TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Daniel Berežnoi 193816IAIB

# WEB APPLICATION FOR DROPLET-BASED EXPERIMENT ANALYSIS DATA VISUALISATION.

Bachelor's Thesis

Supervisor: Evelin Halling
PhD

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Daniel Berežnoi 193816IAIB

# VEEBIRAKENDUS TILGAPÕHISE KATSEANALÜÜSI ANDMETE VISUALISEERIMISEKS.

Bakalaureusetöö

Juhendaja: Evelin Halling
PhD

Tallinn 2023

# Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Daniel Berežnoi

22.05.2023

# Abstract

The goal of this thesis is to develop a user-friendly web application, named EasyFlow, specifically designed for the visual analysis of droplet-based experiment analysis data. The application was commissioned by a group of researchers from TalTech University who recognized the need for an intuitive and accessible tool in the field of research droplet-microfluidics.

EasyFlow aims to provide researchers with a comprehensive environment where they can easily visualize and analyze their experimental data. The application incorporates a range of features that streamline the process of analyzing droplet-based experiments, addressing the limitations found in existing tools. The primary motivation behind creating EasyFlow was to offer a beginner-friendly solution, as many existing tools in this domain require a significant learning curve to operate effectively.

In addition to developing EasyFlow, this thesis also aims to compare and evaluate the development processes using two different technologies: Streamlit, a Python library for building interactive web applications, and the Angular+Flask pair, which handles the frontend and backend components respectively. By assessing and contrasting these approaches, the thesis aims to identify the benefits and drawbacks of each option, shedding light on the most suitable development framework for similar projects in the future.

The thesis is written in English and is 27 pages long, including 7 chapters, 9 figures and 0 tables.

# Annotatsioon

## Veebirakendus tilgapõhise katseanalüüsi andmete visualiseerimiseks.

Selle lõputöö eesmärk on arendada kasutajasõbralik veebirakendus nimega EasyFlow, mis on spetsiaalselt loodud tilgapõhiste eksperimentide analüüsi andmete visuaalseks analüüsiks. Rakenduse tellisid TalTechi ülikooli teadlaste rühm, kes märkasid vajadust intuitiivse ja kättesaadava tööriista järele uurimisvaldkonnas - tilgapõhises mikrofluidikas.

EasyFlow eesmärk on pakkuda teadlastele terviklikku keskkonda, kus nad saavad hõlpsasti visualiseerida ja analüüsida oma eksperimentaalseid andmeid. Rakendus hõlmab mitmeid funktsioone, mis lihtsustavad tilgapõhiste eksperimentide analüüsimise protsessi, käsitledes olemasolevate tööriistade piiranguid. EasyFlow loomise peamine motiiv oli pakkuda algajatele sõbralikku lahendust, kuna paljud selles valdkonnas olemasolevad tööriistad nõuavad tõhusaks kasutamiseks märkimisväärset õppeperioodi.

Lisaks EasyFlow arendamisele on sellel lõpuööl eesmärk võrrelda ja hinnata kahe erineva tehnoloogia, Streamlit'i (Pythoni raamatukogu interaktiivsete veebirakenduste loomiseks) ja Angular+Flaski, arendusprotsesse. Need tehnoloogiad vastutavad vastavalt rakenduse frontend-i ja backend-i eest. Nende lähenemisviiside hindamise ja võrdlemise kaudu püüab lõpuöö tuvastada iga valiku eeliseid ja puudusi, valgustades kõige sobivamat arendus-raamistikku sarnaste projektide jaoks tulevikus.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 27 leheküljel, 7 peatükki, 9 joonist, 0 tabelit.

# List of Abbreviations and Terms

| | |
|---|---|
| API | Application Programming Interface |
| CPA | CellProfiler Analyst |
| REST | REpresentational State Transfer |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| HTTP | Hypertext Transfer Protocol |
| UI | User Interface |
| JWT | JSON Web Tokens |
| JSON | JavaScript Object Notation |
| CASE | Computer-Aided Software Engineering |
| SQL | Structured Query Language |
| IDE | Integrated Development Environment |
| VM | Virtual Machine |

# Table of Contents

# List of Figures

# 1. Introduction

When creating new chemical or biological products, e.g. vaccines, cosmetics, and antibiotics, researchers are obligated to extensively test them for, both their effectiveness and possible side effects. One of the more prevalent methods is tube testing. Tube testing typically refers to a method of analysis where fluids are contained in tubes, and various tests or experiments are performed on the fluids within the tubes.

Since the early 2000s, a new testing method began to gain popularity and attention. Droplet-based microfluidics is a subfield of microfluidics that focuses on the generation, manipulation, and analysis of discrete droplets of fluids, typically on the scale of microliters or smaller. In droplet-based microfluidics, small volumes of liquids are compartmentalised into separate droplets, which act as independent reaction vessels, allowing for precise control and manipulation of individual droplets. By using this method the researchers can dramatically increase the testing throughput while decreasing the amount of wasted liquid. [1] [2]

Naturally, this new method also needs new techniques for experiment analysis and its results' visualisation. One of them is an image-based analysis. Image-based analysis in droplet microfluidics refers to the use of microscopy or imaging techniques to visually analyse droplets generated and manipulated in microfluidic devices. It involves capturing images of droplets using microscopy, followed by image processing and analysis to extract relevant information about droplet size, shape, composition, and behaviour

One group of researchers felt that tools for image-based analysis were not ideal. They often have a steep learning curve or need programming skills or both. Seeing that, they decided to create their application called EasyFlow to simplify the analysis workflow. [3] This application, however, had its limitations. Ranging from framework-related issues to just missing some good-to-have features.

EasyFlow uses a Python framework called Streamlit. It is a sort of two-in-one package, as developers do not need to write HTML or CSS code to create a web application. Nevertheless, Streamlit is a young framework, made for rapid prototyping, and has its issues. The author has worked on EasyFlow as a part of a ITI0303 Software Development Project: procurement (Tarkvaraarenduse tellimusprojekt).

Based on the author's perspective, some of the issues are:

- Streamlit is a fairly rigid framework. Creating complex HTML elements is often overly difficult or impossible. For example, Streamlit does not permit creating any sort of element that needs several consecutive button clicks.
- Streamlit-based applications are slow. When the user changes a single parameter in the app, the whole app must be reloaded.
- Streamlit youth means that the number of specialised libraries is quite low and needs further development.
- Smaller community means there are fewer materials to use during the development.

One possible way to combat both framework-related issues and lack of features is to recreate EasyFlow using more established frameworks.

# 2. Problem Statement

In this thesis the Author will continue working on EasyFlow but on different frameworks.

## 2.1 Clients' needs

The clients of this project are a group of researchers from the microfluidics lab at Taltech Faculty of Science. Clients need an easy-to-use web application for droplet-based experiment analysis visualisation. That means they want to be able to upload their experiment analysis data into the web application and generate a number of dynamic plots and tables. The contact person between that group and the author was the creator of the first iteration of EasyFlow and already had some ideas on what to add and improve. For example, adding the ability to save and share both data and visualisation parameters for later use. To summarise, clients want an easy-to-use web application for experiment analysis visualisation and the ability to save and share both data and visualisation parameters.

## 2.2 Existing application

The first version of EasyFlow was created by clients who worked on Streamlit. The users could upload their data and graphs would be rendered based on the uploaded data. The users then can change visualisation parameters for these plots and tables. The generated plots and tables are interactive.

## 2.3 Thesis goals

This thesis aims to create a working web application for droplet-based experiment analysis visualisation. The web application should be easy-to-use and preferably quickly developed. There should be two types of users in the app: a guest and a standard user.

