Chirp! Project Report

ITU BDSA 2023 Group 20

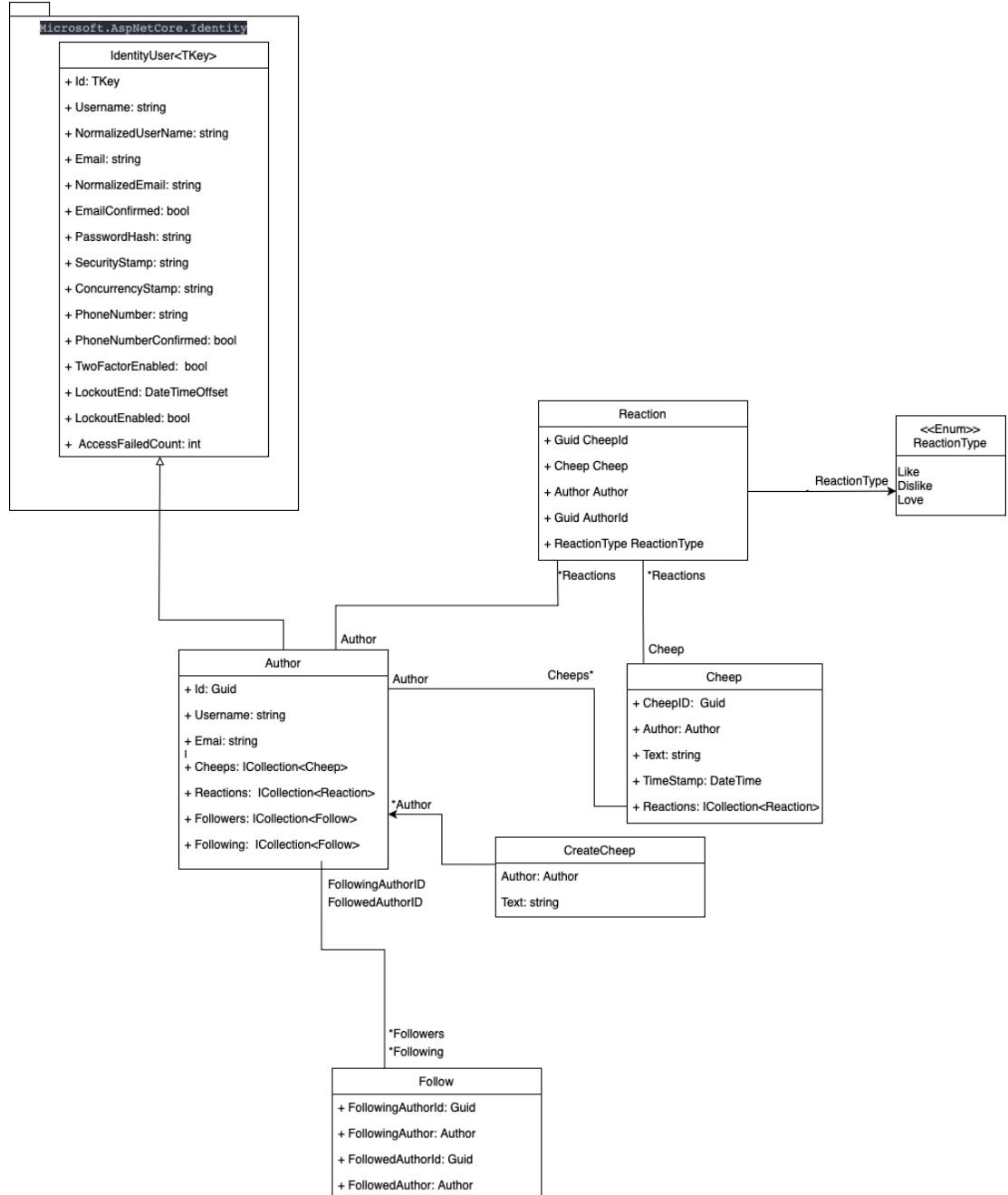Casper Holten casho@itu.dk
Magnus Bertram Kristensen bekr@itu.dk
Mads Dalgaard Andersen madsd@itu.dk
Mikkel Bertram Kristensen mbek@itu.dk
Oliver Laursen olau@itu.dk

1 Design and Architecture of *Chirp!*

1.1 Domain model

IdentityUser<TKey>

+ Id: TKey

+ Username: string

+ NormalizedUserName: string

+ Email: string

+ NormalizedEmail: string

+ EmailConfirmed: bool

+ PasswordHash: string

+ SecurityStamp: string

+ ConcurrencyStamp: string

+ PhoneNumber: string

+ PhoneNumberConfirmed: bool

+ TwoFactorEnabled: bool

+ LockoutEnd: DateTimeOffset

+ LockoutEnabled: bool

+ AccessFailedCount: int

Reaction

+ Guid CheepId

+ Cheep Cheep

+ Author Author

+ Guid AuthorId

+ ReactionType ReactionType

ReactionType

**<<Enum>>
ReactionType**

Like
Dislike
Love

*Reactions   *Reactions

Author

Author

+ Id: Guid

+ Username: string

+ Emai: string
|
+ Cheeps: ICollection<Cheep>

+ Reactions: ICollection<Reaction>

+ Followers: ICollection<Follow>

+ Following: ICollection<Follow>

Author   Cheeps*   Cheep

Cheep

+ CheepID: Guid

+ Author: Author

+ Text: string

+ TimeStamp: DateTime

+ Reactions: ICollection<Reaction>

*Author

CreateCheep

Author: Author

Text: string

FollowingAuthorID
FollowedAuthorID

*Followers
*Following

Follow

+ FollowingAuthorId: Guid

+ FollowingAuthor: Author

+ FollowedAuthorId: Guid

+ FollowedAuthor: Author

The above image is a UML class diagram depicting the domain model of **_Chirp!_**.

