GitHub Username: n8ebel

InTouch: Relationship Assistant

Description

Never fall out of touch again! With InTouch: Relationship Assistant staying in touch with friends, family, and loved ones is a breeze.

Setup automated text messages to your contacts allowing you to stay in contact without always having to remember to text that person you haven't talked to in ages.

Do you sometimes forget to acknowledge the special days in your loved ones lives? Setup automated messages for birthdays, anniversaries, etc. so you never forget to show your appreciation and affection to those who matter.

Intended User

Intended user is anyone with friends, family, or other contacts that they would like to stay in contact with, but commonly fail to stay in touch.

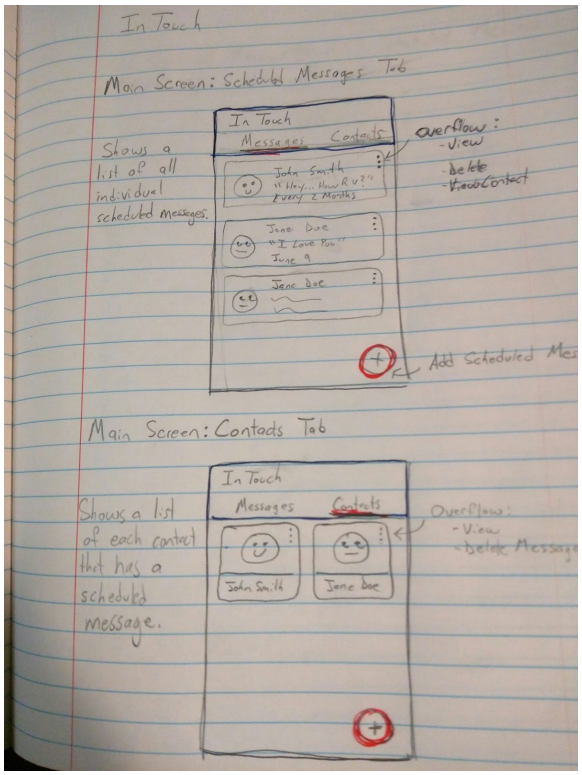
Features

- Widget to list scheduled automated messages
- Schedule notifications with an action button to take you to your sms app
- Schedule automated sms messages
    - At regular intervals
    - On specific dates: eg. birthdays, anniversaries, etc…
- View all messages
- View all messages for a contact
- Edit scheduled messages

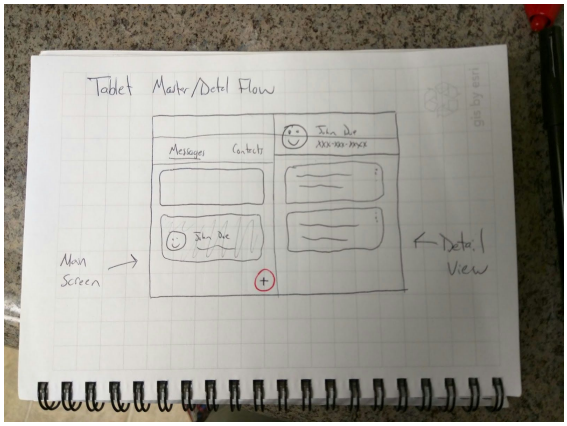
User Interface Mocks

Screen 1

The first screen will present the user with a swipeable view pager that will contain both a list of scheduled messages, and a list of contacts that have scheduled messages. This will provide the user with two different views into the same saved data allowing for quick and easy lookup and data retrieval.



On a tablet, this main screen will be presented in a master/detail flow. The left will contain the view pager while the right will present the details view for a selected item or the create fragment.

