Design Document

[App for Youth Patients in Dialectical Behavior Therapy (DBT) Groups]

Gaveshini Sriyananda 10146093
Yue Cai 10168332
Selena Sun 10152968
Alex Huctwith 10138307

November 19, 2018

CISC 498
Instructor: Prof. Mohammad Zulkernine
Supervisor: Prof. Robin W. Dawes
TA: Abu Faisal
School of Computing
Queen's University

Table of Contents

External Design Document

Introduction

Dialectical Behavior Therapy (DBT) is a cognitive-behavioral psychotherapy to treat people with borderline personality disorder. Borderline personality disorder consists of ongoing patterns of varying behavior and moods. However, therapists identify the difficulty of engaging youth patients in DBT groups as a barrier to treatment. Furthermore, youth patients' unwillingness to speak on the phone or communicate through email is a challenge for the therapists. Alternatives such as text messaging are too personal for use in the therapy context. In this project, we are aiming to create a solution to these problems by creating an app that would allow better communication and enable therapists to provide patients with a set of helpful resources. This app will push out messages/notifications to each DBT group member with DBT information, group starting dates or group cancellations, as well as reminders to practice skills. It will facilitate communication of information between therapists and patients as some youth patients do not check email often.

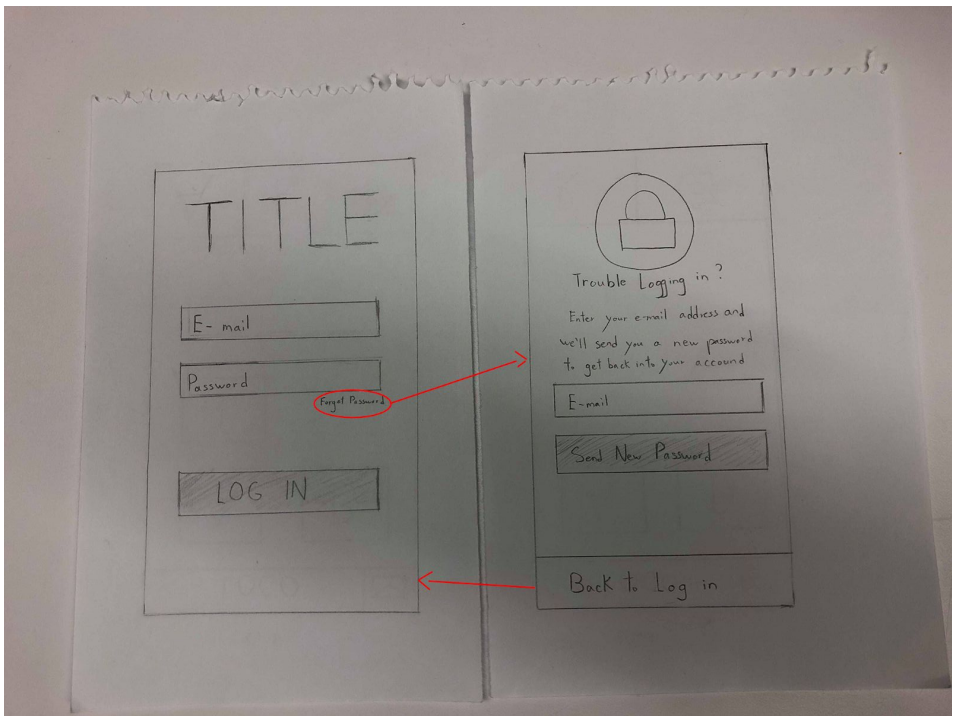
User Interface Design

For this application, there are unique user inputs and lot of interactions happening, for instance, pressing on specific features on content bar and interacting with different windows.

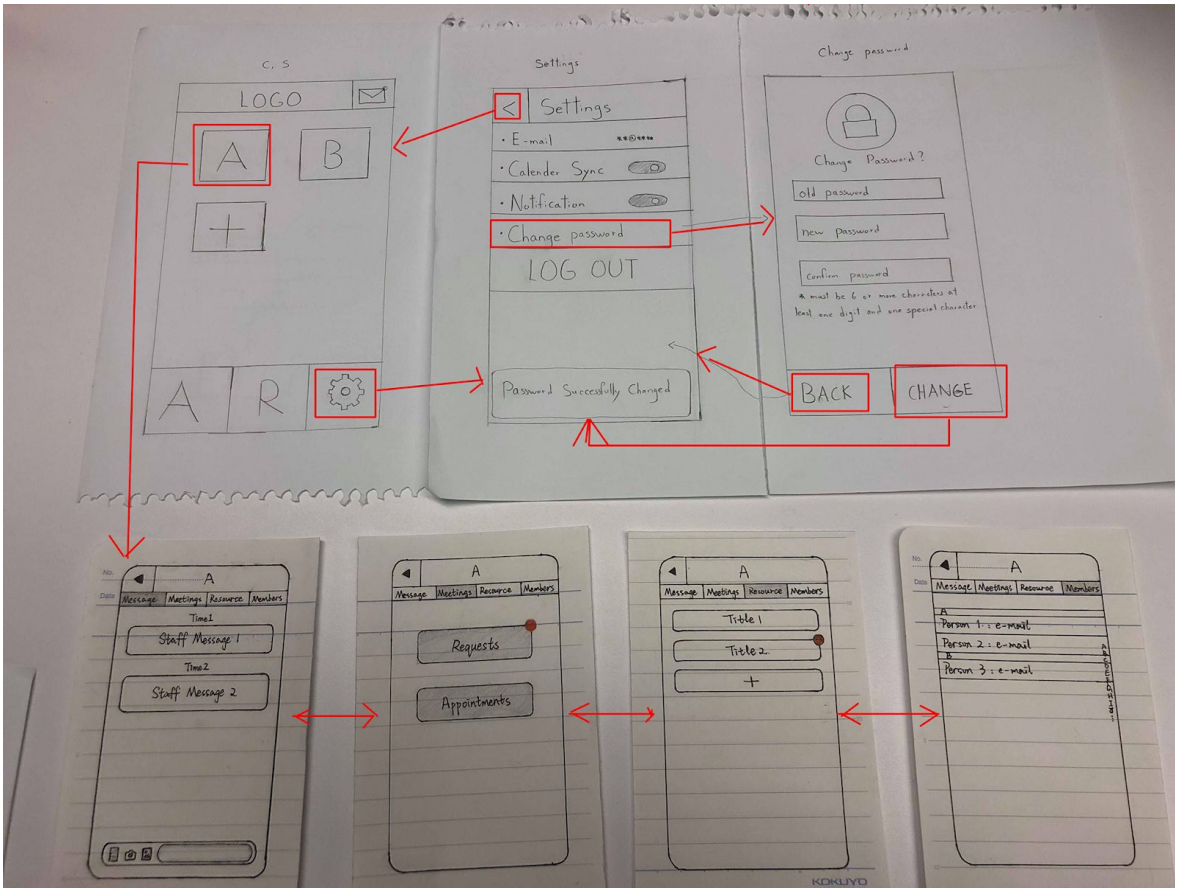
User inputs:

- Email address and password to login to the application
- Old password, new password and confirm password to change a password
- Date, time and reason for adding/modifying appointments
- Adding group names
- Adding resources: title of the resource and its content
- Each individual message sent

User Interactions:



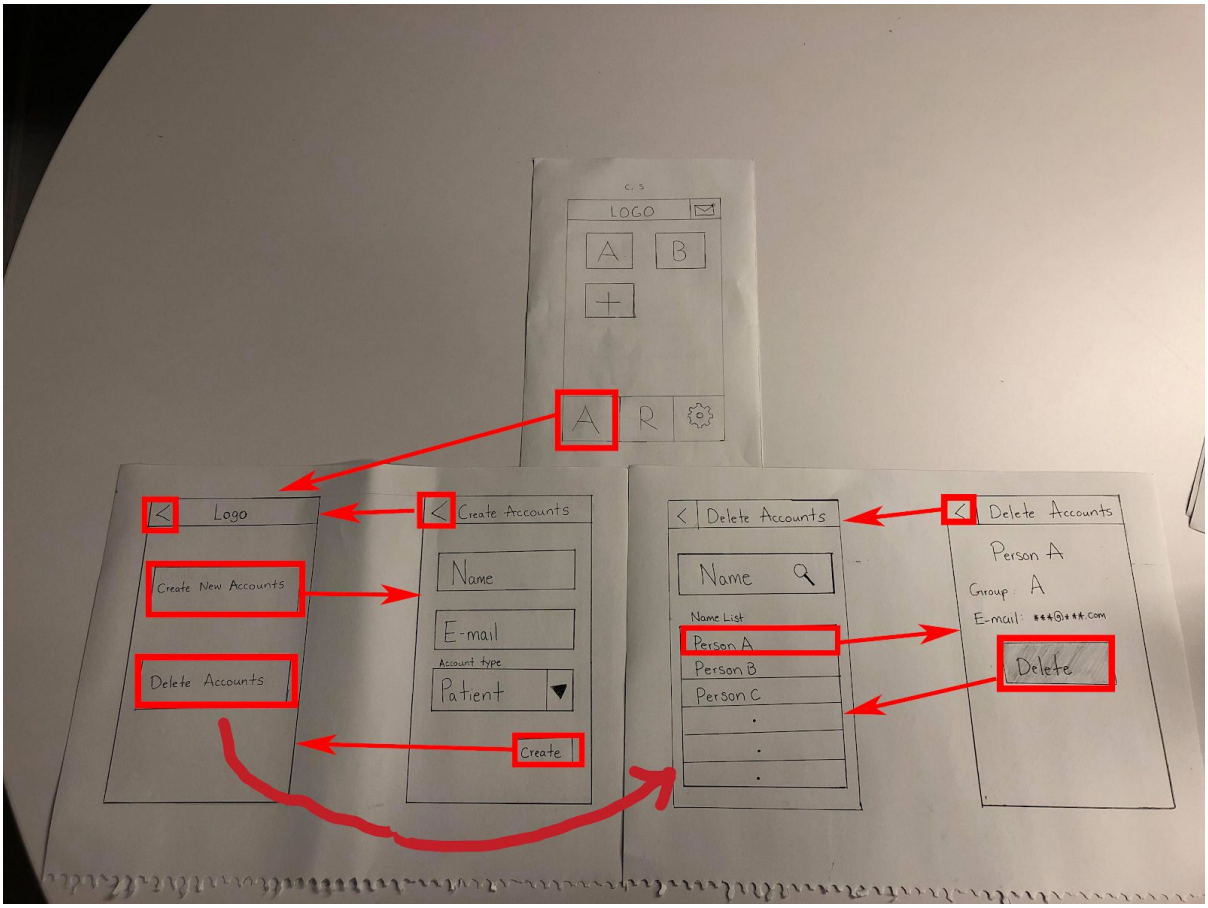
All users login with an email address and password. If password is forgotten, users press “forgot password” on the title page. This leads to a page that lets them enter their email address to reset the password. They get an email with a link to reset the password. Once they press on the link, “change password” window appears with options: “new password,” and “confirm password.” Users type their new password and confirm password into the textbox, thereby resetting their old password.



All users can go to settings by pressing the gear button at the bottom-right of the main page. In the setting page, users are able to turn on/off the calendar sync and/or the notification. By pressing “Change password”, change password page appears. Users type in “old password,” “new password,” “confirm password” and click on CHANGE button to change their password. The BACK button leads to the Setting page. All users can access general resources area. To check a specific resource, users go to “R” on main page which leads to a page with list of resources. Staff/coordinator can access accounts by pressing on “A” in main page. This action leads to a page allowing them to create/delete accounts. Once the staff/coordinator is in delete accounts page, by holding on the button with the group name, a red deleting dot will appear and they are able to delete accounts by pressing on it. To add a new group, they can press the button with the plus sign in the create accounts page.

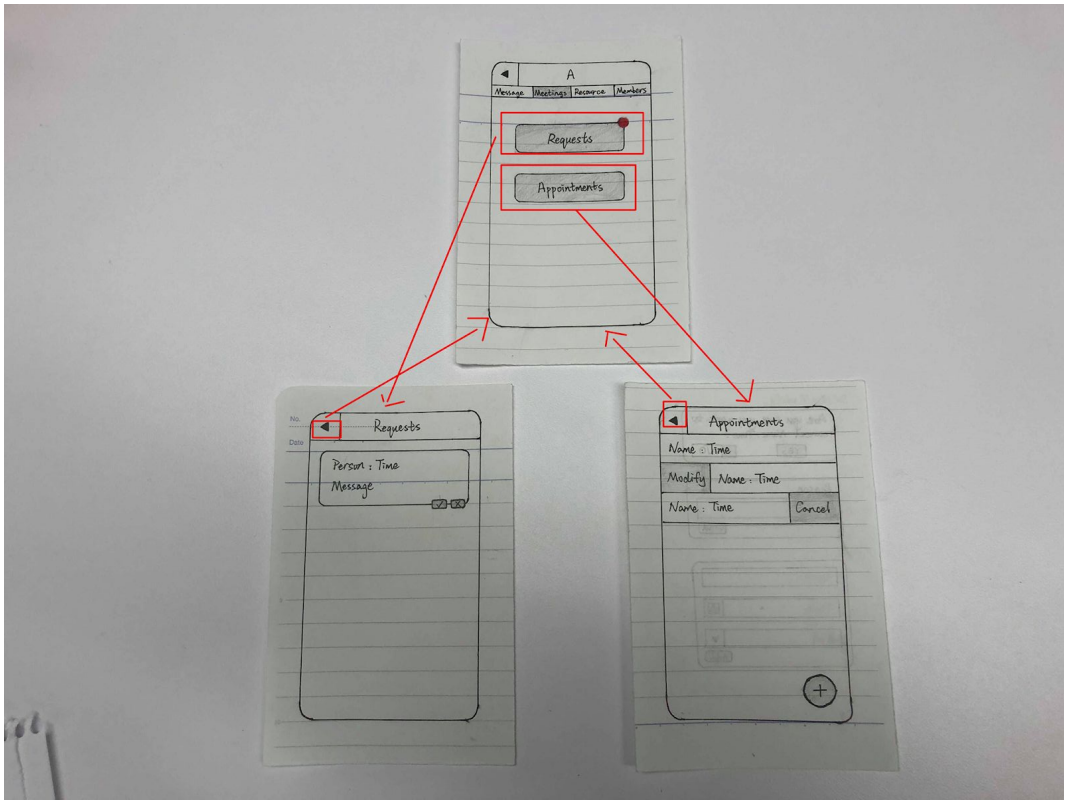
By pressing on the group name(i.e Group A), staff and coordinator go to the message page with a tab bar. The tab bar has four options: message, meetings, resources and members. Each will lead to one page. Every page contains a back arrow which leads to the main page. In the message page, they are able to type in any messages using the text area at the bottom. After they click on

the send icon, the text will show above. They are also able to send files and images by clicking on the icons beside the text area. In the meeting page, the two buttons in the middle will lead them to the request page and appointments page, respectively. If the user get a new request from the patients, there will be a red dot at the top-right of the “Requests” button. In the resource page, there will be a list of the resources, where every list is clickable. In addition, by holding on the title, a red delete button will show at the top-right of the title button and users are able to delete the resource by clicking on it. The members page contains the list of the patients which are enrolled into the group. Every list item is clickable. Staff and coordinator can also find the QR code of the group by clicking on the add button at the bottom-right of the page.

