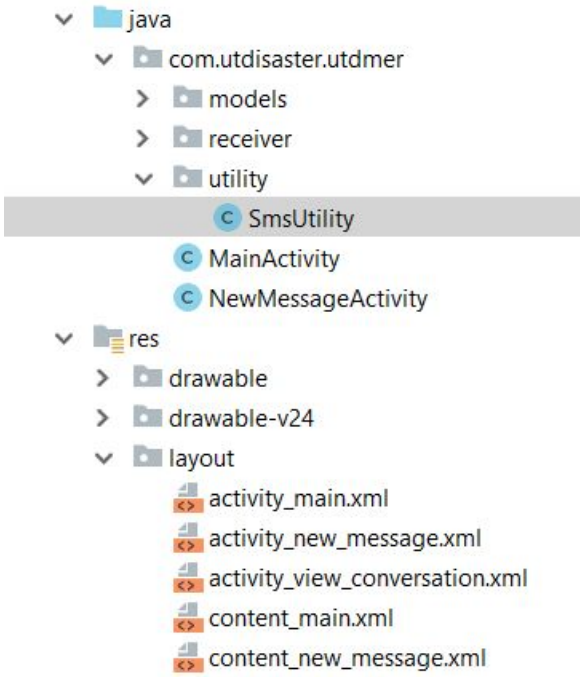
I.) Design Pattern: (SmsUtility.java)

- Created SmsUtility class to manage interfacing with messages
- Model View Controller - ('java' and 'res') → The 'Model (primary component in the pattern) is represented and under the java folder, and the 'View' is contained under 'layout'.



- Extract Method - getAllMessage method added to reduce duplication
    - After refactor

```
// Get SMS messages
public static List<Sms> getInboxMessages(Context context) {
    List<Sms> messages = getAllMessagesInMemory(context);

    // Sort messages by timestamp
    Collections.sort(messages);
    // Most recent message first
    Collections.reverse(messages);
    // Sort the messages and populate conversations hashmap
    sortMessagesIntoConversations(messages);

    return getRecentMessagesFromConversations();

}
```

-

- Before Refactor

```
// Get SMS messages
public static List<Sms> getSmsInbox(Context context) {
    ContentResolver contentResolver = context.getContentResolver();
    // Request sms messages
    Cursor smsCursor = contentResolver.query(Uri.parse("content://sms"), null, null, null, null);
    ArrayList<Sms> messages = new ArrayList<>();

    // process received sms
    if(smsCursor != null) {
        // verify cursor is valid and in good state
        int indexBody = smsCursor.getColumnIndex("body");
        if (indexBody < 0 || !smsCursor.moveToFirst()) {
            return null;
        }
        do {
            // Parse cursor data to build sms obj
            Sms sms = parseSmsCursor(smsCursor);
            messages.add(sms);
        } while (smsCursor.moveToNext());
        smsCursor.close();
    }
    if(messages.isEmpty()){
        return null;
    }
    // Sort messages by timestamp
    Collections.sort(messages);
    // Reverse list to display most recent message on top
    Collections.reverse(messages);

    conversations = new HashMap<>();
    for(Sms message: messages){
        ArrayList<Sms> prevMessages;
```

-

- Rename method- getAllMessages renamed to getAllMessagesFromMemory to avoid ambiguity and confusion

```
147     -
148     -   public static List<Sms> getAllMessages(Context context){
    117 +   public static List<Sms> getAllMessagesInMemory(Context context){
149 118         ContentResolver contentResolver = context.getContentResolver();
150 119         // Request sms messages
151 120         Cursor smsCursor = contentResolver.query(Uri.parse("content://sms"), null, null, null, null);
```

-

II.) Test Classes with Unit Test Cases:

We have two test classes:

Under app/main/test:

1. SmsTest, which tests
    a. compareTo
    b. equals
    c. toString
    d. hashCode
2. SmsUtilityTest, which tests
    a. sendMessag
    b. parseSmsCursor

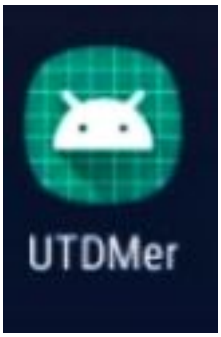
III.) Instructions:

- Building the Software:
    - ❏ Setup:
        - Ensure test device has the SdkVersion equal to or greater than 26

- Using the Software:
    - ❏ Refer to the “UTDMer App User Manual” for specifications

