COMP2100/6442 Group Project | My Educational App

v.01 - Project Description released (24.03.2023)

Assessment weight: This project is worth 30% of your overall mark.

Code due date: Thursday, Week 11 at 11h59 pm (last commit for the code)
Report (group and individual reflection) due date: Friday, Week 11 at 11h59 pm (last commit for the report, Wattle for individual reflection)
Presentation Slides due date: Friday, Week 11 at 11h59 pm (slides must be uploaded to Office365)

Group Presentation: Monday, Week 12 (during lecture time)

No late submission allowed

Table of Contents

Introduction

Key requirements for the App

- Part 1: Basic App
- Part 2: General features of the App
- List of General Features
  - Search-related features
  - UI Design and Testing
  - Greater Data Usage, Handling and Sophistication
  - User Interactivity
  - Privacy
  - Creating Processes
  - Peer to Peer Messaging
  - Firebase Integration
  - How many features should my group implement?
- Suggest a new feature | Voice your feature

Surprise!

Checkpoint

Project Submission

- Code development
- Report (Important!!!)
- All attempted features MUST be documented

Other important items

- Individual Reflection and Peer- and self-evaluation
- Group Project Demonstration
- Best Practices for Using Generative AI Tools for Software Construction

Frequently Asked Questions

- What kind of app should we develop?
- Why and how to simulate a data stream?
- How should I develop the search mechanism?
- What about the data?
- What about the data structures?
- Which design pattern should I use?
- How complicated should my grammar be?
- What should my app look like?

General Information (assessment, submission, presentations, etc)

- Working physically apart, but remotely together!
- What is the number of members per group?
- Due date and late submission policy
- Originality

How will projects be assessed?

- Marking Criteria
- Assessment Rubrics
- Other relevant information

Introduction

The objective of this project is to gain some experience in the process of software construction (the design, specification, documentation, implementation, and testing of substantial software). This project will also give you some practice in designing and implementing a graphical user interface (GUI) application and using several important development tools (particularly Android Studio and Git) with Java. Most importantly, it is an opportunity to reason about and put into practice some of the fundamental concepts covered in this course, such as Data Structures, Tokenizer/Parser, Data Persistence, Design Patterns, Software Testing, etc.

As you complete this project, you should reflect on the overall design along with the software engineering process that you used in bringing this project to completion. Note that this is not an assignment about who could write the best app only, but who could best manifest what they have learned in this course and during the implementation of this app. **The journey is as important as the final app (regular meetings, regular contributions/commits, etc).**

The topic of the project: My Educational App

A mobile educational app is a software designed to help students learn remotely. The structure of learning systems is often complex and involves many of the topics learned in this course.

A few examples of educational platforms are Moodle, a free and open-source learning management system used by several universities in the world including ANU, the Khan Academy that provides personalised learning experiences to students, and the Chinaooc, a course aggregator containing more than 20,000 courses from several Universities in China. Other known platforms are Coursera, Udemy, EdX.

You are encouraged to come up with your own ideas related to this topic and features as a group. (B-creative!)

Although the content for your My Educational App is open, your implementation must meet several basic requirements.

Note that we do not expect a fully operational app - simplifications can be made as long as the core architecture is well-developed. The key assessment aspects include Data Structures, Tokenizer/Parser, Data Persistence, Design Patterns, Software Testing, and Code Quality.

Key requirements for the App

Part 1: Basic App

In the first part of the assignment, you must create an app (with generated APK) that allows users to:
(i) login, (ii) visualise and (iii) search for information within the app.

Marks will be awarded for the following listed requirements. It is advised, although not necessary, that you complete this basic application before proceeding to the additional/general features. Implemented features that do not meet the requirements will net you zero marks. Use the report to explicitly link the features your group implemented to the requirements listed.

- Users must be able to log in (not necessarily sign up). (easy)
  Important: You must include the following two accounts for markers' access to your App:

  Username: comp2100@anu.au Password: comp2100
  Username: comp6442@anu.au Password: comp6442

- There must be data file(s) with at least 2,500 valid data instances. The data files must be used to feed your app, simulating a data stream. For example, every x seconds, a new item is read from a file. An item can be an action (e.g., a new set of lecture notes has been posted by a lecturer; students are added to a course; an assessment submission; a new user signed up; etc). (easy)

  Important: Include the link to the data file(s) in your report for marking purposes.
  If Firebase is used to store the data instances or for any other parts of the project, you must add comp21006442@gmail.com as a Developer to your Firebase project and add the link to your Firebase repository to your report.)