Guest will be able to:

- Upload data in the application, but it will be saved only for the duration of the session.
- Generate various plots
- Change parameters of the plots
- Use public analysis data

- Use public visualisation parameters

A standard user will be able to:

- Everything a guest user can
- Save their experiment analysis files
- Save their visualisation parameters
- Download their saved experiment analysis files
- Share their experiment analysis files or make them public
- Share their visualisation parameters or make them public
- Delete their experiment analysis files
- Delete their visualisation parameters

The end result should be a user-friendly application that could be used without requiring extensive knowledge about its inner workings. This application will be open-source and available for any researcher to use.

After the web application is created the author will compare the development processes

## 2.4 Potential Solutions

Due to droplet-based microfluidics being a rapidly growing field, there are already a number of tools that can be used to visually analyse the experiment results.

### 2.4.1 CellProfiler

CellProfiler is an open-source software designed for automated image analysis in biological and biomedical research. It provides a user-friendly interface for analyzing images of biological samples, such as cells or droplets and extracting quantitative measurements from these images. The software is widely used by researchers in fields like cell biology, genetics, and drug discovery to study various biological phenomena.

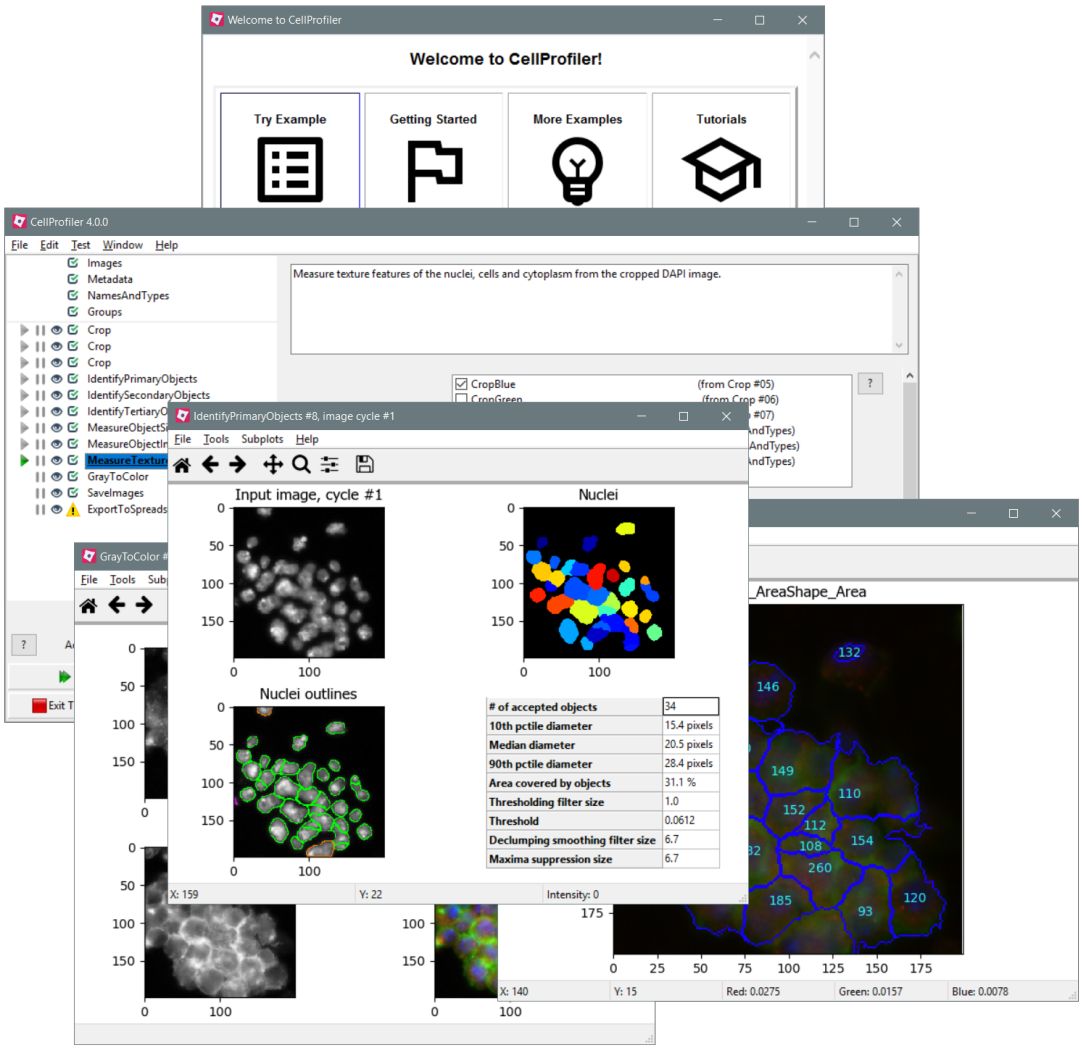



Figure 1. Various windows of CellProfiler.

CellProfiler has several notable advantages. Firstly, it is an open-source project, meaning it is freely available to users, making it accessible to researchers with limited budgets. Additionally, CellProfiler offers automation capabilities, allowing users to automate repetitive tasks and perform high-throughput analysis. This feature saves considerable time and effort compared to manual analysis.

Furthermore, CellProfiler's flexibility is a significant strength. It provides a wide range of customisable modules and tools for image analysis. Users can tailor their analysis workflows to suit their specific research needs, enhancing the software's versatility.

However, while CellProfiler has significant advantages it has its issues. One of the primary challenges for new users is the learning curve associated with the software.

Individuals without prior experience in programming or image analysis may find it difficult to master CellProfiler. Without understanding the underlying concepts and workflows using CellProfiler may feel overwhelming

Moreover, creating specialised analysis workflows in CellProfiler may pose additional complexities. Advanced programming skills are often required to achieve highly specific analysis goals. This aspect can deter users who lack extensive programming expertise from fully realising the software's potential.

In summary, CellProfiler offers numerous benefits, such as its open-source nature, automation capabilities, and flexibility through customisable modules. However, it presents challenges for first-time users due to its learning curve and the requirement of programming skills for specialised analyses.

# 3. Technologies

The application consists of three separate parts: Angular client side, Flask REST API and PostgreSQL database.

## 3.1 Angular client side

The reason behind using Angular framework for the client, or any front-end framework at all, was to make the client more customisable and the fact that Angular is one of the most popular front-end frameworks and has a very big community. The client was done mainly for the desktop, as the application has a lot of various inputs, plots, and tables, making the mobile client hard to design (at least for the author).

### 3.1.1 Angular

Angular 15 is an open-source Typescript-based web application framework developed by Google. It allows the quick creation of complex and flexible single-page web applications. Angular is also compatible with other frameworks. Angular has a simple and organised file structure, where the application is divided into components and each component has its own CSS stylesheet, Typescript file and HTML template.

The author has picked Angular because of several factors. As Angular is very popular, it has a massive community with a wide array of materials available online. The author finds its file structure to be pleasant. The author has some previous experience with Angular.

### 3.1.2 Typescript

TypeScript is a superset of JavaScript, which is a statically-typed programming language. It adds optional static typing and other advanced features to JavaScript, making it a powerful tool for building complex web applications.

The typescript was chosen because it supports interfaces for HTTP responses. Another reason is typing which makes it easier to understand the code and predict the results of running it. The final reason is the author's previous experience with it.

### 3.1.3 BokehJS

BokehJS is a front-end Javascript library for rendering Bokeh plots. Using BokehJS one could render Bokeh plots by calling just a single function with plot data as an argument.

