TalkaTiel Second Implentation System

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

Contents

1 Product Release

1.1 Server Side Implementation

Currently the server code can be downloaded and run locally. By cloning the repository located at

```
https://github.com/B13rg/Talkatiel_API.git
```

one can run the server. There is also extensive documentation detailing all the steps that need to be taken to run the server. There is also a SQL script that will create a database when run. This can be used to create a database for use on a different engine. Currently it is configured to run on the localhost on port 5002. This can be tested to navigating to

```
127.0.0.1:5002/Posts/New
```

where you can see the raw Json output of a post. In the repository there is also an sqlite database that is run alongside the python code. This will connect with the python to respond to different sql queries. It is fully functioning, and the only thing left to do is test it, and find a permanent URL. Additionally, you can test GET and POST reqeusts at our API server URL, aidangrimshaw.pythonanywhere.com, using the documentation located on our API github repo.

1.2 Client-Side Implementation

By midweek, we plan to start permanent hosting with google app engine.

2 User Story

2.1 Add a post

2.2 Add a Comment

2.3 Like/Dislike

2.4 Report

2.5 Delete

2.6 Refresh

2.7 Sort

2.8 View Newly Created Post

2.9 Secure HTTPS Connection

2.10 Save Website as App on Phone

2.11 Responsive Web Page

3 Tests

3.1 Server Side Implementation

4 Design Changes and Rationale

4.1 Server Side Implementation

For the most part the server group stayed very close to original design. To design the database, we worked off the UML design diagrams we designed earlier. This allowed us to see what fields we needed to include for each table. We added an extra table not originally in the design doc to handle reports. We found there was no way to keep track of reports server side, so we added a table to handle it. It keeps track of the user who reported it, the post in question, and the text of the users report. By having its own table, it can also be queried separately instead of having to sort through who knows what. Additionally, we modified our api server implementation so that it was served remotely from a different service than the client side server, and so that the API would be better equiped to handle POST requests.

4.2 Frontend Implementation

We have stayed very close to our original design. The changes we have right now are temporary and easy to change. We do not have some things matching such as colors, but we have every correct button and card in position. Styling the rest will be very easy. Our buttons do not currently send data to our actual server. This was done intentionally in order to build the backend and frontend at the same time. We are currently fixing the color scheme as we get closer to connecting our mobile app with the database.

5 Refactoring

Our project is still being developed in parts. For the most part, we have been working towards building a working application and database, and slowly cleaning up as we go. This has lead to some sloppy, duplicate code. However, we have been making progress building. Once one of our members has completed his/her section, while waiting for others to complete, they have cleaned up their code and make it easier to follow. While this isnt a top priority right now, we believe that refactoring our code as we complete files will help us in the future. It will also allow us to switch duties and work on each other's code without too much confusion.

We ended up doing minor refactoring. While working on the client side development, we ended up making templates for common, reoccurring elements. This has been useful while developing other parts of the application, when attempting to make the styling similar. We have been able to remain somewhat consistent.

In addition to these changes, we have also changed variable names before committing to the shared repository. This helps those who are viewing changes follow along. Many of us have been saving files offline before pushing, forcing us to correct changes before pushing. By doing this, we are not stressed to push our work as a simple backup of our work. Saving and updating locally/remotely before pushing to a shared repo has helped us stay consistent and organized.

Making these changes has helped us stay organized and also limit the turnover time between switching parts and tasks. For instance, when two people switched files to review code, we were able to follow along much easier when the code was simplified.

6 Tests

7 Meeting Report

7.1 Schedule for next week

7.2 Progress made this week

7.3 Plans and goals for next week

7.4 Team Member Contributions