- Users must be able to load data/information (from the data file(s) or Firebase) and visualise it (e.g., a list of courses and/or a list of lecturers and students as users). (medium)

- Users must be able to search for information on your app. (medium)
  The application is dependent on your app theme. For instance, you may want to allow users to search for information such as courses, lecture materials, course participants by certain criteria (e.g. #assignments due: 2023/3--5).
  a. The search functionality must make use of a tokenizer and parser with a formal grammar of your own creation.

The underlying implementation must:

I. contain at least one fully implemented tree data structure taught in this course (e.g., Binary Search Tree, Red-Black tree, AVL tree, B-Tree) for organising, processing, retrieving and storing data. We will also evaluate your choice and use of data structures and your justification (not only trees, but also other data structures such as arrays, lists, maps, etc).
II. Implement at least three design patterns covered in our course.
III. Retrieve data from a local file (JSON, XML or Bespoken) or Firebase.

Can I use pre-made solutions (e.g. databases (including Firebase) or APIs)?

You can and should; software reuse is part of software construction. However, this assignment has some compulsory items (e.g., tree data structure), and we expect you to implement them. So, as long as the pre-made solutions do not impact the core features, there is no problem with using them.

We remind you that we will assess what you have implemented, not what others have implemented. All used third-party code/library/solution must be referenced (read the Originality Section).

Part 2: General features of the App

On top of the basic infrastructure, a software app is enriched by features that better present information and provide useful functionalities relevant to its theme and purposes.

For this assignment, you can adopt features from the provided list and/or propose other features (see next subsection "Voice your Feature"), and present your implementation for each feature in the report.

Note that these features are used for assessment purposes to demonstrate your skills by reference to the core assessment criteria. As opposed to the development of a fully-functioning app, the actual functionalities provided are less important. You do not need to accomplish a lot of features to achieve a good grade, quality over quantity (see "*How will projects be assessed?*" Section).

Greater marks will be attributed to the excellent implementation of a few features as opposed to poor attempts at several features. For example, achieving the entirety of the 'peer-to-peer messaging' category to an excellent level of quality (great code documentation, excellent use of data structures, appropriate incorporation of design patterns, etc) will net you greater marks than poor attempts at several features.

List of General Features

The following is a list of features, each with an [id] (in squared bracket at the start) classified by category and level of difficulty for references. Please note that '[stored in-memory]' refers to storing data in the application's temporary memory (and not storing the data persistently on the device itself) suffices.

Search-related features

1. [Search-Invalid] Search functionality can handle partially valid and invalid search queries. (medium)
2. [Search-Filter] Sort and/or filter a list of items returned from a search, with the help of suitable UI components. For instance, when searching for assignments, include checkboxes for users to select the target course(s); include drop-down field for the selection of sorting methods, etc. (easy)

UI Design and Testing

1. [UI-Layout] UI must have portrait and landscape layout variants as well as support for different screen sizes. Simply using Android studio's automated support for orientation and screen sizes and or creating support without effort to make them look reasonable will net you zero marks. (easy)
2. [UI-Test] UI tests using espresso or similar. Please note that your tests must be of reasonable quality. (hard)
    a. Espresso is not covered in lectures/labs, but is a simple framework to write Android UI tests.

Greater Data Usage, Handling and Sophistication

1. [Data-Formats] Read data instances from multiple local files in at least 2 different formats (JSON, XML or Bespoken). (easy)
2. [Data-Profile] User profile or Course material activity containing a media file (image, animation (e.g. gif), video). (easy)
3. [Data-GPS] Use GPS information (see the demo presented by our tutors. For example, your app may use the latitude/longitude to show information relevant to your app. An example is to display

localised based on the location of the user app. (e.g., English materials if the user is located in an English speaking country)). (easy)

4. [Data-Graphical] Graphical report viewer. Provide users with the ability to see a report of interactions with your app (e.g., summary of assessment results for a course or an individual student, etc), in a graphical manner. (medium)
5. [Data-Deletion] Deletion method of either a Red-Black Tree, AVL tree or B-Tree data structure. The deletion of nodes must serve a purpose within your application. (hard)
    a. Note that this advanced feature will only be considered if the chosen tree is the most suitable data structure (out of the data structures covered in this course) for the App you are developing. Note that the RB-Tree deletion is not covered in lectures (see deletion algorithm in the references of the data structure lecture [readings icon on Wattle]).