BokehJS was chosen because EasyFlow used Bokeh plots from the start and needed some front-end libraries to render them.

### 3.1.4 Bootstrap 5

Bootstrap 5 is a CSS framework, used for creating various HTML elements

### 3.1.5 Angular Material

Angular Material is a UI component library that developers can use in their Angular projects

## 3.2 Flask API

The role of the backend in this project is to receive HTTP requests from the client, process them and return appropriate responses. The backend is running in a virtual environment because Flask can potentially interfere with other programs on the machine.

### 3.2.1 Flask

Flask is a lightweight and flexible Python web framework that makes it easy to build web applications quickly and efficiently. It follows the micro-framework approach, which means it provides only the bare essentials for building web applications, giving developers the freedom to choose and integrate additional components as needed. [4]

Flask was chosen because it is an established framework with a big community, third-party library support, and extensive documentation.

### 3.2.2 PyJWT

PyJWT is a Python library that allows developers to encode and decode JSON Web Tokens (JWT). Generated JWT tokens can be then used to authenticate users and make

the application stateless. While encoding and decoding, developers can use a secret key to verify that tokens are secured.

PyJWT library was chosen because using JWT tokens looked more understandable than alternatives.

### 3.2.3 SQLAlchemy

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language. [5]

SQLAlchemy was chosen to simplify the performing of operations with the database.

### 3.2.4 Bokeh

Bokeh is a Python library used for creating a wide range of plots and tables. Tables are generated as HTML code, while plots are sent as JSON objects. The decision to use the Bokeh library was made because it has been used from the inception of EasyFlow, and switching to another library would be counterproductive.

## 3.3 Database

The database for this project was created using PostgreSQL

### 3.3.1 PostgreSQL

PostgreSQL is an object-relational database management system. It supports a large part of the SQL standard and offers many modern features. And because of the liberal license, PostgreSQL can be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic. [6]

PostgreSQL was chosen because the author has previously worked with it.

## 3.4 Deployment

### 3.4.1 NGINX

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. [7]

### 3.4.2 GitLab runner

GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline. The runner can be used to automate the process of deploying the application. The users can specify the various stages and commands that would be run after the pipeline is started. [8]

### 3.4.3 Gunicorn

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy. [9]

# 4. Used tools

While working on this project the author has used a variety of tools for developing the application and track the progress made. These were a big help and could be used for any other project.

## 4.1 Tools used in the application development

### 4.1.1 Figma

Figma is a widely-used and powerful cloud-based design tool that enables the creation of user interfaces, web designs, mobile app designs, and other visual design projects. It offers a comprehensive set of design features, including vector editing, prototyping, design components, and design libraries. [10]

Using a dedicated design tool like Figma can simplify the development of user interfaces for developers, as designing page views in tools like Figma can be faster and more efficient compared to designing directly in an integrated development environment (IDE).

The author chose Figma because it provides all the necessary tools for designing websites and was the first tool that the author discovered. However, it's important to note that there may be other design tools available that could also meet the author's needs.

### 4.1.2 Enterprise Architect

Enterprise Architect is a comprehensive CASE (Computer-Aided Software Engineering) tool that allows for the creation of conceptual and physical designs and diagrams for databases. It offers support for various database systems, including PostgreSQL, and provides features for designing tables with indexes, constraints, and columns. One of the notable advantages of Enterprise Architect is its ability to automatically generate SQL code for PostgreSQL with minimal modifications. [11]

The author chose Enterprise Architect because it offers a wide range of tools that streamline the process of creating a database, and because the author has had extensive experience using it.

# 5. EasyFlow

This iteration of Easyflow consists of three parts: Flask REST API, Angular frontend and PostgreSQL database.

## 5.1 Backend API

### 5.1.1 Authentication service

One of the features that the client wanted to be implemented was the saving and sharing of the droplet analysis data and visualisation parameters. To do that I needed a database, obviously. However, making this feature available to all users is not optimal, as accessing these saved data and parameters without some account id could prove to be overly complicated. Thus the author and clients decided to add authentication to the application.

A simple username and password pair is used for logging in. When users register the passwords are hashed and salted using Bcrypt. Bcrypt is a password hashing function designed, it uses strong cryptography to hash and salts passwords. Both hashed password and salt are stored in the database user_account table. When a user logs in the inserted password is hashed using the salt attached to the user account and compared to the stored password hash.

The author decided to use JWT tokens as the chosen method for authenticating and validating users in the project due to their robust security features. When a user logs in to the client, a new JWT token is generated, which contains essential user information such as the user ID, username, and role. This information enables the API to run SQL operations on behalf of the user and limit methods that users have access to. Additionally, each token is assigned an expiration date and time, and if the current date and time exceed the token's expiration, the token is deemed invalid.

To encode the JWT token, the PyJWT library's provided methods are utilised, and a secret key is used to secure the token. The secret key is a randomly generated string that is stored in a separate file to ensure its confidentiality.

To validate tokens, a custom decorator is created, which is applied to every endpoint that should be limited to a certain role. This decorator ensures that only users with the

appropriate role can access the restricted endpoints, adding an additional layer of security to the authentication process.

When considering the authentication method, there were two options: JWT tokens and Flask Login. Flask Login is a library specifically designed for Flask, offering session-based authentication. In this approach, user credentials are stored on the server side, and a session ID is stored in a client-side cookie. However, the decision was made to favour JWT tokens due to their straightforward implementation. The only challenging part involved creating a custom decorator for token validation. In contrast, Flask Login required understanding multiple decorators, and while comprehending their purpose might have been easy, it was still more complex compared to JWT tokens.

