**HW6: First Implementation**

**by Team Eupheme:**

Zack Anderson, Adam Clarke, Eric Happe, Albert Le, Kevin Ngo, Jubal Norman, James Tooze and Jacques Uber

**for Customer:**

Kevin Tang

**on Project:**

SMS+

**List of User stories:**

**1) receive message**

**2) send message**

**3) mass message**

**4) read old messages (inbox)**

**5) delete message**

**6) ping of success**

**7) decrypt message**

**8) encrypt message**

**9) import / send key**

**10) sign keys**

**11) user interface**

**User Story #1: Receive Message**  
  
**Requirements:**  
The SMS+ user receives and is able to read a SMS+ protocol message.  
  
due: week 1  
  
1. Receive file from another device.  
Should take minimal time to implement if done after send task 3.  
  
2. Open received file and display relevant information  
Should take minimal time to implement.

**Programmers:** James Tooze and Zack Anderson

**Problems encountered:** Lack of knowledge programming for the Android platform and using github, both remedied through research (Internet).

**Time taken:** approximately 2 hours, most of which was learning how to setup the Android programming platform (Eclipse with Android plugins) and other research

**Current Status:** implemented, minimal hands-on (not automated) testing

**Left to complete:** N/A

For the user story of receiving text message we (James Tooze and Zack Anderson) were able to find a viable open source application that already did this; namely, the text messaging packages of the Android operating system. We were able to pull the relevant pieces required to receive and read text message from this and have implemented our user story using this code.

This task took us only a few hours spread over two days to implement. We had first considered approaching this problem from scratch but our research showed us that it would prove much easier and quicker to use the existing system. From there we only had to find the relevant sections we needed.

The current status of this user story is complete, barring any errors when attempting to interact with the other user stories, all that’s left to do would be a comprehensive test to locate and remove any possible bugs.

**User Story #2: Send Message**  
  
**Requirements:**  
The user is able to send a SMS+ protocol message to another SMS+ user.   
  
due: week 1  
  
1. Accept user input  
Should take a pair of programmers minimal time to implement, time extension may be necessary if programmer does not know how to read input from device.  
  
2. Format input to file  
Should take the pair of programmers minimal time to implement.  
  
3. Send file to target device  
Day or two to research best means of implementation, 1 day to implement the chosen design. 3 days total.

# User story #4: Read old messages

**Requirements:**  
The user’s SMS+ client is able to read messages they have received earlier on their phone.  
  
Due: week 1  
  
1. Store messages on phone  
Should take very little time to implement, after file format is decided.  
  
2. Retrieve file from phone.  
Again, should take minimal time to implement.  
  
3. Open file and display relevant information.  
Very simple to implement after basic send/receive is finished.

**User Story #11: UI**

**Requirements:**  
**Create user interface in which the user interacts with the SMS+ system.**

**due: week 1**

**1. Display relevant data**

**Should display relevant data on each screen, for example on the send screen the UI should (minimally) display a textbox for inputting the destination phone number, a box to enter the message text, and a send button.**

**2. allow user to interact as necessary**

**Should allow the user to use the hardware or software keyboard to input into text boxes, and use the touchscreen or directional-pad as well as other buttons to navigate, make changes and selections. Diagrams:**

Was the spike or diagram useful? Why or why not?

* The spike (installing Eclipse with Android plugins) for the receive user story was useful since it allowed us more time to understand how to program for the Android platform
* The UML sequence diagram was not so useful because understanding how a basic SMS text message would be sent isn’t really worthy of a diagram (IE it’s easy to understand)

Were there any diagrams that you *wish* that you had? Why or why not?

**Refactoring:**

**Integration Testing:**

User story #1: We have not had need to conduct any unit tests yet as this section proved to be quite rudimentary. We did however have some difficulty locating the exact sections of code we needed within the larger scope of the operating system.

**Customer Meeting**

Our customer spoke with our team in class rather than via email; the customer was reasonable about the work that should be done, and the timeframe in which it is due. We renegotiated with our customer on the priorities of the project: we felt as though we should get the basic SMS (send, receive, inbox, delete) working before we began implementing the encryption and ping of success parts of the project.

Questions to customer:

**Priorities and Work Breakout (for coming week):**

Implementation of project will be done using the Android API and SDK (therefore the Java programming language).

Implementation of mass message and delete message

Priority 2

Assigned members: Zack and James

Implementation of ping of success

Priority 1

Assigned members: WHO WANTS THIS?

Implementation of decrypt and encrypt message

Priority 2

Assigned members: WHO WANTS THIS?

Implementation of import, send and sign keys

Priority 1

Assigned members: WHO WANTS THIS?

**Team Responsibilities:**

* Implement User Story 1: James Tooze and Zack Anderson
* Implement User Story 2: Jacques Uber and Kevin Ngo
* Implement User Story 4: Jubal Norman and Albert Le
* Implement User Story 11: Adam Clarke and Eric Happe
* Diagrams and Refactoring: (whole team)
* Integration Testing: (whole team)
* Customer Meeting and Revised Estimates: (whole team)
* Priorities and Work Breakout: (whole team)
* Compilation and Formatting: James Tooze