# *Chirp!* Project Documentation

## Formal Requirements

All your documentation for that report is written in Markdown, a lightweight markup language (https://www.markdownguide.org). In the root of your repository, create a directory called `docs`. Let it contain a file called `report.md`.

In case you include illustrations from image files, put all of these of a subdirectory of `docs`, which shall be called `images`.

Before 21/12/23 14:00, hand-in a PDF file with your report on LearnIT (https://learnit.itu.dk/mod/exam/view.php?id=182186). The filename of your report has to be `2023_itubdsa_group_<no>_report.pdf`, where where `<no>` is replaced with the group number from here.

You can convert your report from markdown to PDF format with `pandoc`:

```
cd docs
pandoc report.md -o 2023_itubdsa_group_<no>_report.pdf
```

Find instructions on how to install `pandoc` here In case you do not want to install `pandoc` locally to build your report PDF file, you can add the GitHub Actions Pandoc Action to a respective workflow that updates the PDF file on pushes to/merges with the `docs` directory in your repository.

Use the report template that is provided next to this description. OBS: You have to adapt the metadata of `report.md` (the block within `---` on top) accordingly!

Note, good project reports are spell- and grammar-checked! This can be done in many different ways. In case you write your report in VSCode, then you might consider the following extensions: for spell-checking in VSCode and for spell- and grammar-checking

OBS: Work on your documentation in the same way that you work on source code. That is, small iterative and incremental steps with respective commits.

When you have to write something in the project report, write concisely. That is, write short and precise sentences that do not contain fluff. Important, all illustrations have to be legible in the produced PDF document. So make sure they do not become too small.

Create all illustrations either with PlantUML or with DrawIO. Store all sources of your diagrams, i.e., PlantUML diagram source code or DrawIO XML files under `docs` in a directory called `diagrams`.

Before handing in your project reports, make sure that your source code is suitably documented in-code.

**OBS**: After handing in your reports, please let your deployed systems be operational until the end of the third week of January 2024.

The following describes the sections that your report has to provide. These sections are also provided in the project template. OBS: This is a preliminary description. Helge might change it slightly before 7/12/23. In case there is a change, Helge will inform you via Teams.

---

## Design and architecture

### 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.

### 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.

### 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.

### 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.

### 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.)

## Process

### 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 briefly the illustration, i.e., how you application is built, tested, released, and deployed.

### 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.

### 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, Rasmus 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.

### How to run test suite locally

List all necessary steps that Rasmus or Helge have to execute 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.

## Ethics

### License

State which software license you chose for your application.

### 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.