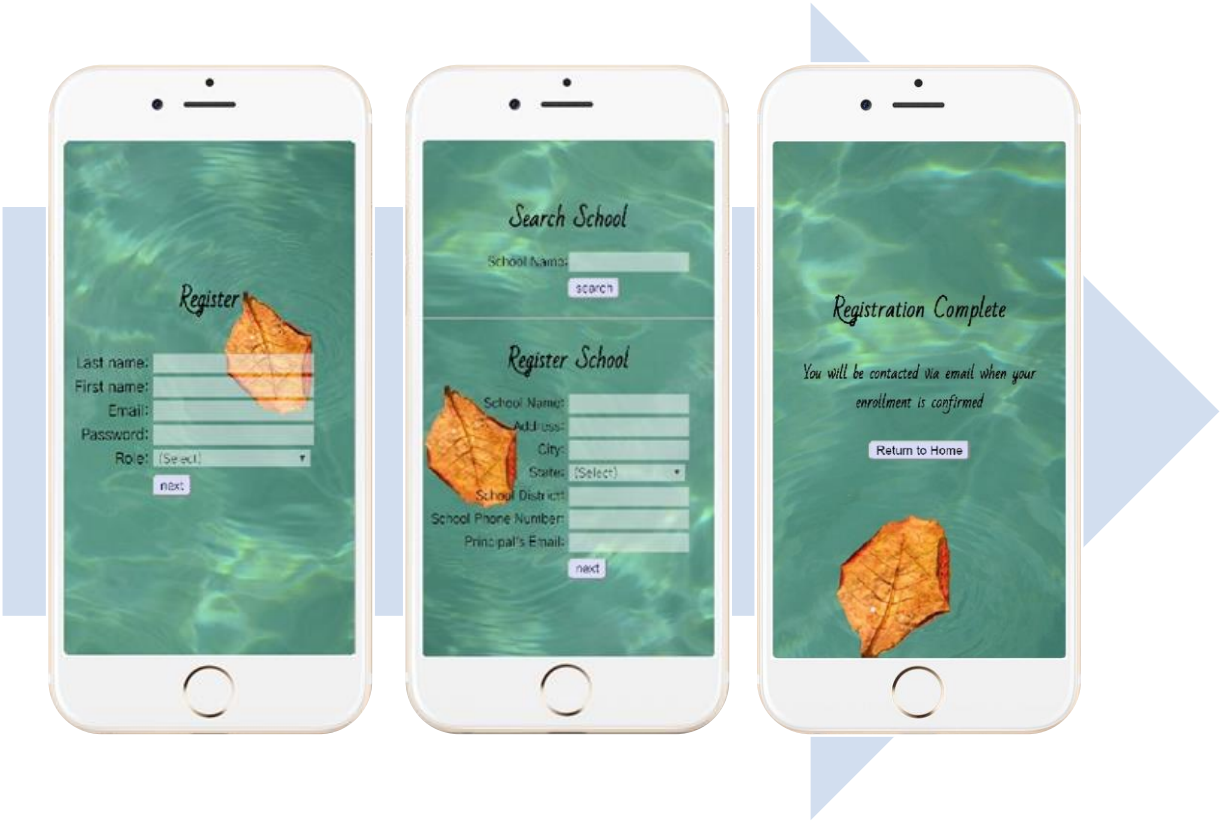

Figure 2: New user registration process.

Document Management

Document management is done primarily through the web application to allow ease of document uploading. From the web application, tier 2 users can search through all the documents present on the main server, assign different documents to instructors, students, or classroom groups, and upload new documents. All documents are stored on the main server. This allows for documents to be accessible even if the computer of origin is offline, deleted the file, or is unreachable for any other reason. Document information such as location on the main server, file type, computer of origin, and locations of use are all stored in the SQL database.

