RCP#0032 Intake 10 Student Internship Summary reports

Table of Contents

Link to Intake 9 Summary Report

RCP#0016 Intake 9 Student Internship Summary reports.pdf that can be used as an example.

AIVE Student Project

Interns: Si Yang (Sean) Chen, Chun-Tung (Chloe) Tsai, Jiawen Deng

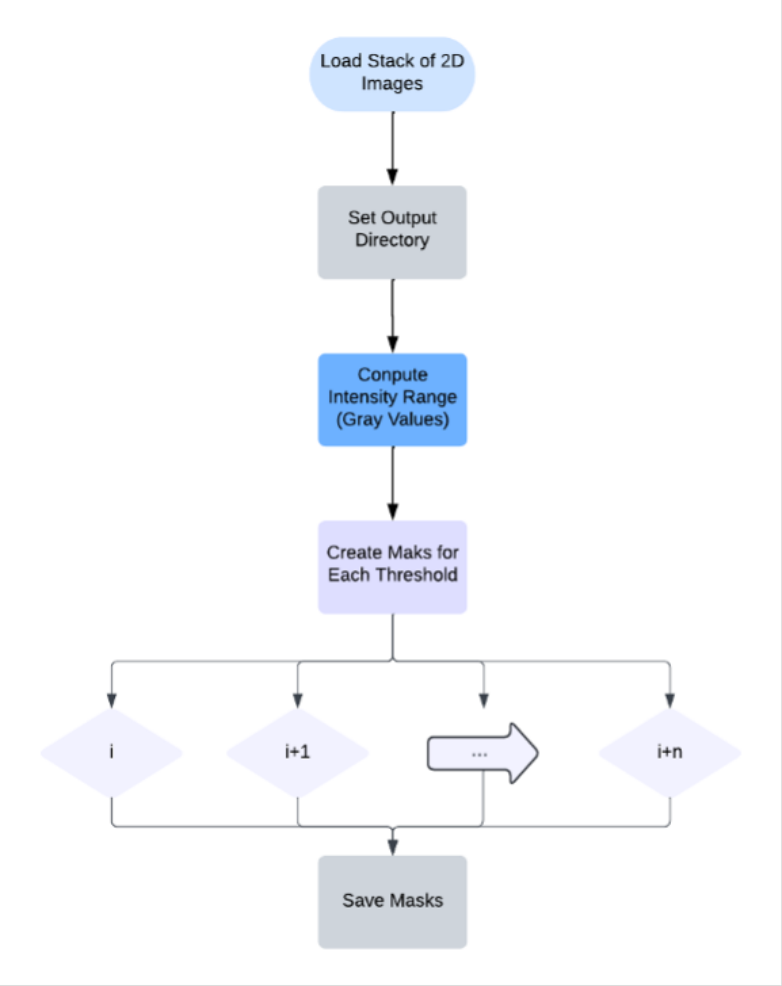
High-Level Domain work

During the first 4-5 weeks, we learned about and tried to understand the AIVE workflow for converting 2D cell image stacks into 3D models and familiarised ourselves with software tools used in the workflow such as ImageJ, MIB and WEKA. Our in-depth understanding was presented in the whiteboard presentation, which detailed key stages in AIVE's workflow. The team also developed high-level flowcharts showing the entire process and the interconnected stages.

Subsequently, we decided to work on the organelle segmentation stage. We converted two ImageJ macros (macros 1 and 1b) from ImageJ Macro Language into Python code for better reproducibility. Building on the public wiki created by previous interns, we added technical documentation and additional notes for both our and previous features, including detailed explanations and flowcharts, to help future interns and people without prior knowledge understand AIVE more easily.

Architecture / Algorithm work

Macro 1:


Load Stack of 2D Images
Set Output Directory
Conpute Intensity Range (Gray Values)
Create Maks for Each Threshold
i
i+1
...
i+n
Save Masks

Macro 1b:

Technical work

Macro 1:
Image stack is loaded into program and all unique pixel values in entire stack are computed. These pixel values are used as thresholds, which are applied to each image slice in stack to create stacks of binary masks. The mask stack is saved and this is repeated for each threshold.

Macro 1b:
The process starts by selecting the input image file (*HeLa.tiff*) and specifying the output directory. A Gaussian blur is then applied with an XYZ radius of X = 3, Y = 3, and Z = 1, chosen to account for the anisotropic voxel dimensions (3 nm in X and Y, 10 nm in Z). These values ensure consistent blurring across the real-world voxel sizes. Users may need to adjust blur parameters or consider kernel size for different datasets. The macro then processes the entire image stack, saving the results in new stacks.