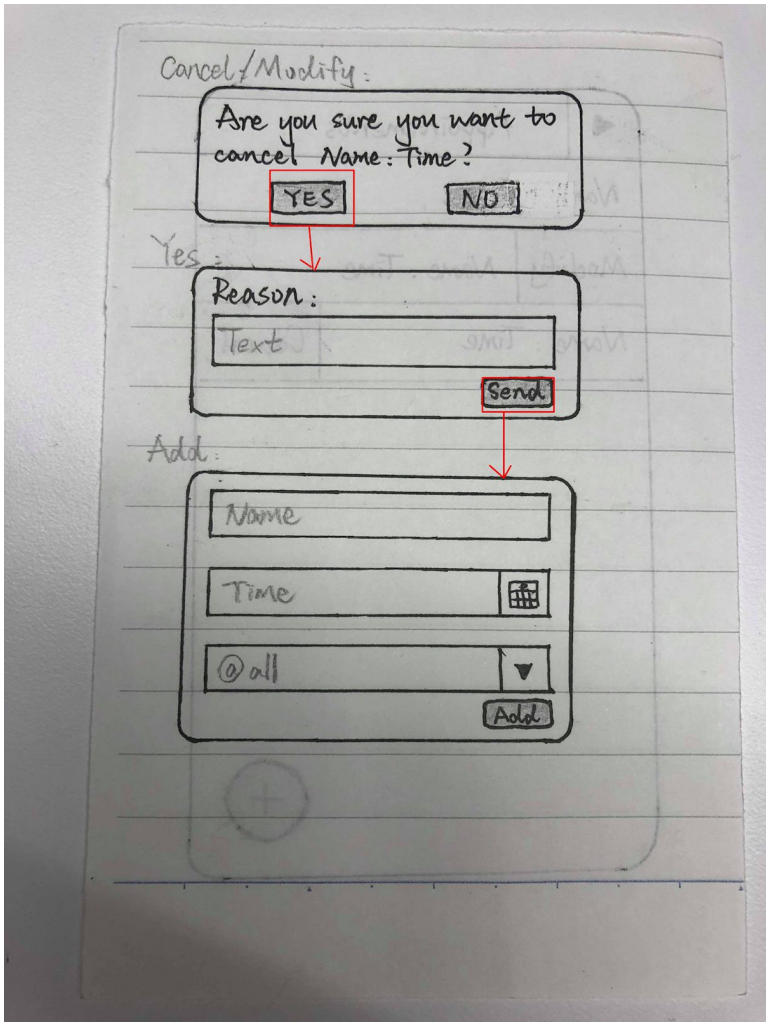


By clicking on the “A” button, staff and coordinator go to the accounts page. In the accounts page, there are two buttons which lead them to the creating new accounts page and deleting accounts page respectively. In the create accounts page, they need to type in the real name and the e-mail address of the user. They also need to select an account type through the combo box. In the delete account page, name of the patients (and name of the staff in the coordinator’s

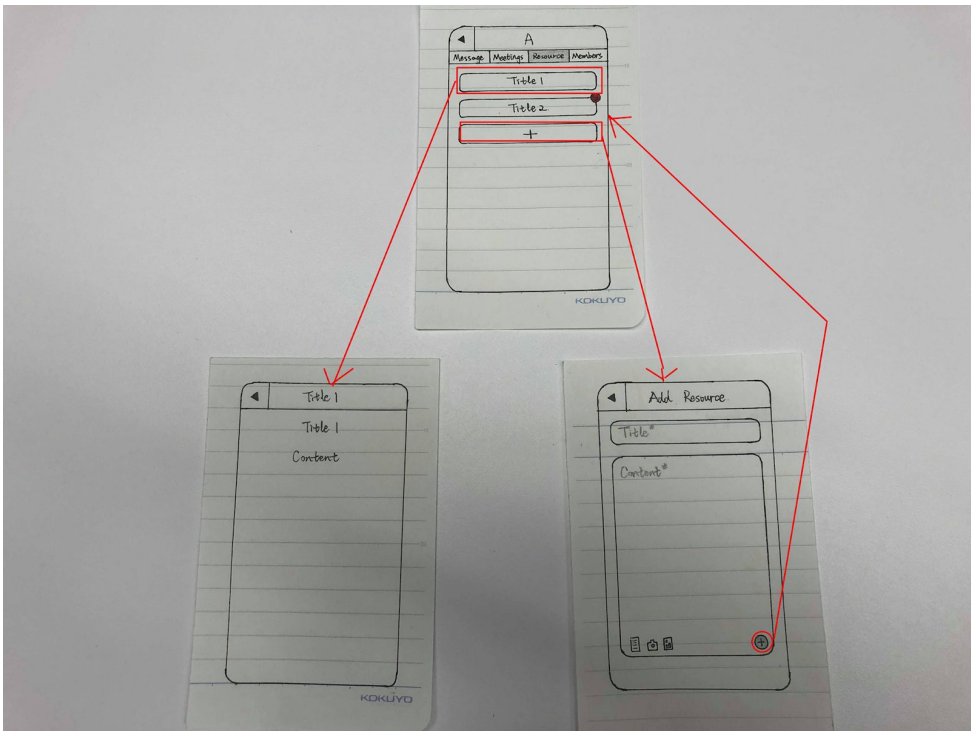
interface) are shown as a list under the search box. If they click on a person's name(i.e. Person A), they will go to a page containing Person A's name, group and e-mail address. If they want to delete the user, they simply click on the delete button.



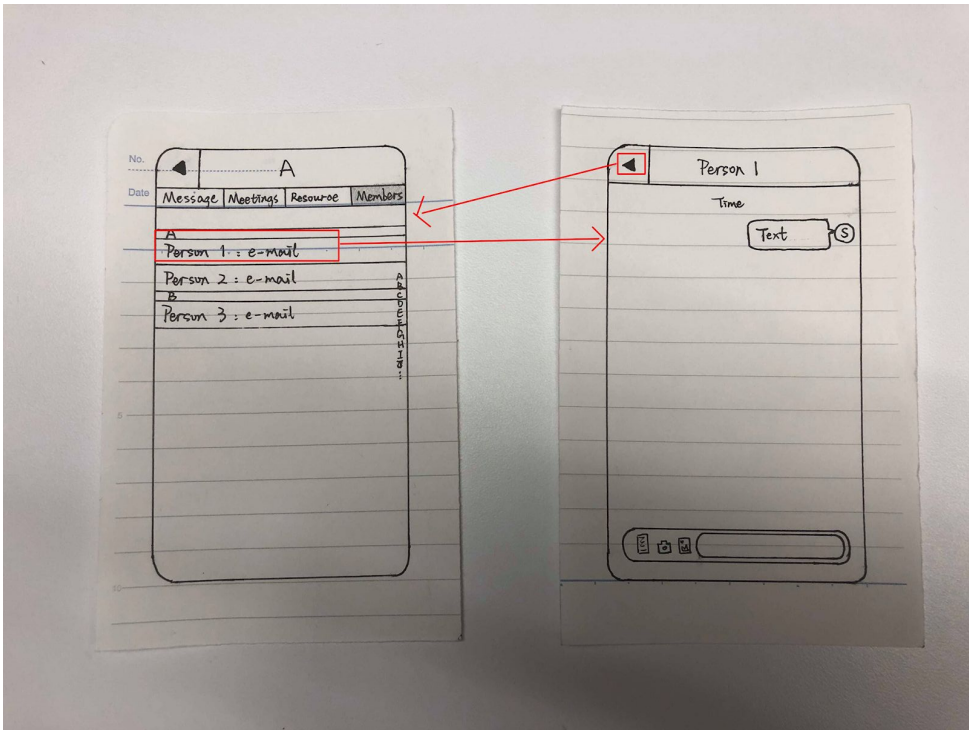
In the meetings page, Staff and Coordinators can go to the requests page or the appointments page by pressing on the button in the middle. Inside the requests page, new requests from the patients will show as lists of blocks. By pressing on the check mark/cross, they are able to confirm/cancel the request and it will disappear after that. In the appointment page, They are able to modify any appointment by swiping left and clicking on the Modify button. They are able to cancel it by swiping right and clicking on the Cancel button. By pressing the add button at the bottom-right, they are able to add any new meetings.



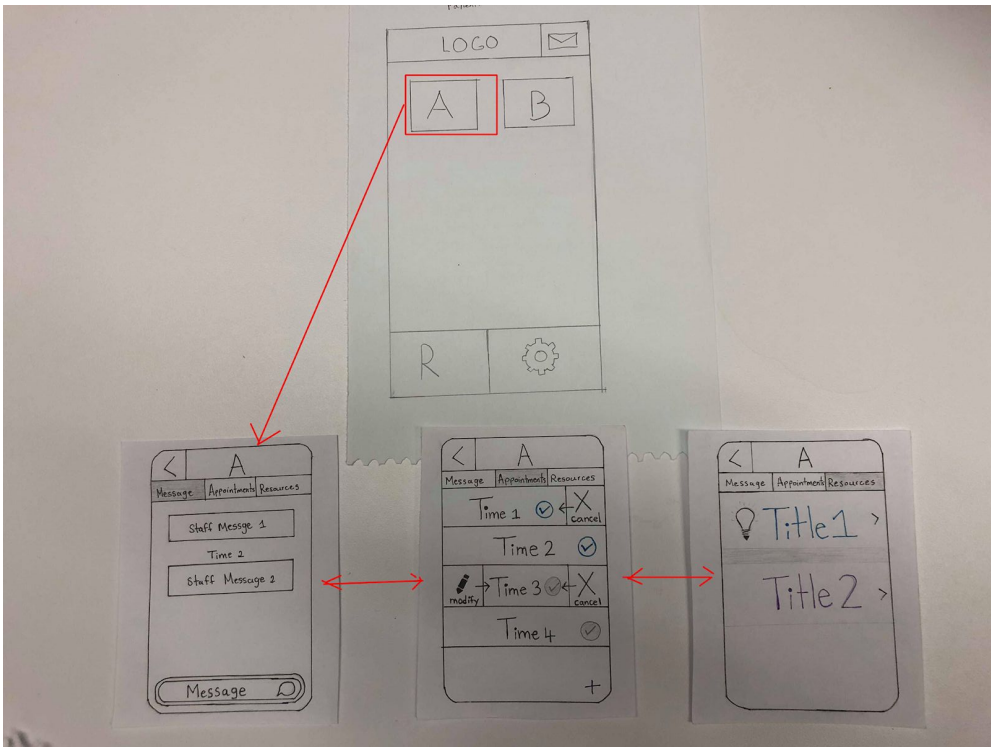
A toast with the confirmation message is shown if they cancel or modify an appointment. Clicking on the NO button will lead them to the original page. Clicking on the YES button will show another toast, asking the user to enter a reason. If they choose to modify an appointment, after they click on the Send button, a name, time and the person they want to meet need to be selected/entered. If they click on the Add button, they will go back to the appointments page with a toast showing that meeting has been successfully modified.



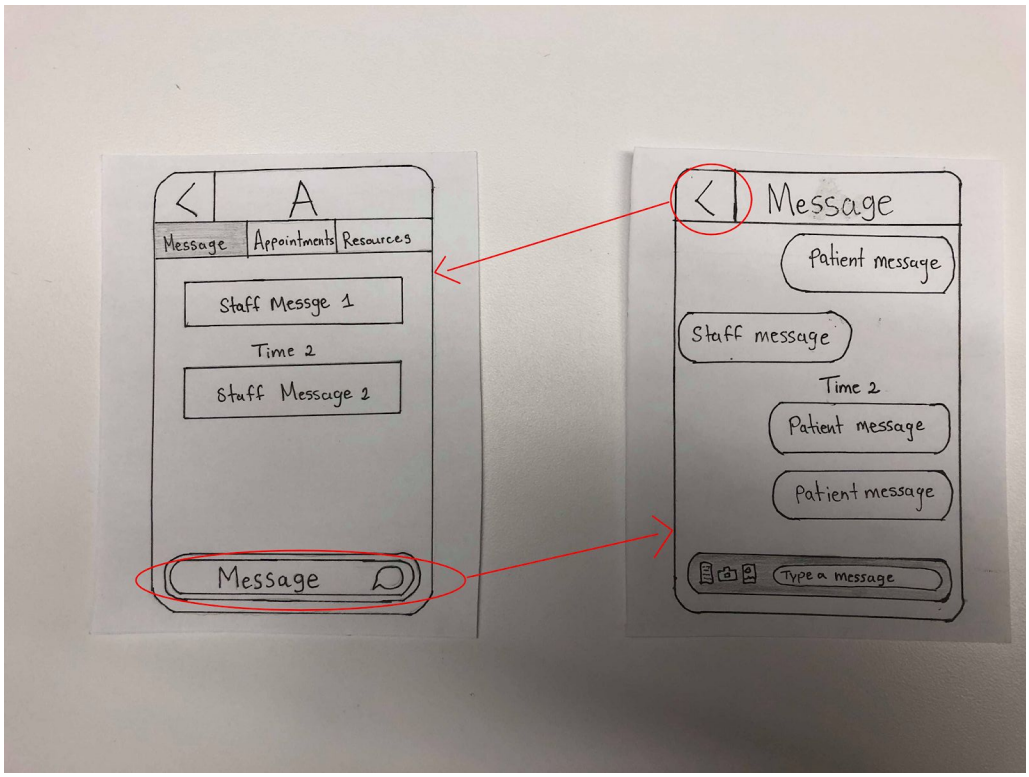
Staff and Coordinator can add resources. To add a resource, Staff/Coordinator press on “+” button at the bottom-right of the general resource page, this leads to a page with options: “title” and “content.” Staff adds a title to the resource and content (either link, document, image file) and press “+” button on the bottom right, thereby adding the resource to the general resources list.