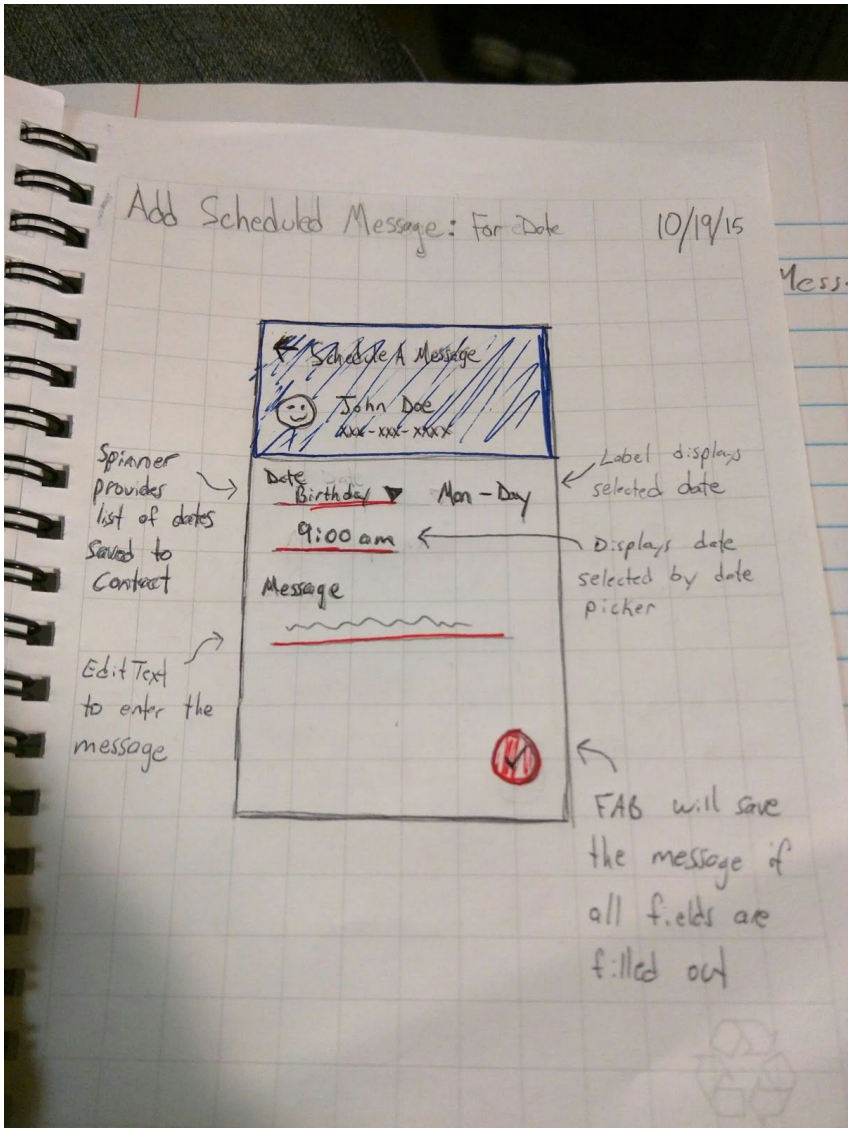


Screen 2

Clicking the FAB on screen 1 can take the user to one of two screens based on which option they choose.

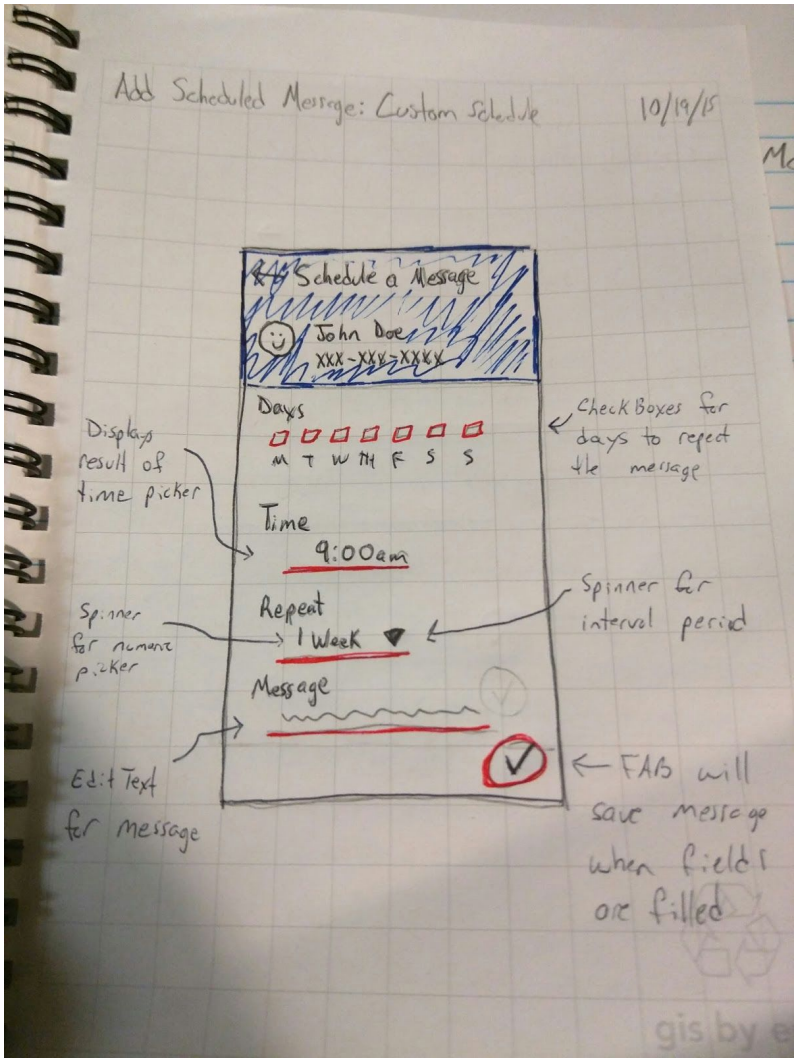
Option 1 will take them to this screen where they can create a scheduled message for a specific date. This date could be an anniversary, birthday, or another date stored to the Contact. The user can choose to automatically send a specified message as a text, or to simply show a notification on the device with an action button to take them to their sms app.

