



TicText Project Documentation

Version 1

Terrence Katzenbaer, Kevin Chen, Chengkan Huang, Jack Arendt, Georgy Petukhov ¹

May 3, 2015

¹Ordered by primary writer, then repository contributors sorted by descending number of commits.

Contents

1	Goals of the Documentation	2
1.1	Final Documentation	2
1.2	Formatting Guidelines	2
2	Welcome & Getting Started	3
2.1	Prerequisites	3
2.2	Setting up your Xcode project	3
2.3	Configuring TicText's Facebook integration	3
2.4	Cloud Code	4
3	What is TicText?	5
3.1	Brief Description	5
3.2	Motivation	5
3.3	Risks/Challenges	5
4	Formal Software Development Process	7
5	Requirements & Specifications	9
6	Architecture & Design	16
6.1	Expiration Picker	17
6.2	Sequence Diagrams	19
7	Future Plans	23

1 Goals of the Documentation

You should write your documentation as though you are explaining your project to a new developer who is about to join your team. You should include enough details so that he or she can get started quickly. On the other hand, the documentation should not overwhelm them with too much information that can be easily discovered by reading the code. Your documentation should include pointers to important facts that you wished you had known when you started the project, e.g., limitations of a framework you used or a workaround for a nasty bug you encountered, etc. These are things that people cannot figure out quickly unless it has been documented.

1.1 Final Documentation

The documentation is worth 20 points. It needs to include the following items (you can use this as a guide for the table of contents):

You should be able to re-use material from the homeworks and your writing assignments (if you are taking CS429) while preparing the documentation.

1.2 Formatting Guidelines

1. All class, method names, code must be in **monospace font**. Use a font like Courier, Monaco, etc
2. All figures must be labeled.
3. Do not draw the diagrams by hand – use a diagramming tool.
4. Your documentation should be at least 5 pages but not more than 15 pages.

2 Welcome & Getting Started

Hello, reader and possible new employee at the prestigious TicText organization. This document exists to help familiarize you with this project and its history. We will first introduce what TicText is, our motivation, and even the biggest risks/challenges we face. Then, you will learn the software development processes we follow. Next, you will read a brief overview of the project's requirements and specifications before diving into the current architecture and design. Finally, we hope a brief outline of future plans will help you find what you want to work on.

Let's get started, Tic Tok!

2.1 Prerequisites

- Xcode 6, preferably the latest version
- iOS 7

2.2 Setting up your Xcode project

1. After downloading the latest copy of the repository, install all project dependencies from CocoaPods by running,

```
cd TicText-iOS
pod install
```

2. Open the Xcode workspace at `TicText-iOS/TicText.xcworkspace`.
3. Set up a, or ask permissions to our development, TicText app on Parse.
4. Copy the environment's application id and client key into `AppDelegate.m`:

```
[Parse setApplicationId:@"APPLICATION_ID" clientKey:@"CLIENT_KEY"];
```

5. Finally, select the TicText target and go to **Build Phases**. Under **Upload Symbol Files**, update line 3 to point to your Cloud Code folder, if any. If you're not using Cloud Code, feel free to remove the **Upload Symbol Files** section.

2.3 Configuring TicText's Facebook integration

1. Set up a, or ask permission to our development, Facebook app at <http://developers.facebook.com/apps>.
2. Set up a URL scheme for `fbFACEBOOK_APP_ID`, where `FACEBOOK_APP_ID` is your Facebook app's id.
3. Add your Facebook app id to `Info.plist` in the `FacebookAppID` key.

2.4 Cloud Code

Add your Parse app id and master key to `TicText-iOS/CloudCode/config/global.json`, then type `parse deploy` from the command line at `TicText-cloud`. See the Cloud Code Guide for more information about the `parse` CLI.