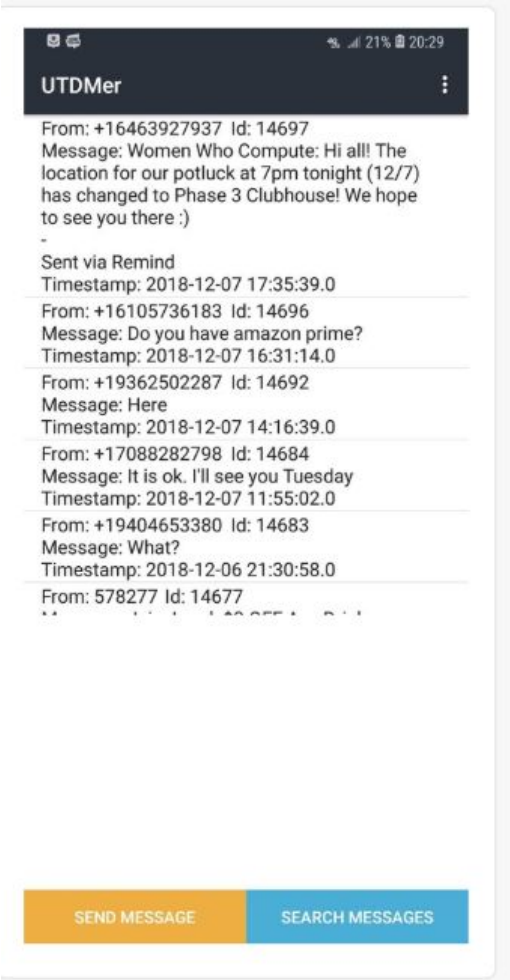
UTDMer App User Manual

An SMS Messaging App Utilizing some Basic Functionalities

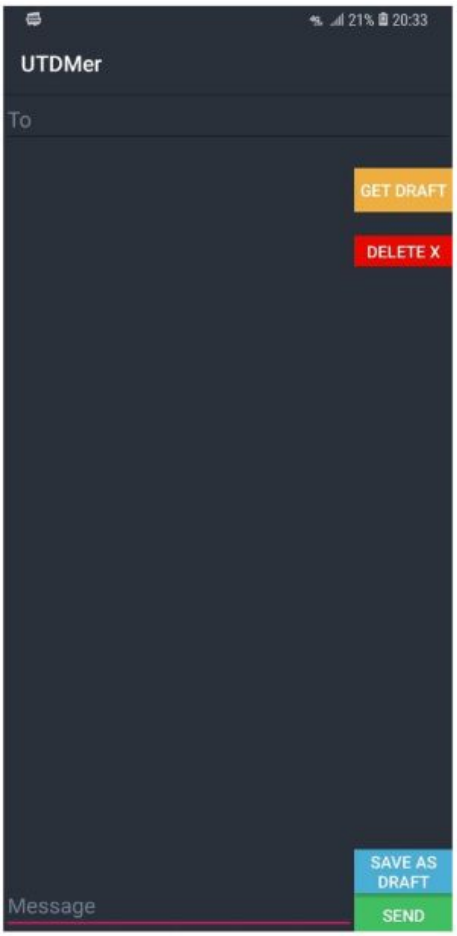


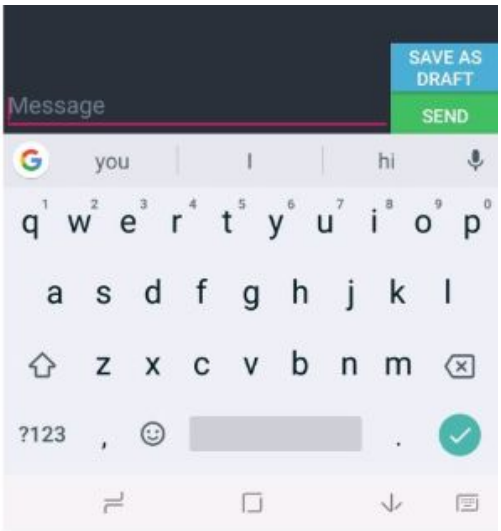
I. Screens

- Inbox View:

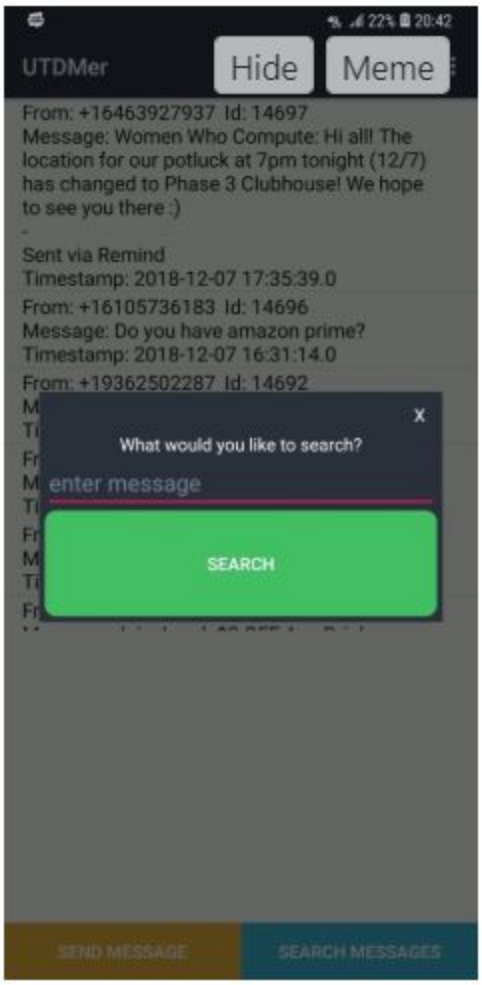


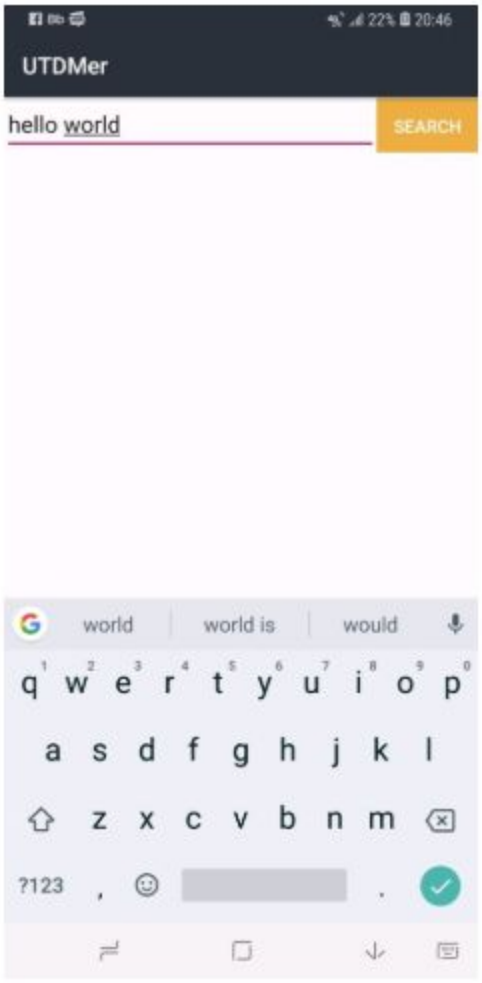
- Send Message:

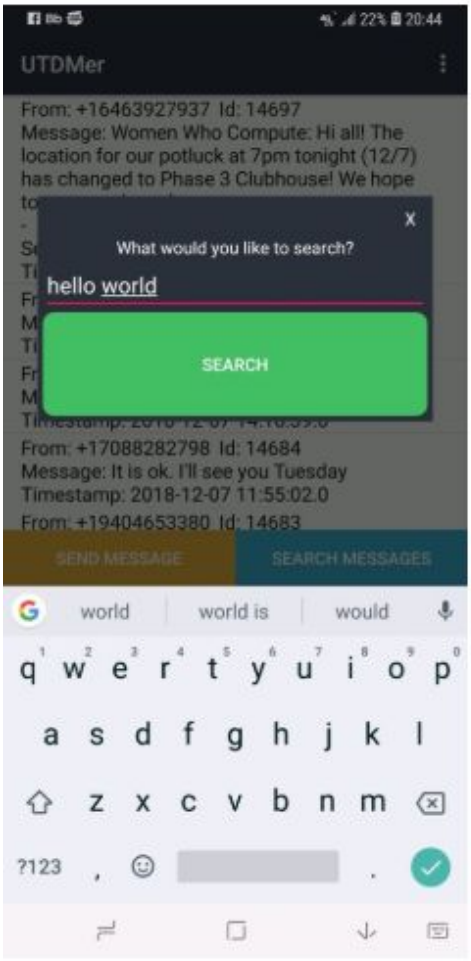




- Search Message (for a saved draft):