Can I use the lab code?

Yes, you can, *but you must reference it and use the report to show us what you've improved in the labs code*. There are several ways to improve the code from our labs. Remember that we will only assess your code, not ours. Only significant improvements can count towards the completion of a feature.

User Interactivity

1. [Interact-Micro] The ability to micro-interact with items/users (e.g. add to todo-list, like/follow a post in the forum, connect to another user, etc.) [stored in-memory]. (easy)
2. [Interact-Follow] The ability to 'follow' a course or any specific items. There must be a section specifically dedicated to 'things' followed (e.g., showing all updates from all courses followed in chronological order). [stored in-memory] (medium)
3. [Interact-Share] The ability to share an item to another user (similar to social media shares with basic information of the item. For instance, for students to share lecture notes (including the title, thumbnail photo, and extract of the description) via private messages). [stored in-memory]. (easy)
4. [Interact-Noti] The ability to send notifications based on different types of interactions (e.g., new announcement, assignment submissions, etc). A notification must be sent only after a predetermined number of interactions are set (>= 2 interactions [e.g., 2 announcements have been made or 2 follow requests have been received). Note: it is not mandatory to use the Android Notification classes. (medium)
5. [Interact-Scheduled] Scheduled actions. At least two different types of actions must be schedulable. For example, a user can schedule an item (e.g., release new lecture materials/the exam page on Tuesdays at 3 pm). (medium)

Privacy

1. [Privacy-Request] Students may send requests which are then accepted or denied by the Lecturers/other users (e.g., a request to join the course or follow another student). (easy)
2. [Privacy-Visibility] A student can only see a course's profile that is set to Public (consider that there are at least two types of profiles: public and private) or after being added to the course as a student (easy).
3. [Privacy-Block] Provide lecturers with the ability to 'block' things. Things (e.g., forum posts containing abusive language, etc) shall not be visible to other students but still visible to the lecturer(s). (medium)
4. [Privacy-Anon] Provide students with the ability to make anonymous posts in a forum. The visibility of author of anonymous posts should be determined by the viewer's role (e.g., visible by lecturers, tutors and the author, but not by other students). (medium)

Creating Processes

1. [Process] Your app may implement a chain of action/steps (at least 3 steps) to follow up on a process designed to be included in your app (e.g., Group Project Submission: each group must have *n* students (step 1), after reaching the target number of members, the process moves on to the next step (step 2) which can be the provision of group information by the students (e.g., group name, topic chosen); afterwhich a presentation date will be set by the lecturer (step 4); and then confirmed by the group (step 5); and prior to the deadline, the presentation powerpoint will be submitted by the group (step 6) and finally, a score will be provided by the lecturer. (hard)
2. [Process-visualise] Process visualisation. Your app may implement a graphical element to visualise the progress of a process (e.g., stages of Group Projects or course completion). (medium)
3. [Process-Interact] Your app may allow users to micro-interact and/or add messages to each step of the process (easy).
4. [Process-Permission] Only users with permission (e.g. all lecturers, or the user who created the process, etc) can view messages in the processes. (easy)

Peer to Peer Messaging

1. [P2P-DM] Provide users with the ability to message each other directly in private. (hard)
2. [P2P-Block] Provide users with the ability to 'block' users, preventing them from direct messaging them. (medium)
3. [P2P-Restriction] Enable users to restrict who can message them by some association (e.g. setting: can message me only if we are currently enrolled in the same course). (hard)
4. [P2P-Template] Template messages or Macros (for peer-to-peer messaging or template posts (e.g. a quick one-tap post)). For example, "Hi %USERNAME%, I am also taking %COURSE_NAME%. Find me at %EMAIL_ADDRESS%, if you please. Cheers, %MY_USERNAME%". The use of tokenizer and parser is mandatory. (hard)

Firebase Integration

1. [FB-Auth] Use Firebase to implement User Authentication/Authorisation. (easy)
2. [FB-Persist] Use Firebase to persist all data used in your app. (medium)
3. [FB-Syn] Using Firebase or another remote database to store user information and having the app updated as the remote database is updated without restarting the application. e.g. User A (a lecturer) posts an announcement, user B on a separate instance of the application sees the announcement appear on their app instance **without restarting** their application. (hard)

> Why and how to implement these features?
>
> Each feature is designed considering one or more concepts covered in the course. Think about what design pattern, data structure, persistence method, … you can use to implement it.

How many features should my group implement?