3 What is TicText?

3.1 Brief Description

TicText is an iOS messaging app that allows you to send expire-able text with your friends. We call this kind of text "Tic", as in the sound of a clock counting down (Tic Tok, Tic Tok...). For each Tic sent to your friends, you will be able to set a time duration, which will be the amount of time before the Tic got expired and becomes unable for your friend to retrieve its content. That is to say, the recipient can only retrieve the content from our server within a certain time period after the sender sent the Tic. Here is an example of a possible scenario of using TicText:

JOHN: *[HE is drunk at 3am and sends a tic using HIS PHONE to MARY with a 60 minute expiration time. The tic reads...]*

Hey babe, I miss you so much and I really regret breaking up with you...

[MARY (while sleeping) receives a notification on her phone.]

MARY'S PHONE: Someone has just sent you a Tic! Swipe to read before it's too late.
Tic Tok...

[The next morning, MARY saw the notification and opened the app. The only thing left was...]

TICTEXT ON MARY'S PHONE: Sorry you've missed a Tic which expired on 4:00 am.

TicText would also let you to send a Tic anonymously - even if the recipient saw the notification in time, he/she still won't be able to see the sender's name.

We believe TicText is also capable of creating romance in many ways. But in the example above, well, hopefully TicText just avoid something awkward or even more.

3.2 Motivation

We would like to build something brings people joy and more importantly, works even with limited amount of users. So we gave up on the ideas related to sharing, social networking, or dating, and focused on single user or 1-on-1 experiences. I believe TicText could be something we, especially the younger generation, would enjoy. Also, building a mobile app with modern technologies will always be challenging and exciting. From my past experiences, I'm sure we could learn a lot and have fun as well.

3.3 Risks/Challenges

One of the biggest challenges would be building a good-looking UI with great user experience. There wouldn't be many features in TicText so UX will be the top priority with no doubts. In the meantime, we have to think about the security of all the Tics as well. The last thing we want is compromising user's privacy. In order to do that, we need more than beautiful code, but also coding in the "right way" using the "right tools". Another possible challenge is learning how to code and deploy on Parse if we choose to use it as our cloud platform.

4 Formal Software Development Process

The process followed by your team (make use of your essay if you are taking CS429) - (5 points)

Make sure to address the issues of iterative development, refactoring, testing and collaborative development (even if you are not using XP, you have to address these issues in your documentation).

In the software development world, every project ultimately descends into chaos. Teams proactively battle against this entropy using formal software development methods. These methods can slow a projects descent, but they can, when used effectively, stall the descent entirely. One method that we learned and applied last semester in Software Engineering was Extreme Programming. Extreme Programming is a branch under Agile Software Development methodology and derives many of its practices from Agile Software Development including fast iteration cycles, use cases, and user stories; but, Extreme Programming differs from a typical Agile Software Development methodology through pair programming and complete unit testing.

While my group last semester followed many of the practices of Extreme Programming, we did not adhere as strictly to pair programming. Why did the students in my group not desire pair programming, especially when I received multiple second-person accounts of groups that flawlessly utilized pair programming? Could group dynamic, individual skill, an unnecessary byproduct of our interspersed and erratic location on campus at any given time, or something else determine pair programming's necessity and success? I believe that all of the aforementioned concerns contribute in peer hesitation to adopt and consistently practice pair programming. Some group atmospheres do not support or encourage their members to admit unfamiliarity or inexperience. Other individuals feel their skill does not necessitate someone else watching over their shoulder as they crank out clean code—line by line. And the sheer size and scatterness of the campus make it difficult for students to easily coordinate a place and time for pair programming. Because of the reality of resistance to pair programming, my group this semester has adopted it on an as-needed basis. We hope that this compromise will let skilled individuals maintain their efficiency while also helping the bottom line—those that may not be experts in the technology.