Key Links

- Whiteboard Presentation: Whiteboard presentation flowchart
- Final Presentation: Final Presentation slides
- Public GitHub: https://github.com/MitochondRuna/AIVE-Intro/wiki
- Python Macros Link: https://github.com/MitochondRuna/AIVE-Intro/ImageJ macros
- Personal Technical Notes: Technical Notes

Clinical Dashboards Student Project

Interns: Kathleen Wongso, Jane Xu, Lucas Valente, Yixin Jiang

High-Level Domain work

REDCap / Simulacrum Aspect:
One of our goals for this aspect of the project was to make use of a larger and more complex dataset (i.e. Simulacrum) than the previous intake used. We aimed to upload as much of the Simulacrum V2 data as we could to REDCap, which shaped how we cleaned our dataset. We also wanted to see if we would break REDCap with the new, larger dataset.

Security Aspect:
x

Architecture / Algorithm work

The structure of this project is demonstrated by the following flow chart:

REDCap / Simulacrum Aspect:

1. Cleaned Simulacrum V2 data
    a. Fixed column names (lowercased all field variable names and added underscores due to REDCap guidelines for field names)
    b. Identified invalid records
        - Assumed 'gender_patient' was based on sex assigned at birth -> records of female patients with prostate cancer & male patients with gynaecological cancer
        - Records where treatment (radiotherapy and chemotherapy /hormonal therapy) occurs after death of patient
        - Patients that died before treatment / diagnosis
        - Treatments that occurred before tumour diagnosis
    c. Removed invalid records across the different tables

2. Defined a data dictionary for REDCap
    a. Defined the instruments: Av Patient, Av Tumour, Av Gene, Sact Regimen Outcome, Sact Cycle, Sact Drug, Rtds (each is a separate table)
    b. Defined repeating instruments (number of treatments and tumours varied across the patients)
    c. Defined valid field value types for certain field variables (e.g. 'gleason_combined' in av_tumour defined to only take type 'integer')

3. Uploaded onto REDCap through the API
    a. Obtained REDCap API token
    b. Obtained importing function from previous intake's git
    c. Uploaded tables via the API

Security Aspect:

Technical work

REDCap / Simulacrum Aspect:

Python was the main programming language that we used. We conducted data preprocessing on the Simulacrum dataset by removing records that are not accurately reflected in the real world (e.g. female patients with prostate cancer, treatments that occurred after death). We also ensured consistent schema for the dataset in order to adhere with the data dictionary defined in REDCap. We had to define repeating instruments in REDCap due to the varying nature of number of treatments or tumours per patient. Finally,

we uploaded (to the best of our abilities) the cleaned datasets onto REDCap using the API. However, due to complications caused by uploading via the API we were only able to upload the av tables and some of the sact_outcome_regimen merged table that we created after cleaning the data. Hence, we broke REDCap and it presents the question: “is REDCap the best way to store this information?”. Also, Simulacrum does NOT include location-specific data (the closest is the type of place a patient died, e.g. a hospital or hospice), so a new visualisation dashboard might need to be made if that’s the aim of a future intake.

Security Aspect:
We created a Django application to act as an authentication wrapper around an Rshiny app. Thus access is controlled by Django’s authentication layer and users log in via SSO, which we paired Django with mozilla-django-oidc to accomplish. Currently our SSO server is auth0 but the aim is to move to AAF. We have set up the application such that all that would be needed to move to AAF would be to set the environment variables for server address & security key once AAF provides those, as well as doign any configurations that AAF require on their end. The authorisation is achieved by Django creating user sessions with the ‘users’ from SSO stored in a PostgresSQL database.
Testing was done locally but the application is packaged in docker and can easily be deployed to a correctly configured nectar server that opens the relevant ports. Currently the embedded application is

Key Links

- Our final presentation
- Our Github repository
- Our technical diary

BioNix Student Project

Interns: Cat Tuong Anh Nguyen, Di Wu, Liam Mclnerney

High-Level Domain work

During the first few weeks of the project, we learned about the Nix ecosystem and the required bioinformatics knowledge for the software that we are going to wrap into BioNix. In particular, terms like metagenomics or taxonomic classification for Kraken2, or tandem repeats for TRF. All our knowledges are then presented in a meeting on a whiteboard for our supervisors to correct any false interpretation.