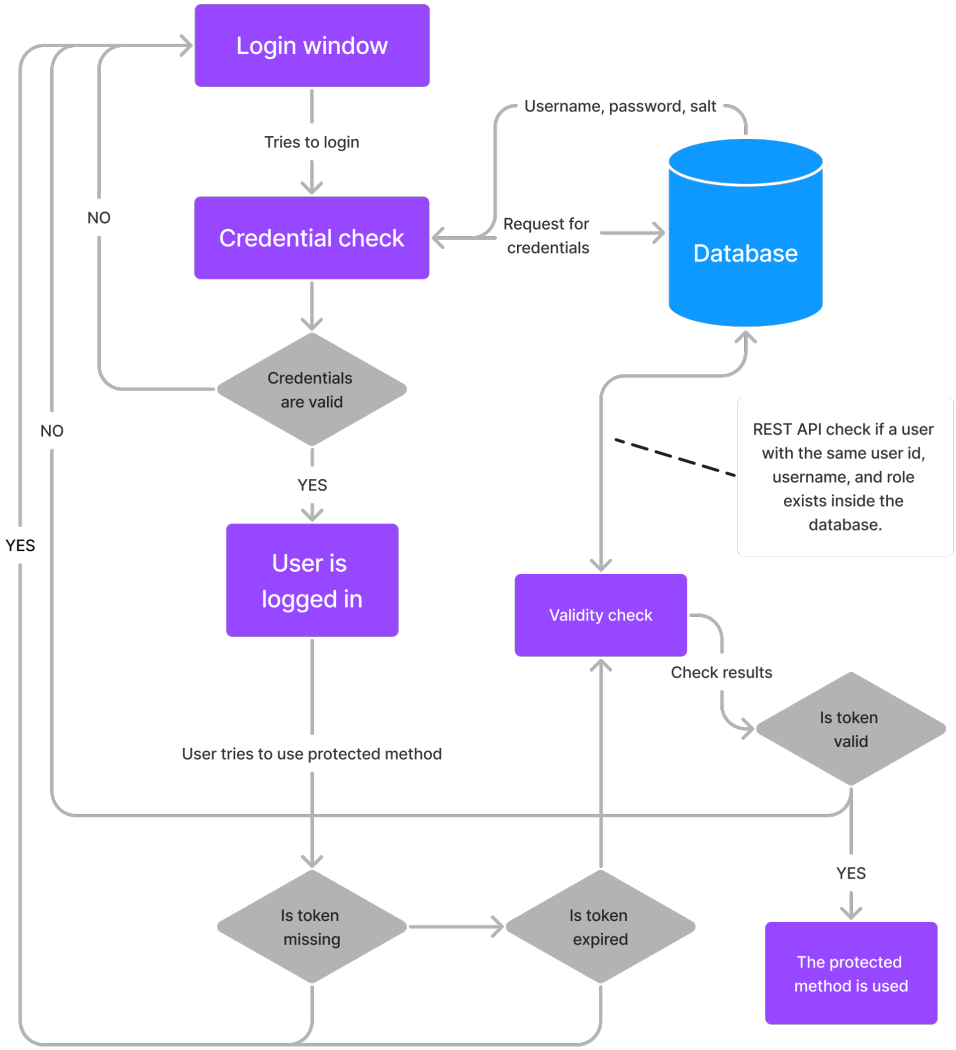


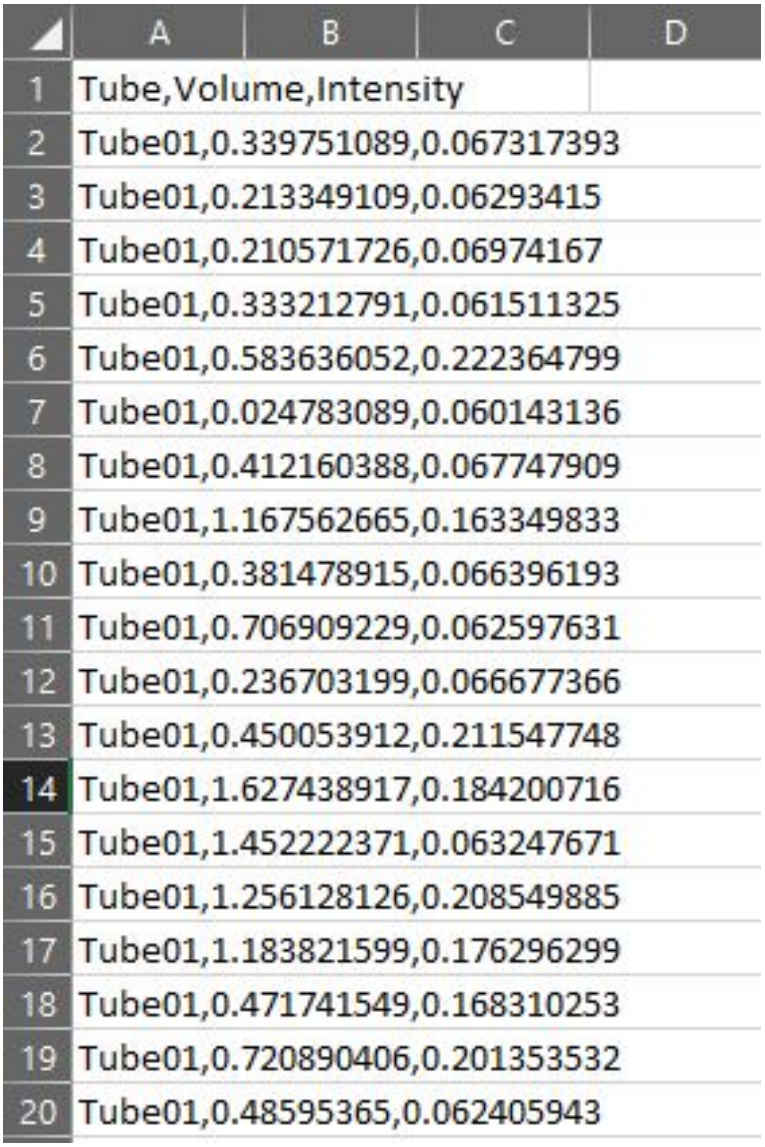

Figure 2. The authorisation and authentication workflow.

### 5.1.2 Experiment data service

The experiment data service plays a crucial role in receiving and processing various requests related to droplet experiment data within EasyFlow. Its responsibilities include reading and saving data, fetching data from the database, sharing data with other users, and deleting data from the system.

To process the output data generated by the image analysis software, EasyFlow relies on

the Pandas library. The application specifically accepts Excel and CSV file formats as input data. When a user uploads or selects droplet data, the data is processed into a Pandas DataFrame and temporarily saved on the backend for further processing.



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Tube,Volume,Intensity | | | |
| 2 | Tube01,0.339751089,0.067317393 | | | |
| 3 | Tube01,0.213349109,0.06293415 | | | |
| 4 | Tube01,0.210571726,0.06974167 | | | |
| 5 | Tube01,0.333212791,0.061511325 | | | |
| 6 | Tube01,0.583636052,0.222364799 | | | |
| 7 | Tube01,0.024783089,0.060143136 | | | |
| 8 | Tube01,0.412160388,0.067747909 | | | |
| 9 | Tube01,1.167562665,0.163349833 | | | |
| 10 | Tube01,0.381478915,0.066396193 | | | |
| 11 | Tube01,0.706909229,0.062597631 | | | |
| 12 | Tube01,0.236703199,0.066677366 | | | |
| 13 | Tube01,0.450053912,0.211547748 | | | |
| 14 | Tube01,1.627438917,0.184200716 | | | |
| 15 | Tube01,1.452222371,0.063247671 | | | |
| 16 | Tube01,1.256128126,0.208549885 | | | |
| 17 | Tube01,1.183821599,0.176296299 | | | |
| 18 | Tube01,0.471741549,0.168310253 | | | |
| 19 | Tube01,0.720890406,0.201353532 | | | |
| 20 | Tube01,0.48595365,0.062405943 | | | |

Figure 3. This is the input data that experiment data service deals with.

Once the DataFrame is generated, it is saved within the code for the duration of the session. This approach enhances the efficiency of data utilization when updating parameters. It allows for quick access and manipulation of the data during the session, optimizing the user experience.

Additionally, if a logged-in user wishes to save the data, the DataFrame is combined with a specified filename to generate a new file. This file can then be saved on the server, providing an option for users to persistently store the DataFrame as a separate file. This

ensures that users can access and manipulate the data swiftly during the session and also have the capability to store the DataFrame for future use.

| | Tube | Volume | Intensity | Label |
|---|---|---|---|---|
| 0 | Tube01 | 0.339751 | 0.067317 | Tube01 |
| 1 | Tube01 | 0.213349 | 0.062934 | Tube01 |
| 2 | Tube01 | 0.210572 | 0.069742 | Tube01 |
| 3 | Tube01 | 0.333213 | 0.061511 | Tube01 |
| 4 | Tube01 | 0.583636 | 0.222365 | Tube01 |
| ... | ... | ... | ... | ... |
| 37294 | Control | 0.177523 | 0.059682 | Control |
| 37295 | Control | 0.318775 | 0.060561 | Control |
| 37296 | Control | 0.134310 | 0.060770 | Control |
| 37297 | Control | 0.592799 | 0.061018 | Control |
| 37298 | Control | 0.565458 | 0.061442 | Control |

[37299 rows x 4 columns]

Figure 4. The pandas generated dataframe as shown in terminal.