To improve team efficiency and facilitate grading, the instructors frequently require students to use Subversion: a program (more accurately “a lifestyle choice”) that helps software developers maintain and track all changes to, or version, a code repository—similar to what I'm doing manually for this essay. My group this semester hopes to improve efficiency adjustments to our project's software development method by versioning our code with Git instead of Subversion. The biggest difference between Git and Subversion is how collaboration is organized. Subversion uses centralized remote repositories, which means that each collaborator syncs every change with a single master repository over the Internet. Git, on the other hand, uses decentralized repositories. In decentralized repositories, there is no

single master repository and each collaborator's local repository is autonomous. Because each local repository is autonomous, collaborators can reliably commit changes to their local repository even without an Internet connection. These changes can then be, when an Internet connection becomes available, pushed and pulled to and from other collaborators. This workflow makes Git's reliability far superior to Subversion. Both Subversion and Git let collaborators create branches to pivot development for a specific feature, but Git further augments collaboration through a different feature: repository forks. When a collaborator forks a repository, they create an isolated copy of the original repository. In this new isolated repository, collaborators can create, delete, and commit changes to branches without worrying about conflicting with another collaborator. In case something goes awry, projects using Git commonly designate a stable branch on the main repository to guarantee a safe point to revert. For a group that wants to maximize efficiency, Git offers our project unparalleled autonomy, reliability, and safety.

While each collaborator using Git has a local copy of the main repository, many projects backup their repository to a remote service, such as Github, and designate their repository on Github as their main repository. Don't be fooled—Github is more than just a service to host a remote Git repository; its website provides many tools to enhance efficiency. One of these tools is the pull request: a proven, efficient, and convenient practice to submitting, review, and merge collections of changes. Pull requests are similar to formal code reviews, a process software development teams use to ensure code quality in incoming changes. Prior to completing a feature, the submitter submits their changes for a formal code review where they receive critique and feedback from other developers. Often a feature must go through many iterations in this stage before becoming "production ready." While Github doesn't natively facilitate formal code reviews, my group is self-imposing a rule that each pull request must be reviewed and approved by two other team members before merging. Reviewers can use Github's web interface to review and comment on the pull request, its files, and even individual lines of code.

But before we can even write a line of code, we need to know what features we want and when we want them. To solve this problem, we create user stories and assign each story a story point value. We hope to relieve some of the stress and inherent instability with assigning a fixed man-hour value while still providing an accurate estimate to facilitate the planning and management of iterations. Through user stories, we hope to be able to accurately portray the needs of the users while also providing realistic time estimates to help us pace ourselves so that we may finish the project within the deadlines.

5 Requirements & Specifications

Requirements & Specifications of your project (make use of HW2 and HW3) - (5 points)
This should include the user stories and use cases which you implemented in your project.
You do not need to mention the requirements which were dropped.

Table 1: Iteration 2, Feb 7 - Feb 19

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
8	8	US1: As a user, I can use Facebook to login.	Unknown
3	3	US2: As a user, I want TicText to automatically sync and connects me with friends already on TicText	Unknown
2+	5	US12: As a user, I want to get a notification when one of my Facebook friends joins.	Unknown
13+	16	<i>Total Story Points</i>	

Table 2: Iteration 3, Feb 20 - Mar 5

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
6	5	US15: As a user, I want to see which of my friends are already on TicText when I join	Jack, Georgy
10	8	US3: As user, I want to see conversations in bubbles like SMS, WhatsApp, etc. (controller)	Kevin, CK
3	3	US11: As a sender, I want to send a Tic (text-only) to a hardcoded TicText friend (backend)	Terrence
3	3	US8: As a recipient, I will get a push notification when a new Tic has arrived.	Georgy
1	3	US5: As a recipient, I want to see the conversation view controller when I open the notification	Kevin
3	3	US6: As a recipient, I want to have to tap a new message before I can see its contents	CK
3	3	US7: As a user, when I'm not in a conversation view I want a tab controller to switch between Conversation List, Contact List, My Profile, and (Settings)	Kevin
3	3	US9: As a user, I want to be able to change my notification settings and access the Facebook page through the Settings VC	Jack
32	31	<i>Total Story Points</i>	

Table 3: Iteration 4, Mar 6 - Mar 19

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
8	5	US4: As a sender, I want a set a time expiration before I send the Tic. (backend & custom view element)	Terrence
8	5	<i>Total Story Points</i>	