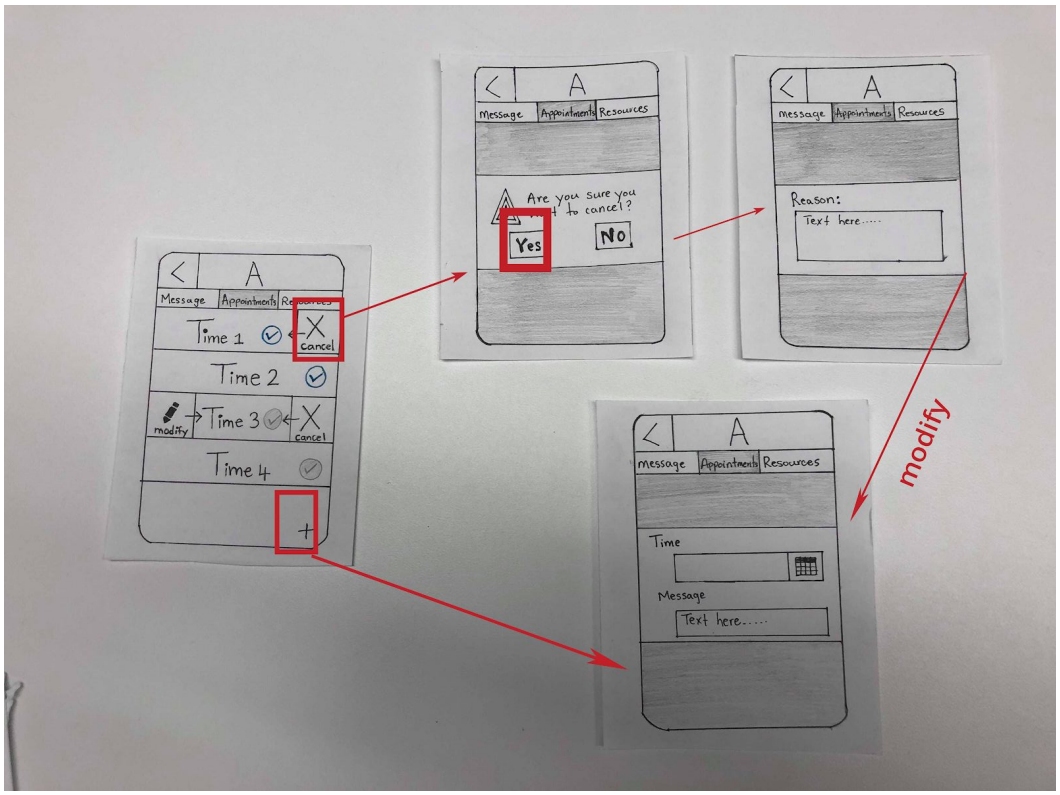
Staff/coordinator go to “Members” on the content bar to access all the members. Members page has a list of names of patients and their email addresses. By pressing on a specific name, the staff/coordinator can send a message directly to that specific person. By holding on it, they can delete the person from the group. Clicking on the add button at the bottom-right will go to a page with the



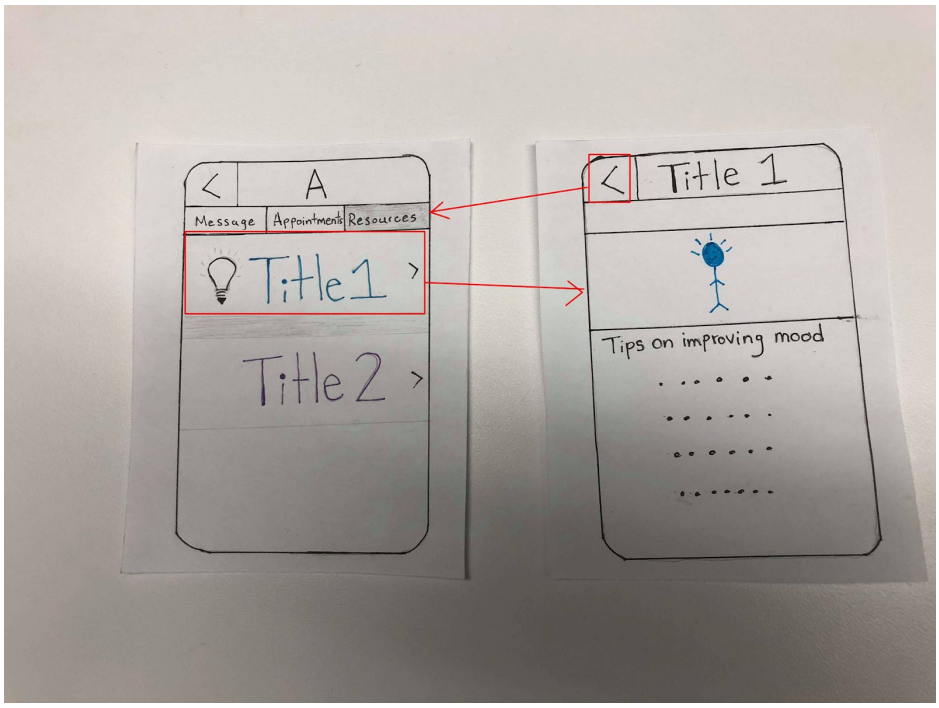
The difference between the main page of the patients/parents and of the staff/coordinator is that patients/parents do not have access to the Account page and they can not add/delete groups. Inside the group, the tab bar have three options: message, appointments and resources. On the message page, patients/parents are not able to send messages here. At the bottom of the page, there is Message button leading them to the message page. The appointment page is similar to the one in the staff/coordinator interface, except that there is check icon showing whether or not their appointment has been confirmed by the staff. Blue check box represents confirmed while white box does not. The resource page is also similar to the one in the staff/coordinator interface, except that they are not able to add/delete resources.



Patient and Parent can access message page with staff messages. To send a message to the Staff (can be as a reply to the staff's message), users press on the "message" button which leads to a message page. Users can send text message or send an attachment.



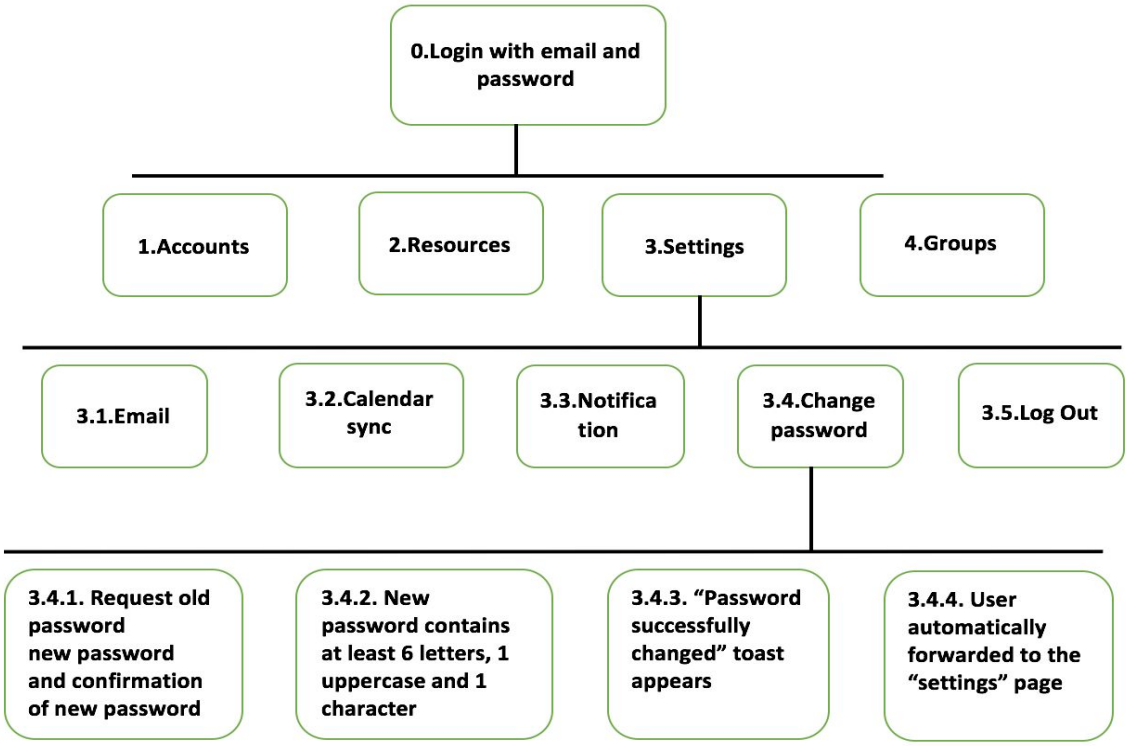
In the appointment page, in order to modify an appointment, users swipe left and press on modify icon. In order to cancel an appointment, users swipe right and press on cancel icon. Once the users press on cancel icon, a confirmation toast with “are you sure want to cancel” appears. Users can press either “Yes” or “No.” If users press “Yes,” reason toast with text box to add a reason for cancelling the appointment appears. If users press “No,” they are taken back the the appointment list page. Similarly, once the user press on modify icon, a confirmation toast with “are you sure you want to modify” appears. Users can press either “Yes” or “No.” If users press “Yes”, they are taken to an interface that allows them to add new time and the reason to modify the appointment. If users press “No,” they are taken back to the appointment list page. To create a new account, users press on the “+” which leads to an add toast with time and message text boxes. Users are able to add a time for the appointment and a message giving the reason for the appointment.



Patients and Parent can access group specific resources page. Once they press on resources on the content bar, a list of resource titles is shown. Users press on one of these resources in the list which leads to new page that shows the content of the specific resource.

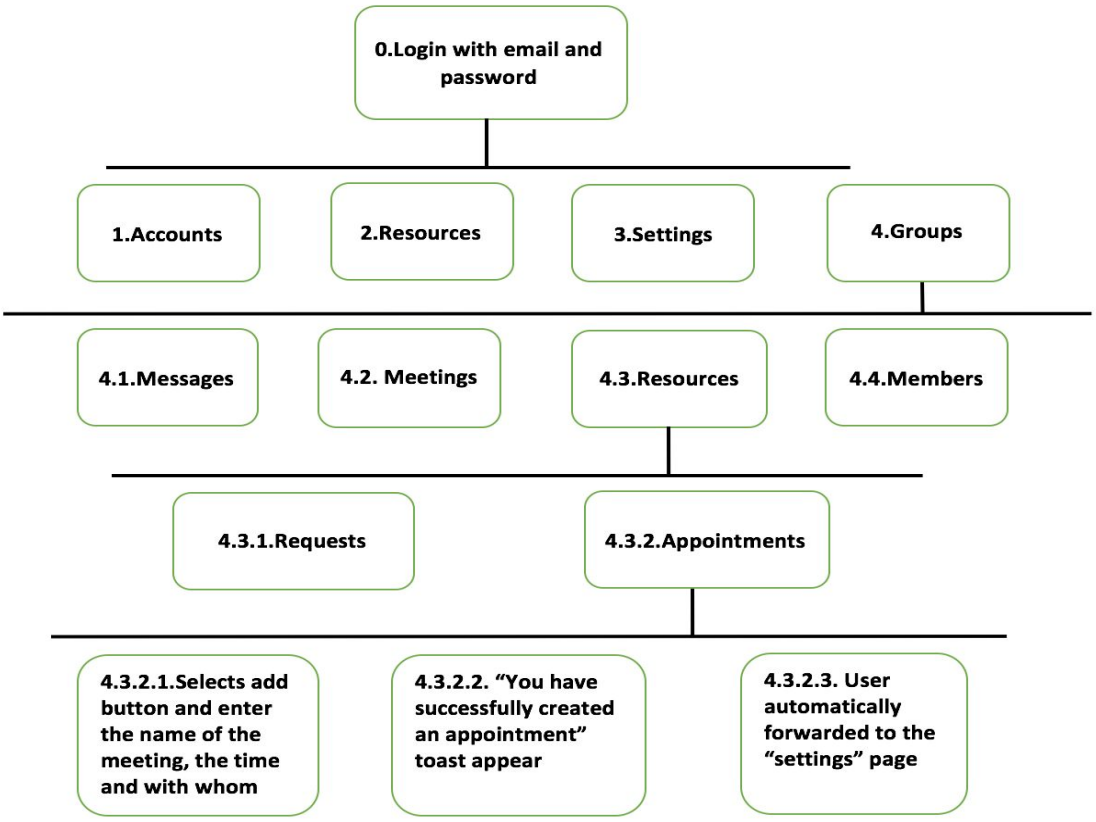
Usage Scenarios

Scenario 1: Loggin and change password - any user



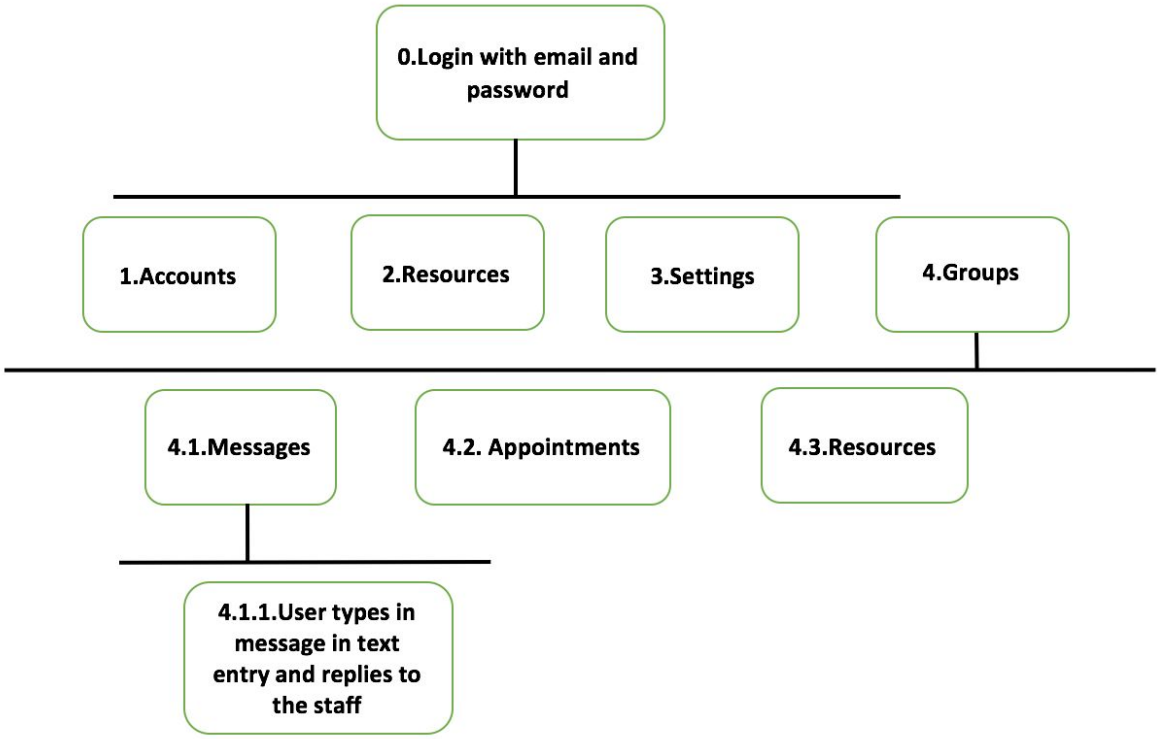
- The user first needs to login with their email address and given password
- The user is then presented with a main page that includes sub menus: setting, accounts and resources
- The user selects "setting" menu
- The user is then presented with a settings page that includes sub menus: Email, Calendar sync, notification, change password and log out
- The user selects "change password" menu
- The user is then presented with "change password" page which requests the old password, new password and confirmation of the new password
- The new password requires it to have at least 6 letters, 1 uppercase and 1 character.
- If the user successfully creates a new password, "password successfully changed" toast will appear and user will be automatically forwarded to the "settings" page

Scenario 2: Making an appointment with a patient: Staff



- The user first needs to login
- The user selects a group from the list of groups
- The user selects "meeting" from the tab bar
- The user is then presented with a meeting page with two buttons: requests and appointments
- To make an appointment, the user has to select the appointments button
- The user is then presented with a appointments page with list of appointments and an add button to add an appointment
- The user selects the add button and gets prompted the name of the meeting, the time and with whom.
- Once the user successfully create an appointment, "you have successfully created an appointment" toast will appear
- The user is automatically forwarded to the appointments page