### 5.1.3 Visualisation service

In the previous iteration of EasyFlow, the visualisation service was the only component that was adapted instead of being completely rewritten. The primary role of the visualisation service is to generate plots and tables based on incoming visualisation parameters and experiment analysis data.

To render plots, the visualisation service sends the plot data to the frontend as a raw JSON string. The BokehJS library is then responsible for rendering the plot based on the received data. For tables, the visualisation service sends an HTML code that is injected into the website document to display the table.

Once the experiment data service receives and processes the experiment data, the generated dataframe is modified by adding a column for categorisation purposes. During the creation of plots and tables, a Python library called NumPy is extensively used. NumPy provides essential functionality for data manipulation thus being extremely useful for plot creation.

By adapting and enhancing the visualisation service, EasyFlow can effectively generate

plots and tables based on user-defined parameters and experiment analysis data, offering users valuable insights and data representation.

## 5.2 Client

The original EasyFlow system did not include a separate frontend application, as all HTML, CSS, and JavaScript codes were automatically generated using a Python library called Streamlit. However, the author found that Streamlit was too inflexible and slow for their requirements, which led them to separate the backend and frontend components.

Based on their prior experience in creating single-page applications, the author opted to use the Angular framework and Typescript to develop the website. He utilised Angular version 15 and Typescript version 4.9.5, with the aim of creating a functional and user-friendly website from the outset. To achieve the desired appearance of the website, the author employed Bootstrap 5 and Angular Material.

During the development process, the author found the design tool called Figma to be an invaluable tool. Prior to adopting Figma, the site creation process was slow. However, Figma enabled the author to rapidly create a design for the application and then generate the necessary components based on that design. This streamlined the development process and facilitated the efficient implementation of the desired user interface for the EasyFlow system.

The author made a decision not to adapt the website for mobile devices, as the intended functionality would have resulted in a clunky user experience on mobile devices anyway.

### 5.2.1 Visualisation page

The visualisation part of the website comprises multiple Angular components and services. The parent component serves as the visualisation page, while each plot has its own dedicated component.

Within the parent component, users have access to various tools for uploading or selecting analysis data, choosing visualisation parameters, and saving their selections if they are logged in. Additionally, users can download sample data for experimentation. The parent component acts as a central hub for these functionalities, coordinating the interactions between the different plot components and the visualisation service.

Each plot component contains useful tables and interactive Bokeh figures. Users can also interact with various inputs within each plot to visually analyse the experiment data. Unlike the Streamlit version of EasyFlow, the combination of Angular and Flask allows users to update the parameters of specific plots without affecting the rest of the application or causing it to reload. This provides a more seamless and efficient user experience for analysing experiment analysis data in the EasyFlow system.

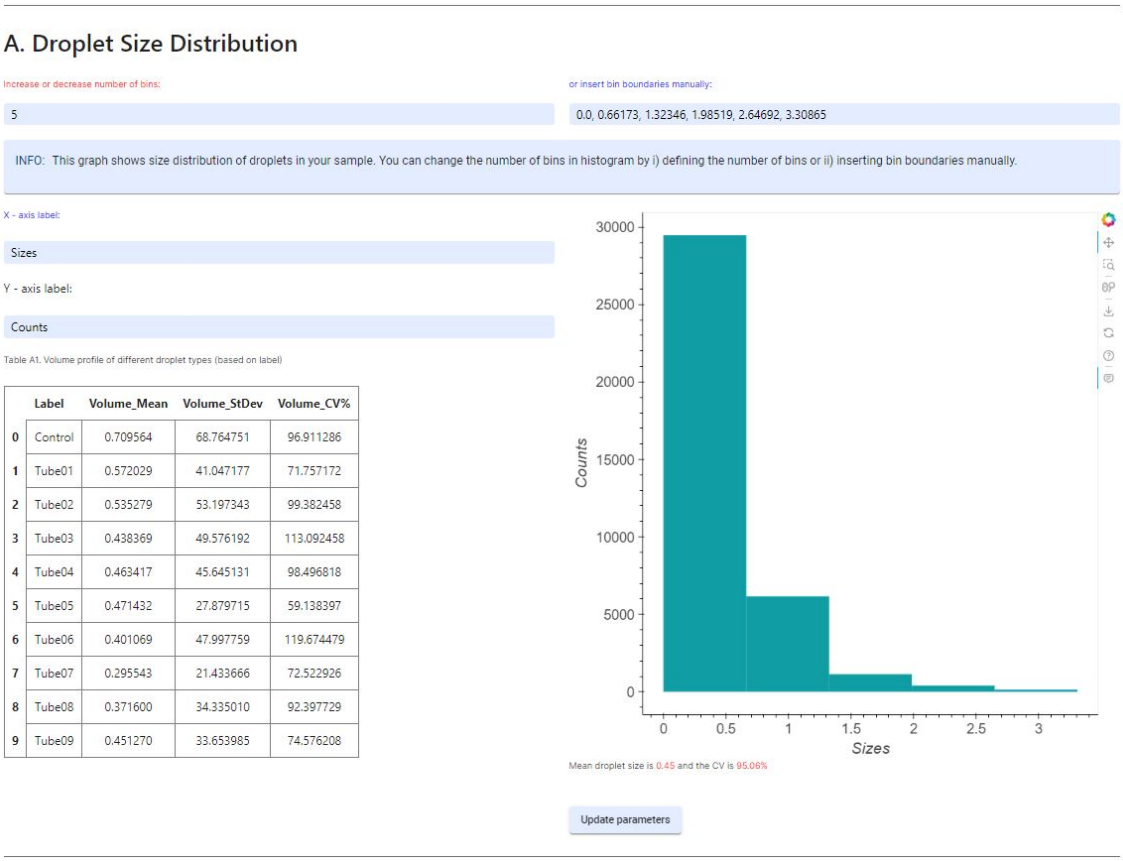



Figure 5. An example of visualisation element. The element is big and had to be zoomed out to be fully captured.

If the user is logged in then they may save their uploaded experiment data file or visualisation parameter set, or both. When they click on the corresponding buttons a pop-up window will appear where they can specify some information about their saved data, e.g. name, description, and public status. Users also could update their visualisation parameters by hitting the corresponding button.

If the users are logged-in, they have the option to save their uploaded experiment data file, visualisation parameter set, or both within the parent component. When the corresponding buttons are clicked, a pop-up window appears where users can specify information about their saved data, such as name, description, and public status. Users can also update their

visualisation parameters by hitting the corresponding button.

This functionality allows logged-in users to store their analysis data and visualisation settings for future reference, making it convenient to revisit and analyse the same data later on. The pop-up window provides a user-friendly interface for managing the saved data, allowing users to provide relevant information and customise their settings as needed. This feature enhances the usability and versatility of the EasyFlow system, providing users with a seamless experience for managing and updating their visualisation parameters and saved data.

### 5.2.2 Experiment Data and Visualisation Parameters Management

The management pages for both visualisation parameters and data management are designed to be nearly identical. To achieve this, the author utilised a single Angular component for the management page. This approach simplifies the process of changing between different management pages and ensures visual consistency throughout.

In the overview section, the author implemented a feature that enables users to quickly delete data or parameters, thereby enhancing the overall user experience. Additionally, on the data management page, users have the option to directly upload their data files, bypassing the need to navigate to a separate visualisation page.

While creating the overview, the author used a table component from the Angular Material library. This library was chosen because it provides all the necessary tools for managing owned data and parameters, such as sorting and pagination. By leveraging the capabilities offered by the Angular Material library, the author was able to enhance the functionality of the overview page.