Each group must implement at least **6 General Features** (in addition to the 4 Basic App features).

For groups that aim at achieving D and HD grades, you could refer to the indicative level of difficulty of features and tackle more features to enrich the app. However, please make sure that you could

manage the number of planned features and review your plans periodically. Note that we expect quality over quantity.

Suggest a new feature | Voice your feature

Is there a feature you would like to implement but is not listed here? No worries, you can post your feature idea on our Wattle "*voice your feature*" forum with the information below. We will assess the proposed features and, if it is approved, will be given a difficulty classification. Please note:

1. Any features which are approved in the forum can be pursued by any group.
2. Any feature that is refused can still be pursued but will NOT be assessed. Although it might count towards the creativity criterion.
3. We will only accept new features until the end of Week 9.

A feature suggestion MUST contain:

- Post subject: feature name
- A description of the feature.
- Details as to what the feature would entail. (e.g., an additional tokenizer, custom B-Tree, etc.)
- Why is this feature relevant to the project and course? (short explanation, link to the course content)
- Suggested difficulty level: (easy, medium, hard, very hard) followed by a short justification.

Please try and keep any features mentioned relevant to the course content and the core software design criteria. Any features that stray too far from either of these will be refused.

Feature Request Example

Subject: Partially valid and invalid search query handling.

Description: The application's search bar will be able to handle both valid and invalid search queries without crashing the application and still provide the user with at least some search results in line with what was valid.
What the feature entails: modifying the tokenizer/parser that the application uses to handle valid and invalid search queries without crashing and still providing a response.
Feature relevance: Tokenization and parsing.
Suggested Difficulty Level: medium.

A kind reminder: A feature that does not work is not a feature, it is a bug.

Surprise!

Building software can be an exciting activity. At some point in Week 10, requirements may change or be added, and you may need to adapt your project to meet them. This is how it works in REAL LIFE, and we will simulate it here.

Be prepared for changes! Build your software in a way that is *easy to extend and change.*

You may be asked to make small changes. This is to practice the software development/construction process. If you successfully develop it, your final mark may be increased by up to 2 marks (your client will "pay" more to have better software).

If a group does not participate in any of the checkpoints, they will not be eligible for this "surprise" (missed opportunity).

Checkpoint

In week 7 and week 10, we will have checkpoints. You must participate in your lab session and show your tutor what stage of development you are in (only ONE member has to present, make sure someone in your group did it). For the first checkpoint, we expect you to have at least a schedule of your development activities, the roles of each member described, and at least one meeting minute besides an initial code structure. For the second checkpoint, we expect your app to be almost fully implemented and tested. You also need to show your report and the completed sections. Your tutor will review your schedule and will provide quick feedback. The checkpoints are mandatory.

*Note that every customer wants to know how the software is progressing. Here's the same, we are your customer, you need to show that the app is being developed and that you will meet the deadline. Every time you do not meet a checkpoint, your final mark for *the group project will be reduced by 5 marks* (you missed an important meeting!).

Note that besides those mandatory checkpoints, if you have any issues at all, please feel free to ask us on the Wattle forum, lab or drop-in sessions.

Project Submission

We will use the school's GitLab server for submitting the project. Your group should create a GitLab repository and continuously update your work from **week 6 to the submission deadline**. We will request you share the link to your repository during the group creation on Wattle (set the visibility of your group project repo to private).

To get started, fork the course group assignment repository which provides some template files:

- **Report.md** (use it as a template for your report); **Checklist.md** (project minimum requirements); **statement-of-originality.yml**; **MeetingTemplate.md** (a template for your minutes).

Code development

All of your work must be continuously and regularly updated to your group repository. You are expected to manage tasks, including merging branches and consolidating your codes well ahead of the deadline.

- You must include @author annotations for each java code file with the UID and name of the key author. You must also include author annotations for methods and classes where significant contribution were involved.
- Git commit: You must commit often and include meaningful commit messages. We expect to see regular commits by each member during the full course of the project from the commit history. Failing to do so will result in significant mark reductions and academic misconduct investigations.
- While you may use branches during the development process, only ONE branch (i.e., normally the master branch of group Gitlab repository) will be assessed, in which you must place all the files, including code files, report, meeting minutes, APK, etc.

- Remove unused code files and/or segments. Do NOT report a feature that was not actually implemented in the final submission.

Use your own GitLab user account (UID) to commit your contributions. Add this information to your report. (Failing to do this may result in zero marks and academic misconduct investigations).