The research is then compiled by us to be added into the existing BioNix wiki by WEHI, including any modifications so that the page is more coherent and user-friendly. We added a project introduction page, added some more questions to the FAQ, and had a scan through the existing information to make sure they are non-repetitive, correct, coherent, uniform and change them if necessary

Architecture / Algorithm work

We had help from our instructor to visualize what an expression to wrap our software would look like, combined with our learning from both the existing package-base, and our personal research. We understood the basic structure of the expression, and how Kraken2 would require no configuration, because all the configs are already in the GitHub repository, or how TRF require thorough management of its dependencies.

In addition, we understood the contribution steps for nixpkgs and ensure that our expressions abide by the rules.

Technical work

- Finalized the expression of wrapping TRF to BioNix.
- Made prototypes of expressions to wrap Kraken2.
- Wrote and tested the prototype expressions for Kraken2.

Key Links

- Fork repo of the wiki
- Trello to manage tasks
- Final Presentation

Conference Organiser Student Project

Interns: Zheyuan Wu, Antonio Wang, Wingyee He, Natasha Ngo, Jiaxi Zheng

High-Level Domain work

The Conference Organiser project aims to streamline the annual RSEAA conference, which fosters inclusivity and recognition for software engineers worldwide. In the second intake, the team focused on two key areas. First, a dynamic HTML page was developed to update key dates on the main website, RSEAA.org.au, enhancing user experience. Second, the team focused on creating a platform that drew inspiration from a popular live polling application called CrowdPurr. Within this polling project, the team was divided among two sub-projects: One focused on extending Claper, an existing open-source live polling tool, while the other group built a custom live polling application from the ground up using React.

Architecture / Algorithm work

Claper architecture:

React Live Polling:
High-level Architecture:
Database Architecture:

Technical work

Live polling application:

Frontend: React Library (HTML, Javascript, CSS)
Backend (Database and functionality):
Testing and integration: NectarVM (used for docking)

Claper:

Frontend: HTML, TailwindCSS
Backend: Elixir, Phoenix
Testing and integration: NectarVM

Webpage:

Simple HTML, CSS and JavaScript

Key Links

- Live polling react application: https://github.com/ngonatasha/WEHI-Conference-Organiser
- RSEAA webpage: https://github.com/ngonatasha/RSEAA.github.io
- Project Claper: https://github.com/Jazzzheng9/Claper.git
- Original Claper: https://github.com/ClaperCo/Claper
- Final presentation: Conference Organiser Final Presentation.pdf
- Whiteboard presentation: Conference Organiser Whiteboard presentation.pdf
- Technical diary: Technical Diary

Data Commons / REDMANE Data Registry Student Project

Interns: Haoxuan Lu, Jeremy Lau, Natasha Mulay

High-Level Domain work

The Data Registry is a web-application that connects Raw and Processed Data to the Data Portals such as cBioPortal, Omero, Aquila etc., to form the Data Commons/REDMANE. It has four main components, the front end, the backend, the database and the virtual machine (Nectar). As the Data Registry subgroup, we addressed the frontend, backend and VM as the main areas of work. The frontend involved the creation of different components such as the Projects components which linked to the Patients and Datasets components (Figure 1).

Architecture / Algorithm work

Figure 1. ER diagram of different components within the frontend of the Data Registry and how they link together.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Technical work

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Key Links

Frontend with React (Current): https://github.com/jeremlll/REDMANE_react.js

Frontend with React (Base): https://github.com/HxLu03/DataCommons

FAST API (Working one): GitHub - jeremlll/REDMANE_fastapi

Data Commons / REDMANE Data Ingestion Student Project

Interns: Xinyu Wang, Boyu Chen, Chelsea Kwan, Bucheng Liu

High-Level Domain work

The aim of the Data Commons Project is to:
Allow for ease of access to various datasets for researchers and stakeholders
Ability to locate raw/processed data stored on local servers through links on the data registry
Access summarised data and visualisations on data portals
Form connections between data registry, local servers and external data portals (cBioPortal, Aquila etc.)
For the Data Ingestion, the primary goal is to Ensure that datasets are validated and ingested into cBioPortal. Key responsibilities would include validating datasets using the validator tool, ingesting validated datasets into the cBioPortal system, and setting up and manage the infrastructure for the ingestion process.