Table 4: Iteration 5, Mar 30 - Apr 10

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
??	5	US: As a recipient, I want to see all my new Tics (without a sender) in an unread list above my conversation list	Jack, Kevin
??	1	US: As a recipient, I want to get sent to a conversation screen when I select an unread unexpired Tic from the unread list.	Jack, Kevin
??	1	US: As a user, I want my conversations with unread messages to be highlighted in the conversation list.	Jack, Kevin
??	3	US: As a recipient, a new Tic only show up in my active messages view if sent within 5 min from the last one; or it goes into the unread list.	Kevin
5	5	US: As a user, I want to see a contact list and be able to filter people by name.	Jack
3	3	US: As a user, I want to be able to send Tics anonymously.	Terrence
1	3	US: As a sender, I want to set the expiration time for the current Tic after tapping on the "expires in ..." button in the toolbar.	Terrence
1	1	US: As a sender, I want to choose if the Tic is anonymous or not within the toolbar.	Terrence
5	5	US: As a sender, I would like to have a toolbar to set the type of the Tic, expiration time, anonymous or not.	Terrence
??	27	<i>Total Story Points</i>	

Table 5: Iteration 6, Apr 11 - Apr 24

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
??	5	US: As a user, I want to be able to view and edit my display name and profile picture in Settings (controller)	Jack
??	5	US: As a sender, I want to select the image the way Facebook Messenger does. (after tapping on the image icon in the toolbar)	CK
5	3	US: As a recipient, I want a timer on my unread unexpired Tics in the messages view before it was tapped.	Georgy
??	5	US: As a sender, I want to take a picture and send it the way Facebook Messenger does after tapping on the camera icon in the toolbar.	CK
??	18	<i>Total Story Points</i>	

Table 6: After Iteration 6 (Last Iteration), Apr 25 - May 3

Story Points (<i>Actual</i>)	Story Points (<i>Estimated</i>)	Description	Assignee
??	3	US: As a recipient, I want to be able to differentiate between anonymous senders using their name (Anonymous Table, Anonymous Car, Anonymous Penguin, etc.)	Unknown
??	3	US: As a recipient, I want anonymous Tics from the same person to be in the same conversation.	Unknown
??	1	US: As a user, I want to be able to delete conversations and individual messages.	Unknown
??	12	US: As a user, I can see if there are unread Tics for me and how many of them in the banner view.	Kevin
??	12	US: As a user, I can view all new Tics (including unread ones and expired ones) in a dropdown view with a summary of the amount of unread and expired Tics.	Kevin
??	7	<i>Total Story Points</i>	

6 Architecture & Design

TicText is an iOS application built with Apple’s Cocoa Touch framework, Facebook’s Parse framework, and many other modules provided through CocoaPods (a dependency manager). The application, when joined with these frameworks, compose our entire infrastructure from front end to back end.

In the front end, the Cocoa Touch framework provides us with many inherent design patterns, such as Model-View-Controller and Chain of Responsibility for touch input, that trivialize most device interactions and let us focus on the application. The Parse framework enables our models to easily implement the Active Record pattern and provides out-of-the-box data caching.

In the back end, the Parse framework eliminates the tedious work of designing a schema and setting up an Apple push notification server. Hence, the entirety of our back end code is composed of triggers and stored procedures written in JavaScript.

1. Include multiple UML diagrams to show the important parts of your system (you must have UML diagrams).
2. Describe in a top-down manner the architecture of your system.
3. Include enough details about the design of your system such that anyone who refers to your documentation can understand the major components of your system and how they are related.
4. Describe how the choice of the framework influence the design of your system.

When a user uses TicText, they will interact with several major workflows:

1. Onboarding (Sign up/Log in through Facebook + Find Friends)
2. Viewing conversations and new Tics
3. Viewing a single conversation
4. Composing a new Tic
5. Receiving a new Tic

Check out Figure 3!!

