Talkatiel Software Requirements and Planning

Brendan Byers, Ryan Sisco, Iliana J, Aidan Grimshaw, Yufei Zeng
byersbr, siscor, javieri, grimshaa, zengyu

Contents

1 Product Description

The purpose of Talkatiel is to create an anonymous message board app. Users will be able to post posts and comments anonymously to each other. This type of communication is similar to Yik-Tak and 4chan. Users will be able to react and respond to other posts, along with reporting if the post in question goes against some rules. This is the basic idea behind our app. Here is a much more in depth description:

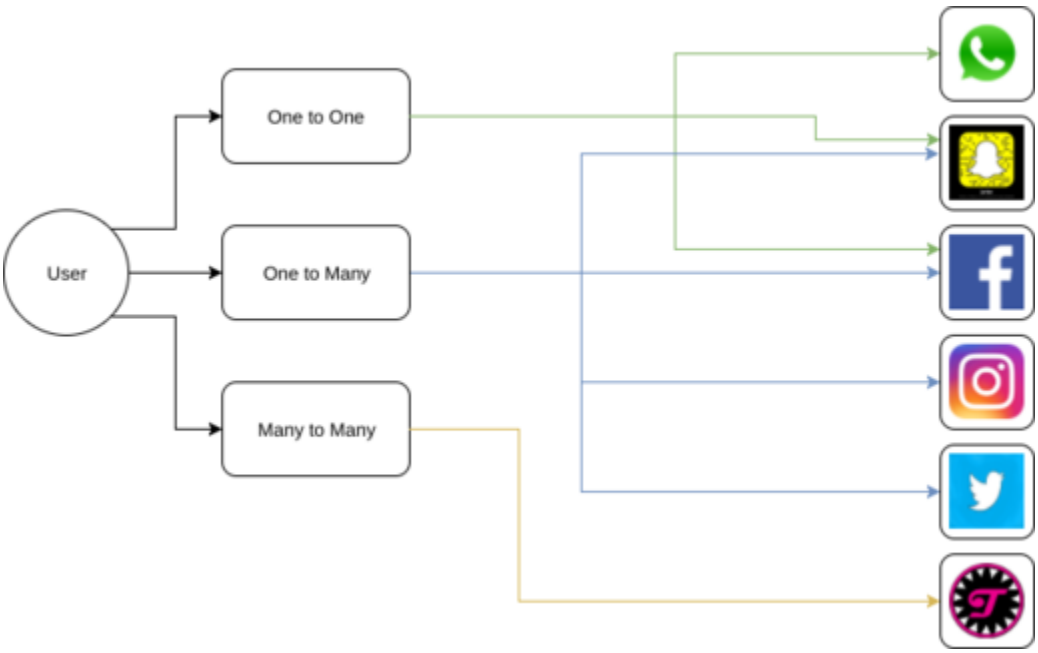

Figure 1: This describes the types of communication present on different platforms. Note that Talktatiel is the only Many-to-Many platform.

When people move to a new area, it can be difficult to interact with new people. People move away from their hometown and have an increased reliance on social networks such as facebook and twitter to feel connected to their friends and families at home. This results in people not feeling part of their new communities. The old communities have been left behind but people have trouble fully integrating into their new towns and neighborhoods. College students are extremely susceptible to this feeling due to the fact that since 1986 the number of out-of-state freshmen has doubled[2].

Talkatiel provides a safe way to engage in a community without actually meeting anyone. Users no longer need to fear going out to social events or community activities and dealing with terrifying or uncomfortable social situations. This application allows people to integrate themselves into a community without great risk to themselves. It can provide the deep seated need to be part of a community while also allowing people to be themselves without feel of social repercussion. Talkatiel solves an issue that other social networks have trouble solving. Traditional social networks are great at one to one or one to many interations, but have trouble with many to many interactions. Talktatiel allows many to many interactions that provide a sense of community and place for users.

Talkatiel will be a Portable Web App, or a PWA. This allows it to behave as an app does while also being much more accessible and easier to use for our end users. The web app itself will consist primarily of 3 elements: the service worker, the app shell, and the backend database. A service worker is a chunk of code that exists outside the website. It stays on a users device and works to cache and manage data for the web app. It will also allow us to send push notifications to the user. The app shell is the skeleton behind the web app. It is stored on the users device, allowing the web app to load quickly. It manages the basic user interface which is then filled with content from the server. The database will store all posts

along with identification and metadata information.