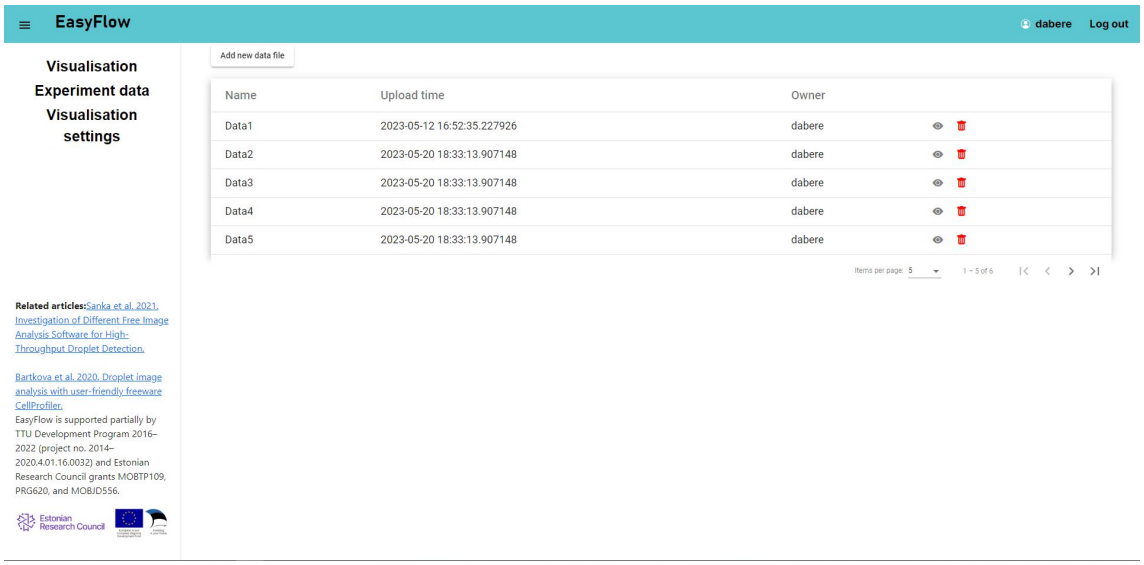



Figure 6. An example of visualisation element. The element is big and had to be zoomed out to be fully captured.

In addition to the features mentioned earlier, the management page also offers users the option to open a detailed view of their data or parameters. This detailed view is displayed as an Angular dialogue window. Within the detailed view, users have the ability to modify the saved item. They can change the item's name, description, and public status (where public items are accessible to unregistered users). Furthermore, users can update the list of individuals with whom the item is shared.

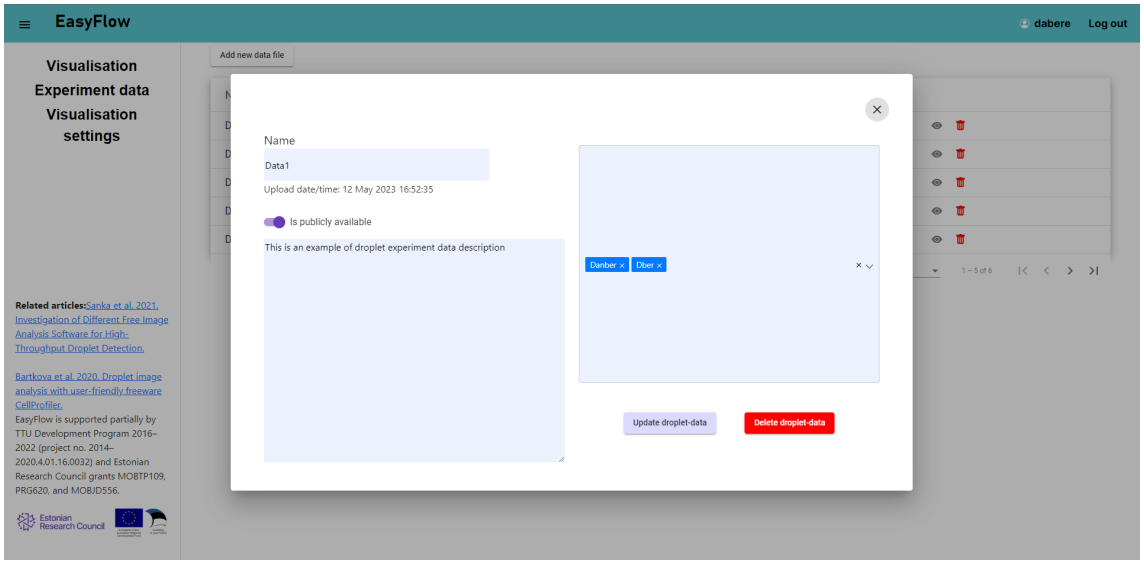



Figure 7. An example of saved experiment data's detailed view in management page.

During the planning phase of the management page, the author initially considered using expansion panel components from the Material library. This approach was based on the

author's previous experience with the Streamlit version of EasyFlow, where expansion panels were employed due to the lack of alternative options that were not overly complex. However, upon further consideration, the author decided to move away from expansion panels. While they would have been easy to implement, their usage would have limited the availability of useful features that material tables offer.

## 5.3 Database

The database for this project was created using PostgreSQL, and the backend is connected to the database using the SQLAlchemy Python library. SQLAlchemy enables developers to define the structure of the database based on the models they create in Python code. In this project, the database was the first component that was developed, as the author had previously created its MySQL version while working on EasyFlow as part of the "ITI0303 Software Development Project: procurement". As a result, SQLAlchemy was primarily used to simplify the task of performing queries.

The author utilised a CASE (Computer-Aided Software Engineering) tool called Enterprise Architect to model the databases, which made the migration of the EasyFlow database from MySQL to PostgreSQL a straightforward process. By using Enterprise Architect, the author was able to easily update the database schema and generate the corresponding SQLAlchemy models in Python code, simplifying the migration process and ensuring consistency and integrity in the data model between the MySQL and PostgreSQL versions of the database.

No data was transferred, as the Streamlit iteration of the EasyFlow the author worked on was just a prototype with no real data in the first place. During the development of the current iteration, the author has performed a number of small fixes, e.g. adding necessary columns, but the overall structure stayed mostly the same.