1.2 Architecture — In the small
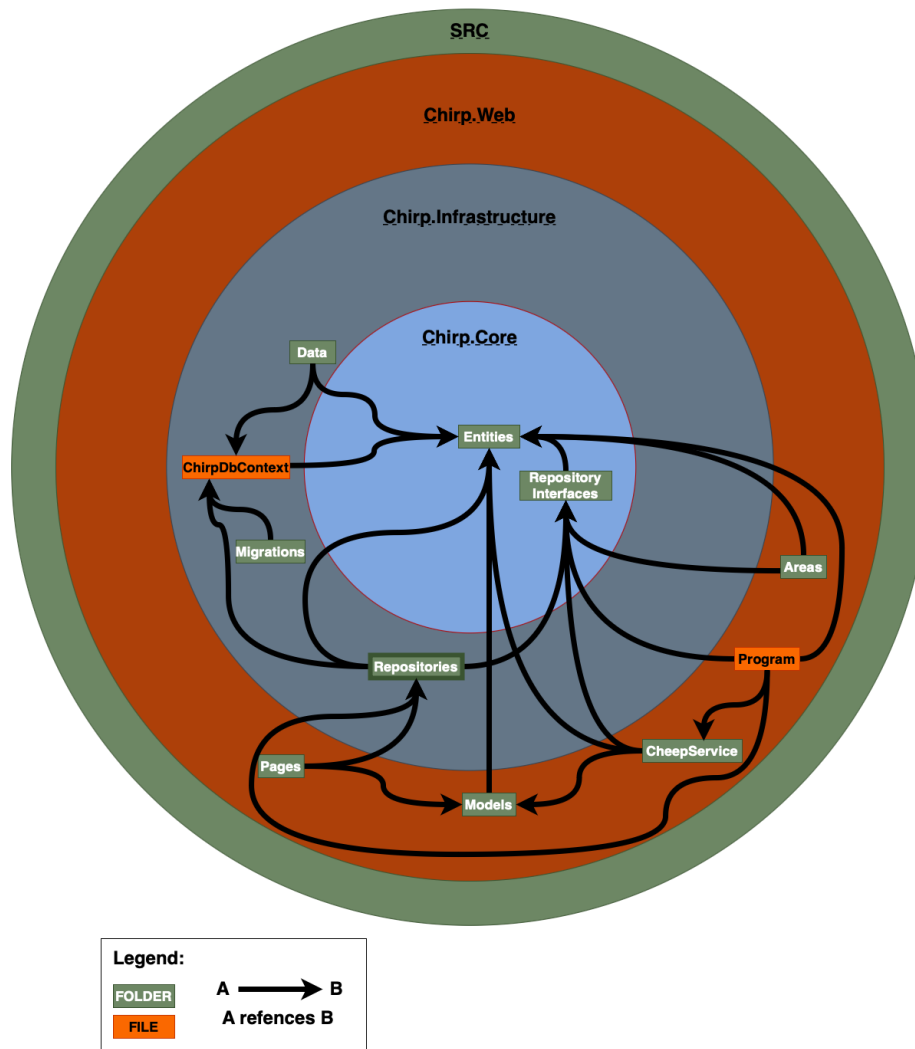
The above image illustrates the architecture of **_Chirp!_**. It's onion based and each onion layer represents a package in the "src" folder of the application.