Previously this idea has been attempted by Yik-Yak. Originally released in 2013, Yik-Yak set out to solve many of the problems we are solving with Talkatiel. One of the biggest issues that Yik-Yak encountered was bullying and harassment on the platform[1]. The only moderation present in the app was an automod that deleted posts that had certain words. This did not effectively deal with hurtful messages, only invited people to figure out better ways to circumvent the moderation. To solve this, Talkatiel will use both word filtering and actual people to moderate. As seen on sites like reddit, using trusted users to moderate other users can be a strong and effective tool.

To build Talkatiel we will use Polymer for the user facing side. It has been proven to be a simple and fast portable web app tool that allows us to quickly move from the idea stage to the testing stages of features. Polymer is built on a mix of HTML, javascript, Node.js (4.x) and Bower. The database will run on the google service FireBase. This allows us to easily store and distribute user posts and settings and integrates easily with Polymer. In addition to simply delivering data, Firebase also has extensive analytic and monitoring tools, allowing us to see the health of Talkatiel at a glance.

We will be designing this for use primarily on mobile device browsers. These include Chrome, Safari, and Firefox. Because of the way PWAs are structured, they should also perform well on desktop browsers such as Chrome, Edge, and Safari.

Talkatiel will use a service worker to precache parts of a web app, allowing it to load quickly the next time the user uses it. This allows us to manage what the user sees while cutting down on network traffic and load times. A service worker can perform its tasks even when the web app isnt open. This means in addition to caching and managing web app data, it also has the ability to allow users to access things offline, intercept HTTP/HTTPS requests and even send push notifications to the user.

1.1 Platforms and Sharing

A portable web app means that the application will be availible from any link online as well as through a link on a users phone.

Our planned platforms are mobile, web
Our application should be shareable through a link

1.2 Functional Requirements

1.2.1 logging

It is extremely important that we have the ability to log both what posts are made and who makes them. Despite being an anonymous message board, users will only be anonymous to each other, not to the server. This allows us to take action against posters that create content that is bullying or hate speach. Without some form of logging there is no way to punish users who continually antagonize other users.

1.2.2 Posting

User will be able to create and post in discussions anonymously. They will be able to read and respond to posts by other users. Users will be able to reply to a discussion, reply to another reply and vote on posts.

1.2.3 Moderation

There will be both computer and real person moderation. The computer run moderation will filter out posts will blacklisted words or phrases, while the human moderation will be done at the best judgment of whoever is moderating.

1.2.4 Geolocation

Users will only be able to post to certain discussions if they are with a range of the discussions marker. If a user is within a certain distance they will be able to post to discussions. If they are out of range of a discussion, they wont be able to view or participate in the discussion.

1.2.5 Viewing Posts

Users will be able to scroll through posts made by others users. They will be able to see the text of the post, and possibly how many votes the post has. A user can then respond, vote, or continue scrolling to new content.

1.2.6 Reporting

Posts will be able to be reported for reason such as spam, abuse, bullying, or other issues. These posts will then be dealt with by a human moderator, who can then choose to remove the post or let it be. This will help cut out harmful speech that isnt wanted on the platform.

1.2.7 Commenting

Users will be able to view comments on posts. They will also be able to make comments on posts. If the post has been removed while commenting, they will be shown an error.

1.2.8 Saving

Users will be able save posts. These will be stored offline in order to view them later on. They will be able to view all saved posts and remove them from the list as they please.

1.3 Non-Functional Requirements

1.3.1 Secure

Talkatiel will need to utilize a secure connection (HTTPS). All content needs to be delivered through a secure connection to confirm that all aspects of the app are controlled by us. This tops malicious 3rd parties from abusing the service.

1.3.2 Network Independent

The app should still work when there is little or no network connectivity. If not showing the different discussions, at least displaying a page notifying the user of the fact. This will allow it to act as though it is a native app, though it is actually a website crafted for their device. Also stops it from appearing "broken" to users, which could result in losing them.

1.3.3 Responsive

Talktatiel should be responsive to users on both phones and tablets. The pwa should react quickly to user inputs, meaning that both the application presented to the user and the back-end database need to do their tasks speedily.

1.3.4 Deep Linking

Each page should have a distinct URL. This means each discussion, post and reply should have it's own individual URL. This allows user to share the post with others, such as on facebook or other social media services.

1.3.5 Standard App Manifest

Talkatiel should include a file that specifies all the metadata associated with it. This allows us to set various settings for how the web app is displayed, such as orientation, display modes, image links, URL settings, and theming. This is an integral part of making a portable web app and signals to browsers how to handle the website.