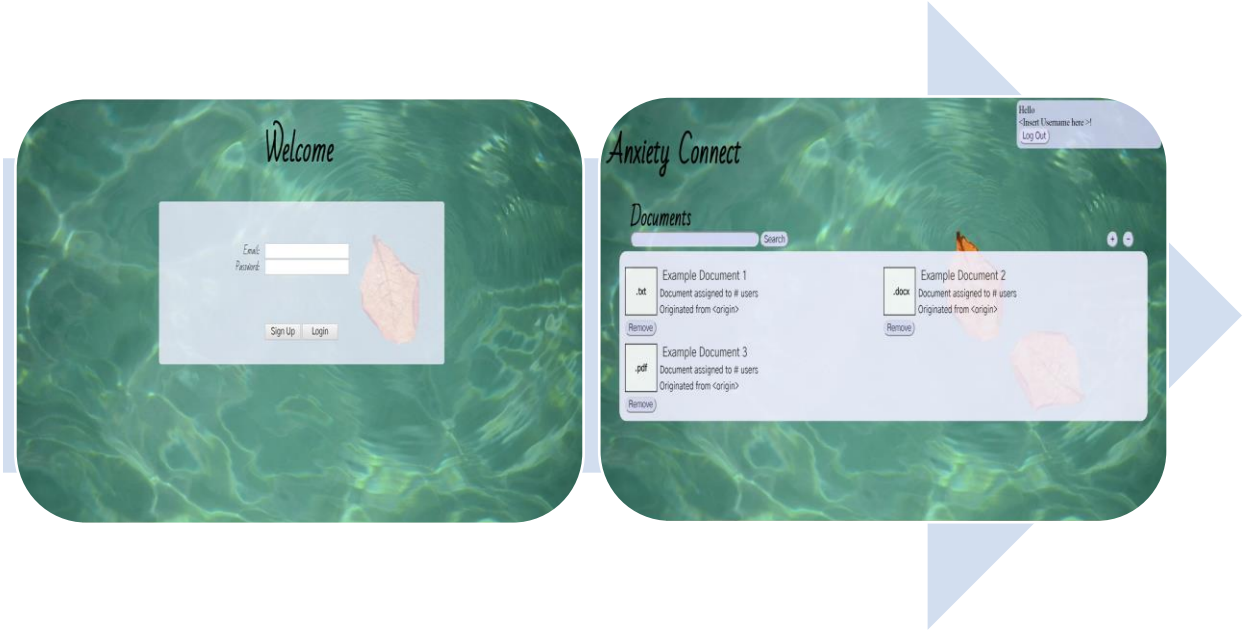

Figure 3: Web application view.

Use Cases

Basic Flow (Mobile Application):

{User Authentication}

1. User enters their email and password.
2. Main Server receives user log in request and looks up the email.
3. Main Server verifies the hash of the password matches the one on file.
4. Main server looks up the role of the user and loads the corresponding layout.
5. Main server starts a new session.

{User Actions}

(Student View Documents)
(Student Contact Teacher)
(Instructor View Documents)
(School Administrator Verify User)
(School Administrator Create Class Group)

{User Log Out}

6. User clicks the log out button.
7. Main server end the user's session
8. Main server loads the login page.

Extend Basic Flow:

Student View Documents:

1. Student opens resource tab
2. Main server queries database for student specific resources.
3. File information is retrieved and loaded into the documents page.
4. Documents page is loaded.
5. Student selects a document to view.

6. Main server locates file on main file server.
7. Main server displays content based on file type.

Student Contact Teacher:

1. Student opens instructors tab.
2. Main server looks up classroom groups student is in
3. Main server loads Instructor data into the page
4. Student selects an instructor to contact
5. Main server loads communication page
6. Student chooses message to send to instructor
7. Main server locates instructor's mailbox in the database and deposits student's message.
8. Main server sends instructor a notification.
9. Instructor receives a notification and responds to student.

Instructor View Documents:

1. Instructor opens resource tab.
2. Main server queries database for affiliated document information.
3. Main server loads all document information into the documents page.
4. Instructor chooses a resource.
5. Main server locates file on main file server.
6. Main server displays content based on file type.

School Administrator Verify User:

1. School Administrator clicks on users tab.
2. Main server queries database for school network users and displays on users page.
3. Main server queries database for school network users waiting to be verified and displays them on users page.
4. School administrator clicks accept on user verification.
5. Main server sets new user's status to verified.

6. New user is notified.

School Administrator Class Groups:

1. School Administrator clicks on Class Groups tab.
2. Main server queries database for all class groups in the school network.
3. Main server displays class groups with number of students and affiliated instructors.
   (Add Class Group)
4. School Administrator chooses a class group.
5. Main server queries database for all students and instructors in class group.
6. Main server displays user information on class page.
   (Add User to Class Group)
   (Remove User from Class Group)

Add User to Class Group:

1. School Administrator click the (+) button.
2. Main server queries database for all users in the school network but not in the class group and loads into *user pool*.
   (Search for User)
3. Main server displays user information.
4. School Administrator selects a user.
5. Main server adds the user to the class group.
6. User is notified.

Remove User from Class Group:

1. School Administrator clicks the (—) button.
2. Main server queries the database for all users in the class group and loads into *user pool*.
   (Search for User)
3. Main server displays user information.
4. School Administrator selects a user.

