Design Document

Conor Kostick (17386616), Vincent Achukwu (17393546)

Link to Demo Video

https://youtu.be/HkChi44GyEI

Introduction

In this design document, we will discuss the general program description, any limitations to our implementation, how we divided the work between us, and briefly explain each part of the program code.

We decided to implement the funding account management application via REST APIs given that we are more familiar with it. We did this using Python's Flask framework. To run our project, you will need to open three terminals, navigate to the directory containing our 3 Flask applications, and run each of the API's in each terminal simultaneously. This can be done with:

```
py dcu_api.py
py funding_api.py
py researcher_api.py
```

Then you need to open a web browser and go to **localhost:5000** to access the researcher api (which will contact the funding api automatically). After you have made a researcher account and had a project approved, you can go to **localhost:5002/login** which acts as the UI for the DCU API. From here you can login with the same name that you used for the project proposal section and access your approved projects. When you see your list of approved projects, you can add/remove researchers to/from any project and also withdraw funds from any of your projects.

General Program Description

For this assignment, we handled the communications between the 3 different entities through a REST API that we made through Flask, a web framework for Python. We have created 3 different programs: **researcher_api**, **funding_api** and **dcu_api**.

The researcher API acts as the initial interface to the researcher. The researcher API takes in the name of the researcher and then asks the researcher to submit a proposal form to the funding agency. The researcher API then informs the researcher if their proposal was

accepted or not. Upon submitting the proposal, the user will be provided with a URL button to login and view their existing approved projects. Essentially, regardless of whether their proposal was accepted or not, a URL button will be displayed (along with the message as to whether or not the proposal was approved) after submitting the proposal which allows the user to login to view their approved projects.

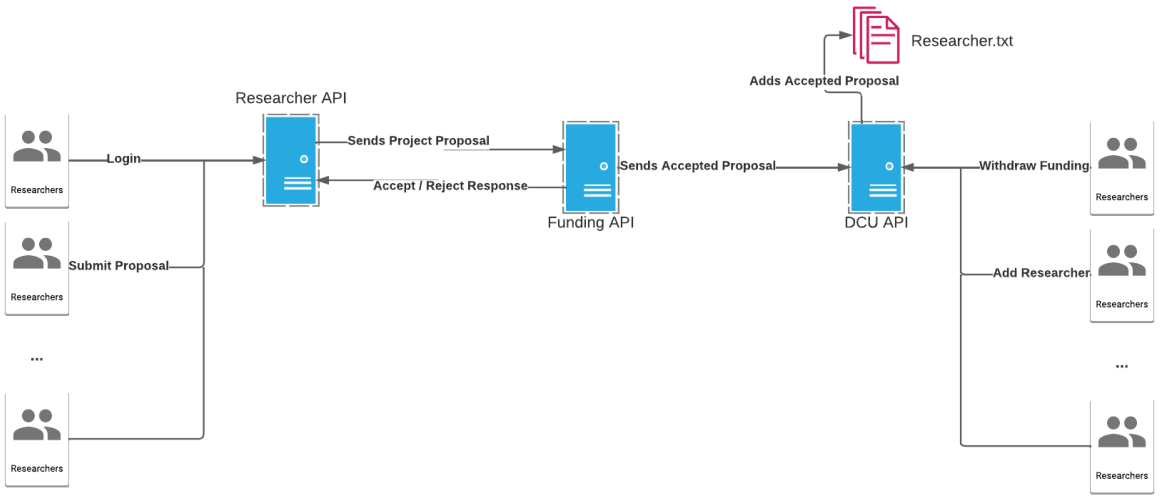
The funding API has no interface. It waits and receives project proposals from the researcher API and returns a message to say if the proposal has been accepted or rejected. A project is approved if the requested funds from the researcher are within the 200k-500k range and the total budget of 2 million (specified in total_budget.txt) has not been exceeded. If the proposal is accepted, it sends on the details of the proposal to the DCU API. The funding API decreases the total budget when the requested funds from the researcher is approved for each project.

The DCU API receives only the approved project proposals from the funding agency. When the DCU API receives this, it makes an account for the researcher if one does not exist and then adds the project details to the researcher's file. The researcher can then log into the DCU API and see their approved projects and withdraw money from the project or add / remove other researchers to the project. When a new member is assigned to a project, the new member will have an account created for them which grants them access to the project page displaying the projects they're assigned to, and the new member will also have their own designated text file describing which projects they've been added to as well as any projects/proposals they created themselves. A member can also be removed from a project after which the text file associated with that user is deleted. There is a flaw with this however. If a project creator deletes a member from a project, but that member also created a project of their own, then that member's account will be deleted entirely (including the project that member created themselves separately).

If the user ever stumbles upon the page which says "Invalid request", this means that the user either removed a non-existing user from a project, or they tried signing in to their account which does not exist. This is how we attempted to implement some form of failure handling in the application.

We found that the Flask API handles most of the concurrency in this assignment. We realised that being able to use a database for this assignment would have significantly simplified the challenge of creating a distributed system since it would have handled all of the synchronisation and concurrency between files when it came to creating/deleting researcher accounts. Due to time constraints, we couldn't quite figure out a way to handle synchronisation within the text files when it came to, for example, withdrawing funds from a project that multiple members were part of. E.g if Bob creates a project and adds Mark to that project, they both see the same project in their accounts and their funds should synchronise if one of them wants to withdraw.

System Architecture



Personal Reflections

Conor's Personal Reflections:

1: What did you learn?

I learned about how to write code based on the REST principles, and gained much more experience on how data can be sent and received in a distributed system

2: How does it relate to the course material?

Both REST APIs / the REST principles and the idea of having a distributed system are covered in the course material

3: What work did you contribute?

Most of my contributions in this project were making the researcher and funding API's and where I could, I did my best to help with making the DCU API as well.

4: What would you do differently next time?

I would take a little bit longer to think about the file structure especially with regards to the researcher accounts and how they can best be formatted as I think there are probably ways to improve upon what we did for that part of the project.

Vincent’s Personal Reflections:

1: What did you learn?

I learned how REST APIs can interact with each other via files rather than using a database as well as how they remain synchronised when multiple distributed systems communicate with each other.

2: How does it relate to the course material?

As well as covering concurrency, this course covers distributed systems, and I found that this assignment also covers both distributed systems and concurrency quite well.

3: What work did you contribute?

While implementing the mechanism for adding and removing members to an approved project, I also tried focusing on the frontend aspect. However, it was more important to get the main elements of this assignment working together rather than having an elegantly designed frontend.

4: What would you do differently next time?

If we were to do this project again, I think if we had an earlier start we may have had more time to fix some of the issues we came across in the end to make the project more complete. Also, instead of using REST APIs, we could have implemented this project using RabbitMQ or RMI.