1.4 Documentations

In addition to a simple and intuitive user interface, there will also be a limited help section present. It will show users how to do different basic functions of the application, such as navigating, posting, voting and the purpose. The help section can walk a new user through the application and introduce them to the features. By keeping the Talkatiel simple we can limit the amount of documentation needed while also improving user retention.

1.5 Major Features

- Post text as part of discussion
- Reply to other user's posts
- Ability to vote up or down other posts
- Both computer and human moderation of posts
- Ability to participate and view discussions depending on location
- Ability to report posts to Moderators or Administrators

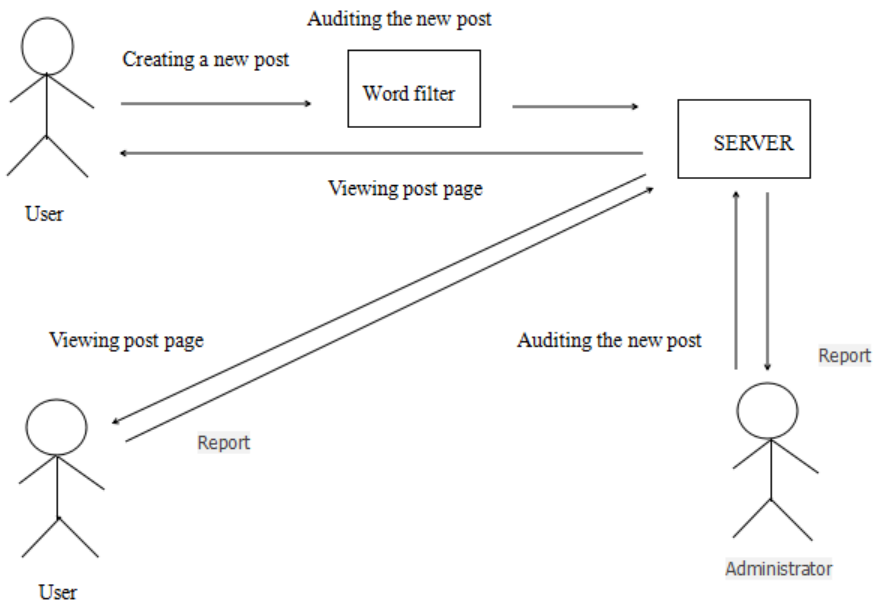

Figure 2: This diagram demonstrates the 3 interactions we cover in our use cases, *Creating a new post, Auditing the new post, and Viewing the post page*

1.6 Minor Feature / Strech Goals

- Ability to save discussions for later viewing.
- Ability for a user to view all of their own posts
- Keep track of the total votes a user has recieved
- Integrate ads into the platform

2 Use Cases

2.1 Creating a new post

2.1.1 Goal

User wants to create a new text or image post and publish it to the web app for others to see

2.1.2 Actors

College student users

2.1.3 Preconditions

The web app must first be loaded and the user must be on the create a new post page. The user must have GPS coordinates that are within the campus area

2.1.4 Postconditions

The web app will send the information to a firebase database of posts, which will update, displaying the new content on other users device

2.1.5 Flow of Events

- user clicks create new post create post page is loaded from server
- user writes post and submits
- post text and title is pushed to posts table
- firebase post table is updated
- other users see updated post table when they load the posts page

2.1.6 Quality Requirements

The app content framework must load in less than a second, maximizing local caching to minimize external CDN requests. The update request created by the posting of a new post must be processed in less than 5 seconds, and other users must be able to see the new post in less than 10 seconds

2.1.7 Error Scenarios

Problem: The user is not connected to the internet, post fails to publish **Solution:** A page will load notifying the user that they have no connection to the internet **Problem:** The database is down for maintenance or DDOS, user unable to post **Solution:** A page will load notifying the user that posting is temporarily down, and to try again in a few minutes

2.2 Auditing the new post

2.2.1 Goal

Illegal posts wont be able to post or they will be deleted later by administrators after receiving reports(break the anonymity rules: posts include personal phone number, email address contain illegal keywords: threatening or offensive language)

2.2.2 Actors

All users administrator , word filter

2.2.3 Preconditions

Once users finish their posts and try to make a post, the word filter will be enabled. If the posts violate above rules, users will receive an alarm asks them to modify their post.

2.2.4 Postconditions

All posts will contain a report button in the bottom, users are expected to maintain community stability by reporting illegal posts.

2.2.5 Flow of Events

- user clicks post
- word filter is enabled and judge if the post legal
- post legal, update to server / post illegal, asks user to modify the post
- other user reviews the post
- other users like the post / report the post
- administrator audits report post
- administrator adds the report post into word filter.