List of Diagrams

1	Expiration Picker Class Diagram	17
2	Expiration Picker Sequence Diagram	18

3	US1 Sequence Diagram	19
4	US4 Sequence Diagram	19
5	US6 Sequence Diagram	20
6	US9 Sequence Diagram	21
7	US11 Sequence Diagram	21
8	US15 Sequence Diagram	22

6.1 Expiration Picker

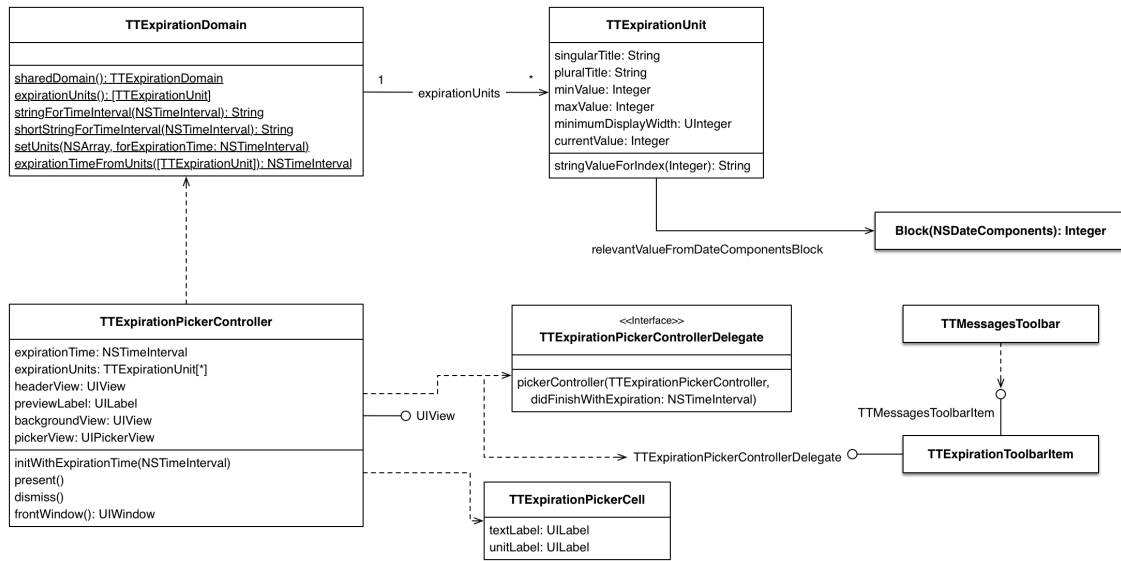


Figure 1: Expiration Picker Class Diagram

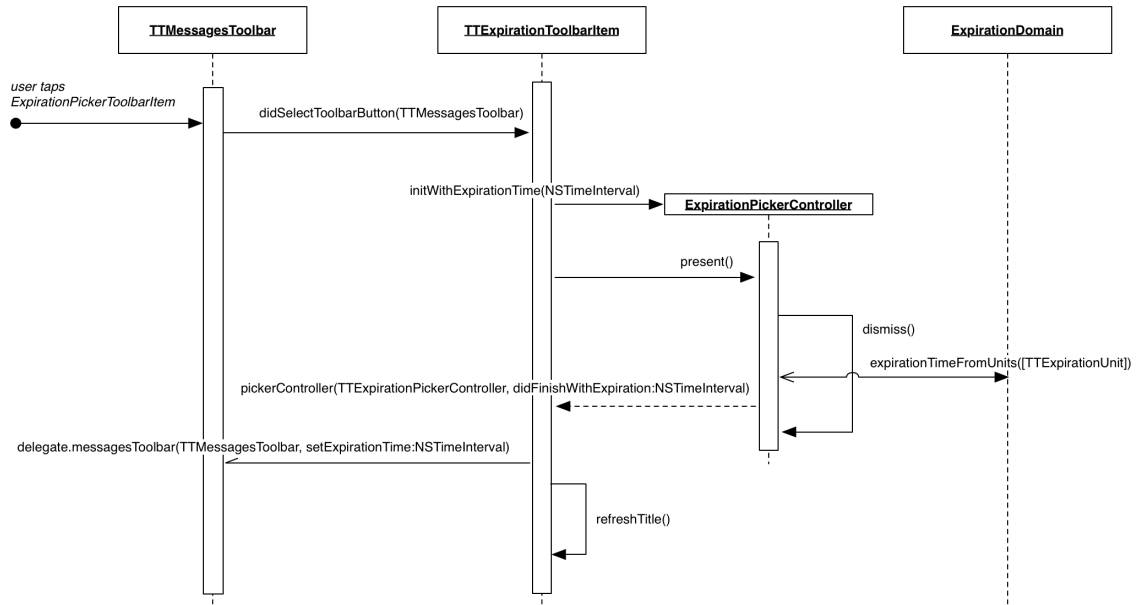


Figure 2: Expiration Picker Sequence Diagram

6.2 Sequence Diagrams

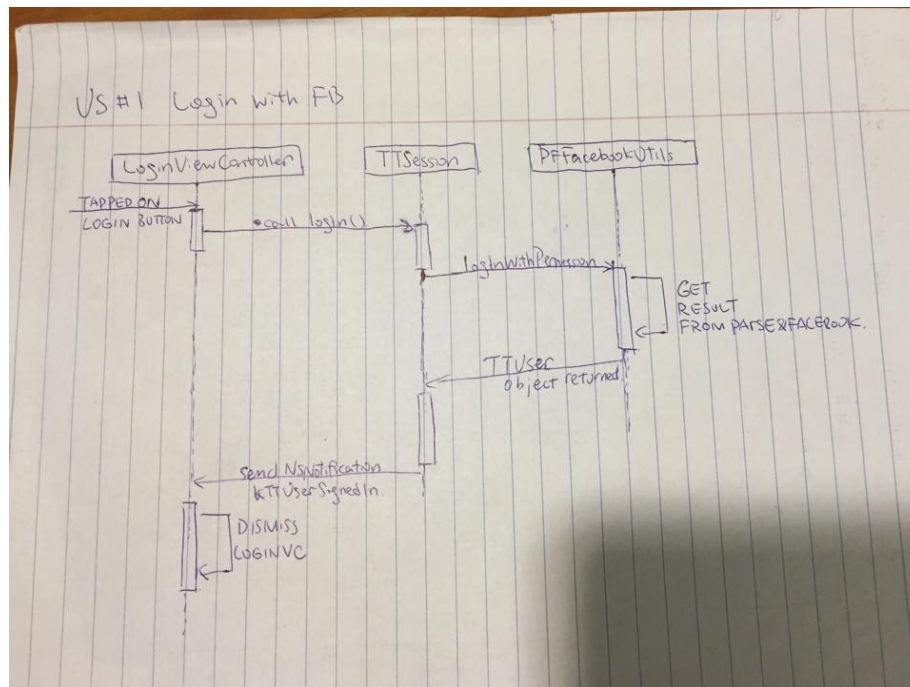