1.3 Architecture of deployed application
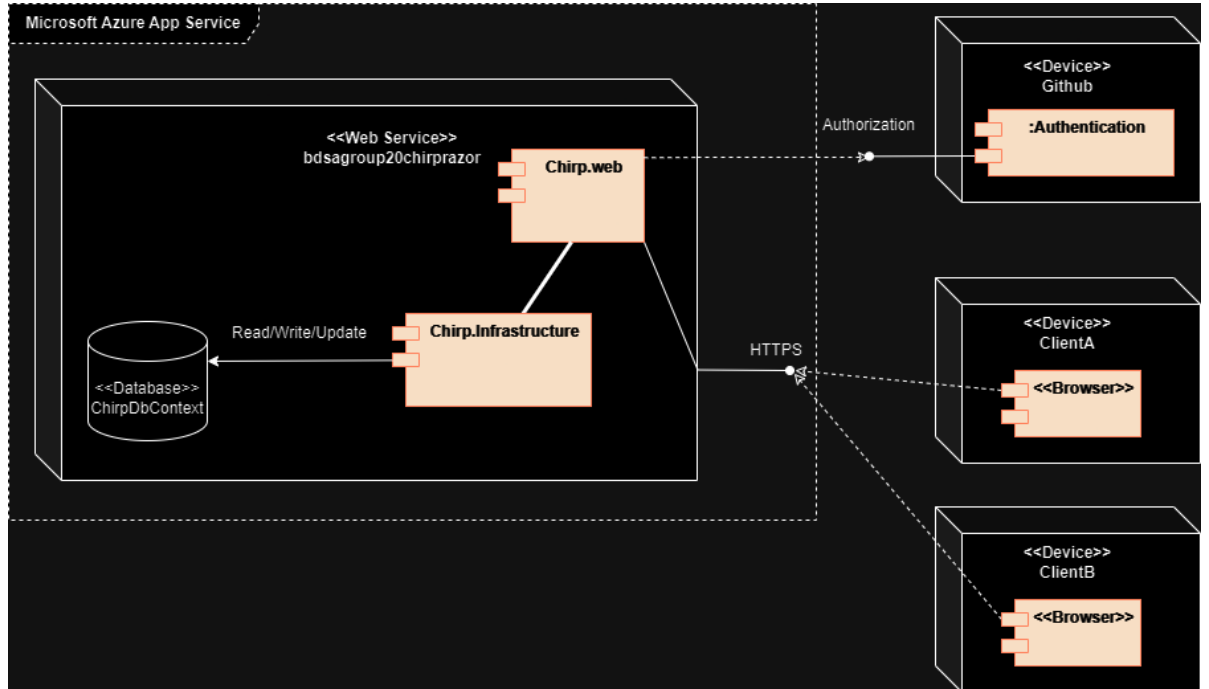
As we ran out of credits on Azure, we had to switch back to using an Sqlite database. Which is the setup the deployment diagram above illustrates. The first commit hash after reverting back to Sqlite database is: eb9b72d3ee52408b700a1912c9be30edc007488e.
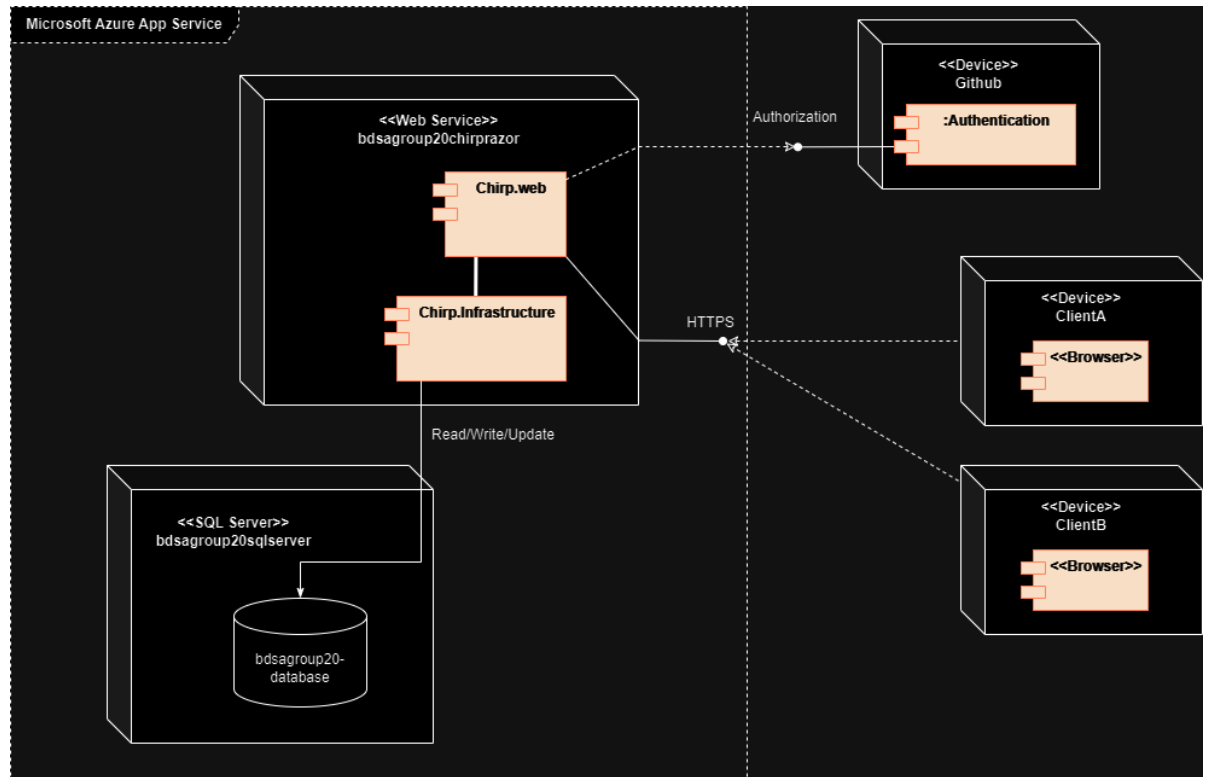
This deployment diagram illustrates our former setup, where we used an Azure hosted SQL server. The hash of the latest commit using Azure SQL server is: e79d7c8bece998633b05939e62f4fe29495a410a.

1.4 User activities
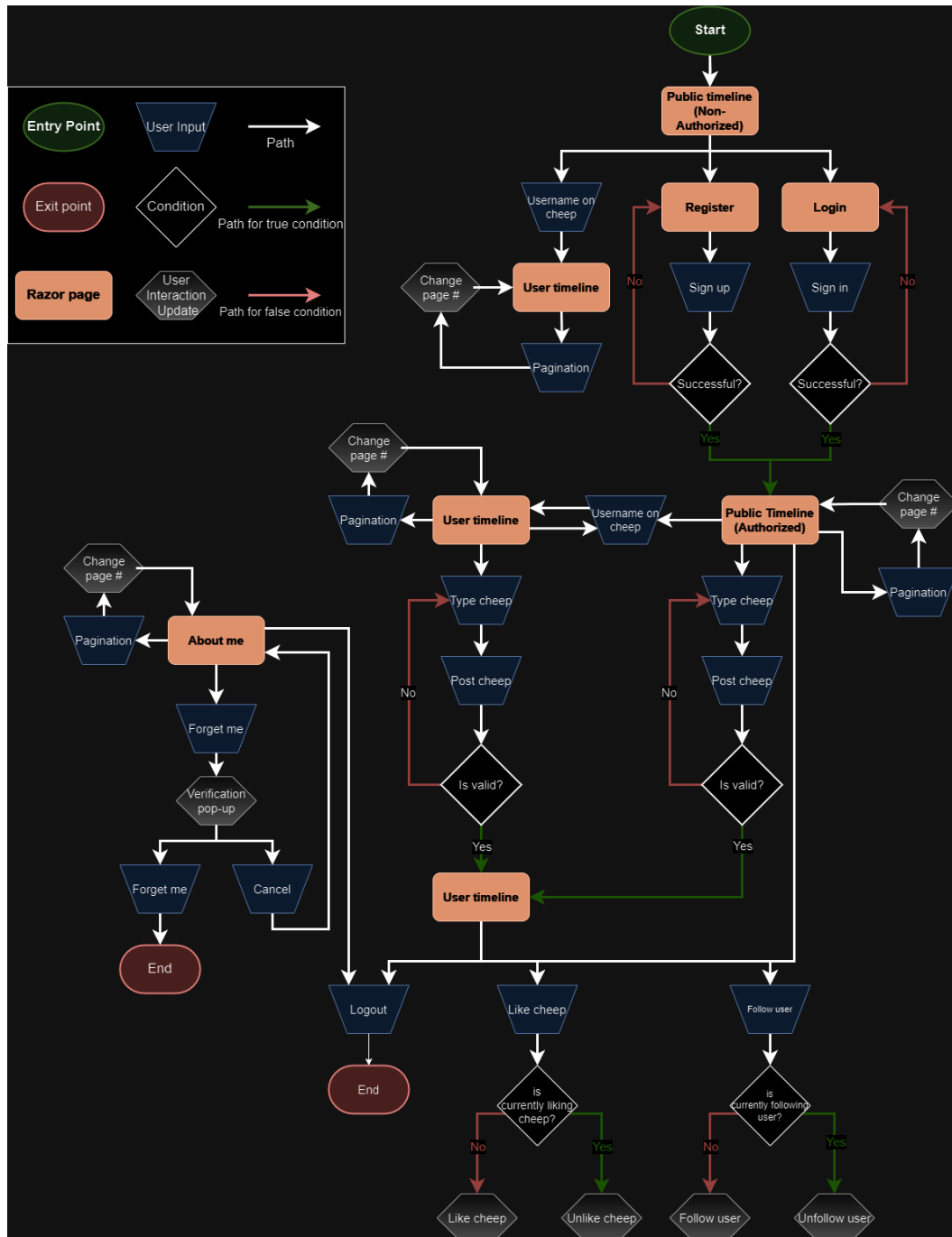
The flowchart above illustrates the user journey through the Chirp!
application