Architecture / Algorithm work

Summary Diagram:

Architecture:
Platform Data Workflow

Technical work

Setting up the virtual environment:
Ubuntu VM Setup: Create an Ubuntu virtual machine (VM) on the server to run the validator and manage the ingestion process.
Portainer for Docker Management: Install and use Portainer to manage Docker containers for deploying the ingestion environment.

Key Links

Data Ingestion Documentation

Data validator repo: https://github.com/Morning-NFX/DataCommons-cBioPortal

Example dataset repo: https://github.com/Morning-NFX/DataCommons-dataset

https://github.com/cBioPortal/cbioportal
https://docs.cbioportal.org/using-the-dataset-validator/
https://docs.cbioportal.org/file-formats/

Data Commons / REDMANE Authentication Student Project

Interns: Trung Ngo, Lucas Speak

High-Level Domain work

The primary focus of the project was to establish a modern and seamless authentication system to protect sensitive data within a medical research data registry.

The data registry is a web application which was concurrently developed by the Data Commons Data registry team. Authentication ensures that only authorized users can access the system, preventing unauthorized access and data misuse. This protection is critical because it safeguards not only WEHI's research interests but also the privacy rights of patients whose medical data is being managed.

The choice of authentication protocols had to align with industry standards and requirements to ensure compliance with security and privacy regulations while supporting Single Sign-On (SSO) for ease of access.

High level work consisted of:

- Researching what resources were available and suitable for this project.

- Understanding how different authentication protocols worked, and the advantages/disadvantages of each.
- Identifying the requirements of the authentication system.

This work resulted in making the following decisions:

- OpenID Connect (OIDC) was selected as the primary authentication protocol due to its suitability for research organizations, as recommended by the Australian Access Federation (AAF). OIDC supports both SSO and social login, offers flexibility in integrating different identity providers, and is simpler to implement compared to alternatives like SAML.
- Keycloak, an open-source identity and access management system, was chosen to run on the authentication server. It offers ease of setup, sufficient security for non-critical use cases, and eliminates the licensing costs associated with platforms like Auth0. Keycloak also allows integration with AAF as an identity provider, leveraging OIDC to provide seamless access for Australian researchers.

Architecture / Algorithm work

The following diagram depicts the typical authentication sequence for a user signing into the data registry using their Okta credentials:

Technical work

Luke:

- Researched OIDC and authentication tools.
- Discovered, installed and configured Keycloak
- Implemented OIDC authentication flow with demo Python script and keycloak.

Trung:

- Deployed Keycloak with test web application to implement the OIDC authentication protocol.
- Modified Keycloak realm to accept Auth0 as an Identity Provider.
- Created and performed authentication demo in final presentation.

Key Links

Authentication set-up docs:

https://wehieduau.sharepoint.com/:f:/r/sites/StudentInternGroupatWEHI/Shared%20Documents/Data%20Commons/2024%20Semester%202/Authentication?csf=1&web=1&e=54AxBl

Final presentation (slides):

https://wehieduau.sharepoint.com/:p:/s/StudentInternGroupatWEHI/EYQh5qAhApVJjh51r84NQgcBKihFHMF2nj8x3ctK39fQjw?wdOrigin=TEAMS-MAGLEV.null_ns.rwc&wdExp=TEAMS-TREATMENT&wdhostclicktime=1728513382398&web=1

^OUTDATED

OIDC demo python script (rough):

https://wehieduau.sharepoint.com/:f:/r/sites/StudentInternGroupatWEHI/Shared%20Documents/Data%20Commons/2024%20Semester%202/Authentication/OIDC%20Python%20demo?csf=1&web=1&e=dlAegm

Genomics Invoicing Student Project

Interns: Changyuan Ni, Jiayi, Li, Iffat Azeez, Ramon Felipe

High-Level Domain work

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Architecture / Algorithm work

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Technical work

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Key Links

-

Haemosphere Student Project

Interns: Michele Meliala, Lady Feren Pangjaya, Thanh Pham, Thi Hong Minh Dao, Zachary Iskandar

High-Level Domain work

Our group's work was the continuation of the ongoing migration of Haemosphere from Python 2 to Python 3, due to Python 2 no longer being supported. We mainly focused on improving overall speed on the site, and improving the scalability of the site to accommodate multiple simultaneous users without crashing. After extensive testing, we determined that a caching-related approach was the best solution for performance enhancement. In addition, we made minor improvements to the existing tutorials and documentation, so that hopefully future intakes will have a smoother and less confusing onboarding experience.