The user can either create a new message from this screen or edit an existing message, but changes won't be saved until the save button is clicked.



Screen 3

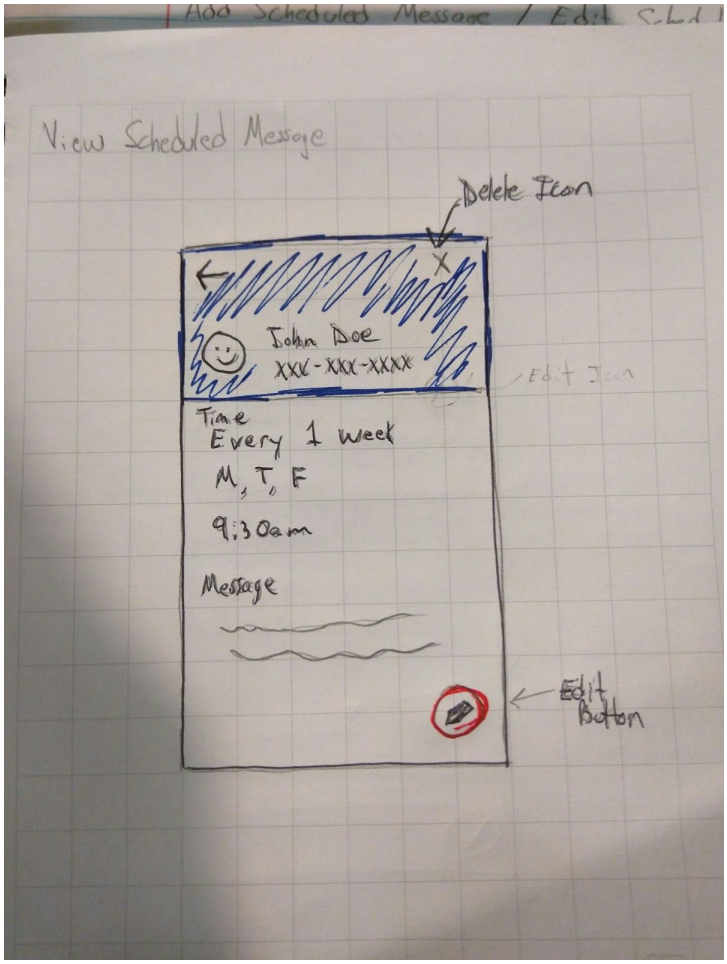
FAB option 2 takes the user to this screen where the user can schedule a message to repeat on a custom schedule. The user can either create a new message from this screen or edit an existing message, but changes won't be saved until the save button is clicked. The user can choose to automatically send a specified message as a text, or to simply show a notification on the device with an action button to take them to their sms app.



Screen 4

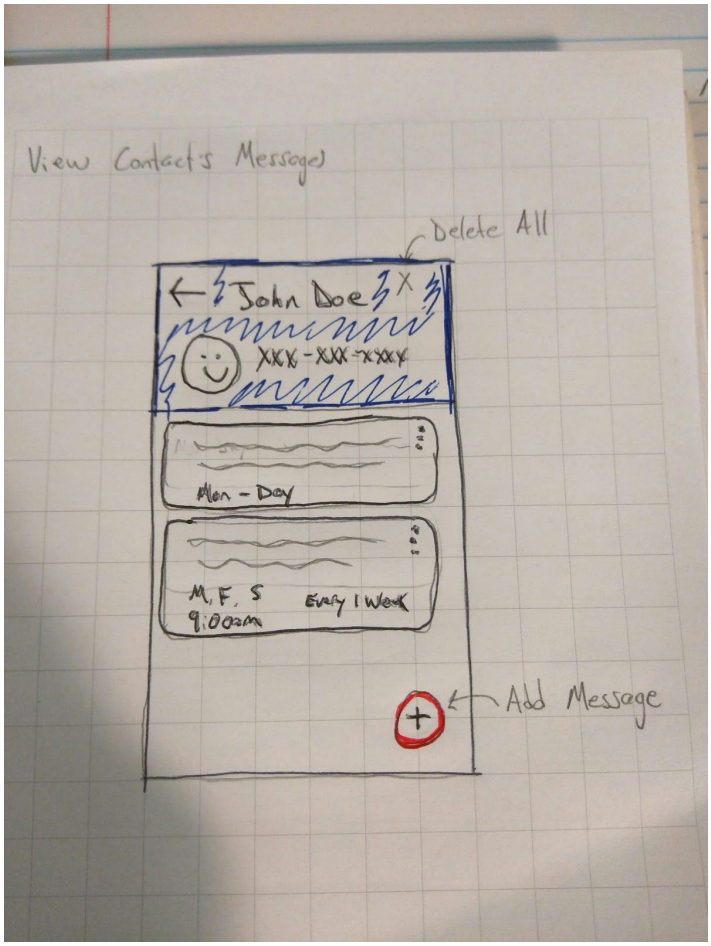
Clicking on a scheduled message card will take the user to a message details screen. This presents the user with a read-only version of a scheduled message.

