Chirp! Project Report

ITU BDSA 2024 Group 4

Allan Petersen allp@itu.dk Bergur Davidsen berd@itu.dk
Lucas Venø Hansen lucasveha@gmail.com
Mikkel Clausen mikcl@itu.dk Victor Sforzini visf@itu.dk

0.1 Design and architecture

0.1.1 Domain model

0.1.2 Architecture — In the small

Below a diagram can be seen, showing the onion architecture of the Chirp program. Were the outer circles depend on the inner circles:

Figure 1. Onion architecture of Chirp program

In the center of the onion one finds Chirp Core, this is were the most primitive code lies, like objects and interfaces.

One step out of Chirp Core, one finds Chirp Infrastructure. This is were the handling of the database is done, this includes retrieving, deleting and updating data. Defining the database and giving it some initial data is also done in Chirp Infrastructure.

In the third layer of the onion, Chirp Web lies. This is were the webpage HTML is found, along with all the styling. The API which the web pages communicate with lies here as well.

Lastly there is the outer layer, naturally here the test lay. The test suit includes Unit-, integration- and end2end test. The end2end test are done using Playwright.

0.1.3 Architecture of deployed application

0.1.4 User activities

The goal of this chapter is to show some core interactions from both **unauthenticated user** and **authenticated user**. We make use of UML activity diagrams, these will visualize the states triggered by a users actions.

First off we want to show what a unauthenticated user can do, and how the journey is for such users to get to register.

Figure 2. unauthenticated user journey and register

This diagram show that a user can authenticate with both Email, and GitHub. Also if you like a cheep from a user on the public timeline. It will simply not like it, but instead put you on the register page. Registering this way will give the same result as just navigating to the register page using the navigation bar.

When you are authenticated / logged in, we have 4 primary action a user can do, repectively: Cheep, Like, Follow and Delete the account from the Chirp service.

The process of cheeping is show in this diagram:

Figure 3. Cheeping journey and validation of cheep

A cheep is valid if its length, as show in the diagram, is between 0 and up to and including 160 characters. If you were to click the Share button, with and empty text field, a warning will pop up. A warning pop up wont explicitly be shown to the user for cheeps longer that 160 characters, we simply show the length counter on screen, and dont allow for more characters, in both front- and backend.

The users also need to like cheeps, for that action we have this diagram:

Figure 4. Liking cheeps

The 'heart' button we have besides each cheep is essentially a toggle for likes on the given cheep. And as showed in the diagram, each user can only like any given cheep once. It is important to note, as of now the liking of a cheep will result in the page redirecting you to the root page (page 1), even though you might be on for instance page 6. This is an obvious room for improvement and is currently a task in the project board.

Next up we want to show the journey of a user following another user.

Figure 5. Following users

The flow of following a user, is close to the same as liking cheeps, as both are 'toggles'. The only difference is that we decided to show the newly followed users profile after the follow action. Which eleminates the issue we are having with liking cheeps far down on the public timeline, and wanting to scroll beyond that point afterwards. This does then create the issue with wanting to continue scrolling after following. But this navigate to the private timeline of the newly followed user, is a conscious decision.

Lastly it is important for us to show how the user can delete and see the data we have gathered.

Figure 6. Deleting the user and download data

The linear diagram is pretty much self explanatory, but we feel it's important to show either way, since this is last key feature for a user to experience.

The diagrams provide a clear overview of user journeys, including registering, posting a cheep, liking cheeps, following other users, and deleting an account. Additionally we have highlighted some areas for improvements.

0.1.5 Sequence of functionality/calls trough *Chirp!*

0.2 Process

0.2.1 Build, test, release, and deployment

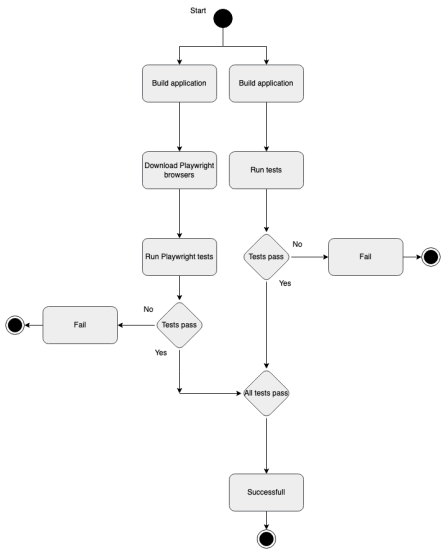

Figure 1: Build and test flow

some text

Figure 2: Playwright test flow

some text

Figure 3: Deployment flow

some text

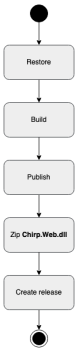

Figure 4: Release flow

some text

0.2.2 Team work

0.2.3 How to make *Chirp!* work locally

0.2.4 How to run test suite locally

0.3 Ethics

0.3.1 License

0.3.2 LLMs, ChatGPT, CoPilot, and others

In the development of our project we used ChatGPT, and when we did so, we made sure to add ChatGPT as a co-author in our git commit message like so:

`ChatGPT <>`

ChatGPT was very helpful when used to create simple code parts and debug some. On the other hand the ChatGPT was not helpful with complex code questions. Therefore we ended up finding it mostly useful for us to understand parts of the code and guide us on where to start on complex tasks.

However we also experienced some negatives when using ChatGPT. It could sometimes go in a spiral, in cases like this we would look at the slides and on the web for help. We also gave Gemini some use sometimes when ChatGPT was not helpful, we did however never use any of the provided code, so it never got to be a co-author.

For the most part the use of LLMs sped up our development, however some times the were send into a spiral and hallucinated, which sometimes could confuse us more. So we experienced the limitations of LLMs and got to learn more on know how to use them more efficiently.