We implore future intakes to explore this caching solution in greater depth to assess its feasibility, and to debug some datasets such as Schultze and Immgen to make them compatible with the new Python 3 framework, as well as any other necessary improvements to the site to further enhance the user experience. In the longer term, we also recommend Haemosphere to be migrated to a supported version of Python, as support for the destination version (Python 3.6) ceased in December 2021, almost three years ago at the time of writing.

Architecture / Algorithm work

We trialled several different solutions and measured the impact each solution had on improving speed and performance. These included increasing the RAM of the virtual machine, pre-loading data onto RAM cache and utilising disk cache. Through performance testing, we found that a large memory virtual machine with caching was the most efficient in terms of speed and reliability. We also explored changing file formats using .h5 and .parquet, but this method was overcomplicated and only yielded minimal improvements compared to other methods.

Technical work

Fixed minor bugs that prevented us from running the server (typos, missing dependencies during the onboarding process, tweaks to config files)
Took the opportunity to explore the option of a higher memory VM after the hosting arrangement with the University for the publicly accessible version of Haemosphere was changed
Conducted performance testing on various differential expression datasets to assess performance and detect errors for future intakes to fix, as well as to compare different server configurations
Implemented RAM and disk caching, which significantly improved performance

Key Links

- Haemosphere website
- Public Github repository
- Sharepoint
- YouTube link for presentation (add link when uploaded)

Student Organizer Student Project

Interns: Jiaman(Mandy) Xu, Xiaoqing(Betty) Hu, Yiyang Chen, Yovela Budiman

High-Level Domain work

The team analyzed the web app's code, setting short, mid, and long-term goals. **Short-term** tasks, like improving the navigation bar, creating filtered student lists, and updating the public data schema and database, were completed. **Mid-term** goals, such as linking project and intake pages and refining the UI, are underway. **Long-term** goals, including database optimization and platform migration, are set for future interns. Documentation in the project wiki and SQL scripts ensure smooth handovers for upcoming cohorts.

Architecture / Algorithm work

The project uses a **modular architecture**, making it easy to test and integrate new features. A responsive **navigation bar** adapts to different screen sizes, with dropdowns for better organization. Key features include dynamic student lists, email copy functionality, and direct links between project, intake pages, and filtered student lists, streamlining navigation.

Technical work

- Frontend (UI/UX):

    - Designed a responsive navigation bar using HTML and CSS for seamless navigation across devices, incorporating dropdown menus for different sections.
    - Developed subpages for student lists (current, future, past, all students) and added buttons linking intake and project pages with pre-filtered data.
- Backend (Database & Functionality):
    - Generated fake test data and updated database schemas accordingly.
    - Created upgrade_database.sql to document and maintain database changes for future students.
- Testing & Integration:
    - Tested the system using Nectar VM to simulate real-world user behavior and ensure smooth functionality.

Key Links

1. Link to the wiki:
   https://github.com/WEHI-ResearchComputing/student-intern-organiser/wiki
2. Link to the Technical notes:
   Technical Notes 2024 S2
3. Link to the Final Presentation slides in SharePoint:

Genomics Invoicing Project:

Interns: Changyuan (David) Ni, Jiayi (Joyce) Li, Ramon Felipe, Iffat Abdul Azeez

High-Level Domain work

The team first learns in detail the client's original workflow, in which the client wishes to have the tedious manual process of producing invoices automated. The team drafted and refined the wireframe for the concept of automation app in generation of genomic invoices. The team considered architectural limitations of the deployment environment and develops software architecture of the app based on the limitations.

Architecture / Algorithm work

Here is the link to the architecture diagram the team developed:

https://lucid.app/lucidchart/295c39f2-77b2-4cd8-9a64-bced69e188b7/edit?viewport_loc=379%2C407%2C4469%2C2110%2C0_0&invitationId=inv_91729030-3e45-4bcf-aac6-ce3bb4682304

Technical work

The team configured and tested the local development environment, server-side deployment environment and the protocol for connecting to the server at WEHI. The team develops an upload interface, with testing files and correct output ready.

Key Links

1. Link to wiki, **all relevant links are found in the wiki:** Genomics-invoicing wiki · Ayacolyte/genomics-invoicing Wiki · GitHub