Scenario 3: Patient replies to a staff's message



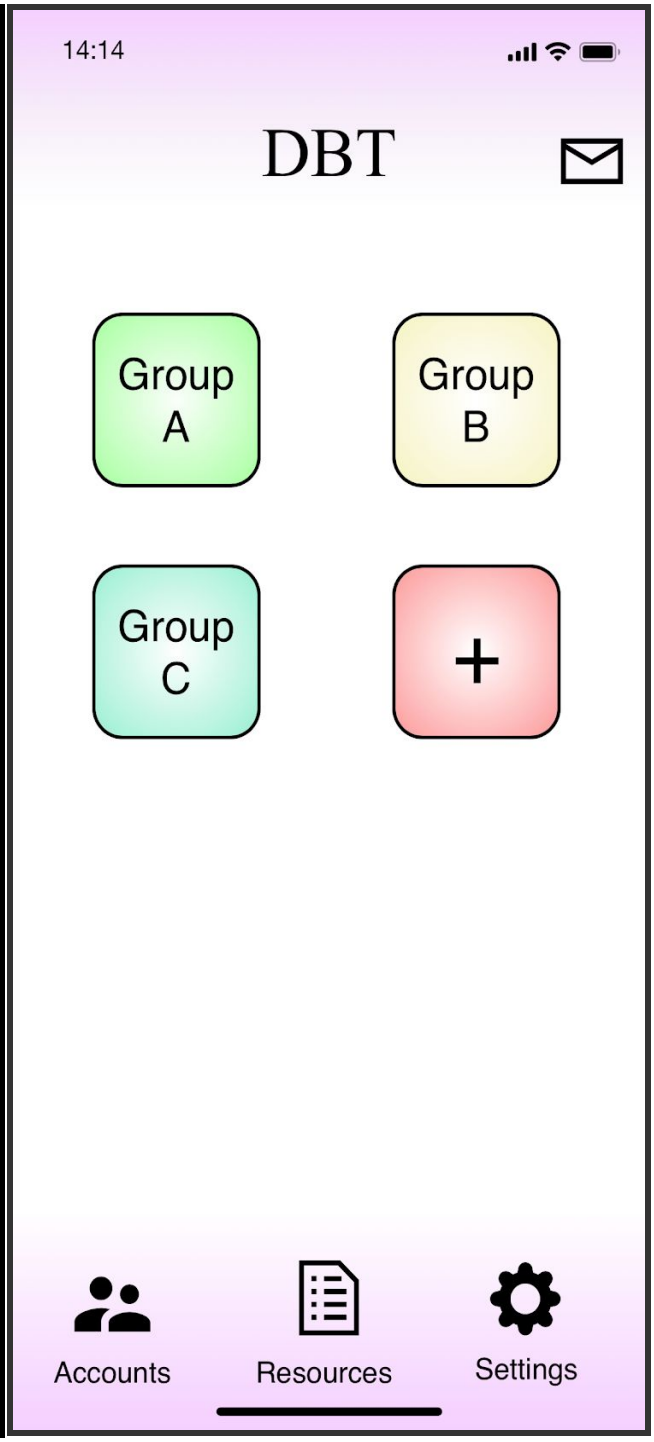
- The user first needs to login
- The user is then presented with the main page
- The user selects the specific group from the list of groups
- The app automatically loads the messages page with a tab bar: message, appointments, resources
- The user selects the "message" tab to reply
- The user types in a message in the text entry and replies to the staff

Prototype

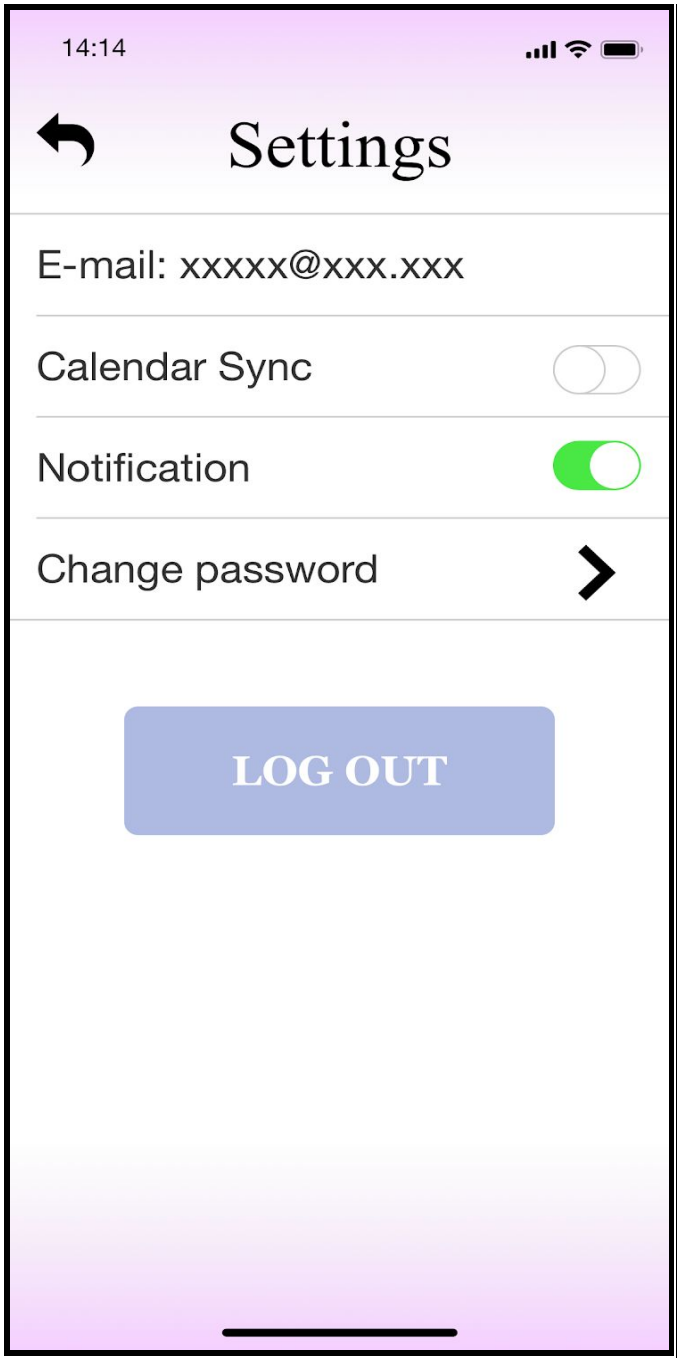
We created a low-fidelity prototype using proto.io app design software. It is running on an iphone X, however our final app would work on either iOS or Android devices, and the prototype does not contain any non-cosmetic device-specific design elements.
Our prototyping approach provides a look and feel of the application because we added critical functionality of the app including the ability to login, access groups, check messages, make appointments and check resources. Although we didn't add specific color schemes or font types to our prototype, we were still able to discover the issues of the design and get them resolved at an early stage, with much less time invested in it.

The Screenshots for the staff/coordinator:

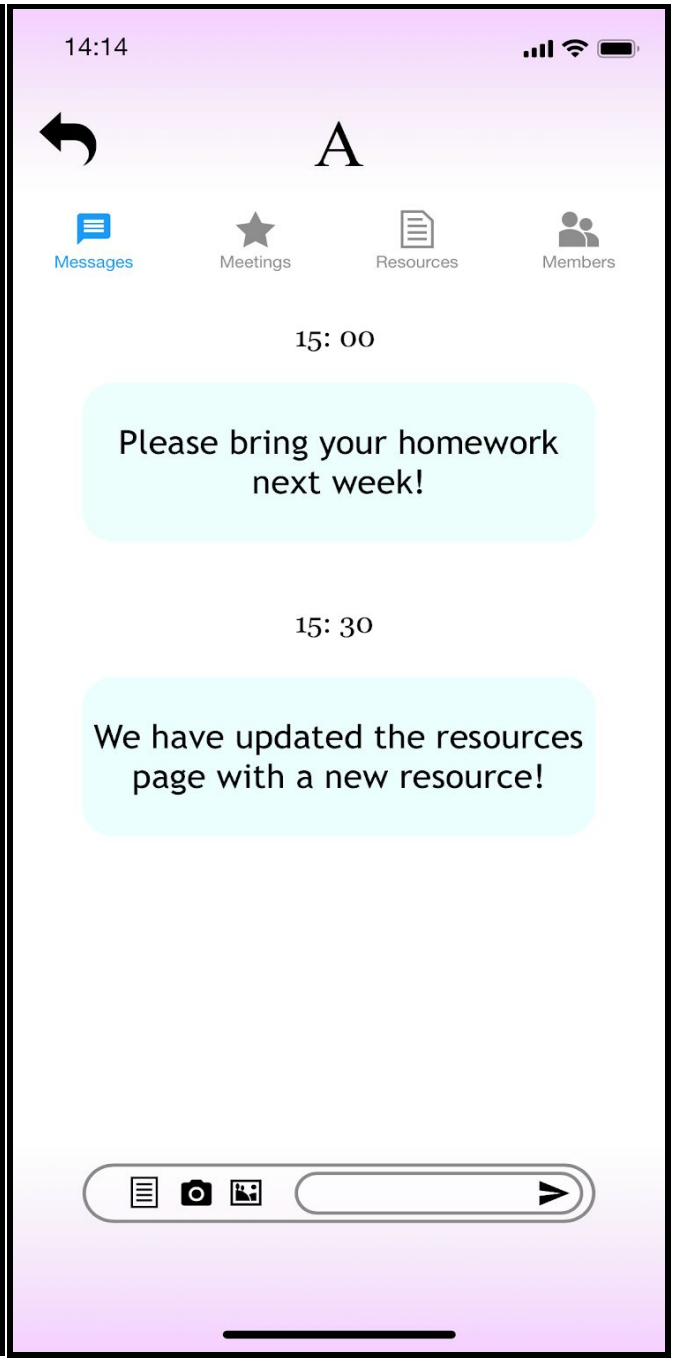
Login page



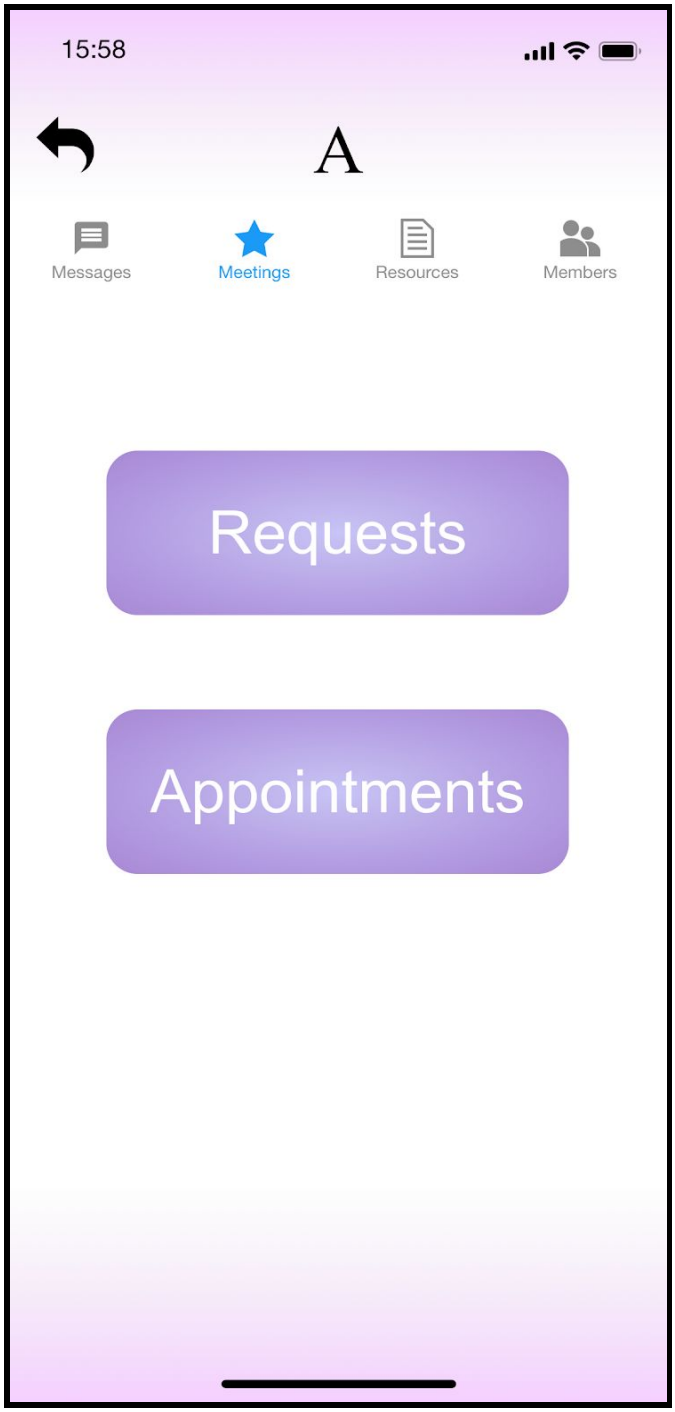
Main page



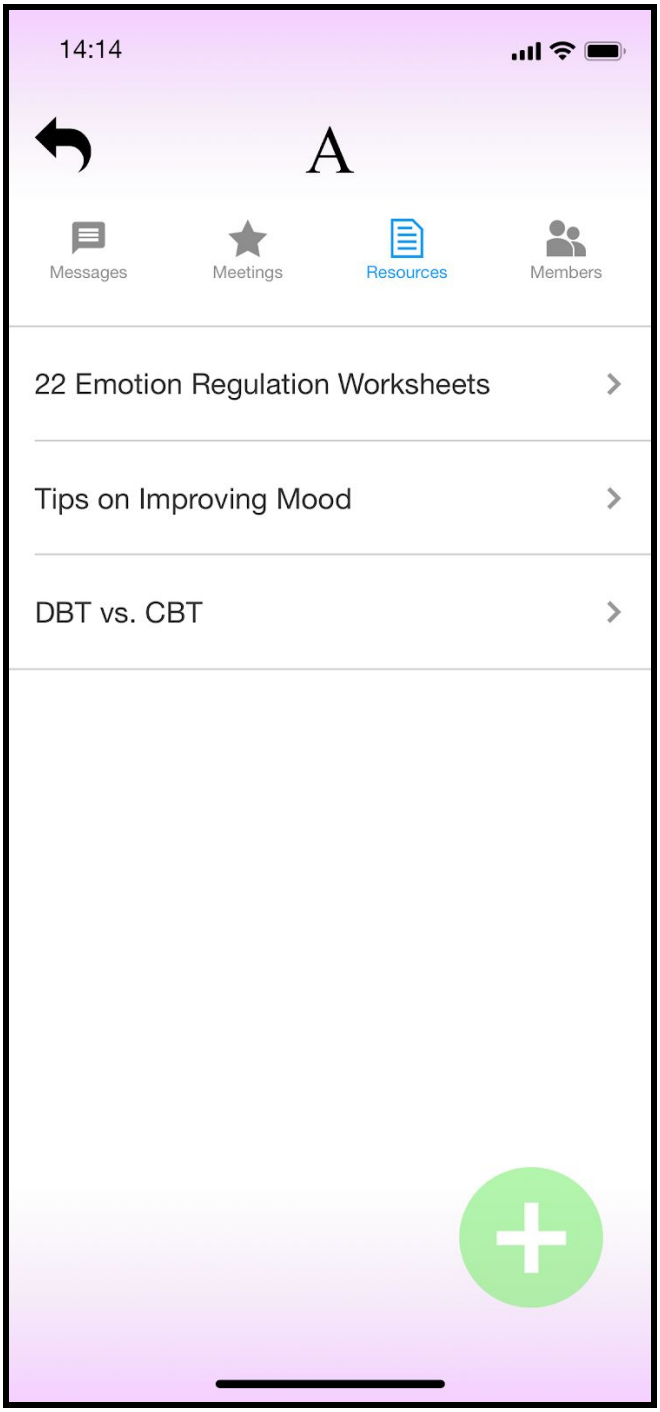
Settings page



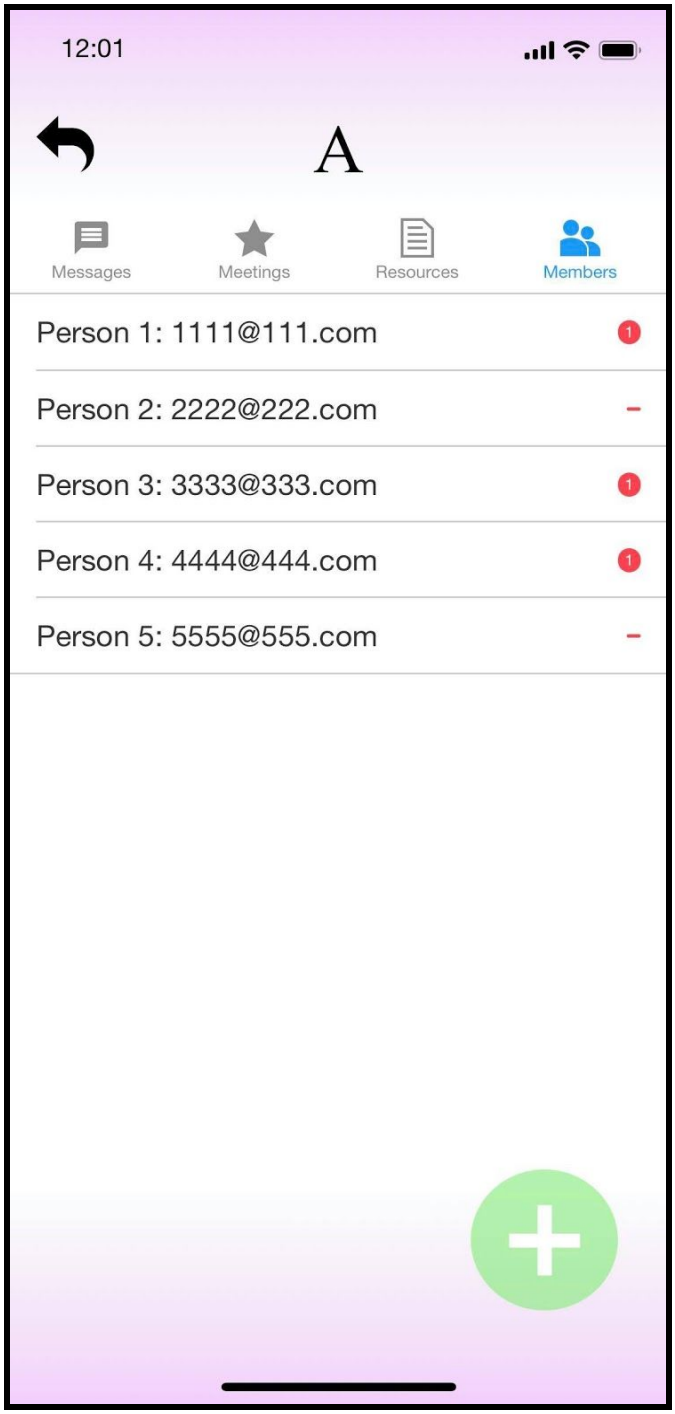
Group messaging page



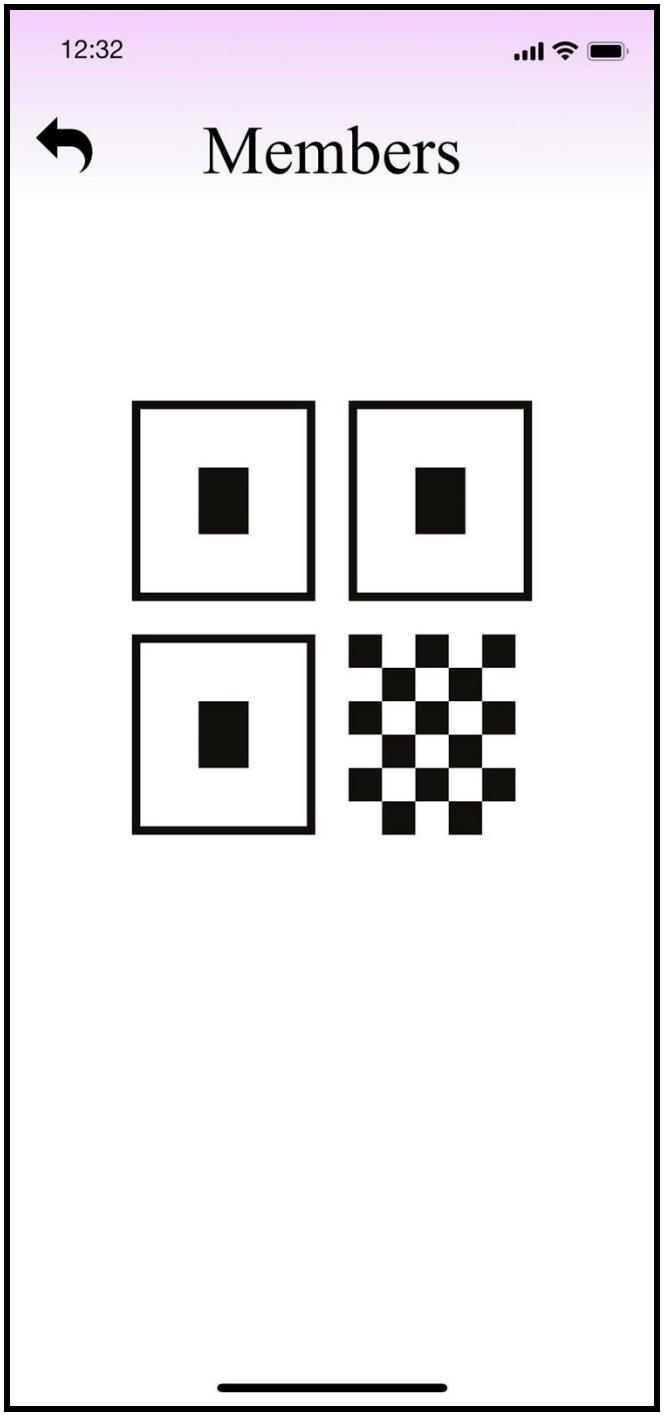
Group meeting page



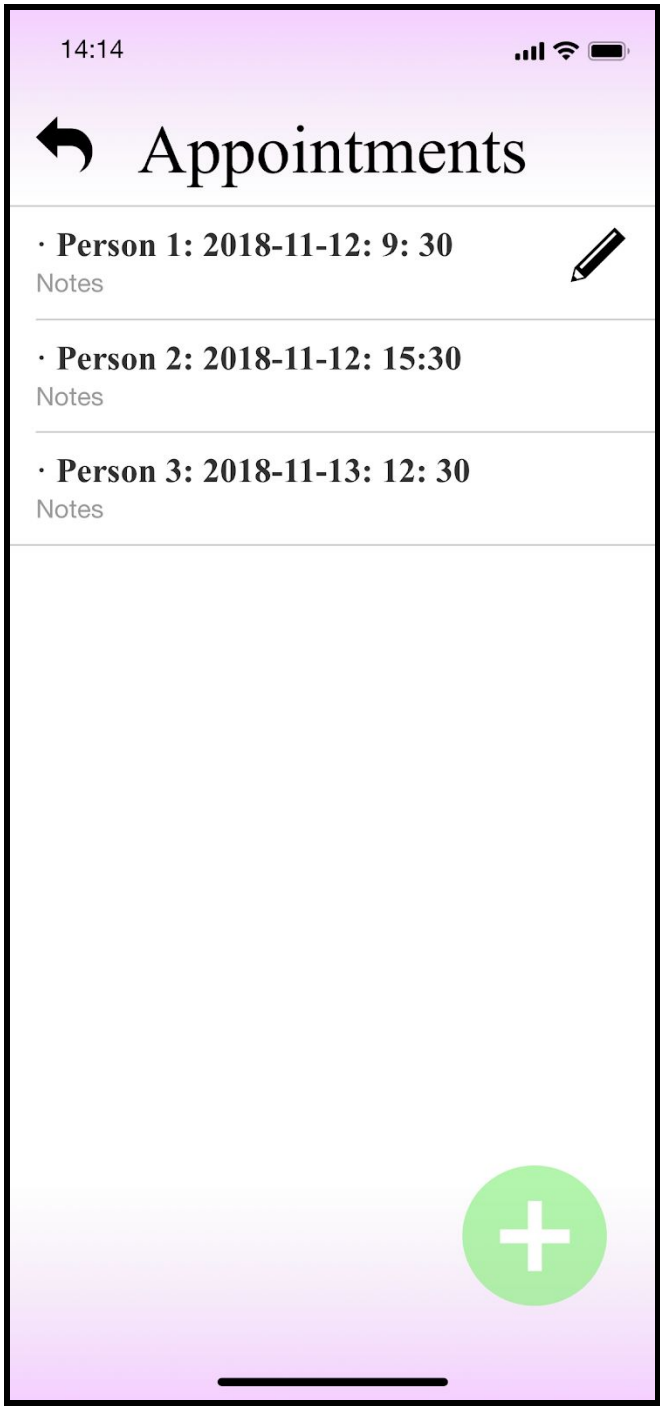
Group resources page



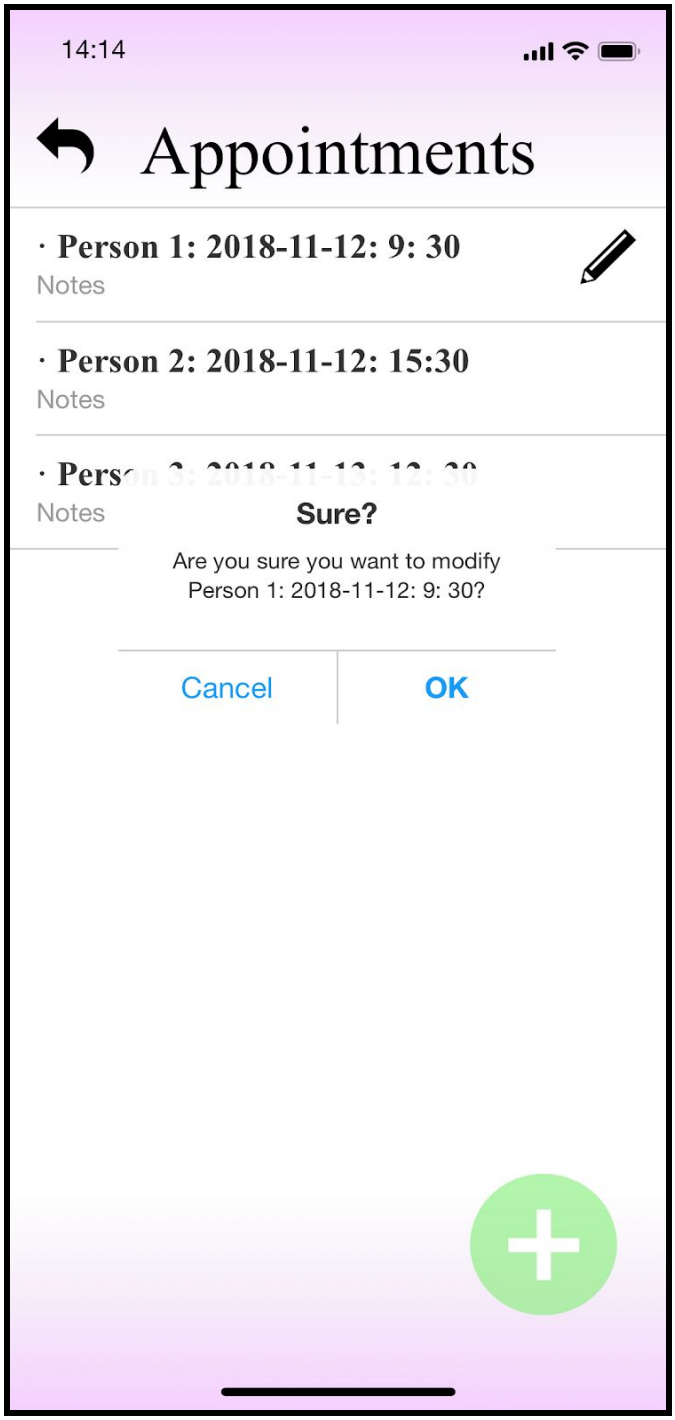
Group members page



Group QR code page

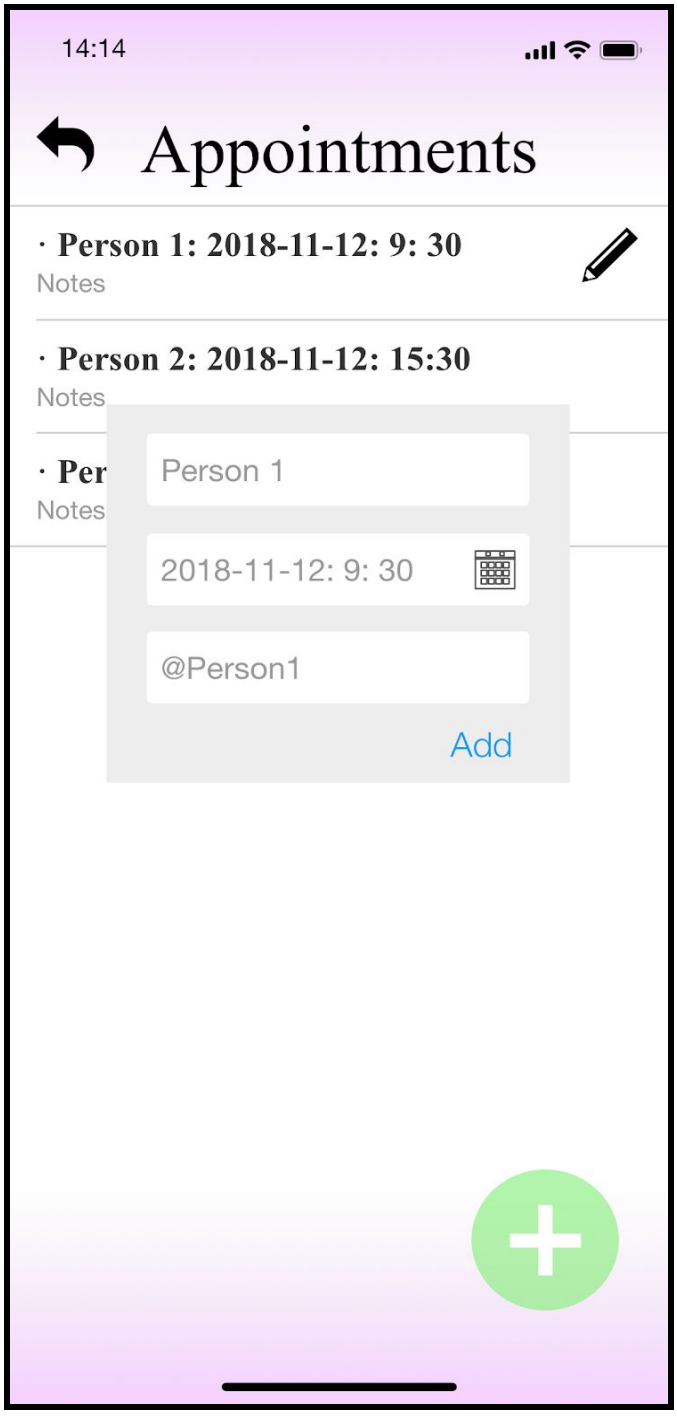


Appointments page