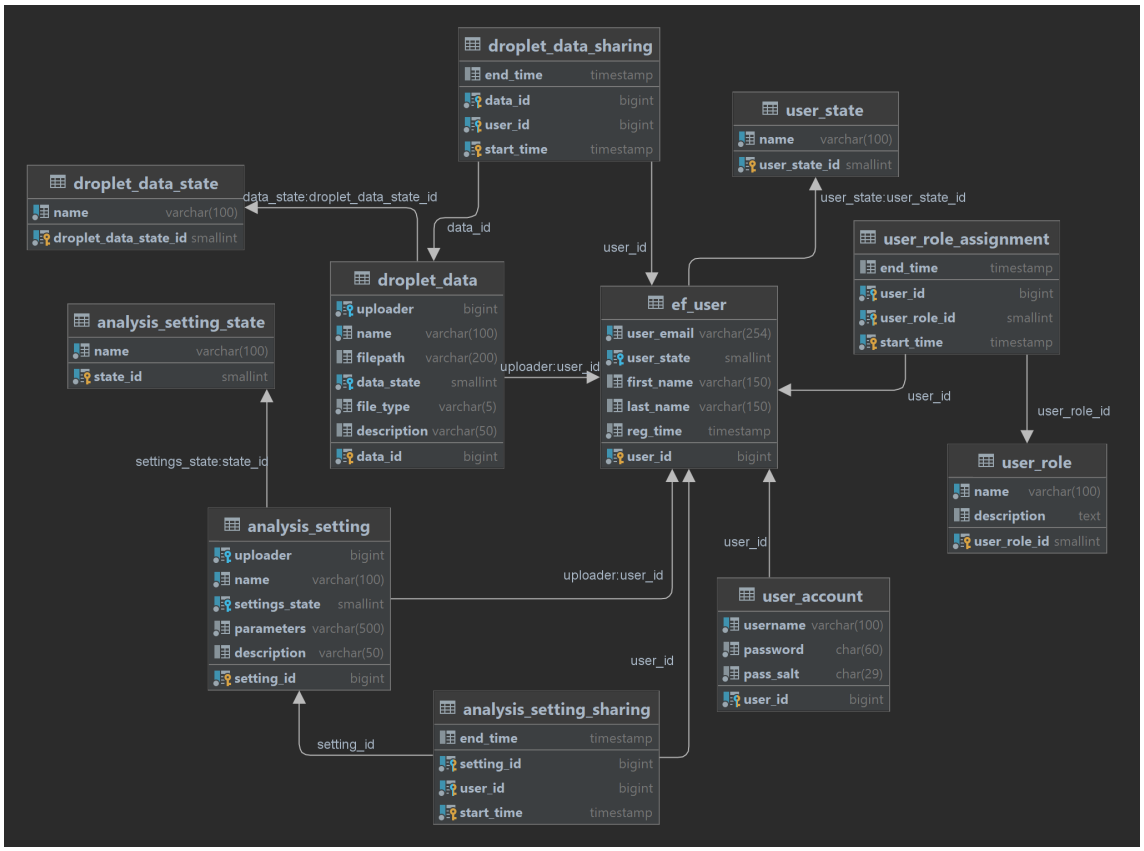



Figure 8. The database diagram. Diagram was created using JetBrains DataGrip.

## 5.4 Deployment

The application is deployed on a Linux server provided by Taltech. This server is originally intended for clients to conduct their own studies, but the author has been granted access to it. All components of the application, including the database server, backend, and frontend, are hosted on this Taltech server.

To facilitate continuous deployment of the frontend, the author utilized GitLab runner. This decision was made to simplify the frontend deployment process, which would have been more complex without the runner. By leveraging GitLab runner, the author automated the application of any changes to the Angular app. The frontend application is served by an Nginx server.

However, for the backend deployment, the author chose not to employ GitLab runner. Since the backend is mostly completed and does not require frequent updates, the runner is not utilised in this case. Instead, the Flask app runs within a virtual environment and is served by a WSGI server. In order to enable external requests to reach the backend, a reverse proxy is configured using Nginx.

When initially considering the deployment strategy, the author contemplated utilising Docker containers. However, after careful consideration, the decision was made to disregard dockerization for this project. The author determined that the benefits of using Docker containers would not significantly impact the project at its current stage. Since EasyFlow is still in the development phase, running it within containers could potentially complicate further development processes.

By opting not to utilise Docker containers, the author aimed to maintain a simpler development environment and facilitate a smoother ongoing development of EasyFlow. This decision allows for a more streamlined development process without the additional complexity introduced by containerisation.

# 6. Conclusion, Analysis, and Potential Future

The end result of this project is an applications that allow users to visualise the data of droplet experiment analysis.

The created collections consist of the following:

- Backend API, created using Flask framework. It conforms to the design principles of the REST architectural style. Its main purpose is to generate various plots and tables for visualisation and perform operations with the database.
- Client application, created using Typescript language and Angular framework.
- PostgreSQL database that holds all needed information for EasyFlow to function

## 6.1 The benefits and drawbacks of using Flask and Angular compared to Streamlit

One of the main reasons for choosing a different framework for recreating EasyFlow instead of using Streamlit was to address performance issues. In Streamlit, whenever a single variable or input field changed, the entire code would run from top to bottom, resulting in slow performance. This had a negative impact on the user experience, especially since EasyFlow's main objective is to provide updatable plots for visualisation with customisable fields. The frequent code reruns in Streamlit made the process of updating these fields slow and inconvenient.

By transitioning from Streamlit to Angular and Flask, the performance issue in EasyFlow was effectively resolved. In Angular, the code reruns only occur when users manually refresh the page or when developers explicitly trigger a rerun. Similarly, in Flask, the reruns of the application are manually controlled. This change significantly improved the user experience by providing faster updates and rendering of plots, as well as more responsive field customisation.

The migration to Angular and Flask also introduced the ability to update plots individually, which further enhanced the user experience. However, it's important to note that updating the threshold, which requires re-rendering several plots, still requires rerunning those specific plots.

Another issue was the rigidness of Streamlit. It prevents developers from creating even basic elements such as pop-up windows. Such limitations prevent developers from creating intuitive website layouts. Meanwhile, Angular exists purposefully to create websites, coupled with its long life its flexibility is incomparable to Streamlit.

The constant reruns that were occurring in Streamlit also made working with inputs in the code a chore. Each time an input was updated the code was rerun and the input had to be saved in the session state. Additionally, each input had to have a callback function that detected the change in the input field, increasing the amount of code and its complexity.

While Angular and Flask resolved the performance issues in EasyFlow, they brought their own challenges, particularly with Angular being more complex compared to Streamlit. One of the main benefits of Streamlit is its simplicity and ease of use, allowing even inexperienced developers to rapidly prototype websites. In contrast, Angular requires developers to create their own HTML, CSS, and TypeScript code. Additionally, there are numerous libraries to learn and study. This makes development with Angular more complex and time-consuming compared to Streamlit. On the other hand, creating a Flask application was relatively straightforward, which could be appreciated by developers of all levels of experience.

In conclusion, the decision to recreate EasyFlow using Angular and Flask addressed the performance issues and allowed for more flexibility in terms of complex website elements. However, it also introduced increased development complexity and a steeper learning curve compared to the simplicity and ease of use offered by Streamlit.

If the researchers prioritise simplicity and a quick development process, Streamlit may be the more optimal choice. Streamlit provides a user-friendly interface and requires minimal setup, making it suitable for creating a simple web application for data visualization without significant programming experience. On the other hand, if the researchers have the time and resources to invest in a more polished and versatile solution, using more established frameworks like Angular and Flask can offer long-term benefits.

## 6.2 Validation

During most of the first half of the semester, there were no meetings with the client after the initial planning phase. This was because the development process was slow and no significant progress was made until the discovery of Figma in the 6th week. However, after adopting Figma, the work progressed more smoothly, and periodic meetings with the client were initiated.

During these meetings, the project details were discussed, and certain features, such as adding groups and data comparison, were removed from the scope to prioritize the development of the core functionality. The decision was made to prioritize functionality over visual aesthetics. The author maintained regular communication with the clients outside of the meetings to ensure informed decision-making regarding the application's functionality.

In the final week of the semester, a final meeting was held with the clients. During this meeting, the author presented the new version of EasyFlow and explained its functionality. The clients provided feedback on added features and proposed new potential features that could be implemented, for example selecting a single tube (refer to Figure 4) to visualise. Due to the time limit these proposals would not likely be implemented, however, they could be implemented by other people.

Overall, the collaborative approach throughout the semester played an important role in ensuring that the final product of EasyFlow met their requirements and expectations. The feedback was positive, some of the frustrations experienced them, e.g. constant reloads in the Streamlit version, were resolved and new features improved the overall user experience.

## 6.3 Possible future

EasyFlow still has a number of possible features that could help it perform its purpose better.

Adding a group feature to EasyFlow could greatly enhance its collaborative capabilities. This feature would allow users to create and join groups, where they can share experiment data and visualisation parameters with other group members. This could be particularly useful for team projects or classes, as it would enable team leaders and teachers to easily share data and parameters with their team or students. Group invites could be sent manually from the management page or by sharing a group link, making it simple to collaborate and share information within a group.

