**Group setup-**

* Capabilities and preferred responsibilities
  + James –
  + Jorge-
  + Justin- Scheduling,
    - Current Status- Attempting to schedule interview for m or w after class 5:20-5:50
  + Ricardo-
  + John-
* Preferred collaboration software –Github
* Server setup
* Android studio

**Project 1- Option 1: Secure Messaging Android App**

Our company wishes to inquire about the creation of a mobile messaging app. Due to the nature of our business and clientele, we regularly exchange highly sensitive data and information. Because of this, we wish for an app to be developed with several security options for any message (e.g. - text message, picture, etc.) that our employees or clientele may send or receive. Below is a list of possible security options we would like to see. First and foremost, however, we would like to indicate that no messages should EVER be stored on our (or any) servers. While we understand that a back-end may need to be utilized for the transfer of messages (if it is possible, we would like no back-end to be utilized for that actual transfer of messages), we see storage of this data anywhere as an unnecessary security risk. Also, we understand that each phone will need an identifying account of some sort to not only verify the identity of a user, but also to identify where messages need to be sent. Our personal solution for this account is the following. An account cannot be registered for through the app by a normal user. An account may only be created by a system admin (who will be stationed at one of our company’s headquarters and only register an account after an individual passes our security checks). For a username, the account registration software shall generate a unique random number between 10 and 15 digits long. The user will also be given a randomly generated password that must be at least 16 characters long and contain at least: one lowercase letter, one uppercase letter, one number, and one special character (!, @, #, or $). Each password should also be unique. These usernames and passwords should be stored using normal security techniques (e.g. - only the encrypted version of a password should be kept on the server). Users should only know, and be able to see, IDs as the sender/receiver of their messages. (If message transfer is utilized without using a back-end, then a device ID -or other necessary information- may also be stored in the account so senders have a way of finding the device(s) the receiver is using in order to connect to it). For each sent message, a list of security options may be selected and implemented as part of the sending of that message. As many security options may be specified as the user sees fit, but at least the message Time-Out Deletion security option must be specified. A sent message cannot (and must not) be saved on the sender’s mobile device.

**Message Time-Out Deletion**

For this option, when a message is sent a deletion timer should be specified (not to exceed 5 minutes). The timer will begin to countdown once the message is opened and viewed. When the counter reaches zero, the app should automatically (instantly, atomicly, and unstoppably) delete the message. Upon the message’s deletion, a reply message should be generated and sent back to the sender (this reply message should not contain any security options).

**Message Encryption by Key**

For this option, a message should be encrypted before sending. The sender should provide an alphanumeric encryption key (no less than 15 characters long) to the app at the time the message is to be sent. Only the encrypted message should be sent to the receiving mobile device. When the recipient attempts to open the encrypted data, the recipient will be prompted for the encryption key. The recipient will give a key, which will be applied to the message and then the message should be displayed. As a recipient may accidentally type a key incorrectly, the recipient should be asked if the message was unencrypted successfully after the key is applied and message is displayed. If the answer is “yes”, then no further action should be taken by this security option and the Message Time-Out Deletion security option shall begin. If the answer is “no”, then the message should be re-encrypted with the same key the user just entered, then reprompted to enter a security key. The recipient should be given only 3 chances to unencrypt their message. If they fail the third time, the message should be deleted and a reply message should be generated and sent back to the sender indicating that the message was unsuccessfully unencrypted (this reply message should not contain any security options). In NO WAY should this encryption key be kept, saved, or sent. It should be assumed that the key has been shared previously (or will be shared in the near future) between the sender and receiver of the message.

**Message Encryption by Pattern**

For this option, a message should be encrypted before sending. The app should contain a graphical display to take a pattern as input (possibly similar to the display used to unlock a mobile device for pattern-type password). The sender should provide a pattern on the display to the app at the time the message is to be sent. The pattern should be mapped, in some way, to an encryption scheme (possibly an alphanumeric key). The pattern should be at least 10 points/edges long. Only the encrypted message should be sent to the receiving mobile device. When the recipient attempts to open the encrypted data, the recipient will be prompted to input the pattern. The recipient will give the pattern, which will be applied to the message and then the message should be displayed. As a recipient may accidentally input a pattern incorrectly, the recipient should be asked if the message was unencrypted successfully after the pattern is applied and message is displayed. If the answer is “yes”, then no further action should be taken by this security option and the Message Time-Out Deletion security option shall begin. If the answer is “no”, then the message should be re-encrypted with the same key the user just entered, then reprompted to input the pattern. The recipient should be given only 3 chances to unencrypt their message. If they fail the third time, the message should be deleted and a reply message should be generated and sent back to the sender indicating that the message was unsuccessfully unencrypted (this reply message should not contain any security options). In NO WAY should this pattern be kept, saved, or sen. It should be assumed that the pattern has already been shared previously (or will be shared in the near future) between the sender and receiver of the message. Note, that if the Pattern and Key encryption options are both selected, the recipient should be prompted to input both the pattern and key, the message should be unencrypted by both, and the message displayed before asking if the message was successfully unencrypted (still only 3 chances should be given to unencrypt the message successfully). The above security options are only suggestions and all or some of them (at least the Message Time-Out Deletion) may be implemented. It would be greatly appreciated if the developers were able to also implement their own ideas for security options as well and present them to us. We hope that this app will be able to help our employees and clientele better communicate with each other. This would greatly increase our company’s ability to operate more efficiently and effectively as well as better serve our clients.