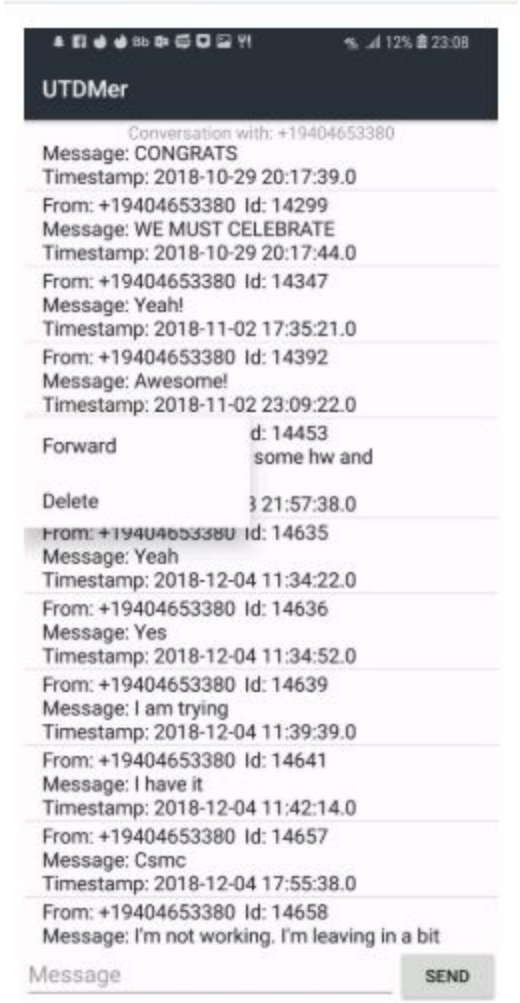


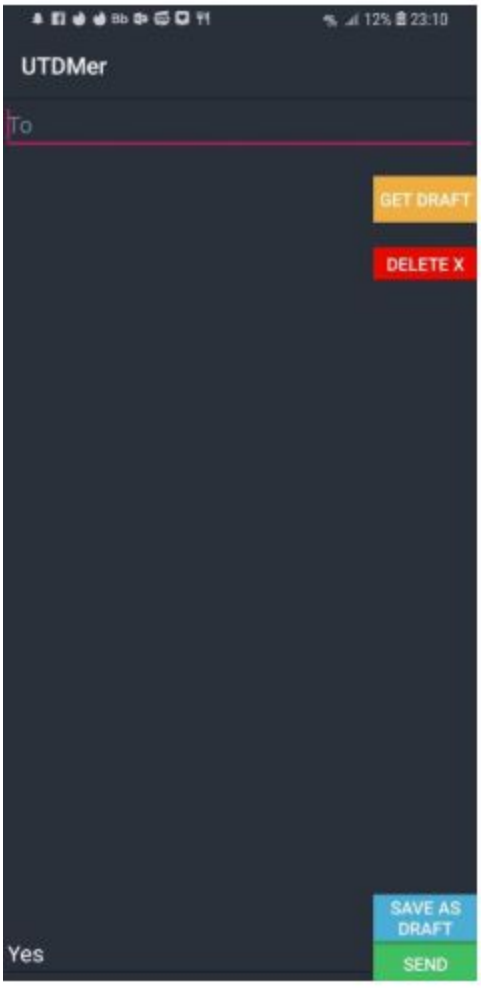




- Forward a Message:

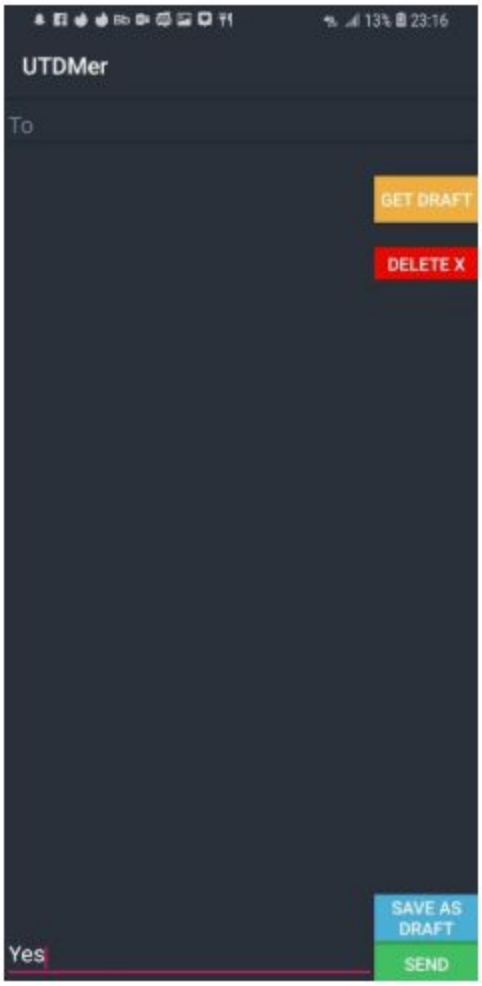
*Note: You can forward a message only from within the conversations, and not from the inbox!