5. Main server removes the user from the class group.
6. User is notified.

Search for User:

1. User clicks on search bar.
2. User enters name or email of user to search for.
3. Main server scans *user pool* for searched user.
4. The user information matching the given parameters is displayed.

Alternatives:

Instructor View Documents:

1a: Instructor looks up student specific document

1a1. Instructor clicks on student tab.

1a2. Main server queries database for instructor's classroom group.

1a3. Main server queries database for students in classroom group and loads into *user pool*.

1a4. Main server displays student data on students page. **(Search for User)**

1a5. Instructor selects a student.

1a6. Main server queries database for student's affiliated document information.

1a7. Main server displays student information and affiliated documents on student page.

System Administrator Verify User:

4a. School administrator denies new user

4a1. School administrator clicks deny on user verification.

5a. User was denied

5a1. Main server removes new user from school network.

Remove User from Class Group:

5a. User is only instructor in class group.

5a1. Main server removes the user from the class group.

5a2. Main server queries database for all users in class group.

5a3. Main server notifies all users in class group of class group deletion.

5a4. Main server notifies all school administrators of class group deletion.

5a4. Main server deletes class group.

End of Basic Flow (Mobile Application

Basic Flow (Web Application):

{User Authentication}

1. User enters their email and password.
2. Main Server receives user log in request and looks up the email.
3. Main Server verifies the hash of the password matches the one on file.
4. Main Server loads document management page.
5. Main server starts a new session.

{User Actions}

(Add Document)

(Search Document)

{User Log Out}

6. User clicks the log out button.
7. Main server end the user's session
8. Main server loads the login page.

Extend Basic Flow (Web Application):

Add Document:

1. User clicks on (+) button.
2. File explorer opens.
3. User navigates to file to upload.
4. Main server copies file contents.
5. Main server adds file to database.

Search Document:

1. User clicks on search bar.
2. User enters title, keyword, or origin of document.
3. Main server queries database for any documents with matching parameters.
4. Main server loads documents to document page.
5. User clicks on document.
6. Main server loads contents of file and displays them based on file type.
7. User click download.
8. Main server adds connection between user and document.

Alternatives:

4a. User not authorized.

4a1. Main server loads error screen.

4a2. Main server logs user out.

End of Basic Flow (Web Application)

Alternative Flow: Registration:

1. User clicks sign up button.
2. First registration pages is loaded.
3. User enters first name, last name, email, password, and role.

4. Main server verifies given information.
5. Second registration page is loaded.
6. User enters school name, location, or district.
7. Main server queries database for all schools matching the given parameters.
8. User selects school.
9. User is added to school network as unverified.
10. All School Administrators are notified of a new user request.
11. Confirmation screen is displayed.

Alternatives:

4a. Password format in correct

4a1. User is prompted to re-enter password with correct format.

6a. User registers new school network.

6a1. User enters school name, location, phone number, and principal email.

6a2. Main server logs a new school network request to be handled by application administrators.

6a3. Confirmation screen displayed.

End of Registration Flow

Deficiencies and Further Work

Functionality Improvements

The framework being used to adapt html to mobile platforms, PhoneGap, uses html pages as templates for screen views. I was unable to load my html templates into the PhoneGap framework while maintaining the formatting and back-end integration needed for proper application behavior. Further work is needed on the understanding of this software in order for me to create a functioning application. The back-end, while mostly complete, still has some bugs. Because of the complicated nature of the relationships between users and the storage of documents, some queries were not fully polished. Further debugging is need specifically in document handling and message sending. Finally, an adaptor is needed to link the main file server to the different school networks. Options for this include the Node.js server or an external java server. While the Nod.js server would be the simplest option, creating a java server would allow for more customization.

Security

Right now there are some basic security measures in place. The user hierarchy protects user information among every-day users, basic input sanitization protects against SQL injection, and password requirements and hashing gives a layer of protection over user accounts. However, because of the sensitive nature of the data dealt with in this application, further security measures would be needed to protect against malicious attacks. This security would include encrypting of user data and the messages sent between them, obfuscation of the current code, and more through verifying of the users requesting access to the application. This was beyond the scope of this project as I was focusing on the building of an application on multiple platforms, not application security.

Code Statistics

Due to the possibility of future publishing, the exact code will not be included in this write up. However, the current code statistics are as follows:

Number of Files	19
HTML	6
CSS	5
JavaScript	2
PHP	2
SQL	4
Lines of Code	1193
HTML	382
CSS	348
JavaScript	118
PHP	194
SQL	151