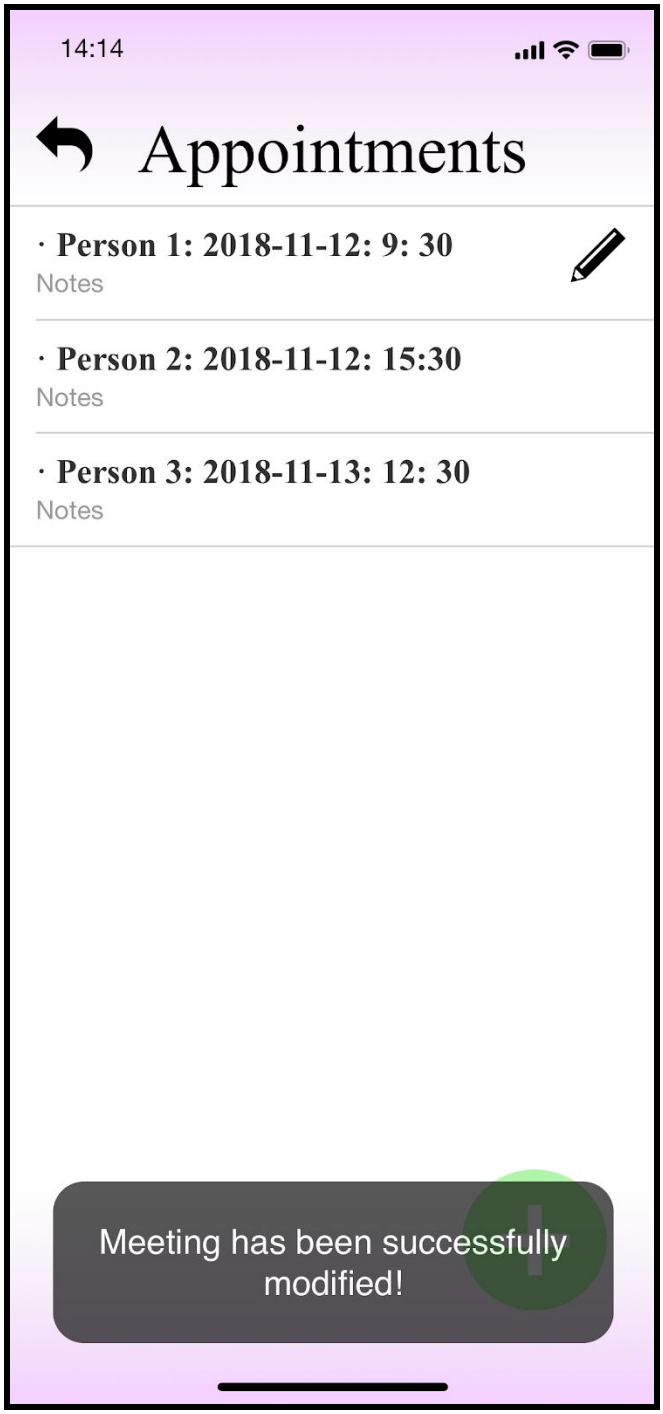


Confirm toast to let users confirm that they want to modify an appointment

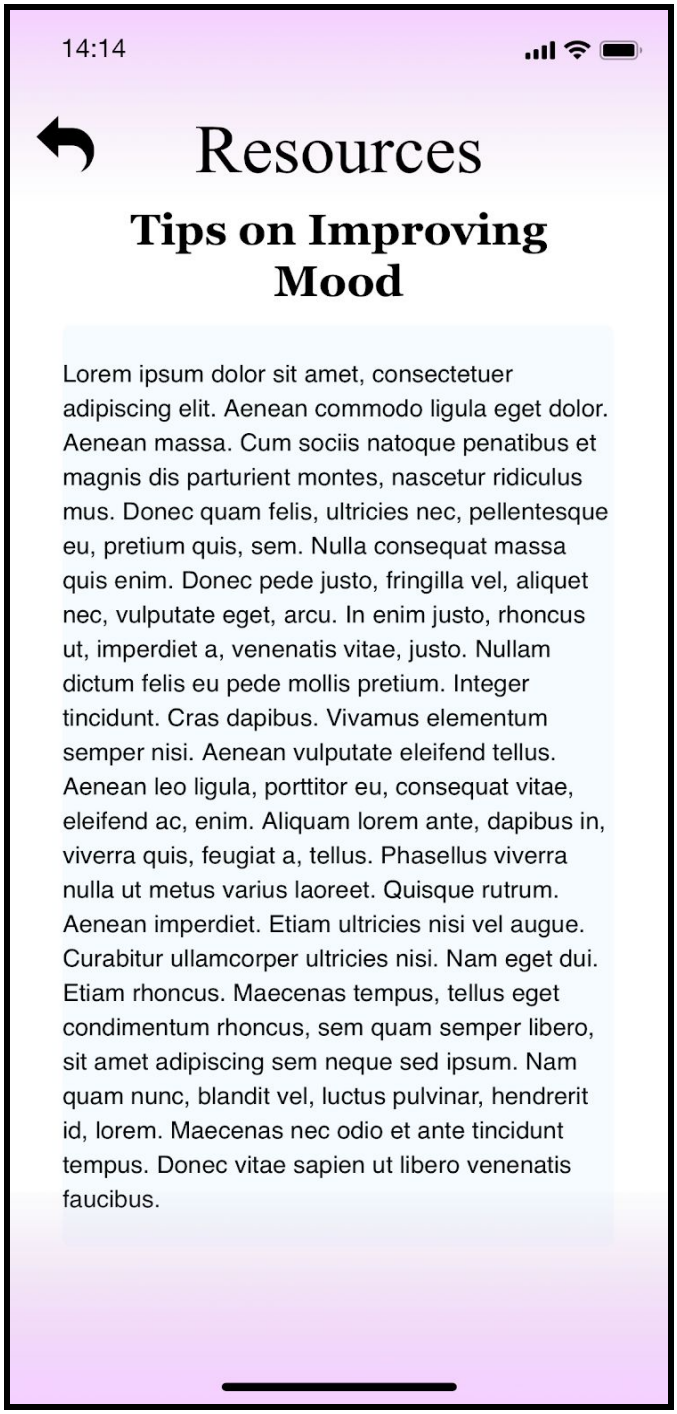
Reason toast to let users give a reason for modifying an appointment



An add toast to modify/add an appointment

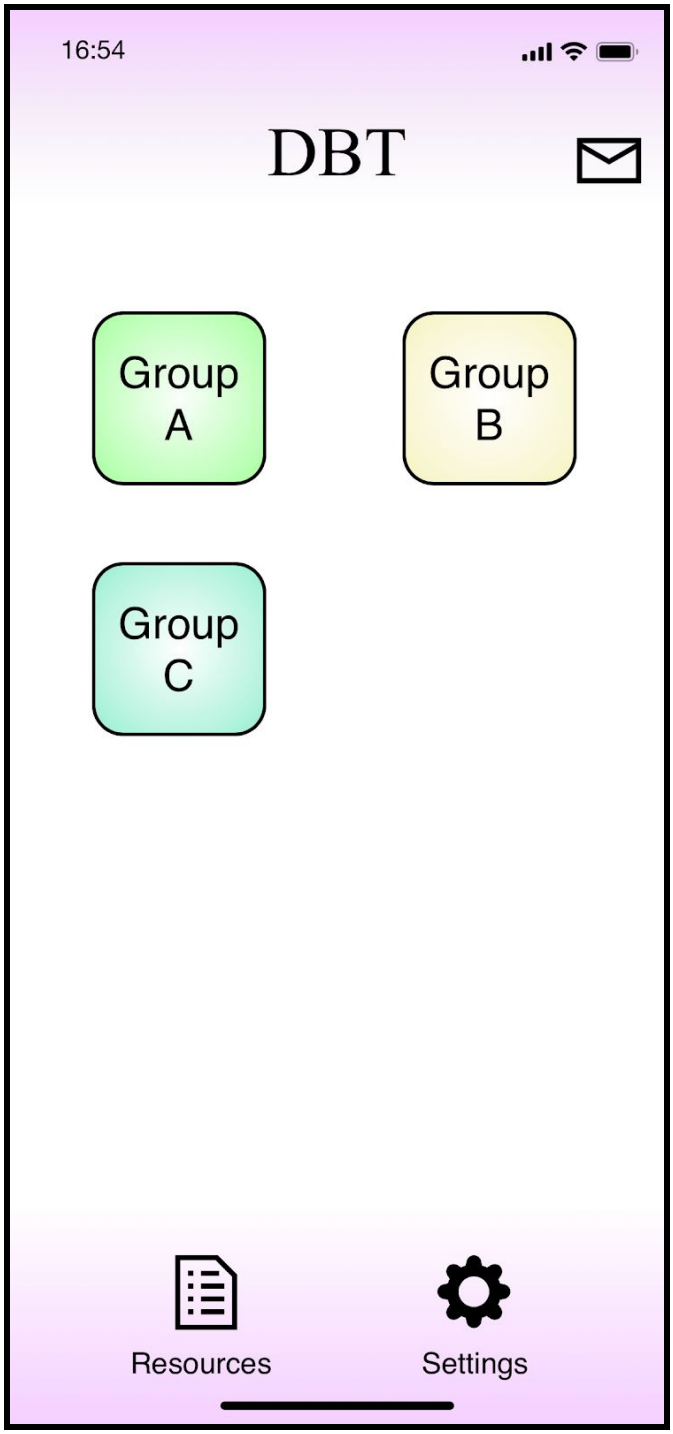


Successfully modified toast

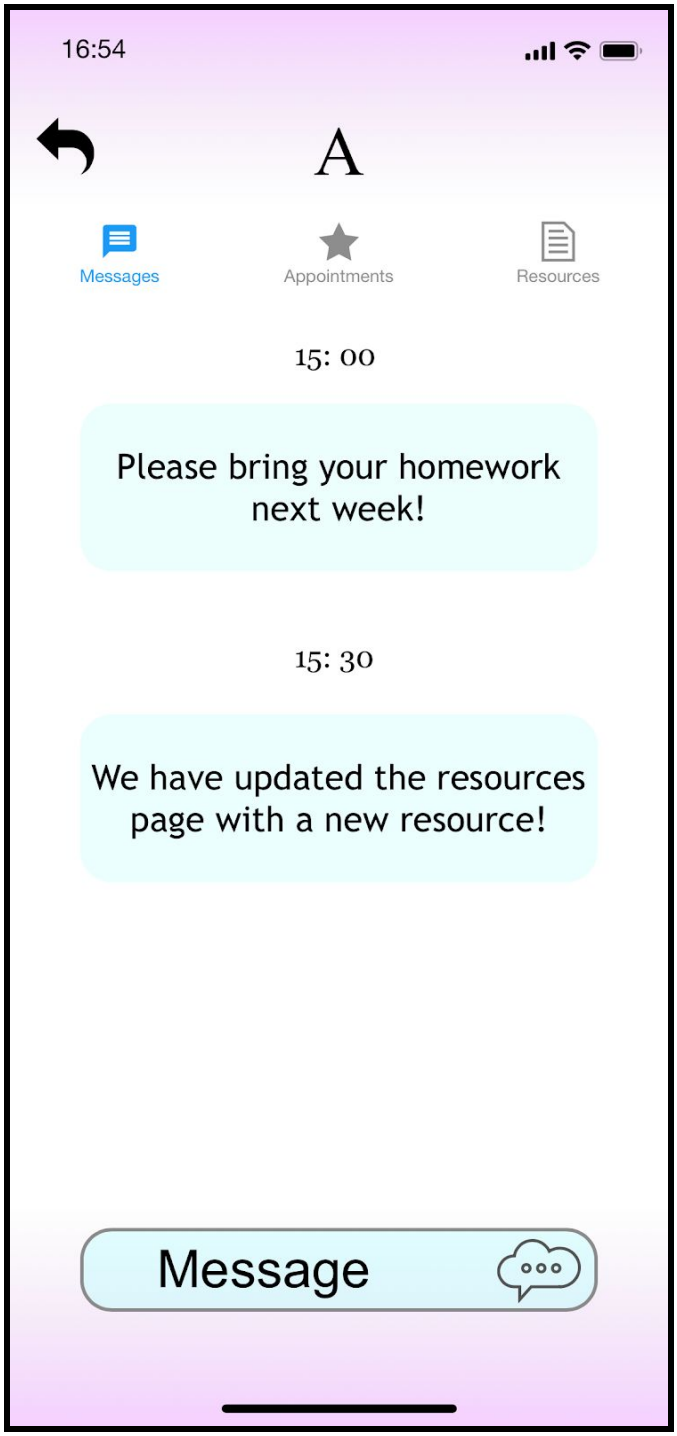


Page containing one example resource

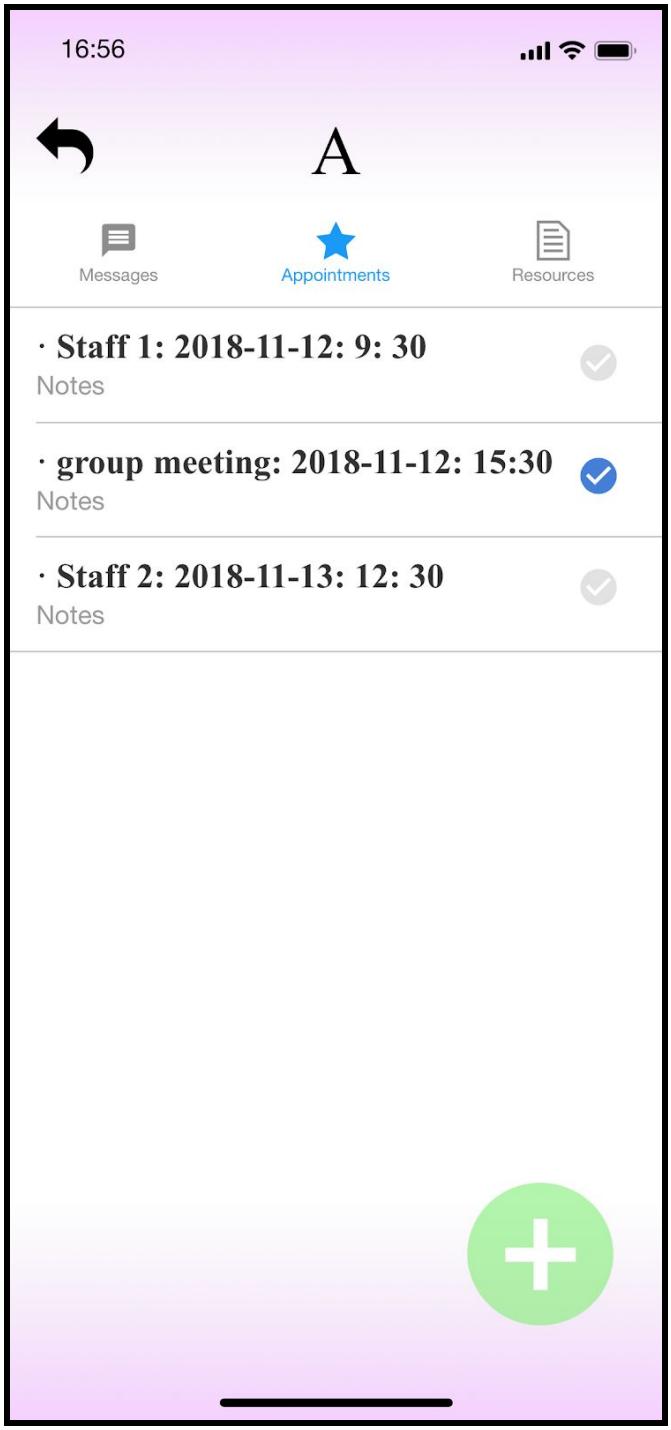
The Screenshots for the patient/parents:



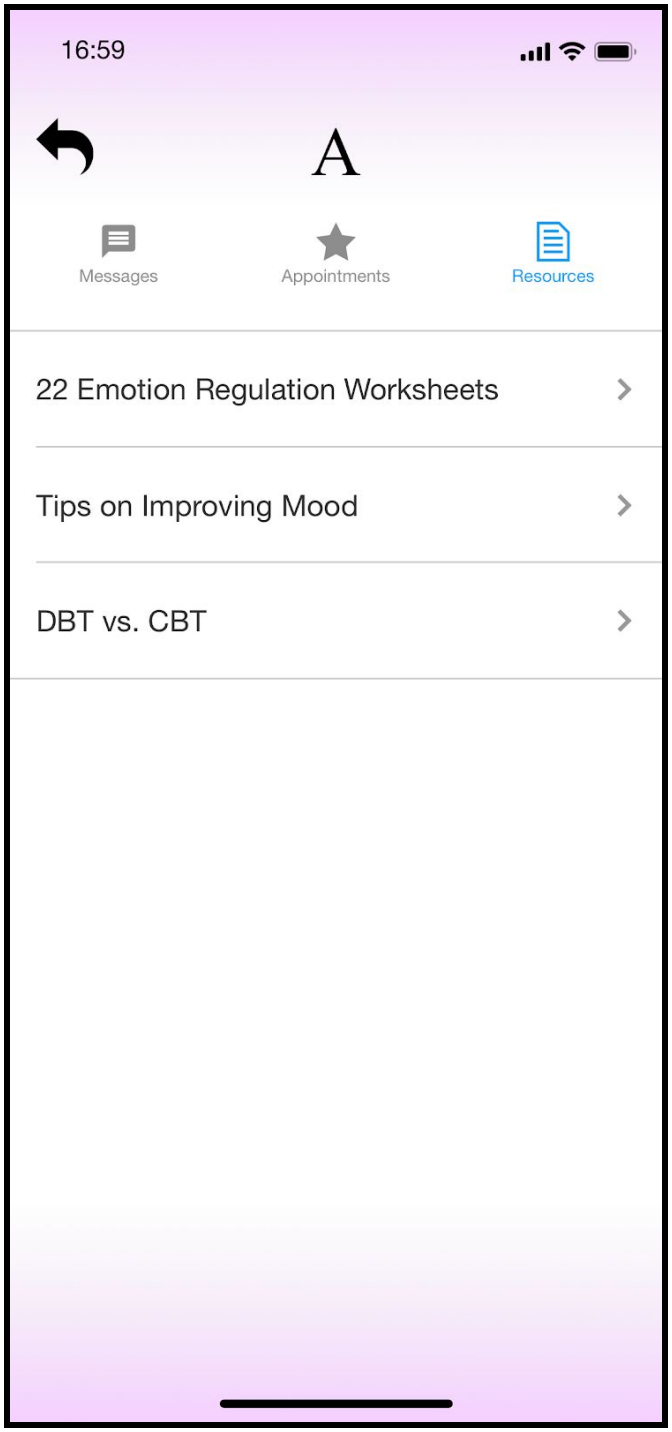
Main page



Group message page



Group appointments page



Group resources page

Work Products

Resource files provided by the staff is required by the application as input files. These files are used in the general and the group specific resources area.

Feedback from customer

We created two prototypes using two different technology:

- Paper prototype was done as a rough draft of our design and the customer was very satisfied with it. As part of the feedback, the customer wanted to be able to disable patient's ability to send attachments. In terms of colours, they wanted to have a colour gradient (i.e transitioning from light red to orange, light purple to blue) to achieve visual effects. Although we had thought of few logo designs (as shown in figure 1), the customer was interested in a mandala logo design and so we changed the application's logo to reflect that.
- Prototype using proto.io.app was done to do user testing. We asked some questions related to the app's functionality before users are given the prototype, then we asked some design centered questions after interaction with the prototype:
    - What would you expect to be able to do with the app?
    - How would you expect it to look?
    - Did you have any trouble interacting with any of the features?
    - Does anything seem out of place or unnecessary? Any features missing?
    - Patient specific questions:
        - Login and make an appointment
        - You get a message from the staff reminding you to do homework but you have a question you need to ask about the homework - how would you ask that question?
        - Go to Group A resources page and select resource # 2 from the list
        - Can you check how many appointments you have?
        - Can you make another appointment?
        - From the appointments page, go to main page and logout
    - Staff specific questions:
        - Go to resources page and add a new resource
        - Message to Group A
        - How many appointments do you have?
        - How many appointments are confirmed?
        - One patient is sending inappropriate attachments, what would you do?
        - From the resource page, go to main page and logout

Observations:
We were mainly interested in how easy the user found interacting with our application and how useful they found it.

- Most of the users had no problem interacting with the prototype and seemed to find the functionality intuitive, needing no help or prompting
- Two users, however, found the "modify" feature confusing. It took them awhile to figure out that they are supposed to swipe right and click on the pencil icon to get a pop-up window that lets them modify the appointment
    - We figured that we should change it to make the app more user friendly: instead of having to swipe right, we would have the pencil/pen icon present on the appointments window and users would then press on the icon to modify the appointment.

Requirements update

- A new component is added to the settings to include the privacy policies of the app
    - Explains data retention/deletion policies and describe how a user can revoke consent and/or request deletion of the user's data
- Both patient and parent role is given the same privileges - they are both able to check messages, send messages to staff, create appointments and check general resources and group specific resources area
        - In the message interface, both roles are able to send attachments such as images, videos and document files but if patients don't use it the right way (i.e send pictures of them trying to harm themselves), staff is able to disable sending attachments by going to the patient's messaging window and pressing on the disable icon.

Internal Design Document

Introduction

This application is designed to facilitate communication of information between therapists and patients. It will push out messages/notifications to each DBT group member with DBT information, group starting dates or group cancellations, as well as reminders to practice skills.

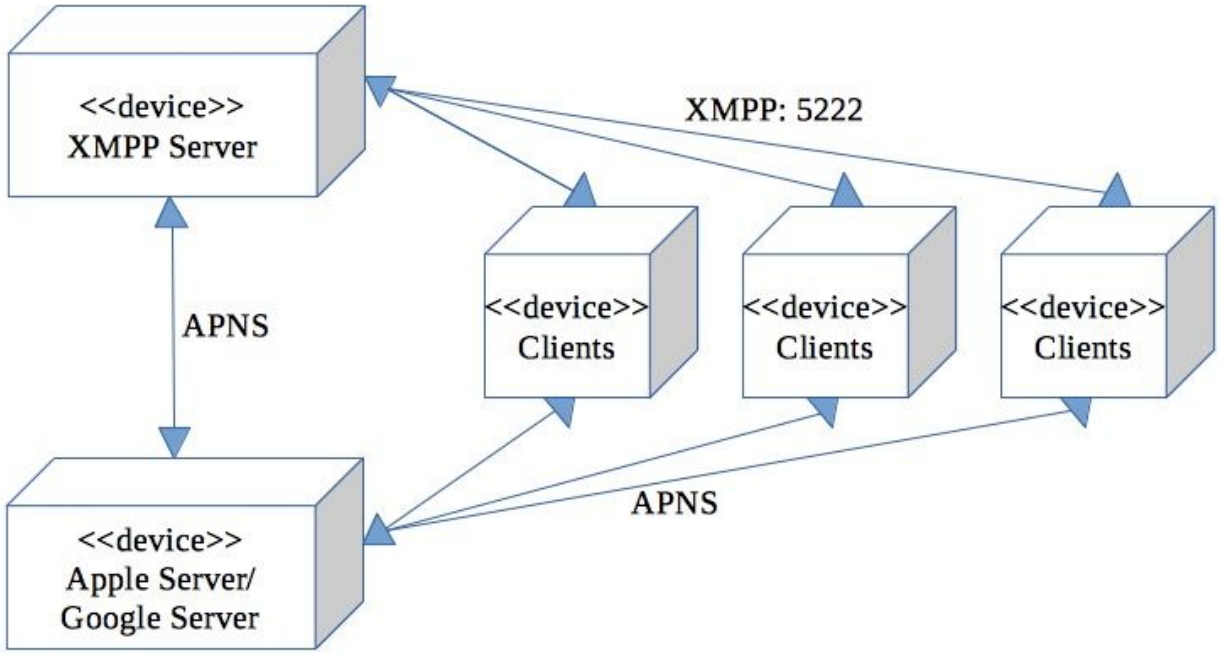
The Internal design document will start by going through the programming tools and languages used to create the system, as well as the derivation of the software architecture. The software architecture component will consist of deployment view of the system identifying the major components of the system and class based view, with input and output, showing the structure of the system. It will also identify the architectural software and design styles. Subsequently, we will talk about the interaction with persistent data, more specifically, format of the data messages and database architecture design/patterns. Lastly, we will finalize the internal design by explaining notable tradeoffs and any modifications to our milestones.

Programming environment

We will be using Android Studio as our programming environment for the Android version, and Xcode for the iOS version. Xcode is an integrated development environment containing a suite of software development tools developed by Apple for developing iOS software, and Android Studio is an integrated development environment built on jetBrains' IntelliJ IDEA software for developing Android software. We will be using the newest versions of Xcode and Android Studio - Xcode version 10.2 and Android Studio version 3.2. In terms of programming languages, the mobile applications will be implemented in Java and Objective C - Java is for the Android version and Objective C for the iOS version. On Android we will be using the Java-based Smack XMPP library. On iOS we will attempt to use the Smack library as well, converted to Objective C using Google's J2ObjC, and failing that will fall back onto the native Objective C XMPPFramework library. MySQL will be used as our database management system and git with the gitea web interface as our source control system.

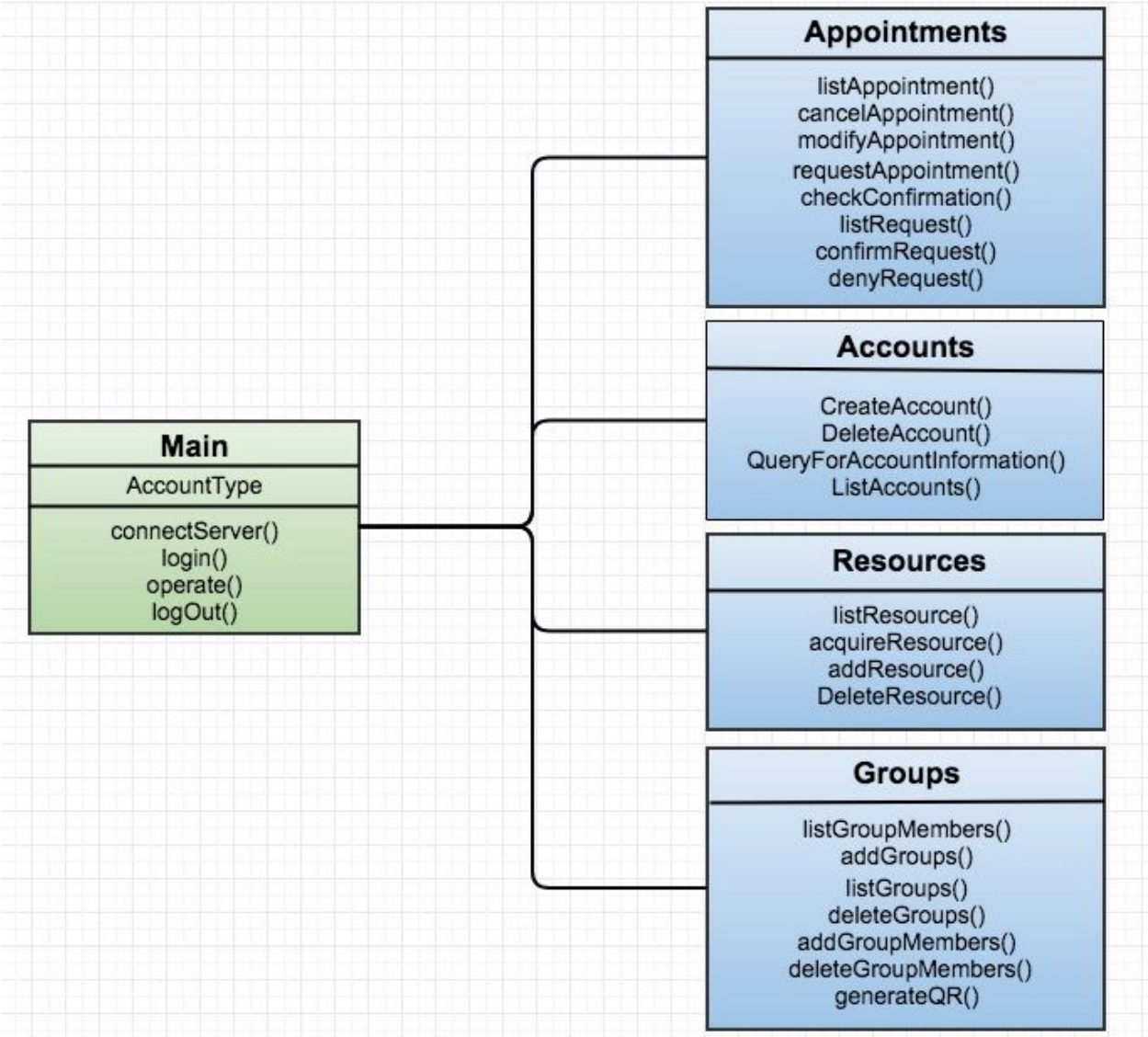
Software architecture

Deployment view



The basis of the backend system is a standard, unmodified XMPP server (Prosody in current testing). The client connects to the XMPP server, which handles all the messaging protocols. A client can send messages to another client through the XMPP Server. A bot running on the server connects using this user account and sends data to the clients using a hidden chat channel. Patients, who are not allowed to message other patients, will have all other patients added to their XEP-0016 privacy lists on the server. File transfers will be accomplished using XEP-0363 HTTP file upload. Password changes will use XEP-0077 In-Band Registration. The use of these existing XMPP extensions allows the app to run reliably on an unmodified standards-compliant XMPP server. The Apple/Google server push out notifications to the clients through the relevant push notification protocols. Push notifications for user messages will be handled directly by the XMPP server using XEP-0357 Push Notifications, and notifications related to non-chat functions will be handled directly by the backend bot. Non-message app data is also sent via the XMPP protocol, to a designated user.