Figure 3: US1 Sequence Diagram

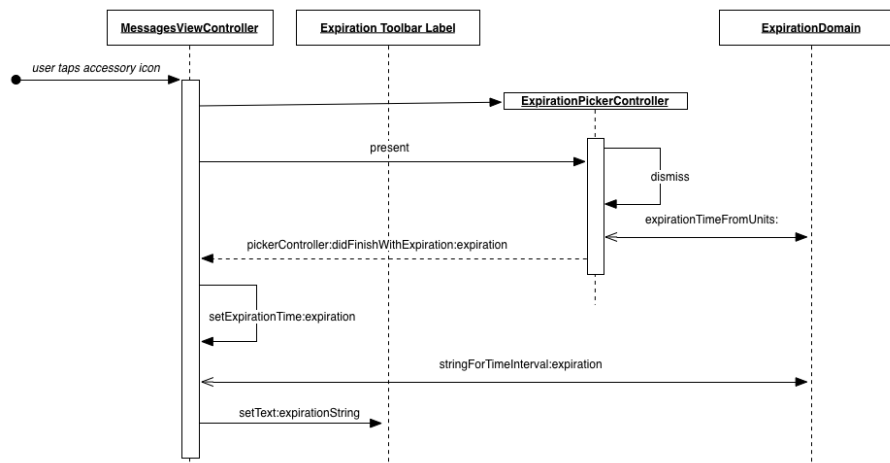


Figure 4: US4 Sequence Diagram

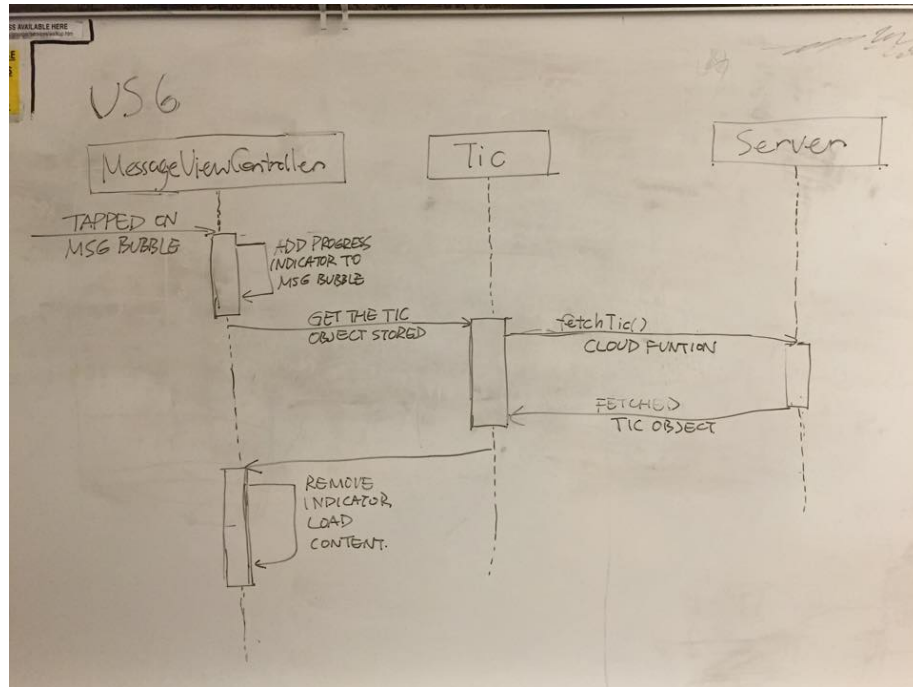


Figure 5: US6 Sequence Diagram

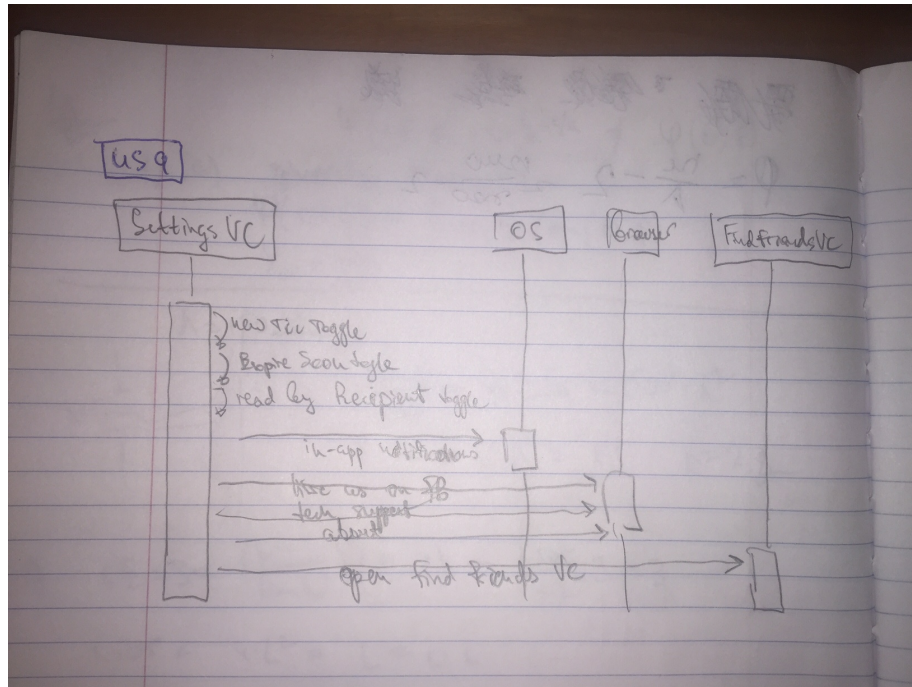


Figure 6: US9 Sequence Diagram

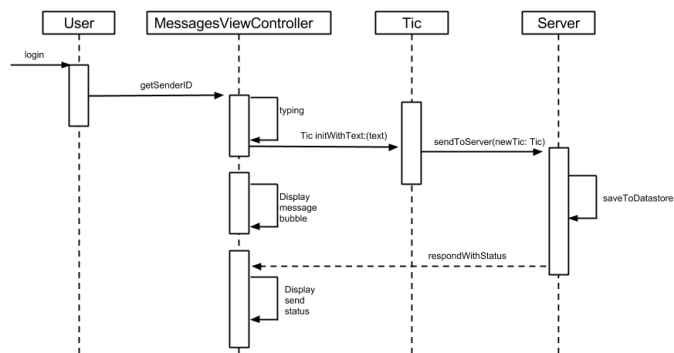


Figure 7: US11 Sequence Diagram

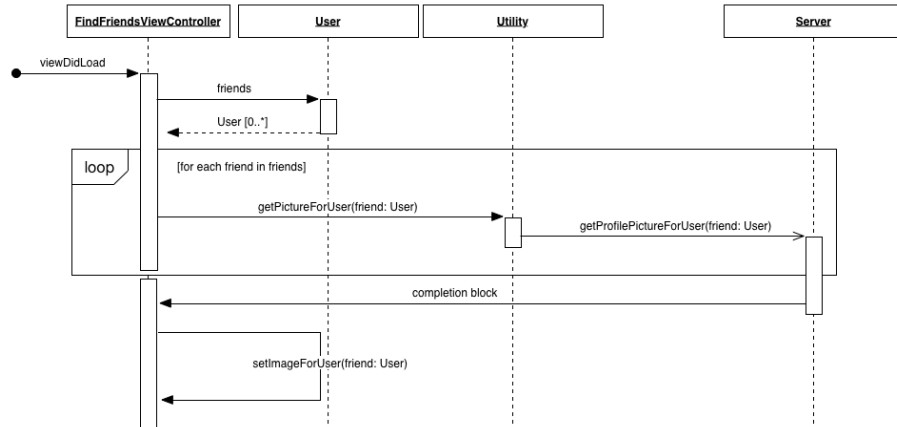


Figure 8: US15 Sequence Diagram

7 Future Plans

Future plans - (3 points)

Include personal reflections on the project and process by each team member. Describe what you learned.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(Terrence Katzenbaer)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(Kevin Chen)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(Chengkan Huang)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(Jack Arendt)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(Georgy Petukhov)

Table 7: Future Stories

Story Points	Description
5	As a user, I want to send a voice message Tics to a TicText friend.
3	As a user, I want to be able to set the app's color scheme in the in-app Settings
5	As a user, I want to be able to set and sync a background image in a conversation (MsgVC) with my friend
8	As a user, I want to be able to enter my phone number and have the app use my Address Book to automatically match us.
3	As a user, I want to be able to check someone's profile from inside the conversation view.
3	As a user, I want to be able to send a text with an invitation to join TicText to a person in my Address Book