Report (Important!!!)

The report is an essential part of the group assignment and should give us evidence and details of the functioning of your group, project decisions and implementation. The technical outcome is a key aspect of the evaluation of this project and your reasoning process and teamwork are just as important.

Therefore, you are required to produce a detailed report which presents the justification for your design and team management of your project. A template for the report is available in our GitLab Repo. It must be in the form of a markdown file titled 'Report.md' and include (but not limited to):

- Team structure and roles
- Summary of individual contributions
  - This section must contain each individual contribution to the project, including classes, methods, lines of code, and link to the code files implemented by each team member.
  - Each team member is responsible for writing their own subsection.
  - A generic summary will not be *acceptable* and may result in team members losing a significant amount of marks.
- An app summary with screenshots.
- A list of examples/use cases of your app.
- A UML diagram (e.g., class diagram).
- A design summary page (include justification for decisions made, diagrams, etc). The design summary is an important item of your report and should include all decisions made and information required to understand your decisions. This includes, for example,
  - Details about the parser (describe the formal grammar and language used)
  - Decisions made (e.g., explain why you chose one or another data structure, why you used a specific data model, etc.)
  - Details about the design patterns used (where in the code, justification of the choice, etc)
  - If you implement the surprise item, explain how your solution addresses the task.
- A summary of the basic App and implemented features
  - List all items/features implemented in your project (Separate features into their categories).
- A summary of known errors/bugs (list of bugs).
  - A list of known errors and bugs demonstrates you tested your app. We may deduct some marks based on the bugs listed in your report. However, if a bug is found by us but reported, the part of the software containing the bug will not award you any marks.
  - How to write a bug report? Here is an example: https://blog.instabug.com/how-to-write-a-bug-report-the-ideal-bug-report/
- A testing summary section (including e.g., number of test cases, description of tests, coverage, etc.)

Be concise (just in case, according to the Oxford dictionary, concise means "*giving a lot of information clearly and in a few words; brief but comprehensive*").
*If you are not familiar with YAML (for markdown files), you can refer to: https://commonmark.org/help/.

All attempted features MUST be documented

> Note that if it is not in your report, we may not consider the feature, data structure, design pattern, etc, as completed. It is your responsibility to report everything that was implemented in your project. We are not playing treasure hunt and appeals will be rejected if what was implemented is not explicitly described with reference to method names, line numbers, etc., in your report.

Please list all features you have attempted with the feature ID. Preferably, we would like you to separate them by category and classify them for difficulty (this helps with marking).

For example (*Check assessment rubrics for more details):

Feature Category: Privacy

Implemented features:

1. [Privacy-Request]. Description of the feature and your implementation (easy)
    - Class X, methods Z, Y, Lines of code: 10-100
    - Class Y, methods K, L, M, Lines of code: 35-150
2. [Privacy-Block]. Description … … (medium)

Feature Category: Firebase Integration

Implemented features:

3. [FB-Auth]. Description of the feature and your implementation (easy)
    - Class A: methods A, B, C, lines of code: whole file

...

Other important items

You should place all non-android documents into one folder named Items at the root of your repository. Make good use of links in your report to make reference to these documents in the appropriate sections. These include:

- Team meeting minutes (at least 4 (this is the minimum!))
    - You should develop a clear plan of how the work will be divided and document the same in the minutes of the first meeting.
    - Right after every meeting (max 2 days after, otherwise they will not be considered), upload the minute to your project repo. Note that we will check the commit dates in your repo.
- Your conflict resolution protocol - Define this protocol in your first meeting (mandatory). This shall include an agreed procedure for situations including (but not limited to):
    - e.g., if a member fails to meet the initial plan and/or deadlines
    - e.g., if your group has issues, how will your group reach consensus or solve the problem?
    - e.g., if a member gets sick, what is the solution? Alternatively, what is your plan to mitigate the impact of unforeseen incidents for this 6-to-8-week project?
- A statement of originality (all members must be listed there and sign the document by adding your own names to your group repository, using your own Gitlab account).
- You must bundle your app into a standalone APK that can be loaded and executed correctly on an AVD (see video on Wattle in the Group Assignment Section).
- Create a short video titled `features.(mp4|mov|wmv)` that demonstrates each implemented feature. The video should contain a visual representation of every feature showcased in the running app with clear voiceover narration or written captions that explain the current feature being demonstrated. Include both Basic and General features, in the same order as listed in the report.