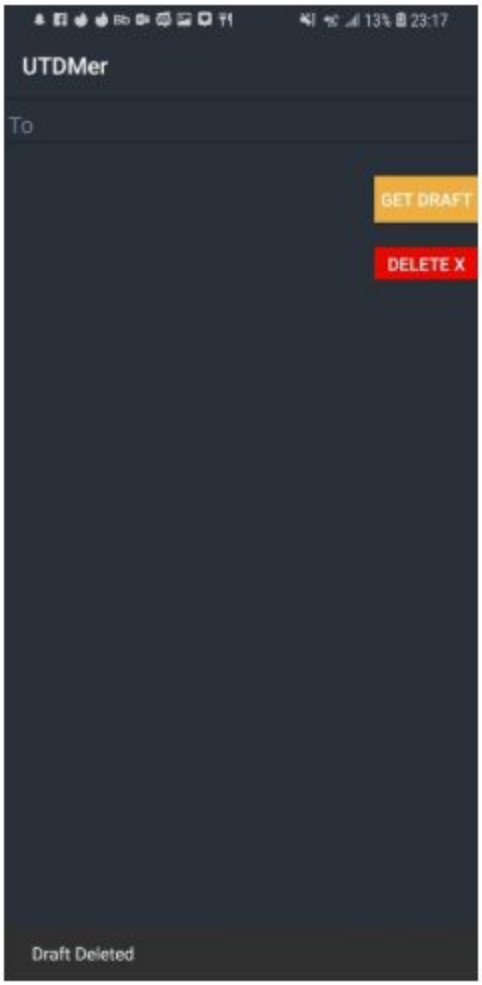




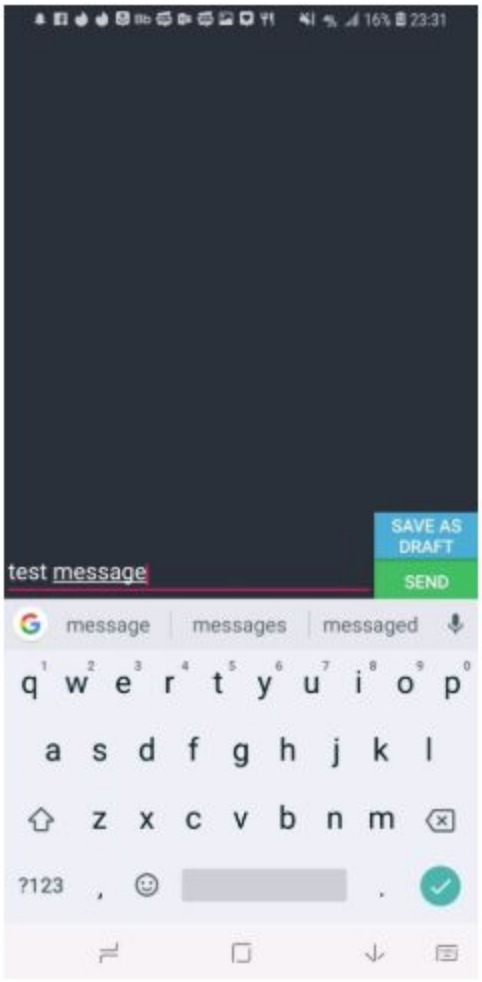
- Delete a Message:

*Note: You can delete a message only from within the conversations, and not from the inbox!





II. Save Message as Draft:



III. App Features

- View and edit messages
- Save message as draft
- Send & receive messages
- Delete messages
- Reply to messages
- Forward Messages
- Search messages
- List Views
    - Inbox (sender contact, sender id, message, and timestamp)
    - Conversation View