TicText Project Documentation

Version 1

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

May 3, 2015

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

Contents

1 Welcome

Hello, reader and possible new employee at the prestiguous TicText organization. This document exists to help familiarize you with this project and its history. We will first introduce what TicText is, our motiviation, 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 Getting Started

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 that 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 one-on-one experiences. We 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. We're 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 aren't many features in TicText, so User Experience is a top priority. In the meantime, we have to think about the security of all Tics as well. The last thing we want is to compromise a user's privacy. In order to do that, we need more than beautiful code; we also need to code in the "right way" using the "right tools."

4 Formal Software Development Process

4.1 Motivation

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 project's freefall, or when used effectively stall it entirely. One method we learned and applied in Software Engineering I was XP, or Extreme Programming. XP is a branch under the Agile Software Development methodology and derives many of its practices from Agile Software Development including fast iteration cycles, use cases, and user stories; but, XP differs from typical Agile Software Development methodology through pair programming and complete unit testing.

While most groups in Software Engineering I followed many XP practices, fewer groups adhered strictly to pair programming. Why were many students repulsed by the idea of pair programming when the remaining groups reported success with it? 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? We 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.

In software development, investors and stakeholders want to see visible results and often pressure developers to focus on functionality. To please the stakeholders, beginner developers will write code that only meet the requirements. In school and personal projects, this procedure is often adequate to achieve full marks or a deployable product, but quickly deteriorates in a collaborative and large-scale projects. In such projects, new code is introduced and old code is refactored so frequently that it's inevitable for a code defect to appear. Under Agile Software Development methodology, development teams simultaneously use unit testing to verify code being committed and to guard against future refactors and design changes.

To improve team efficiency and facilitate grading, the University of Illinois instructors 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. An alternative versional control system to Subversion is Git. 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 project 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."

4.2 Our process

Because of the reality of resistance to pair programming, we have 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.

Our project uses XCode's native XCTest framework for unit testing, OCMock for comprehensive testing, and Automation for interface testing.

Our project hopes to improve efficiency adjustments to our project's software development method by versioning our code with Git instead of Subversion.

While Github doesn't natively facilitate formal code reviews, we self-impose 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.

Following Agile Software Development methodology, we are always refactoring the codebase. We refactor while coding, while writing tests, and while revisiting code when implementing a new feature.

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 *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

Table 2: Iteration 3, Feb 20 - Mar 5

Story Points *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

Table 3: Iteration 4, Mar 6 - Mar 19

Story Points *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

Table 4: Iteration 5, Mar 30 - Apr 10

Story Points *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

Table 5: Iteration 6, Apr 11 - Apr 24

Story Points *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

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

Story Points *(Actual)*	Story Points *(Estimated)*	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	*Total Story Points*	

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

1. Onboarding (Sign up/Log in through Facebook + Find Friends)
2. Setting the expiration time for a Tic
3. Composing a new Tic