2.2.6 Quality Requirements

All posts which get reported much be reviewed by administrator within 24 hr. Word filter must be updated monthly.

2.2.7 Error Scenarios

Problem: When illegal posts are prevented by word filter, but user doesnt know what was wrong. **Solution:** highlight the illegal words.

2.3 Viewing post page

2.3.1 Goal

Users should be able to view all posts in a specified range ranked by a variety of chosen factors, (time posted, number of upvotes in different time range)

2.3.2 Actors

College student users

2.3.3 Preconditions

The web app must first be loaded and the user must be on the posts page. The user must be connected to the internet to load content.

2.3.4 Postconditions

The app must be able to connect to the posts database.

2.3.5 Flow of Events

- User opens post page
- post page cached DOM framework loads
- web app queries server for posts with specific tags (time posted in range, topic)
- web app sorts posts by number of upvotes
- web app displays posts in DOM.

2.3.6 Quality Requirements

Post page must load in less than 3 seconds Individual posts must load in less than 2 seconds

2.3.7 Error Scenarios

Problem: The user is not connected to the internet, post fails to publish **Solution:** A page will load notifying the user that they have no connection to the internet **Problem:** The database is down for maintenance / DDOS, user unable to view posts **Solution:** A page will load notifying the user that posting is temporarily down, and to try again in a few minutes

3 Planning

3.1 Milestones

Each milestone builds on the last, which was how the order was chosen. Much of the 10 weeks of the term will be spent documenting and planning the project as follows:

3.1.1 Software Requirements and Planning (SRP):

Implementation Goals:

- Meet with customer to discuss goals
- Agree with customer on functional and nonfunctional requirements
- Brainstorm 3 initial use cases, showing important product functionalities

- Create rough project timeline and milestones

3.1.2 Software Design Specification and UI (SDSUI):

Begin creating UI prototypes, create class and sequence diagrams to plan components of the software from both the developer and customer point of views. Requires access to photoshop, a developer to create the prototypes, and the rest of the team to create the diagrams. *Depends on early stage plans defined in SRP.*

Implementation Goals:

- Create prototypes of each page of the app using photoshop
- Begin class diagrams
- Begin sequence diagrams

3.1.3 Design and Implemented System (DIS):

Further flesh out the requirements and design of the software. Add further detail to the UML class diagrams created in SDSUI. Use class diagram to decide how the classes will be packaged (coupled) together, the interfaces needed to implement the classes, and begin analyzing possible risks in the system. Begin brainstorming exception handling for these risks. Requires whole team to design and agree on product design. *Depends on designs created in SDSUI stage.*

Implementation Goals:

- Complete class diagrams, final internal system design
- Decide what type of server and database will be used for implementation
- Create Risk Analysis paper

3.1.4 User Stories and Setup Assignment (USSA):

Think about how the system will be used. Create 20 user stories relating to these scenarios. Each story should correspond to a piece of implementation. Set dates for when the features in these stories need to be implemented in code, and for each story create a new sequence-of-events diagram to clarify to the developers what classes/parts of the system are used for the story and in what order. Requires each teammate to handle about 3 user stories and for whole team to agree on their priorities. *Depends on diagrams from DIS, Sequence diagrams brainstormed during SDSUI.* Implementation Goals:

- Create list of 20 user stories
- Decide priority of each story
- Create sequence diagram for each story

3.1.5 First Implementation of System (FIS):

Begin implementing the user stories in code. Choose what is implemented this week based on the deadlines and dependencies for each user story outlined in USSA. For each user story, create unit tests to ensure the correctness of the code. Create Product Release paper to explain to the costumer where the program is located and include instructions on how to run it. Review and revise deadline estimates based on progress made up to this point. Requires at least one computer to run server and database, one other device to connect with server, and each developer having own computer to work on their own assigned code. *Depends on user stories created in USSA.* Implementation Goals:

- Implement final prototype of UI in HTML/CSS
- Set up a server
- Set up a database
- Implement 10 user stories using the UI, server, and database created above

3.1.6 Second Implementation of System (SIS):

Complete implementation of user stories. As in FIS, create tests for each user story as it is implemented. Final revision of deadline estimates and explain which features (if any) could not be implemented within the timeline. Requires at least one computer to run server and database, one other device to connect with server, and each developer having own computer to work on their own assigned code. Implementation Goals:

- Implement all user stories
- Ensure server can identify individual users while keeping them anonymous
- Test program on multiple devices
- Present final product

3.2 Timeline

3.2.1 HTML/CSS

02/12/18- This should be enough time to develop a clean, fully functioning look to the app. It needs to be mobile friendly as well. This is assigned to Ryan, and Aidan.