- Your final submission at the default branch of your GitLab repository and the commit history will be assessed. The project cannot be updated after the deadline (otherwise, you may get zero marks).

Individual Reflection and Peer- and self-evaluation

Each student much submit an individual reflection via Wattle, with the same due date as the group report in Week 11. This includes:

- Individual reflection: write 100-120 words related to your experience during the group project (be concise and direct). For example: how was your experience working in a team? Reflections on what your team could have done better, what worked and what did not work? How was the work divided and was that fair?
- Peer and self-evaluation: an evaluation of each member of your group:
    - Name - UID
        - Contribution: 25%
        - Justification: This member was responsible for the classes …, collaborated with others, wrote sections X and Y in the report.
    - Name - UID
        - Contribution: 10%
        - Justification: This member was unavailable most of the time within week 7-9, did not deliver some of the modules promised and recorded in the minutes (meetings 1 and 2). However, this member completed some modules (list …) and handled the minute madness presentation.
    - …

*The "contribution" measures the percentage contribution of each member and must add to 100%. For example, a group with 5 members who contributed equally to the project must have a contribution of 20% for each member.

Group Project Demonstration

In week 12, there will be a hybrid group demonstration during our lecture time. This will be an opportunity to showcase your project and also to get feedback and marks on what you have done. This Project Demonstration will follow the Minute Madness format where each group will present a set of timed slides within a predetermined time (slides will change automatically every *x* seconds). The total number of groups will determine the duration of each presentation and the number of slides.

Your presentation must clearly present your topic and convince the audience that your app is innovative and could be used in the real world. You must briefly talk about the structure of your project, decisions made, and solutions for the problems faced during the project (use of a particular data structure, design pattern, etc).

Your presentation must be clear, convey your ideas and give an overview of the software construction process you and your team experienced.

I would recommend using some screenshots to show how your app looks (images can be better than words sometimes).

More details related to the project demonstration will be released in Week 10. Slides must be uploaded to the link to Microsoft PowerPoint in Week 11.

Best Practices for Using Generative AI Tools for Software Construction

Generative AI tools, such as ChatGPT and Co-pilot, offer immense value to software developers and can significantly enhance their efficiency. These tools have shown the ability to generate code snippets, suggest improvements, and provide contextual guidance, thereby simplifying the coding process and reducing the time and effort required to complete some tasks. By leveraging these AI-based tools, we expect you to focus more on designing and refining your software rather than spending valuable time on mechanical tasks automated by AI tools.

It is important to note that you and your group are solely responsible for any output generated by AI tools that you have used to complete your group assignment. It is also mandatory to explicitly acknowledge where and how these generative AI tools were used to assist in completing the assignment. Failing to acknowledge the use of these tools is a violation of academic integrity rules.

Important:

1. Any output generated by Gen AI tools in your assignment must be explicitly acknowledged and attributed to the individuals or the whole group for using them. This includes any pieces of code, comments, text, or other content created with the assistance of Gen AI tools. Gen AI tools cannot be listed as authors, as they are not accountable for their outputs.

2. The explicit acknowledgement must be given wherever these outputs are used in your assignment and also in the statement of originality file. Explain how Gen AI tools were used, as well as the proportion of the work done by each member and the AI tools.

3. Gen AI tools can be used in a variety of tasks, including debugging, documentation and commenting, and assisting with other writing and coding tasks. These tools can help to streamline the coding process, improve code quality, and increase overall efficiency. However, the output generated by these tools should be critically assessed as they are not always accurate. It is part of your job to review, validate and refine the output generated by Gen AI tools.

4. You must have a complete and thorough understanding of all aspects of your submission, including any part of the code and report contents generated by you, any member of your group or an AI tool. You must be able to explain and justify your choices and decisions at any time if requested to do so.

5. You must ensure that any output meets the project requirements.

6. Do not use generative AI tools to bypass learning or understanding. Use them to improve efficiency in non-core tasks. Over-reliance on external tools may compromise your grasp of code structure and relevant concepts, hampering your learning and performance in more complex tasks.

7. The use of Gen AI tools is optional.

IMPORTANT
We would like to ensure that you are reading the entire document carefully. Please go to the Wattle course site and find the forum thread titled "Group Assignment Mystery". Without revealing anything, please leave a message in the thread stating the following "B, I've just solved the mystery". This will let us know that you have read and understood the entire document. Thank you.

Frequently Asked Questions

What kind of app should we develop?

You could adopt a wide range of themes and features relevant to Education, including but not limited to those example apps in the project introduction.

