Introduction

The group project is worth 40% of the final grade for both CITS3403 and CITS5505 students and is due at 11:59pm on Friday, May 16th**.** The groups will consist of 4 students and a group must be comprised exclusively of either CITS3403 or CITS5505 students. The project description will be released by the end of the Week 4.

High-level requirements

For this project you are required to build a data analytics application, which allows users to upload private data, view automated analysis of their data and then selectively share the results with other users. You may interpret the concept of "data" and "analysis" *very flexibly*. For example, the application could be:

- An exercise tracking application, where users track their exercise habits, can view stats about their habits, and share information about their achievements with their friends on the system.
- A statistical analysis tool, allowing users to upload certain types of datasets (e.g. timeseries, tabular data), and then run standard statistical analysis algorithms (e.g. regression, clustering, outlier detection) on them, and share their findings and datasets with their colleagues.
- A course-selector tool, allowing users to upload details (e.g. times, duration, credit hours) of courses they are interested in taking, and then run scheduling algorithms to generate a plausible selection of units, and allowing users to share suggested schedules with their classmates.
- A tournament management system, where users can input results of sports/board games tournaments, see player stats, and then share results with other users.
- An infectious disease monitoring system where users can upload datasets of infections, and the application will plot them on a map and you can share areas with high outbreaks to other users.
- A news sentiment analysis tool, allowing users to upload or input news content, and then analyse the text using NLP algorithms (e.g., sentiment scoring) to determine overall sentiment, and share the analysed data with their colleagues.

Any method of data entry is fine. For example data could be manually entered by the user, or uploaded in batches in a suitable file format, or automatically sourced from other devices and services (e.g. fitness trackers, smart watches, publicly available feeds and datasets).

Please think carefully about the design of the application. It should be:

- *Engaging*, so that it looks good and focuses the user on important elements of the application.
- *Effective*, so it produces value for the user, by providing information, entertainment or community.
- *Intuitive*, so that it is easy for a user to use.

Any concept obeying the guidelines above is fine as long as the website offers the following views:

1. An "introductory" view, describing the context and purpose of the application, and allowing the user to create an account or log in.
2. An "upload data" view, allowing the user to add new data.
3. A "visualise data" view, allowing user's to view analysis of either their data or other people's data which has been shared with them.
4. A "share data" view, allowing the users to selectively share their data with other specific users.

Do not yet start coding!! The technical specifications and marking rubric for the project will be released directly after the lecture in Week 6. This will include what technologies are allowable and how you will be assessed for teamwork. As we will directly assess your development methodology and teamwork, **if you start coding now you will end up losing marks**!

What you need to do now: Until after the lecture in Week 6, the only thing you need to do is discuss and decide within your group about the purpose of your application. If you are unsure of whether your project idea is appropriate, please talk to a facilitator in the lab sessions.

Technical specification

The list of core allowable technologies and libraries are the following:

- HTML, CSS, JavaScript
- One of Bootstrap/Tailwind/SemanticUI/Foundation (no others allowed).
- JQuery
- Flask (with plugins described in lectures)
- AJAX/Websockets
- SQLite interfaced to via the SQLAlchemy package

You **may not** use any other core technologies, this includes frameworks (e.g. React/Angular), database systems (e.g. MySQL), or advanced CSS frameworks (e.g. directly using SASS yourself). However, you may freely use any JavaScript or Python libraries that implement non-core functionality that is particular to your application, e.g. libraries that provide bindings for ChatGPT or displaying graphs is fine! Font and icon libraries are also fine.

The creation of the web application should be done in a private GitHub repository that includes a README containing:

1. a description of the purpose of the application, explaining its design and use.
2. a table with with each row containing the i) UWA ID ii) name and iii) Github user name of the group members.
3. instructions for how to launch the application.
4. instructions for how to run the tests for the application.

Assessment

For the detailed mark scheme, please click on the "Submission" item below, and then click the "View rubric" link on the right of the page.