Clicking the edit icon will take the user to the edit/create message screen.



Screen 5

Click on a contact card on the main screen takes the user to the contact details screen. This screen includes the contact's thumbnail, phone number to send messages to, and a list of messages scheduled for that contact.



Key Considerations

How will your app handle data persistence?

A ContentProvider will be used to expose an underlying database. The database will store entries for scheduled messages. These entries will store a contact_id retrieved from a user contact (retrieved from the Contacts provider), a message, and schedule information such as repeat interval, repeat days and time of day.

Describe any corner cases in the UX.

Will need dialogs for date, time, interval values.

On a tablet, the left panel has a view pager allowing the user to swipe between the messages and contacts lists. The right panel will only show the detail fragment for whatever was clicked witch could be out of sync with left panel. A shared element transition could provide context clues that the right panel is based on whatever is selected in the left panel.

When the user is creating/editing a message, clicking back should notify the user that their changes will be lost.

Describe any libraries you'll be using and share your reasoning for including them.

AppCompat, CardView, RecyclerView and Design libraries from Google to bring consistent material design to the app in a backwards compatible way.

Dagger2 for dependency injection to make development and testing easier.

Butter knife to remove boilerplate UI code.

Play services for ads and analytics

Next Steps: Required Tasks.

Task 1: Project Setup

- Create new project
- Add library dependencies
- Add Dagger2 compiler dependency
- Setup project in Google Developer Console for Ads and Analytics api

Task 2: Implement ContentProvider

- Create stubbed out layout with an add button to save a new item to content provider
- Create ContentProvider / DBHelper
    - Define MessageContract
    - Create Uri matcher
- Test that clicking save button properly saves an item into the db with dummy content

Task 3: Implement main screen

- Populate database with dummy content
- Build the main screen layout
    - Create layout with viewpaper
    - Create fragments for messages and contacts lists
    - Add FAB to layout
- Load content into each fragment through a loader

Task 4: Implement add message for date fragment

- Create layout and fragment
- When user clicks button select a contact using system intent
- When contact returned, show fragment
- Populate layout with contact info
- Retrieve saved contact dates and bind them to spinner so user can select one
- When user clicks the save button, save a new entry into the database with the values from the ui elements
- If set to show notification, create a pending intent to show the notification
- If set to send text, create intent to automatically send text to contact's number
- If user clicks back button, notify user with a dialog that their edits will be discarded

Task 5: Implement add message with custom interval fragment

- Create layout and fragment
- When user clicks button select a contact using system intent
- When contact returned, show fragment
- Populate layout with contact info
- When user clicks the save button, save a new entry into the database with the values from the ui elements
- If set to show notification, create a pending intent to show the notification
- If set to send text, create intent to automatically send text to contact's number
- If user clicks back button, notify user with a dialog that their edits will be discarded

Task 6: Implement view message fragment

- Create layout and fragment
- Pull message id from arguments and load database entry
- bind data
- implement edit button that will take the user to the create/edit fragment for the message type
- implement delete button to remove the saved item and unschedule the automated message

Task 7: Implement contact details view

- Create layout and fragment
- load all database entries for contact_id
- bind data to ui
- implement “delete all” button to delete and unschedule all items

Task 8: Implement tablet master/detail layout

- Create tablet specific layout
- Add logic in MainActivity to add necessary fragment if is running on a tablet
- Implement selected item state for messages fragment and contacts fragment

Task 9: Implement widget

- Create widget
- Regularly update widget’s data from content provider
- Implement pending intent to start app if widget is clicked

Task 10: Incorporate Google Analytics

- Add analytics to several UX elements
    - Add message for date button
    - Add custom scheduled message button

Task 11: Incorporate AdMobs

- Add interstitial ad when changing between main screen and edit/view fragments
    - ensure app is only using test ads

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”