From the notion of "education", you could think about what purposes could be fulfilled, or what issues could be resolved with the help of a mobile platform. Alternatively, you could also select an existing application as your starting point, and bring improvements and innovations with some changes in features. You can also think of different target users and sectors (e.g., industry training, military training) while deciding on the theme of your App.

We expect you to think outside the box and develop your own idea! Creativity and Novelty are part of the marking criteria. It does not need to be complex, simple ideas are also good.

On the other hand, your application is only a prototype that showcases how your idea could be developed and implemented in a real application. Please prioritise your effort on the core aspects of this project before developing additional features. In particular, visual appeal is only one of the assessment criteria within "User Interface". We suggest you not overly focus on the artistic design of the App beyond simple alignment and the use of basic UI components and toasts.

Why and how to simulate a data stream?

To simplify the development of the app instead of using a client-server model, we allow you to create a file to simulate users interacting with your app. You may want to create a method or module to read from a file every x seconds and feed the new information into your application. Note that there is a Wiki article written by a former student that may be helpful for this task. (You should also consider writing a good article to share your knowledge with your colleagues. Wikis are also an example of learning platforms).

How should I develop the search mechanism?

Your search mechanism is responsible for tokenizing the query, parsing and evaluating it. You must define your own grammar and document it. You can use the CFG Stanford tool to help you create your grammar.

Example: `Search: java and #advanced and @active`

> In this case (with a corresponding grammar and parser implementation), the user is looking for courses with "java" in their name or description field, annotated with the tag "advanced", and has an active status currently.
>
> Hint: Which data structure would be most appropriate (or efficient) for this case? Your group must decide how to implement it for efficiency, generalisability, extensibility, etc.

What about the data?

The data must be in a file and be structured in a way that is easy to be retrieved/processed.

Let's say the data for your app is stored in a local XML file structure as follows. Here is an example for representing reviews for courses. Note that this may not necessarily be the best format, and you must decide the best format for your app, which will be evaluated. For example, the following example records the information for a course. In this example, each course is considered a single instance.

```
...
<courses>
   <course id="143">
        <lecturer id="abd1253122"/>
        <semester>2023-S1</semester>
        <created>2023-02-20T18:10:11Z</created>
        <code>...!</code>
        <name>...!</name>
        <description>Great Java course!</description>
        <capacity min="3" max"300">260</rating>
        …
   </course>
</courses>
...
```

*Note that you need to generate the data instances for your application as well the data format/structure. You are free to choose any file format (JSON, XML, Bespoken). You can create a script to generate your data, a script to scrape from the Web or manually create it. You can also get data provided by some API (e.g. Khan Academy API). This and only this script can be written in any programming language (Java, Python, etc). Please, make sure you are allowed to download the data from external sources and that the script is included in your repository.

**As an option, you can use Firebase (and JSON) if you feel comfortable with it and this is the best choice for your app.

What about the data structures?

You must know where, which, how and when to use it. It depends on how you design your app. Think about what data structure you would use if you needed to search for something. For example, if you go for a tree, what would be the key of your tree? Is this the most appropriate data structure for your app? Discuss with your group.

Which design pattern should I use?

Again, you must know where, which, how and when to use it. It depends on how you design your app and the features you implement. Several features (!!!) were proposed thinking about the most appropriate design pattern to be used. You can use design patterns that were not covered in the course, but do not forget to explain/justify them in your report.

How complicated should my grammar be?

It does not need to be complicated, but it must be unambiguous and easily extendable. Most importantly, you have to demonstrate that you know how to implement it and be consistent with your app theme.

What should my app look like?

You are free to design your app. Be creative and check the rubrics.

Do you have any advice?

Read this document and do not leave it to the last minute. Choose your teammates carefully (run interviews, align expectations, check their availability and commitment to this assignment). Be upfront about your expectations and agree on that with your teammates. Define tasks for each member of the group. Regularly check if your group is advancing with the tasks given to them. Read and discuss each item in the list (parts 1 and 2) with your group before implementing it. Be prepared for a plan B if something unexpected happens. Don't leave it to the last minute (yes, I intentionally repeated it).

General Information (assessment, submission, presentations, etc)

Working physically apart, but remotely together!

Some of you will need to work remotely to complete this project. No worries! It is very common to have remote collaboration on software development and there are many tools to help you accomplish this work. Use this opportunity to practice collaborative tools, like Git, Zoom, Microsoft Teams, Google Doc, Slack and others to plan, design and implement your project. This is exactly what you will find in the industry, therefore, use this project as an opportunity to get some experience in these tools.