Please note, that unauthorized users can access the 'Public Timeline,' 'Register,'
and 'Login' pages directly from the navigation bar. Authorized users can navigate
to the 'Public Timeline', 'User Timeline', and 'About Me' pages, and logout
from the navigation bar. Clicking 'Logout' ends the flow. This detail is omitted
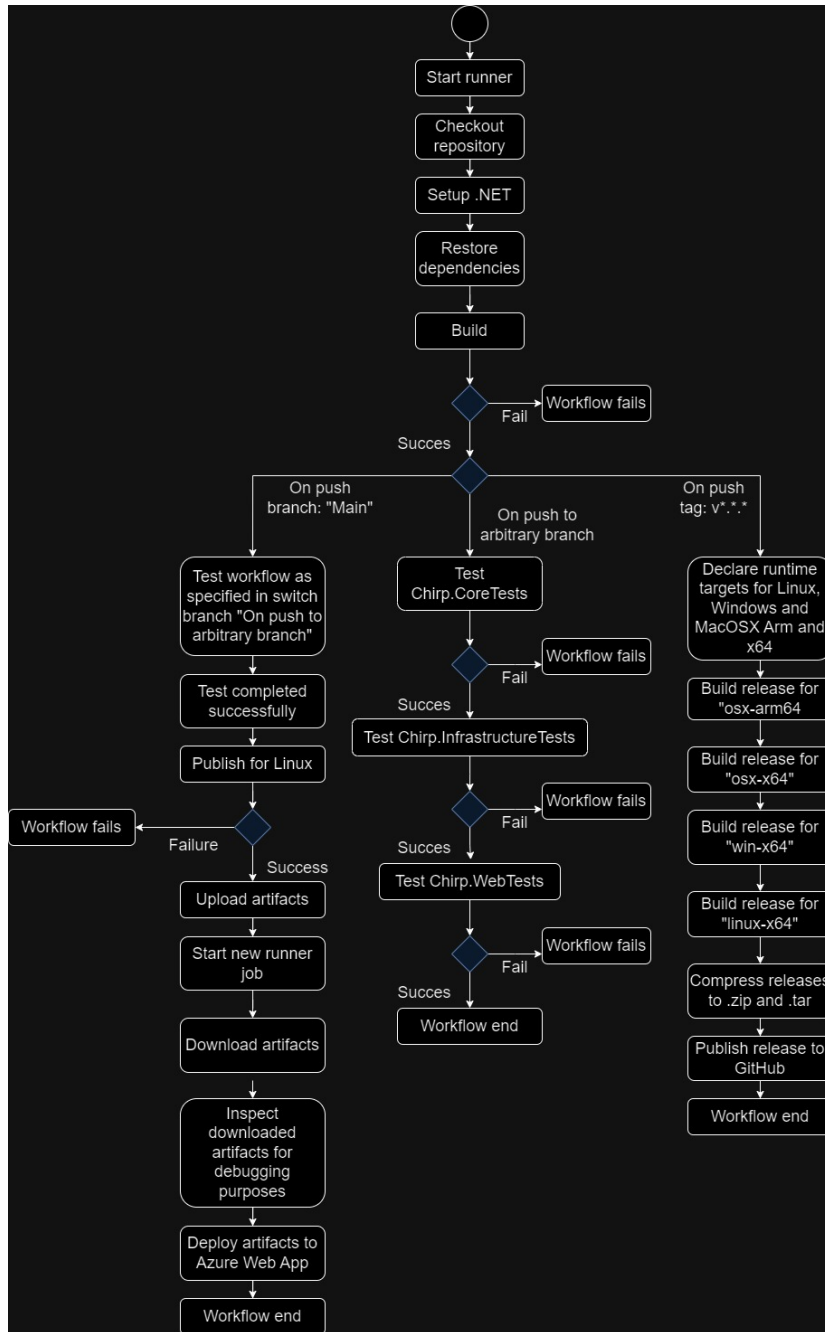from the User Journey flowchart for clarity and readability.

