Chirp! Project Report

Edward Rostomian Thorbjørn Pepe Author 3
Author 4 Author 5

December 17, 2023

Table of Contents

Contents

1. Design and Architecture of Chirp!

Domain model

Architecture — In the small

Architecture of deployed application

User activities

Sequence of functionality/calls through Chirp!

2. Process

Build, test, release, and deployment

Teamwork

How to make Chirp! work locally

How to run the test suite locally

To run the test suites locally, first you will have to start your docker container.

MAC: `docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=DhE883cb" -p 1433:1433 --name sql1 --hostname sql1 -d mcr.microsoft.com/azure-sql-edge`

Windows:

Linux:

Next, open up a terminal in the project. Assuming you are in the root of the repository Chirp, direct to either:

```
cd Test/Chirp.Razor.Tests
```

or

```
cd Test/UITest/PlaywrightTests
```

In both the Chirp.Razor.Tests and PlaywrightTests folder, to run the tests:

```
dotnet test
```

The project contains two test suites, Chirp.Razor.Tests and UITest. The first test suite contains unit tests, integration tests and end-to-end tests. **Har vi det? The unit tests are testing the functionality of the isolated components in our application, that is testing methods within our application of core, infrastructure and web components. The integration tests are testing the interactions of different components in our application, that is testing when using logic from e.g. the infrastructure layer in our web components. The end-to-end tests. . . ?

The second test suite contains our UI tests. These are UI automation tests, using Playwright to simulate a users interactions with the user interface. These

are implemented such that we can ensure that the UI behaves as expected, performing actions and receiving expected output, when doing all types of interactions with our application from the UI.

3. Ethics

License

License: WTFPL

LLMs, ChatGPT, CoPilot, and others

The LLMs used for this project during developments are ChatGPT and GitHub CoPilot. ChatGPT has been used carefully, mainly for asking questions about the code or errors in the code. It has also been used for generating small pieces of code, mainly in the cshtml files. Likewise, CoPilot has been used for generating some of the code in cshtml, but has also been used for helping with code, partly making some of the methods in the repositories and creating outlines for tests. Generally, the responses of the LLMs has been helpful for better understandment of the code and speeding up the development. It has not really created code that we would not have done ourselves, but it do have provided some logic in the methods, which has been helpful in terms of taking inspiration for futher method extensions. The application of LLMs has sped up the development process. Especially, CoPilot has made coding much faster, as it for most parts provides the code needed, e.g., if we already made a test for a method FollowAuthor, in no time CoPilot can make the same one for UnfollowAuthor. However, there has indeed been a few times, when ChatGPT or CoPilot does not understand the requests as intended, and therefore not providing useful outputs. But, for most of the time, they have been helpful tools for development.