Another potential enhancement could be the ability to compare different experiment analysis data. This feature would allow researchers to compare plots side by side or overlay them on a single plot for easy visual comparison. This could aid in the analysis of experiment results and help researchers identify patterns or trends in the data more effectively.

Implementing a chat or comment function within EasyFlow could also be beneficial. This

would allow users to leave comments for themselves or others, facilitating communication and collaboration around experiment data. Users could leave comments to discuss findings, share insights, or ask questions, helping to streamline the process of working with data and encouraging collaboration among team members.

During the final presentation of the project, the clients proposed another feature. In their word providing users with the ability to choose which part of the data (specific tube or control data) they want to visualise may prove to be useful for researchers.

# 7. Summary

The goal of creating EasyFlow with Angular, Flask, and PostgreSQL was twofold. Firstly, it aimed to provide scientists with a web application for visualising the results of their droplet-based experiment analysis. The application allows users to generate dynamic plots and tables, save and manage their experiment data and visualisation parameters, and share their results with others. The integration with a connected database, unlike existing solutions such as CellProfiler, provides users with a centralised environment to store and share their data and visualisations.

The transition from Streamlit to Angular and Flask brought several benefits and drawbacks. On the positive side, it resolved the issues that arose from using Streamlit, such as slow performance due to frequent code reruns. Angular and Flask improved the application's performance and allowed for the inclusion of more complex website elements. The flexibility and extensibility offered by Angular and Flask provided a solid foundation for building a robust and scalable application.

However, there were also drawbacks to consider. The migration to Angular and Flask introduced increased development time and complexity compared to Streamlit. Angular, in particular, requires developers to create their own HTML, CSS, and TypeScript code, which can be more time-consuming and require a higher level of expertise. The learning curve associated with these technologies may be steeper for developers who are not familiar with them.

Despite the challenges faced during the transition, the decision to move away from Streamlit and adopt Angular and Flask was motivated by the goal of improving the performance and flexibility of the EasyFlow application. The author believed that the benefits of enhanced performance, the ability to create more complex website elements, and the integration with a connected database outweighed the drawbacks of increased development complexity and time.

It is important to note that the conclusion reached is based on the author's perspective and may not necessarily apply to everyone. Streamlit can be an optimal choice, especially for individuals without programming experience or for projects with specific requirements that align well with Streamlit's simplicity and ease of use.

Overall, the transition to Angular, Flask, and PostgreSQL improved the EasyFlow application's performance and provided a comprehensive environment for scientists to visualize, store, and share their experiment data.

# References


[1] Somayeh Sohrabi, Nour kassir, and Mostafa Keshavarz Moraveji. "Droplet microfluidics: fundamentals and its advanced applications". In: *RSC Adv.* 10 (46 2020). [Accessed: 21-05-2023], pp. 27560–27574. DOI: 10.1039/D0RA04566G. URL: http://dx.doi.org/10.1039/D0RA04566G.

[2] Immanuel Sanka et al. "User-friendly analysis of droplet array images". In: *bioRxiv* (2023). DOI: 10.1101/2021.12.21.473684.

[3] Immanuel Sanka et al. *EasyFlow: User-friendly Workflow for Image-based Droplet Analysis with Multipurpose Modules*. [Accessed: 21-05-2023]. Dec. 2021. DOI: 10.1101/2021.12.21.473684.

[4] *What is Flask Python*. [Accessed: 21-05-2023]. URL: https://pythonbasics.org/what-is-flask-python/.

[5] *The Python SQL Toolkit and Object Relational Mapper*. [Accessed: 21-05-2023]. URL: https://www.sqlalchemy.org/.

[6] *What Is PostgreSQL?* [Accessed: 21-05-2023]. URL: https://www.postgresql.org/docs/15/intro-whatis.html.

[7] *Welcome to NGINX Wiki!* [Accessed: 21-05-2023]. URL: https://www.nginx.com/resources/wiki/.

[8] *GitLab Runner*. [Accessed: 21-05-2023]. URL: https://docs.gitlab.com/runner/.

[9] Benoit Chesneau. *Gunicorn - WSGI server*. [Accessed: 21-05-2023]. 2019. URL: https://docs.gunicorn.org/en/stable/.

[10] *Figma Official Webpage*. [Accessed: 21-05-2023]. URL: https://www.figma.com/design/.

[11] *Make Your Vision a Reality: Enterprise Architect*. [Accessed: 21-05-2023]. URL: https://pythonbasics.org/what-is-flask-python/.

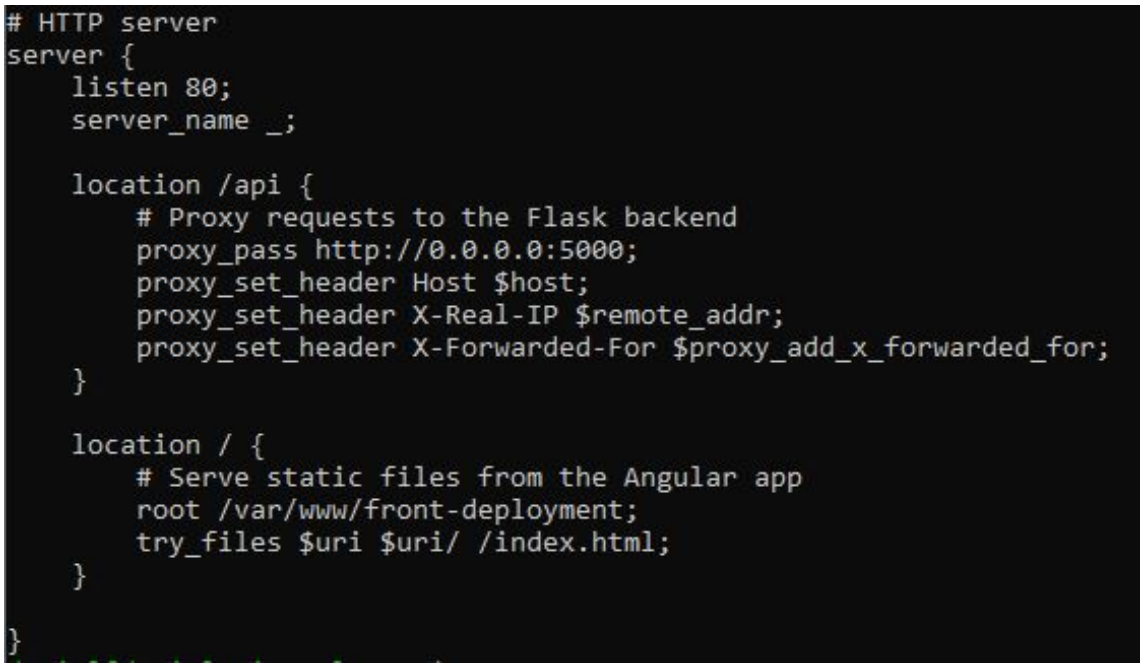
# Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Daniel Berežnoi

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Web application for droplet-based experiment analysis data visualisation.", supervised by Evelin Halling
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

22.05.2023

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.



```
# HTTP server
server {
    listen 80;
    server_name _;

    location /api {
        # Proxy requests to the Flask backend
        proxy_pass http://0.0.0.0:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location / {
        # Serve static files from the Angular app
        root /var/www/front-deployment;
        try_files $uri $uri/ /index.html;
    }

}
```

Figure 9. Nginx server config file for EasyFlow.