Class-based view



Main Class

The main class connects to the XMPP server enabling the chatbot to login. The class then determines what other classes to initiate depending on the type of message it receives. No method inside Main class directly communicates with the clients, and therefore none require message data input or output.

Method Description
connectServer() **Description:** This method is called to start the connection with the server. When the server program is launched, it stays launched forever, waiting for requests. An error message is thrown if the connection fails and then the program exits.
login() **Description:** Log the chatbot into its designated user account and bring it online
operate() **Description:** Chatbot loop and wait for message requests from a client. Determine the type of operation it is required to perform and send the results to the client.
logOut() **Description:** Finish all active data transactions and log the chatbot out of the XMPP server. Should never be called autonomously, only run during system-requested exit (for shutdown, etc.)

Accounts Class

This class handles all the account related operations (i.e account creation and deletion) The tag items for input and output will be explained in data design.

Method description
createAccount() **Description:** This method is called for creating new accounts and storing them into the server database. A confirmation message will be sent back to clients after that. **Input:** <DBTemail>, <DBTname>, <DBTaccountType> **Output:** <DBTconfirmation>
deleteAccount() **Description:** Delete existing account from the server database (by email). A confirmation message will be sent back to clients after that. **Input:** <DBTemail> **Output:** <DBTconfirmation>
queryForAccountInformation() **Description:** List all existing information of the user's account (i.e which groups they are enrolled in etc) . Send details for user back to client **Input:**

<DBTemail> **Output:** <DBTemail><DBTname><DBTaccountType>
listAccounts() **Description:** List all the existing accounts with their names and email addresses. **Input:** No inputs required **Output:** <DBTaccountList>

Groups Class

This class handles all the group related operations (i.e group creation and deletion)

Method description
listGroupMembers() **Description:** List all the existing members of a specific group with their names and email addresses. **Input:** <DBTgroupID> **Output:** <DBTaccountList>
addGroups() **Description:**

Enables a group to be added with a group name

Input:
<DBTgroupName>

Output:
<DBTconfirmation**>**

listGroup()

Description:
List all the existing groups

Input:
No input is required

Output:
<DBTgroupList>

deleteGroups()

Description:
Delete an existing group from the server database along with the members that are enrolled in the group

Input:
<DBTgroupID>

Output:
<DBTconfirmation**>**

addGroupMembers()

Description:
Adds a user to a specified group using the group ID from a QR code. A confirmation message will be sent back to clients after that.

Input:

<DBTgroupID>, <DBTemail>

Output:
<DBTconfirmation>

deleteGroupMembers()

Description:
Removes a user from a group. A confirmation message will be sent back to clients after that.

Input:
<DBTgroupID>, <DBTemail>
Output:
<DBTconfirmation>

generateQR()

Description:
Sends QR code generation information for group ID

Input:
<DBTgroupID>

Output:
<DBTconfirmation>

Appointments Class

This class handles all the appointment related operations (i.e appointment creation and deletion)

Method description
listAppointment() **Description:** List all the appointments with name of the person the user is meeting and the meeting's date/time **Input:** No input is required. **Output:** <DBTappointmentList>
cancelAppointment() **Description:** Removes an appointment from the database and notifies participants. **Input:** **<**DBTappointmentID**>, <**DBTappointmentReason**>** **Output:** <DBTconfirmation>
modifyAppointment() **Description:** Modifies the details of an appointment in the database and notifies participants **Input:** <DBTappointmentID>, <DBTappointmentName>, <DBTappointmentTime>, <DBTappointmentReason>, <DBTappointmentPatient>

Output:
<DBTconfirmation>

requestAppointment()

Description:
Request an appointment with the person's name and date/time.

Input:
<DBTappointmentName>, <DBTappointmentTime>, <DBTappointmentReason>, <DBTappointmentPatient>

Output:
<DBTconfirmation>

checkConfirmation()

Description:
Check everything of the appointment including seeing whether it has been confirmed or not.

Input:
<DBTappointmentID**>**

Output:
<DBTappointmentList>, <DBTappointmentID>, <DBTappointmentName>, <DBTappointmentTime>, <DBTappointmentReason>, <DBTappointmentConfirmation>, <DBTappointmentPatient>, <DBTappointmentStaff>

listRequest()

Description:
List all appointment requests with requested user's email and date/time of the meeting

Input:
No input is required.

Output:
<DBTappointmentList>

confirmRequest()

Description:
Confirms a requested appointment and notifies requested user

Input:
<DBTappointmentID>

Output:
<DBTconfirmation>

denyRequest()

Description:
Denies a requested appointment and notifies requested user

Input:
<DBTappointmentID>

Output:
<DBTconfirmation>

Resources Class

This class handles all the resources related operations (i.e resources creation and deletion)

Method description
listResource() **Description:** List all the resources with resource titles **Input:** <DBTgroupID> **Output:** <DBTresourceList>
acquireResource() **Description:** Downloads resource file from server using XEP-096 **Input:** **<**DBTresourceID**>** **Output:** <DBTresourceTitle>, <DBTresourceContent>
addResource() **Description:** Enables a resource to be added with its title and content, transferred using XEP-0363 **Input:** **<**DBTresourceType**>, <**DBTresourceGroupID**>, <**DBTresourceTitle**>,** <DBTresourceContent>

Output:
<DBTconfirmation>

DeleteResource()

Description:
Enables a resource to be deleted from the database

Input:
<DBTresourceID>

Output:
<DBTconfirmation>

Architectural style and design patterns

Client-Server

We chose client-server architecture as our architectural style. Clients and servers exchange messages in a request-response pattern. The client sends a request and the server returns a response. Server part of the architecture manages most of the resources and services that the client consumes and the client does not have to be concerned with what the server is doing to fulfill its request We chose this architecture over peer-to-peer because peer-to-peer has no central data server so each process shares its files equally with others. This is problematic for our design because there is no authentication of users. In contrast, server in client-serve model determines which users can access files.

Design patterns

- Iterator

The iterator lets you access elements of aggregate objects sequentially without exposing the internal structure. It is an object pointing to some element in a range of elements which has the ability to iterate through elements of that range using set of operators. For example, if B obtained an iterator from A, it can keep getting new elements from the iterator without mentioning A and A doesn't need to know about B either.

- Observer design pattern - Event Listener

Object maintains a list of its dependents called observers. When the observer is changed, a method of the observer is called. Observer pattern uses three actor classes. Subject, Observer and Client. Subject is an object having methods to attach and detach observers to a client object.

Data design

When the main class of our program connects to the XMPP server and listen for messages, these messages may contain custom XML tags. The XMPP protocol allows for custom tags and other nonstandard extensions to the protocol. We have defined custom tags for each backend operation type.

Format of the data messages

Tag
<DBToperationType> **Description:** Type of operation that the data message is a request to perform **Type:** General
<DBTconfirmation> **Description:** Marks message as an acknowledge message sent to confirm that an operation has been performed if the operation has no other output **Type:** General
<DBTemail> **Description:** The email address of the account that is the subject of the data message. Example: the account being created in createAccount **Type:** Account
<DBTname> **Description:** The user's name of the account that is the subject of the data message. Example: the name of the owner of the account being created in createAccount **Type:**

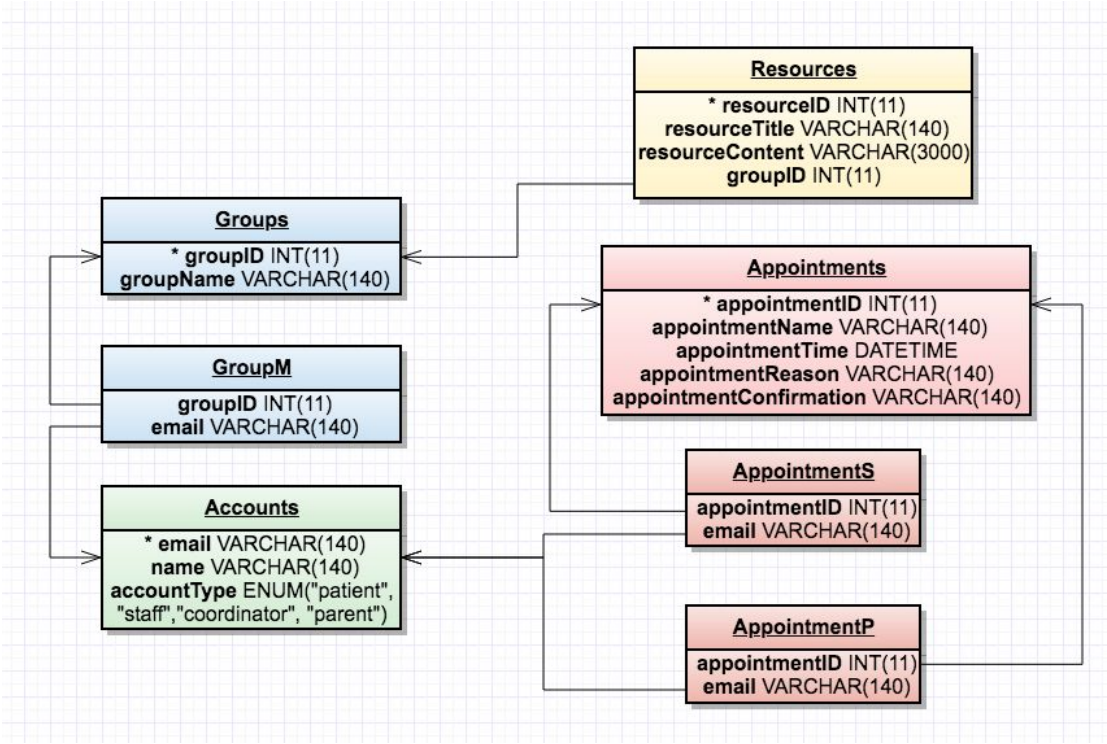
Account
<DBTaccountType> **Description:** The type (role) of the account that is the subject of the data message. Example: the type of the account being created in createAccount **Type:** Account
<DBTaccountList> **Description:** A list of accounts that are the subject of the data message. Example: all accounts in the system in listAccounts, all members of a group in listGroupMemberNames **Type:** Account
<DBTgroupList> **Description:** A list of all groups present in the system **Type:** Groups
<DBTgroupName> **Description:** The group name of the DBT group that is the subject of the data message **Type:** Groups
<DBTgroupID> **Description:** The group ID of the group that is the subject of the data message **Type:** Groups
<DBTappointmentList> **Description:**

A list of all appointments made by the user **Type:** Appointments
<DBTappointmentID> **Description:** The ID of the appointment that is the subject of the data message **Type:** Appointments
<DBTappointmentName> **Description:** The name of the appointment that is the subject of the data message **Type:** Appointments
<DBTappointmentTime> **Description:** The time of the appointment that is the subject of the data message **Type:** Appointments
<DBTappointmentReason> **Description:** The reason text of the appointment that is the subject of the data message **Type:** Appointments
<DBTappointmentConfirmation> **Description:** The confirmation status of the appointment that is the subject of the data message **Type:** Appointments
<DBTappointmentPatient> **Description:**

The email of the patient(s) of the appointment that is the subject of the data message, comma separated if multiple **Type:** Appointments
<DBTappointmentStaff> **Description:** The email of the staff of the appointment that is the subject of the data message **Type:** Appointments
<DBTresourceID> **Description:** The ID of the resource that is the subject of the data message **Type:** Resources
<DBTresourceList> **Description:** A list of all resources made by the user **Type:** Resources
<DBTresourceType> **Description:** The type of the resource(i.e. General resource or group resource) that is the subject of the data message **Type:** Resources
<DBTresourceGroupID> **Description:** The group ID of the resource that is the subject of the data message (if the resource is a group resource) **Type:** Resources

<DBTresourceTitle> **Description:** The title of the resource that is the subject of the data message **Type:** Resources
<DBTresourceContent> **Description:** The text content of the resource that is the subject of the data message **Type:** Resources

Database architecture of the chatbot database



MySQL database is a relational database which recognize relations among stored items. Data is stored in tables contained in the database. The primary key (*) is unique and are used to identify individual records. The foreign key in one table identifies rows of another table (i.e groupID is a foreign key). Each ID has the type INT and the length of an INT is 11 digits. Type of the appointmentTime is DATETIME while other attributes have the type VARCHAR. DATETIME has no maximum length but each instance of VARCHAR has maximum length of 140. The only exception to that is the resourceContent VARCHAR, which has a maximum length of 3000 characters. Moreover, accountType has an enumeration data type with modes: patient, staff, coordinator and parent. This ensures that every account in the table have a type that is one of those four.

Notable tradeoffs

- Use of XMPP messages as a data channel significantly simplifies networking and API design, but it involves the use of custom tags
- Use of the QR code is convenient but a user can enroll in another group if he/she finds the QR code of that group

Updated milestones

ID	Date	Milestone	Description
M1	Dec 01, 2018	**Setting up**	Setting up test server and private git server with a web interface will have been done
M2	Jan 14, 2019	**Finalizing software quality assurance and deployment plan**	Identification of covered software process activities and metrics used will be discussed. Furthermore, description of assurance activities applied and detailed plans for delivering system to customer, hardware and software issues, security plan, plan for acceptance testing will have been finalized
M3	Jan 20, 2019	**Programming environment**	We will have familiarized ourselves with iOS and Android programming languages and environment
M4	Feb 20, 2019	**Implementation**	Basic implementations will have been done
M5	Mar 28, 2019	**Create the final project documentation**	The implementation of the app will have been completed. The user manual, installation guide, maintenance suggestions, limitation and workarounds, as well as necessary updates to the system's internal design will have been finalized
M6	Apr 01, 2019	**Deliver and demonstrate system to customer**	The final poster will have been finished

Contributions

Name	Contributions
Alex Huctwith	Prototype, data message format, deployment, database architecture, backend
Gaveshini Sriyananda	Introduction, prototype, usage scenarios, programming language, software architectural designs and patterns
Selena Sun	Prototype, user scenario, programming language, architectural style, software architecture
Yue Cai	User interface design, Prototype, Usage Scenario, software architecture, database architecture