Chirp! Project Report

ITU BDSA 2023 Group 15

Daniel Sølvsten Millard dmil@itu.dk
Frederik Lund Rosenlund frlr@itu.dk
Jacob Vortscir Pærregaard jacp@itu.dk
Mads Christian Nørklit Jensen macj@itu.dk
Rasmus Lundahl Nielsen raln@itu.dk

- Design and Architecture of *Chirp!*
  - Domain model
  - Architecture — In the small
  - Architecture of deployed application
  - User activities
  - Sequence of functionality/calls trough *Chirp!*
- Process
  - Build, test, release, and deployment
  - Team work
  - How to make *Chirp!* work locally
  - How to run test suite locally
- Ethics
  - License
  - LLMs, ChatGPT, CoPilot, and others

1 Design and Architecture of *Chirp!*

1.1 Domain model

Here comes a description of our domain model.

Illustration of the *Chirp!* data model as UML class diagram.

Figure 1: Illustration of the *Chirp!* data model as UML class diagram.

1.2 Architecture — In the small

1.3 Architecture of deployed application

Illustrate the architecture of your deployed application. Remember, you developed a client-server application. Illustrate the server component and to where it is deployed, illustrate a client component, and show how these communicate with each other.

- Clients:
  - Web browser
  - Mobile app
- Web server:
  - ASP.NET Core
  - Docker
  - Azure
- Database:
  - MSSQL
  - Docker
  - Azure

Clients -> Web server <-> Database

1.4 User activities

1.5 Sequence of functionality/calls trough *Chirp!*

2 Process

2.1 Build, test, release, and deployment

2.2 Team work

When a new issue is created, it is automatically assigned to the "new" column in the project board. Members of the team can then assign themselves to the issue, with the amount of people working on it, being mainly dependent on the complexity of the issue. When a member assigns themselves to an issue, they move the issue to the "in progress" column. A new branch is created to work on the issue, and a pull request is linked to the issue, to track the progress on it. When the issue is considered completed, by the working members, the pull request is reviewed by the other members of the team. Members then review if they find the solution satisfactory. If the solution is not found satisfactory, they provide feedback, throught their review and await the working members to consider feedback and submit a corrected pull request for review. If the solution is found satisfactory, the pull request is merged into the main branch. The issue is then moved to the "done" column.

2.3 How to make *Chirp!* work locally

- Probably something about docker? The program cant run without docker I think?

To clone the project run the following command in the terminal:

- git clone https://github.com/ITU-BDSA23-GROUP15/Chirp.git

From the root of the project run the following command to run the project locally:

- dotnet run –project src/Chirp.Razor/
- Open a browser and go to http://localhost:5273/
- Do we need to do any migrations?

You should now expect to see the public timeline, stating at the top of the site, that you need to login to cheep, a login button should be available at the top right. The rest of the features on the site, will only become avaibalbe after logging in, which is done using your github account.

2.4 How to run test suite locally

To run the test suite locally, you need to have a local instance of the program running. This can be done from the root of the project by entering

- cd src/Chirp.Razor and dotnet run. This will start the program at http://localhost:5273/

The test suite can then be run by entering cd src/Chirp.Tests (maybe not, can just do dotnet test from root of project) and afterwards dotnet test. This will run the test suite and output the results in the terminal.

3 Ethics

3.1 License

We've chosen the MIT License, which is a permissive free software license, because of its limited restriction on reuse. In this project we wanted to encourage reuse of our code, and therefore we've chosen a license that allows for this. The MIT License is also a very common license, which makes it easy for others to understand the terms of the license.

3.2 LLMs, ChatGPT, CoPilot, and others

"State which LLM(s) were used during development of your project. In case you were not using any, just state so. In case you were using an LLM to support your development, briefly describe when and how it was applied. Reflect in writing to which degree the responses of the LLM were helpful. Discuss briefly if application of LLMs sped up your development or if the contrary was the case."

- Co-pilot: Has been useful for auto completing code that has already been made previously, especially the repository database functions.
- Co-pilot: Was not good at creating new code with new logic, almost always faulty and spent more time debugging autocompleted code than what it benefitted.
- ChatGPT: Has been used a lot as a starting point in debugging, when everything seemed overwhelming.
- ChatGPT: General questions about project topics, that helped narrow down the scope of the task and therefore researching became a lot easier.
- ChatGPT & Co-pilot: Has been used with varying succes, to understand errors provides by the compiler. Sometimes it was helpful, other times it simply did not understand the error and provided wrong suggestions.