Chirp! Project Report

ITU BDSA 2023 Group 12

Markus Brandt Højgaard mbrh@itu.dk
Rasmus Søholt Rasmussen rhra@itu.dk
Allan Sigge Andersen asia@itu.dk
Mads Voss Hvelplund mhve@itu.dk
Lukas Brandt Pallesen lupa@itu.dk

1 Design and Architecture of *Chirp!*

1.1 Domain model

Our Application builds on two main consepts: Cheeps and Authors.

1.1.1 Cheep

A Cheep is the understanding of a post, which holds a message from the author of the post. It has a refernce to its author and a knowlegde of when it was posted.

1.1.2 Author

An Author represents the user in our program. The user can write posts (Cheeps), which is why they are named Author. An Author has references to all their Cheeps, as well as their name and email. It is possible to follow an Author; therefore, the Author also has references to the Authors who follow it and the Authors it follows.

Our domain model consists of two data entities, which depict the attributes of a 'Cheep' and an 'Author' in the context of our application. Figure **??** shows the two classes, 'Cheep' and 'Author,' with their fields and cardinality relationships between each other and within themselves. The 'Cheep' class has a one-to-one relationship with an 'Author' class, and an 'Author' class has a zero-to-many relationship with the 'Cheep' class. Furthermore the 'Author' class has a zero-to-many relationship with authors who follow it and a zero-to-many relationship with authors whom it follows.

1.2 Architecture — In the small

1.3 Architecture of deployed application

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

2.3 How to make *Chirp!* work locally

2.4 How to run test suite locally

3 Ethics

3.1 License

3.2 LLMs, ChatGPT, CoPilot, and others

A large level model is used for language understanding and generation. We have used ChatGPT and Co-Pilot in this project. We used these to expedite the coding in certain areas. We tried asking ChatGPT when hitting an error that we could not figure out why we recieved. ChatGPT is often helpful with both finding but also explain what the error is to understand *why* we get the error. This has been great to get a better understanding of what we need to fix to make good and functional code. Furthermore ChatGPT has proven to be a solid tool for discovering keywords and to figure out what documentation needs to be delved into. ChatGPT is also good at comprehending how components work together, whereas documentation for each part (being Azure, .NET or even GitHub) is really good at focusing on the specific domain. When "integrating" these domains, ChatGPT has been helpful with sharing its insights and filled the gaps between documentation. This has definitely saved us time. Co-Pilot were used on the fly, as we were coding it would suggest what we might write, and often it was right and expidited the coding. It was also helpful, as we could write a comment stating what we want and Co-Pilot will then suggest the code for this. We have always been careful when using these tools as they may be wrong, inaccurate, etc. and researched upon information it gave that we were going to use. The way these tool were used in the process, made the code writing a bit faster, and sometimes **way** faster to debug.

test quote Author/Organization (2012)

References

Author/Organization. 2012. “Title of the Article.” https://some_website.com.