1.5 Sequence of functionality/calls trough *Chirp!*
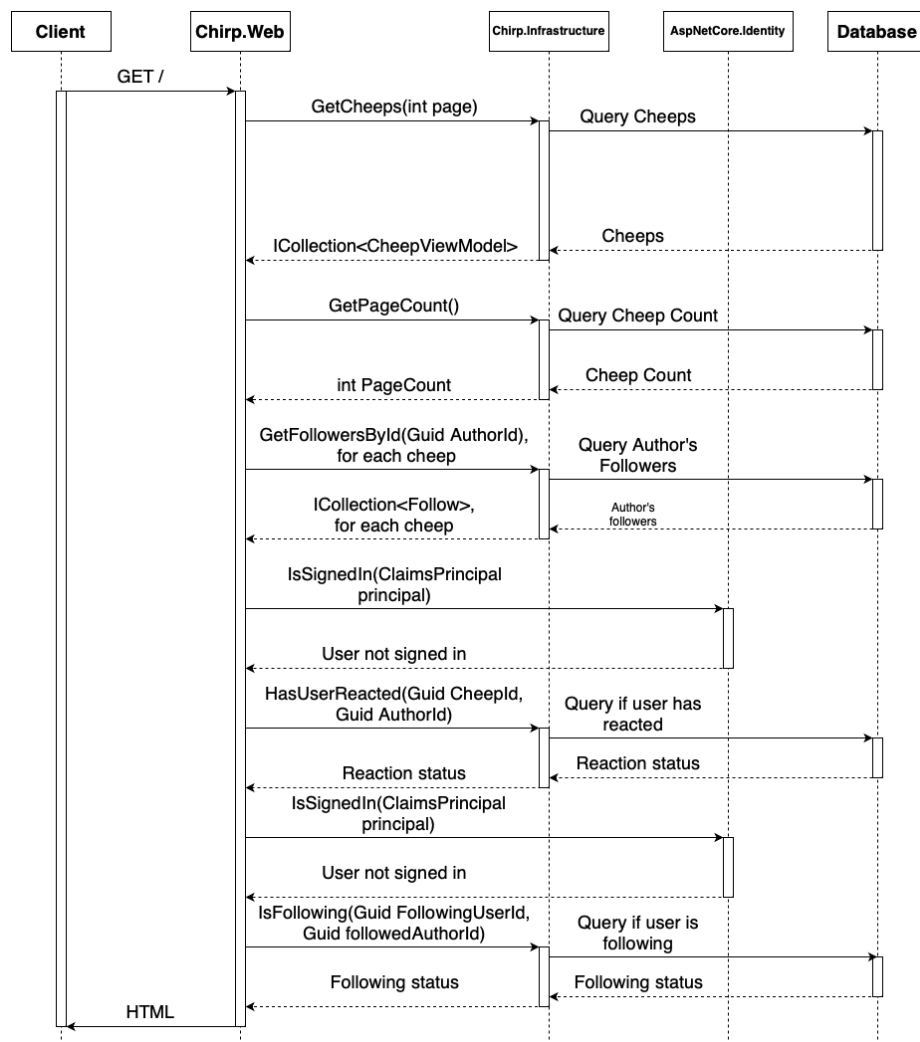
Sequence call diagram for an un-authenticated user requesting the
root endpoint of the application.

2 Process

2.1 Build, test, release, and deployment

2.1.1 Build, and Test

On pull, and push-requests, the program is built, and tested using the provided test-suite.
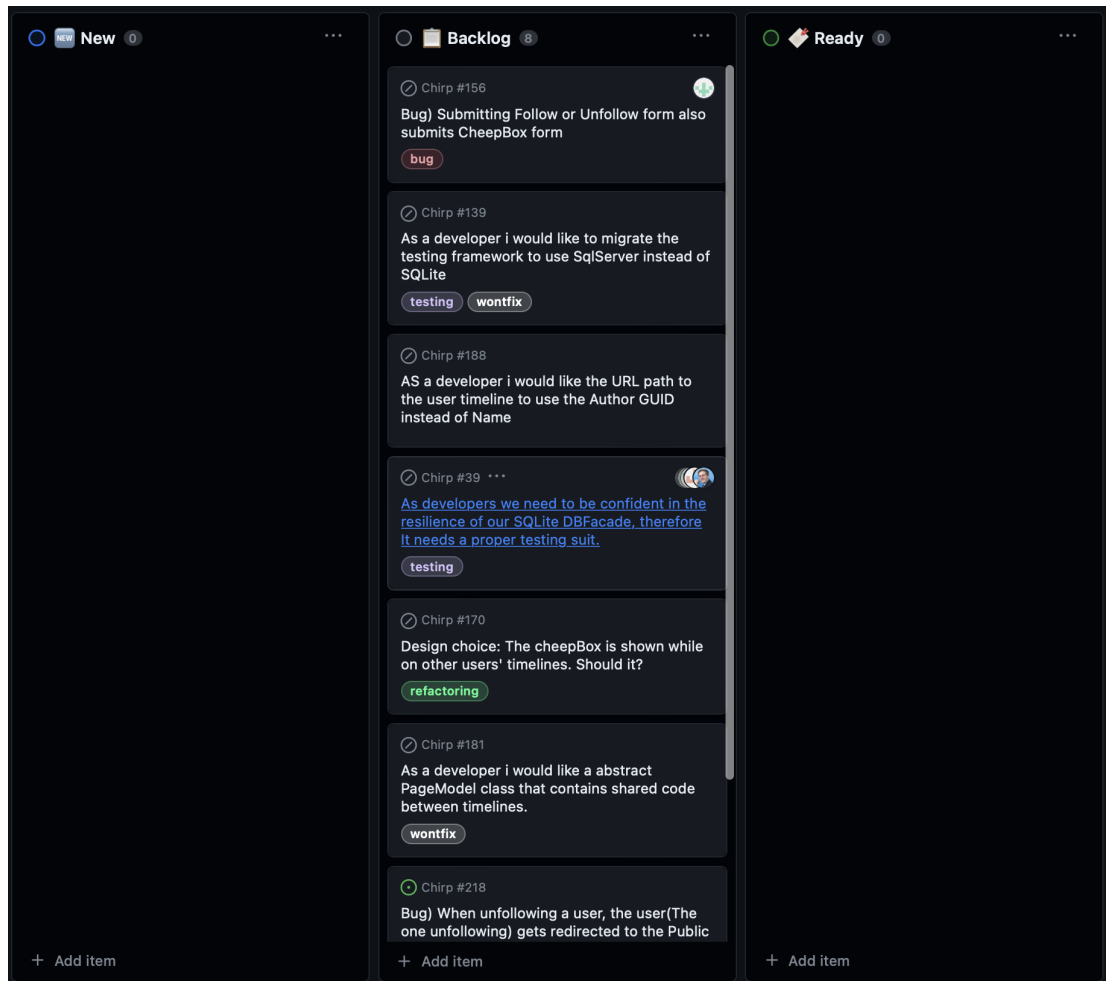
2.1.2 Deployment of the application

On push to the "main" branch, the program is built and tested. If building and testing of the program succeeds, the program is compiled for Linux and uploaded as an artifact to the deployment job.