Intermediate deliverables

In keeping with Agile development, throughout the project there will be two preliminary deliverables which will take the form of presentations. No code needs to be submitted. Facilitators will act as clients and will give some brief feedback.

1. In the week 28th April-2nd May, at least one person from your group will need to turn up to a lab session for a 5 minute presentation. This should involve describing your idea to the facilitator and presenting designs for your GUI. These should consist of static HTML pages with some minimal CSS styling. No actual functionality required (e.g. button presses required, and complex graphs etc. can have placeholders).

2. In the week 5th May-9th May, at least one person from your group will need to turn up to a lab session for another 5 minute presentation. This should involve demonstrating a product with some minimal dynamic functionality (e.g. persistantly storing data in the database and altering the webpage based on it).

Final deliverable

To submit your group project:

1. Make your repository public on GitHub so that the markers can see it and so that the `Insights` tab is enabled.
2. Create a zip file of your complete Flask application:

- All source code, with comments and attributions for any external libraries, **including your tests directory**
- A `requirements.txt` file, listing all packages used. To build the requirements.txt file for your virtual environment, use the command: pip freeze > requirements.txt while your virtual environment is active.
- The `README.md` file
- You may incude a small database for testing purposes

3. Click on the word "Submission" above and upload the zip file.

The project should be runnable by 1) downloading the code, 2) installing the packages in the requirements.txt file in a new virtual environment, 3) following the instructions in the README.

Do not submit the virtual environment directory, or the .git directory. These are overly large, not required and will result in a penalty if they are submitted. Submit your zipped file by clicking on the title of the item in the LMS and attaching the file. Only one person per group needs to make this submission.

Group project presentation

In the last week of term after you have submitte your project, there will be full project presentations and all members are expected to attend. You will have 12 minutes to demonstrate your final project and the marker will ask 3 minutes of questions afterwards.

5 minutes before your scheduled time, please make sure:

- Your entire team is present.
- You have your app launched on a machine.
- Your demonstration database set up.

You will be ask to demonstrate the features of your app. You do not need to guide the assessor through the source code unless asked. This should take no more than 8 minutes, and it is worthwhile having a rough plan or script. All team members must be involved in the presentation. You will then be asked some questions from the assessor, and may need to show some additional functionality, or source code etc.

The sign-up sheets for the group project slots will open later in the term.

Marking criteria

HTML	Valid HTML code, using a wide range of elements, clearly organised with appropriate use of Joomla templates.
CSS	Valid, maintainable code, using a wide range of custom selectors and classes, web page is reactive to screen size.
JavaScript	Valid, well formatted code, including validation and DOM manipulation/AJAX that uses JavaScript best practices.
Design	Good website navigation flow that is intuitive to the user with a strong visual design. The website's purpose is clear and brings value to the user.
Content	All the features requested in the project brief are implemented appropriately.
Flask Code	Formatted, commented and well organised code that responds to requests by the client by performing non-trivial data manipulation and page generation operations.
Data Models	Well considered database schema, good authentication, and maintainable models. Some evidence of DB migrations.
Testing	Comprehensive test suite, containing 5+ unit tests and 5+ selenium tests. The latter should run with a live version of the server.
Security	Passwords correctly stored as salted hashes in database. Use of CRSF tokens to prevent CRSF attacks on the website's forms. Correct storage of environment variables in configuration files.
Commits (individual)	Regular commits with high-quality messages providing meaningful but not overly detailed description of and reason for the changes.

Issues (individual)	Excellent use of the Github Issues tab, with issues regularly used both to describe both bugs and missing functionality. The former have detailed instructions about how to reproduce.
Pull requests (individual)	Excellent use of the Github Pull Requests tab, with meaningfully named pull requests regularly used to add both individual features and fix bugs.
Teamwork (individual)	Evidence of collaboration between team members on Github, including in-depth discussion on other people's issues, and helpful code reviews on other people's pull requests with the feedback being taken into account before merging.