3.2.2 Javascript

02/19/18- Creating javascript is needed to make user interaction clean and send user data to the database. This should be secure and checked for vulnerabilities. This is assigned to Ryan and Aidan.

3.2.3 Database

03/01/18- The database should be set up and functioning in 1 month from now. This means that the app will be functioning and ready for testing. This is assigned to Yufei, Brendan, and Iliana.

3.2.4 Mobile Development

03/01/2018- This should be developed to work alongside the database and serve an identical copy of the website for PC. Mobile will be the main focus of this app, so it is crucial that it works well and is secure. This needs to be done by all members, and is a group effort between the frontend team and backend team.

3.3 Project Tracking

To properly track and organize the project, we will use a variety of services to coordinate our efforts. These include GitHub, Slack, and Google drive. Using these we will be able to coordinate who works on what.

Using Slack we can communicate with each other and discuss what needs to be done.

Google drive will keep track of scheduling and timeline items. There we will keep track of where we are at, and what sort of project goals we will have for each week. When planning to do things we will reference the documents there to decide what to do next. On a side note, it is important that we keep track of updating the schedule with current events. If we finish a real faster or slower than expected it should be recorded so everyone is about to know.

Finally, GitHub will be used to contain the project code. There will be three branches: server side, client side, and project assignments. Because server and client side will be running separately, so it is best to have them be separate projects. Then in order to update code, we don't need to download extraneous code we don't need.

As different user stories and tasks get done and committed to the GitHub repositories the progress taking pages should be updated on the Google drive while also making other team members aware of the changes. That way what was done is recorded, and other group members can reference the repository to view changes. Some tasks depend on others, so the foundation tasks need to be done first. By using these three devices we can properly track the progress of this project.

3.3.1 GitHub

We will be using GitHub to keep our code clean and organized. We have created a separate branch for these assignments and will be merging when complete. Github will allow us to store files neatly, and a seperate folder may be used for different teams.

3.3.2 Slack

Slack has been very useful for many of us in the past. It allows us to communicate and stay on topic. We have everyone added and have been commenting and organizing meetings.

3.3.3 Drive

A Google Drive folder has been created for our team. This allows us to manage files outside of github, such as photos, icons, useful links, notes, and written plans.

3.4 Risk Management

3.4.1 Yik Yak

This idea was done by a company that has shutdown. Yik Yak made serious changes to their app in their final years, including trying to make a transition to a social media website. They added usernames, forced users to be accountable. We need to stay away from making the same mistakes, and make sure our users are hidden, otherwise we will fail.

3.4.2 Storing User Data

Storing user data is very important. Users need to be able to trust our security before they use it. The best way to make sure the data is safe is to not collect much at all. This means that users will have very little to lose if our lack of experience with data protection results in a breach.

3.4.3 Managing Content

Learning how to manage all of the content needed to make the website functioning could be difficult. Many different parts written by different people will need to be combined and work together flawlessly. THis means that we will need to communicate well and make sure that we are all on the same page.

4 Meeting Report

4.1 Progress Made This Week

This week we completed the software requirements and planning phase of the project. We worked on defining both functional and non-functional requirements along with use cases. Finally, we created a project plan that outlines the different phases of the project, including backend development, frontend development, user interface, user experience, and potential issues. We will continue to communicate on Slack to coordinate another meeting time.

4.2 Plans for Next Week

In the coming week we will start mapping out the architecture of the pwa. Work will start with planning out the different views in the application. Along with this we will start looking at how we should store the data, e.g. data structures and such. Team members are going to start experimenting and getting comfortable with both Polymer and Firebase so once we get into the coding portion everything wont be new. Along with this we are going to work on the requirements more and get a plan for each one, detailing what needs to go into each one.

4.3 Team Member Contributions

Brendan Byers worked on the description and bibliography section. Aiden Grimshaw worked use cases, table of contents, and latex formatting. Ryan Sisco worked on the planning section Yufei Zeng worked on use cases and latex formatting

4.4 Customer Meeting Status

We met with the customer and worked on getting general requirements set. We talked about what the end goal should look like, along with optional goals that we may want to work on if we have time at the end. The customer is very happy with how things are coming along.

References

References

[1] Lindsay Bramson *Yik Tak bullying leads district to ban app* 2014: KXAN.com. http://kxan.com/2014/09/29/yik-yak-bullying-leads-districts-to-ban-app/

[2] Nick Strayer *The Great Out-Of-State Migration: Where Students Go* 2016: New York Times. https://www.nytimes.com/interactive/2016/08/26/us/college-student-migration.html