The deployment job downloads the artifacts, and deploys them to the Azure Web Service.

2.1.3 Release of the application

On pushes with the tag "v..*", the program is Built. After building of the program succeeds, runtime targets for"osx-arm64", "osx-x64", "win-x64", "linux-x64" are defined, and the program is subsequently compiled for these platforms. After compiling for the platforms, the outputted folders are compressed to .zip or .tar formats, depending on what is appropriate for the targeted platform. The compressed folders are now released on GitHub.

2.2   Team work

New 0

Backlog 8

Chirp #156

Bug) Submitting Follow or Unfollow form also submits CheepBox form

`bug`

Chirp #139

As a developer i would like to migrate the testing framework to use SqlServer instead of SQLite

`testing`   `wontfix`

Chirp #188

AS a developer i would like the URL path to the user timeline to use the Author GUID instead of Name

Chirp #39 ...

As developers we need to be confident in the resilience of our SQLite DBFacade, therefore It needs a proper testing suit.

`testing`

Chirp #170

Design choice: The cheepBox is shown while on other users' timelines. Should it?

`refactoring`

Chirp #181

As a developer i would like a abstract PageModel class that contains shared code between timelines.

`wontfix`

Chirp #218

Bug) When unfollowing a user, the user(The one unfollowing) gets redirected to the Public

Ready 0

+ Add item

+ Add item

+ Add item

◯ 🏗 In progress 0   ···

◯ ✅ Done 201   ···

⊘ Chirp #45

1.d) As developers we want to showcase a working application, therefore we need to seed our database.

⊘ Chirp #35

1a) As developers we want to switch our database to support an SQLite db

refactoring

⊘ Chirp #86

1. Part 2) The User needs a clear way to login/signout and register inorder to use the login functionality

enhancement

⊘ Chirp #36

1b) As developers we want to ensure the resilience of our program by paginating cheeps.

⊘ Chirp #7

Prepare repo for software testing

enhancement

⊘ Chirp #32

1a)As developers we want to update CheepService to support SQLite inorder to refactor our program to use the file-based SQLite Database.
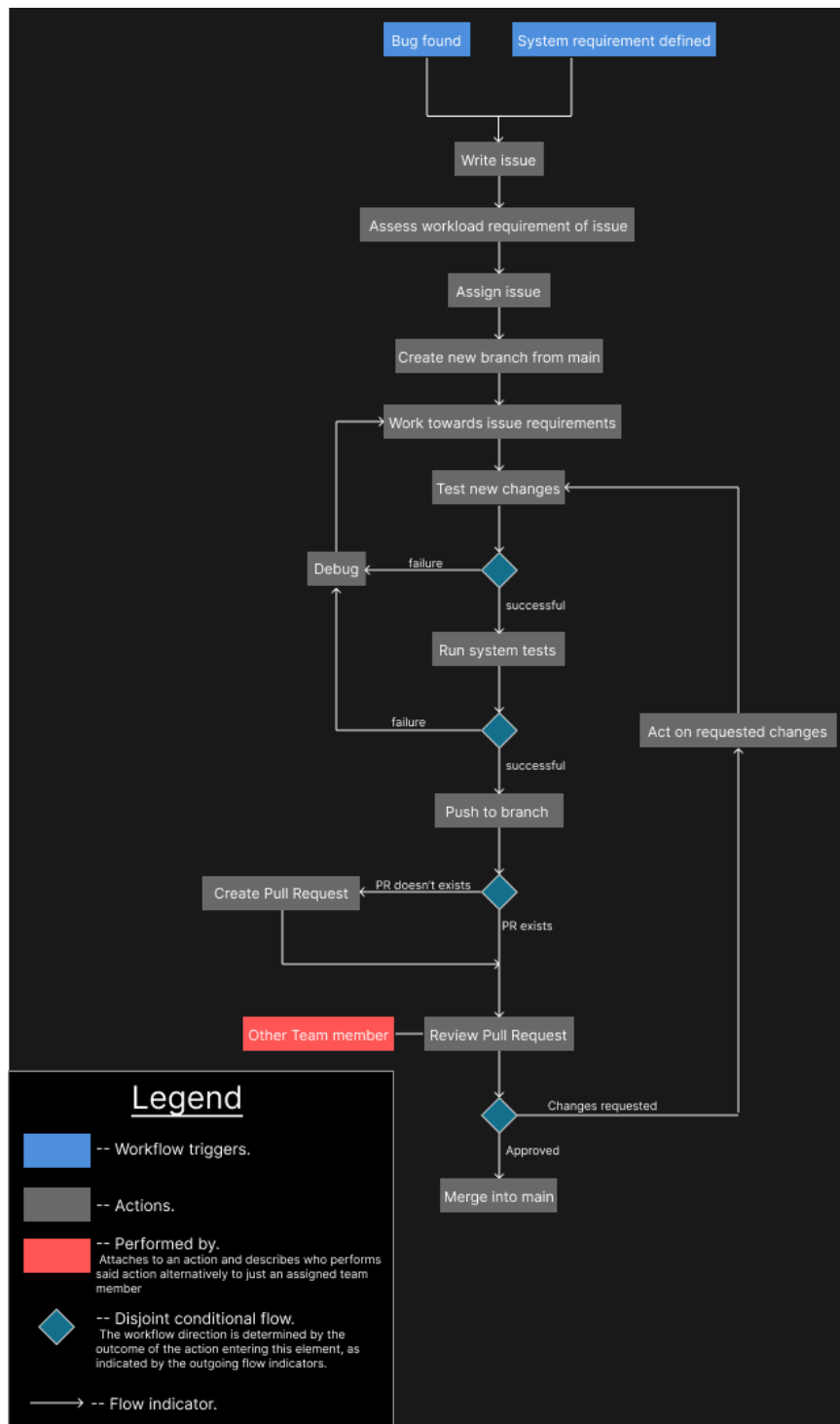
enhancement

