Chirp! Project Report

ITU BDSA 2024 Group 13

Alexander Hvalsøe Holst alhh@itu.dk
Andreas Løvsgren Nielsen anln@itu.dk
André Racraquin Birk arbi@itu.dk
Peter Aksel Skak Olufsen polu@itu.dk
Símun Larsson Løkke Rasmussen simra@itu.dk

1 Design and Architecture of *Chirp!*

1.1 Domain model

<!– Provide an illustration of your domain model. Make sure that it is correct and complete. In case you are using ASP.NET Identity, make sure to illustrate that accordingly. –>

asdasdasd

1.2 Architecture — In the small

<!– Illustrate the organization of your code base. That is, illustrate which layers exist in your (onion) architecture. Make sure to illustrate which part of your code is residing in which layer. –>

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. –>

1.4 User activities

<!– Illustrate typical scenarios of a user journey through your Chirp! application. That is, start illustrating the first page that is presented to a non-authorized user, illustrate what a non-authorized user can do with your Chirp! application, and finally illustrate what a user can do after authentication.

Make sure that the illustrations are in line with the actual behavior of your application. –>

1.5 Sequence of functionality/calls trough *Chirp!*

<!– With a UML sequence diagram, illustrate the flow of messages and data through your Chirp! application. Start with an HTTP request that is send by an unauthorized user to the root endpoint of your application and end with the completely rendered web-page that is returned to the user.

Make sure that your illustration is complete. That is, likely for many of you there will be different kinds of "calls" and responses. Some HTTP calls and responses, some calls and responses in C# and likely some more. (Note the previous sentence is vague on purpose. I want that you create a complete illustration.) –>

2 Process

2.1 Build, test, release, and deployment

<!– Illustrate with a UML activity diagram how your Chirp! applications are build, tested, released, and deployed. That is, illustrate the flow of activities in your respective GitHub Actions workflows.

Describe the illustration briefly, i.e., how your application is built, tested, released, and deployed. –>

2.2 Team work

<!– Show a screenshot of your project board right before hand-in. Briefly describe which tasks are still unresolved, i.e., which features are missing from your applications or which functionality is incomplete.

Briefly describe and illustrate the flow of activities that happen from the new creation of an issue (task description), over development, etc. until a feature is finally merged into the main branch of your repository. –>

2.3 How to make *Chirp!* work locally

<!– There has to be some documentation on how to come from cloning your project to a running system. That is, Adrian or Helge have to know precisely what to do in which order. Likely, it is best to describe how we clone your project, which commands we have to execute, and what we are supposed to see then. –>

2.4 How to run test suite locally

<!– List all necessary steps that Adrian or Helge have to perform to execute your test suites. Here, you can assume that we already cloned your repository in the step above.

Briefly describe what kinds of tests you have in your test suites and what they are testing. –>

3 Ethics

3.1 License

This project uses the MIT 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. –>

The LLMs which were used throughout the developement process were: ChatGPT, GitHub CoPilot and Codium.

All three LLMs were used for debugging, and ChatGPT for generation for most documentation.

Interms of the value of there responses it varied. Sometimes, it was a smnall human error which was overseen. In other more complex cases, it required a greater understanding of the program which the LLMs, espically ChatGPT lacked. In these situations, the LLMs which are built in to text editors, GitHub CoPilot and Codium, were able to gather more information, but were still not always able to solve errors. This may have lead to some spirals throughout the development process and over-relying on an LLM to find a solution.