**Deliverables due on March 26, 2017.**

• An introduction of the Patient Portal SRS

• A description of the Patient Portal

• Develop a prototype of the system in terms of interfaces showing the entire GUI and widgets. This must include the purpose of each page and each widget and the effect of interacting with each widget (thus showing the navigation among pages).

• Develop/consolidate a UML use case diagram showing all identified use cases, along with a brief description of the use case diagram.

• For each use case, build a sequence diagram for its normal scenario.

• Build/consolidate a UML class diagram for the whole system, showing all identified classes, their attributes and operations, all actors, relations, and multiplicities. Include a brief description of the class diagram.

• Build a state machine for at least two classes that display interesting behavior. Include a description for each state machine diagram.

• Identify and document non-functional requirements.

• Create a backlog to containing all use cases (stories). Write a brief description of each story and assign story points to each story. All the stories will be prioritized.

• Document stakeholder meeting minutes as an appendix to the SRS.

**Meetings with Stakeholder (Customer)**

For this project, your TA, John Heaps (sey967@my.utsa.edu), will serve as your main stakeholder/customer. As a real-life customer, he may not know or be able to articulate what they really want. You must communicate with him and help him decide. Do not assume your customer knows what a complete system looks like. You must arrange at least one meeting with your customer before project 1 due date. The format of these meetings is question & answer.

1. Every meeting with the customer must be scheduled at least 48 hours in advance.

2. You shall send the customer a questionnaire 24 hours in advance of the meeting.

3. The length of each meeting should be between 30 and 60 minutes.

4. All group members must be present for the meeting.

5. The customer will provide responses to your questions during the meeting. You must take notes during the meeting. You must submit a copy of the meeting minutes as an appendix of the SRS

**Questionnaire/Questions for TA-**

* App name?
* Design style?
* Messaging style(sms?, internet?)IP messaging
* Where should the app be accessed from(icon on screen, dummy icon, link to access app)?
* On failed login, what would you prefer the app do? (lock till admin resets it, time lock)
* How should the app flow when proceeding from opening through to messaging(login->contacts/id entry->message->security options->sent->contacts/id entry)
* Is storage on back end until receiving device is available to receive message an acceptable method?(IP messaging)
* Who manages the admin accounts? Are there any other capabilities they should be have?
* Are there any other features you would want implemented in the optional security options list?
* Confirm unlock like pattern password
* Confirm desired encryption scheme for pattern unlock

**Requirements Checklist: F-Functional, NF-Non Functional**

* F- Encryption method must be developed(check with unencrypted retrieval methods)
  + NF-Key for messages should not be kept, saved, or sent(sender and receiver should have set up the key previously outside of the app)
* F-Send Message
  + F-Message must be deleted after sending
  + F-Security optional security options list
    - F- Time-Out Deletion must be specified
    - F- Encryption by key-Message should be encrypted before being sent
      * NF- Only the Encrypted message should be sent
      * F-Sender is prompted to provide an alphanumeric encryption key(no less than 15 char long) to the app at the time of sending
    - F- Encryption by Pattern- Message should be encrypted before sending
      * F-Graphical display to take a pattern as input(like phone unlock)
        + NF-Pattern should be mapped to an encryption scheme(possibly an alphanumeric key)
        + NF-at least 10 points long
      * F-Only encrypted message should be sent
    - Other options(time limit to read the message?)
* F-Receive Message
  + F-Message Timer on read
  + F-Delete Message after timeout
  + Un Encryption Methods
    - F-Receiver should provide key to read encrypted message
      * F-Key is applied to message and decoded
      * F-Prompt for confirmation of correct decryption
        + F-if answer is yes-begin timeout deletion
        + F- if no- the message should be re-encrypted with the same key the user just entered, then re-prompted to enter a security key
        + NF-3 chances

NF-On 3rd failure- delete message

NF-Send failure message to original sender (no security options)

* + - F-Receiver should provide a pattern to read encrypted message
      * F-Pattern is applied to message and it is decoded
      * F-Prompt if message was decoded successfully
        + F- if answer is yes –begin timeout deletion
        + F- if no, re-encrypt password and prompt for next attempt
        + NF-3 chances

NF-On 3rd failure- delete message

NF-Send failure message to original sender

* + - F-Receiver should provide key and pattern in event both were selected by sender
      * NF-The message should be unencrypted by both
* NF-Messages should not be stored on any server or anywhere else
* NF-No back end if possible(otherwise only for use in transferring message)
* F-User/Phone account
* F-System Admin Account-Only way to create user phone account
  + NF-User Must have Security clearance for admin to create account
  + F-Database containing All users and their Security status which will be used when deciding if an account can be created
* F- Username- Randomly Generated by account registration software
  + NF- 10-15 digits long
* F- Password- Randomly Generated password
  + NF- at least 16 char long
    - NF-Contain at least:
      * 1 lowercase letter
      * 1 Uppercase letter
      * 1 number
      * 1 special char
  + NF-Unique(should not match any other)
  + F-Should be stored using normal security techniques
    - NF-Only the encrypted version of a password should be kept on the server
* F-User should only be able to see the IDs as sender/receivers of messages
  + F-(without back end)device id or other necessary information may be stored in the account for connection purposes