⊘ Chirp #34

As a developer I would like to add new API tests to ensure the functionality of the new API

+ Add item   + Add item

2.2.1 Unresolved tasks

- There are a few bugs remaining in the application:
  - Chirp #222 - Sometimes when logging in, the user is not redirected to the home page.
  - Chirp #156 - Whenever a user follows another user, or likes a cheep, the SendCheep validation is triggered.
  - Chirp #218 - When a user unfollows another user, the user(That unfollows) gets redirected to the public page.
- Features we would have liked to improve on:
  - (No issue created) - Extend the functionality of Reactions, so that a user is presented with a selection of reactions to use.
  - Chirp #170 - Adjust the visibility of the cheepbox(Where the user writes a cheep).
- Chirp #181 - A larger refactorization of the razor pages' models, based on introducing a base Page to hold shared functionality.

Bug found **System requirement defined**

Write issue

Assess workload requirement of issue

Assign issue

Create new branch from main

Work towards issue requirements

Test new changes

Debug ← failure ◆ successful

Run system tests

failure ◆ successful

Push to branch

Create Pull Request ← PR doesn't exists ◆ PR exists

Act on requested changes

Other Team member — Review Pull Request

◆ Changes requested

Approved

Merge into main

Legend

-- Workflow triggers.

-- Actions.

-- Performed by.
Attaches to an action and describes who performs said action alternatively to just an assigned team member

-- Disjoint conditional flow.
The workflow direction is determined by the outcome of the action entering this element, as indicated by the outgoing flow indicators.

→ -- Flow indicator.

The process of implementing changes to the system starts by defining the wanted changes or the problem at hand and writing an issue describing it. The required workload is assessed for the issue. The issue is then assigned to an appropriate amount of team members. The team member(s) then creates a new branch from main and starts working towards the issue acceptance criteria. The new changes are then tested to see if it works as expected. Once it works, run the system tests to ensure no other functionality has been broken in the process. If any test fails, debugging commences and the test phase repeats. If everything works, the changes are pushed to the branch and a pull request is made. Another team member reviews said pull request. If any changes are requested, those will be acted upon and the test phase repeats. When a pull request is approved, the branch will be merged into main and deleted.

### 2.2.2   Discrepancies in commit amounts of group members

When observing the amount of commits made by each group member, some anomalies can be seen. If a team member has a noticeably larger amounts of commits than others, it should be taken into account, that some or many of these extra commits are in regards to the deployment and release workflow files, that were the cause of a lot of commits relating to debugging these workflows.

## 2.3   How to make *Chirp!* work locally

2.3.1   1. cloning the project

One way of cloning the project is through the Github Desktop application.

- In the Github Desktop application navigate to the '*file*' tab in the top left corner and press 'Clone repository'.
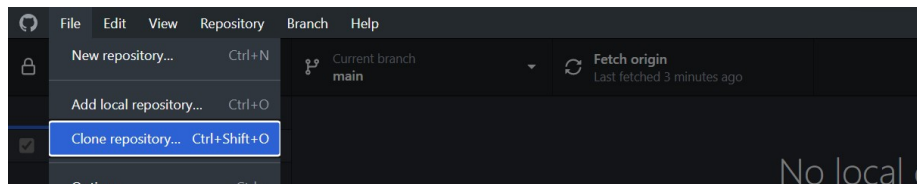
- Navigate to the 'URL' tab.

- Input the following URL in the 'URL' input field.

  `https://github.com/ITU-BDSA23-GROUP20/Chirp.git`

- Input the appropriate filepath for where you want to store the files in 'Local path' input field.
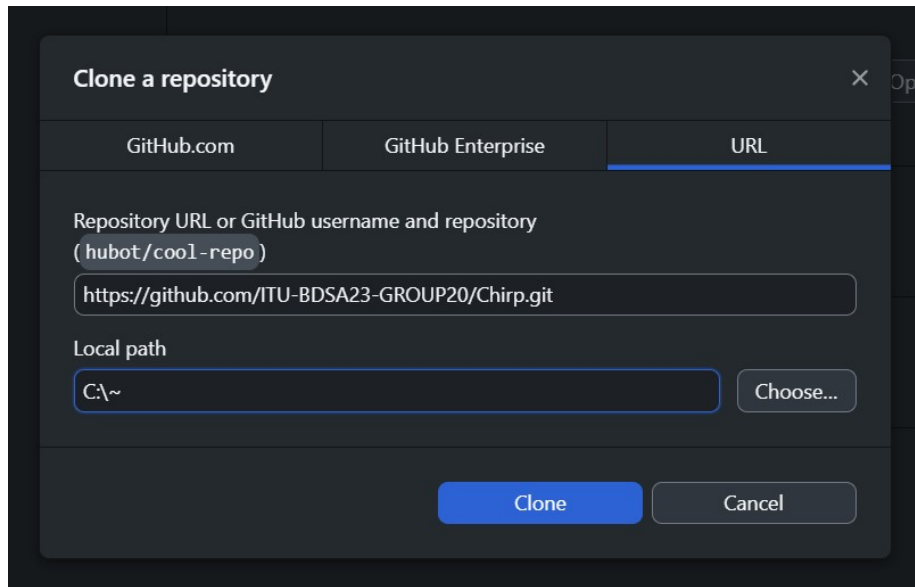
- Press the clone button. Alternatively, the project can be cloned through the terminal with the following command:

  ```
  git clone https://github.com/ITU-BDSA23-GROUP20/Chirp.git
  ```

2.3.2   2. Setting up user secrets

Once you have the project cloned, make sure that you have your GitHub OAuth secrets ready. Once that is done, open a new terminal and navigate to the path `Chirp/src/Chirp.Web`, then run these commands:

```
dotnet user-secrets init
```

Then

```
dotnet user-secrets set "authenticationGithubClientId" "<YOUR_CLIENT_ID>"
```

Where `<YOUR_CLIENT_ID>` is your generated client ID. Then

```
dotnet user-secrets set "authenticationGithubClientSecret" "<YOUR_CLIENT_SECRET>"
```

Where `<YOUR_CLIENT_SECRET>` is your generated client secret.