List of Diagrams

6.1 Expiration Picker

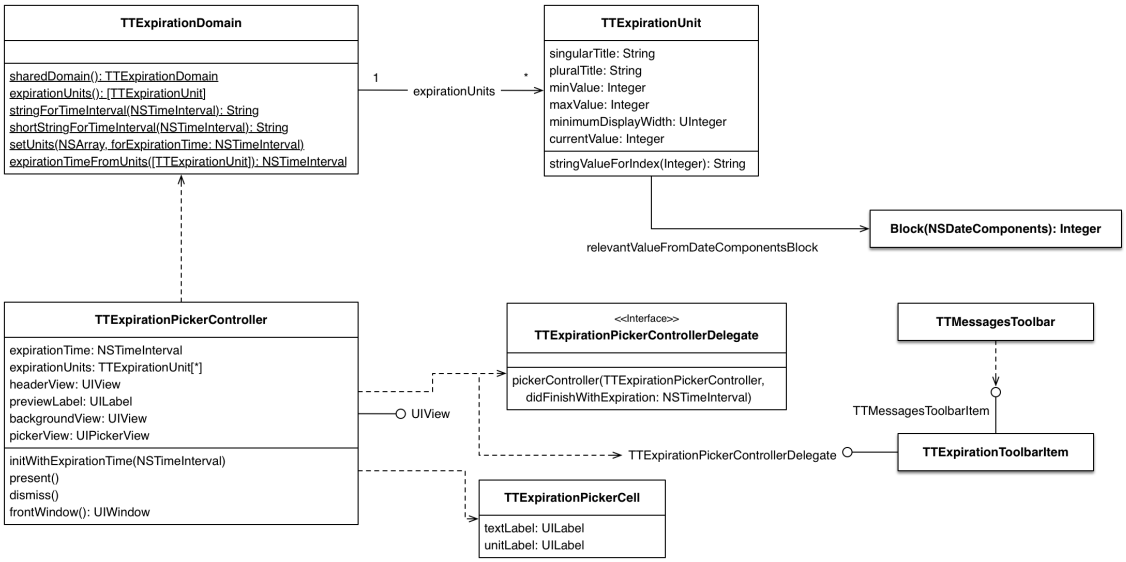

Figure 1: Expiration Picker Class Diagram

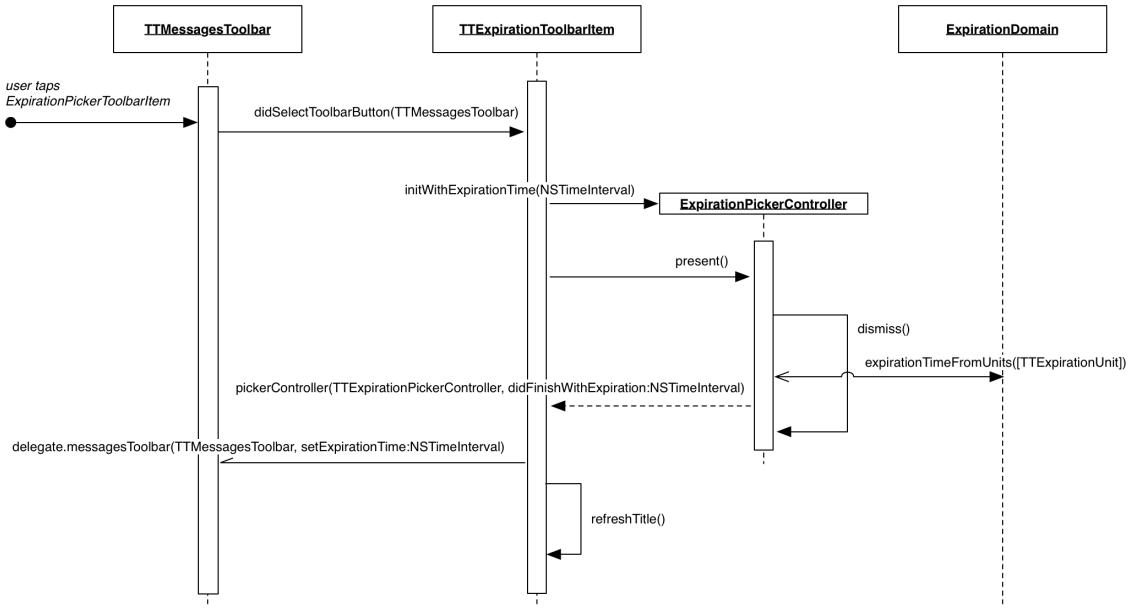

Figure 2: Expiration Picker Sequence Diagram

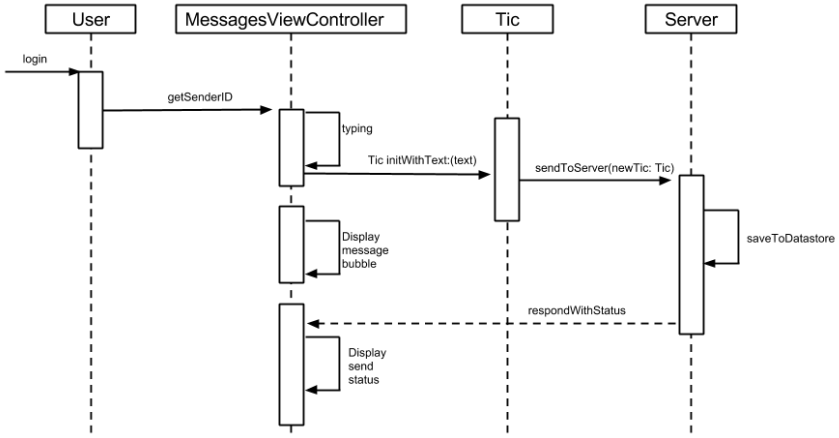

Figure 3: Sending a Tic Sequence Diagram

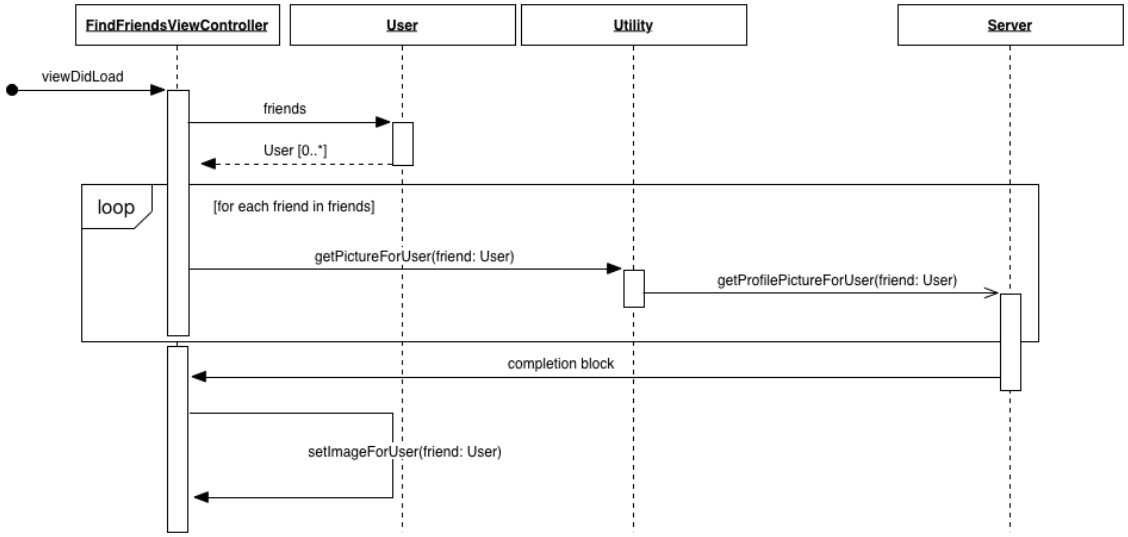

Figure 4: FindFriendsViewController Sequence Diagram

7 Future Plans

While developing TicText, some features had to be dropped. Here are some features that, if given time, we would have loved to implement to enhance the user's experience with the application.

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

Here are some personal reflections on the project and process by current or former team members:

> Working on TicText with the team was an excellent experience. It was fun working with a team with a variety of skills and where each member had something to contribute. In every weekly meeting, someone would bring something new that we couldn't wait to share at the next iteration presentation. On a personal level, I have enjoyed improving my teamwork and formalized software development skills. I will miss expressing myself through emoticons and the often moments of, "Huh!" when learning something new in code reviews. But ultimately, seeing this project grow from a single log-in view controller to the final product has given me the greatest satisfaction. (Terrence Katzenbaer)

> I learned Parse framework to facilitate our development of mobile backend. I spent time on considering potential security issues and anything that might affect user experience. I tried hard minimizing coupling between the views and controllers. In order to give a better UX, I created customized view with decent animations. In terms of process, we strictly followed XP. We held team meeting each week, discussing on unresolved issues and plan for the coming iteration.

For every new feature, have people working on branches and merge into master after pull request and code review. Had well documented code and tests with mocks. (Kevin Chen)

I learned a lot by working on this project, in terms of both iOS development skill and software engineering technique. Before this semester, I have never done any iOS development. Now, by learning from my experienced team members and online sources, I am more proficient in this field. I wrote code following consistent code convention with documentation and tests. We followed XP, always creating pull request before merging into master and reviewing/commenting others' code. And we have user stories done after every iteration.
(Chengkan Huang)

I learned how to incorporate my project into parse and use different testing frameworks. I worked on the FindFriendsViewController and the original conversation view. I also worked on creating the header above the settings view controller. I liked using parse, I thought it was a great framework, I enjoyed the code reviews, it made me a much better coder and more careful of what I was writing. Having branches that weren't merged into master after iterations ended up causing a lot of headaches down the road. (Jack Arendt)

While working on this project I learnt a lot about iOS development and about Parse framework. I found it useful to meet each week to discuss our work and plan for the iteration meeting with the TA (who is simulating a client). That definitely helped everyone to be on track with the current state work. Also using Github and its Pull Request system made code review possible. Everyone who has access to the repository can review the history of commits, which makes our work very transparent to an outsider. (Georgy Petukhov)