2.3.3   3. Running the system

- Open a new terminal and navigate to `Chirp/src/Chirp.Web` in said terminal.

- Type the following command to run the system locally.

  ```
  dotnet run --launch-profile https
  ```

15

Note: The launch profile argument is only necessary if you intend to use github authentication.

The terminal should output the following lines:

```
Building...
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (11ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT "u"."UserId", "u"."LoginProvider", "u"."ProviderDisplayName", "u"."ProviderKey"
      FROM "UserLogins" AS "u"
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (0ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT EXISTS (
          SELECT 1
          FROM "Users" AS "u")
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (0ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT EXISTS (
          SELECT 1
          FROM "Cheeps" AS "c")
warn: Microsoft.AspNetCore.Server.Kestrel.Core.KestrelServer[8]
      The ASP.NET Core developer certificate is not trusted. For information about trusting the ASP.NET Core developer c
ertificate, see https://aka.ms/aspnet/https-trust-dev-cert.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
```

- Open your browser of choice and open a page with the following URL.

  `localhost:5001`

  You should be greeted by the following page.
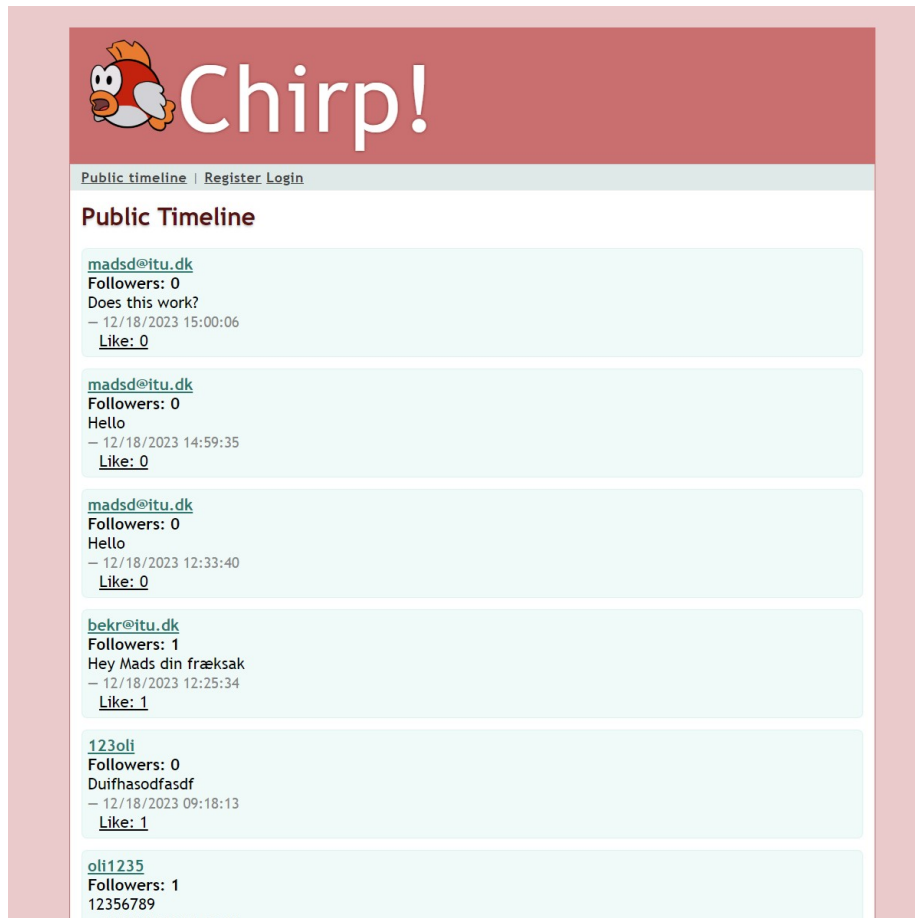
You are now running the system locally.

2.4 How to run test suite locally

To locally run the test suite, simply enter the following commands while being in the root directory of the project:

```
dotnet test test/Chirp.CoreTests;
dotnet test test/Chirp.InfrastructureTests;
dotnet test test/Chirp.WebTests
```

The test suite can be located in the `Chirp/Test` folder. Here you will find 4 folders, `Chirp.CoreTests`, `Chirp.InfrastructureTests`, `Chirp.WebTests` and `Test_Utilities`. The `Test_Utilities` folder contains helper classes and methods used in the test suites. The other folders contain the test suites for the respective packages.

2.4.1   Contents of test suites

2.4.1.1   Chirp.CoreTests

- Testing that records and classes in fact contain the types and attributes defined in the class and record definitions.
- Testing of constraints on class and record members.
- Testing of correct instantiation of objects as defined in the `Chirp.Core` package.

2.4.1.2   Chirp.InfrastructureTests

- Testing that repositories(Base-,Cheep-, Author-, Follow- and Reaction-repository) retrieve, delete, update and add data correctly.

2.4.1.3   Chirp.WebTests

- Testing that the application can transition data from data rich entities, residing in the database, to view-models suited for frontend.

The tests located in Chirp.CoreTests and Chirp.InfrastructureTests are all Unit tests.

3   Ethics

3.1   License

This application uses the GPL-2 software license.

3.2   LLMs, ChatGPT, CoPilot, and others

In our project, we have attempted to strategically leverage LLMs to enhance efficiency where reasonable. The LLMs used in the project are Chat-GPT and Github Co-Pilot. They served primarily as sparring partners during debugging processes, where they occasionally have proven helpful in interpreting cryptic error messages or identifying the cause of an unexpected result.

In our experience, the replies received from the LLMs have often been inadequate or inaccurate, which is likely due to the LLMs incomplete scope of the project. Despite this, the LLMs have helped the development of the project although it has not been substantially.