Use the forum on Wattle to help you find teammates: Introduce yourself, explain how you can contribute to the project, give some details of the theme of the app you want to develop, etc.

What is the number of members per group?

Each group should consist of 5 students. Groups can contain a mix of undergraduate and master students and students from different labs. In exceptional cases, we will accept groups of 4 students, which will be assessed in the same way.

**Your group must be created and registered before Week 7. Register your group on Wattle.
Do not start late, it may impact your project outcomes.**

Due date and late submission policy

The project is due on the Friday of Week 11 at 11:59 pm (hard deadline). As the project will be done iteratively over the second half of the semester, every group should have something that will gain a pass mark well before the due date. No late submissions shall happen. Whatever your group has done up until the due date will be assessed. You are not allowed to submit/update any files after the deadline.

Originality

The project must be your own original work. If you make use of any code that is not your own, it must be clearly referenced. This can be done by adding a simple comment next to the code stating where you obtained the code from. You must also add this to the statement of originality document. This is very important, as any breach of this needs to be investigated and reported. You are much better off not doing this project than copying a small part of code and risking academic misconduct. Remember, we are assessing your code, not someone else's code.

Every person in the group is responsible for the originality of every part of the project (regardless of who actually wrote or contributed to it). Any violation of the academic honesty and plagiarism policy will result in the entire group receiving the mark of zero for the group project.

*Any images, or other assets that you copy from the Web must be attributed (e.g. where did you obtain them from? Splash.com). This must be added to your statement of originality. Ideally, you should only use assets that you have the right to copy, such as ones you create yourself, are in the public domain, or under a creative commons licence. With the statement of originality, you are safe and will be assessed only based on work done by you and your group.

How will projects be assessed?

1. Everyone in the group must implement and commit a non-trivial amount of code (we will check it!).
2. You will be assessed individually and as part of the group. You must make sure that the work is divided up so everyone has the opportunity to undertake some coding and contribute to the overall project. Marks will be reduced for those group members who contributed much less than others, and possibly for the group for poor group management (remember, this is a group project and you must work as a team and collaborate effectively). Note that each group is expected to contribute approx. 80 hours for this project. By default, the marks WILL NOT be the same for each member of the group. Marks will be based on several criteria (check rubrics). Group members who contributed much less than others may receive an extra [10,80]% penalty. Members who did not contribute to the group assignment will receive zero marks in this assignment.
3. Based on your software and the quality and depth of your report and code documentation. Please refer to the rubrics for the key assessment aspects.
4. Based on your App (via your provided APK), implemented features and quality of decisions.

Marking Criteria

The indicative weight for each criterion in determining the final grade is displayed as follows.

Final Group Project (FGP) =
(3*Basic Features + 4*General Features + 4*Report + 1*Presentation + 2*Teamwork + 1*Creativity)/15

Each criterion will be marked as indicated in the Assessment Rubrics. To score 'excellent', you must satisfy all the 'very good' criteria. To score 'very good', you must satisfy all 'satisfactory' criteria.

The individual contributions must be in the meeting minutes (tasks assigned and completed by each member), report (detailed list of contributions for each member), as well as reflected in the Git commit history. Peer and self-evaluation will be used as references and must be submitted to Wattle by each group member (see individual reflection). Only examiners have access to the individual reflections.

We expect that each group member contribute significantly to the code development throughout the period. Specifically, this entails being the key author for at least TWO features (either Basic or General features), on top of having contributing to the report, meetings, and any other tasks.

We also expect you to commit (reasonable amounts of code) frequently (e.g., twice every week from week 7!). Last-minute commits and lines implemented are not expected in large projects and, therefore, marks will be deducted for poor project and time management. We have assessed hundreds of projects before and we know that those who start early, finish early and deliver a better product. This is our way to help you do better :)

Note that despite the indicative formula presented above, violations of the rules for assignment submissions may result in additional penalties, depending on the severity. Examples include:

- Poor GIT practices (e.g., pushing significant amount of codes to the repository at once);
- Late/missing submissions in any items (e.g., individual reflection)
- Misleading content in the submissions (e.g., reporting completion of features that are not implemented, misrepresentation of individual contributions)

Assessment Rubrics

Remember this project is worth 30% of the overall course mark. Teamwork is a key learning outcome for this project, so I would encourage you to prioritise working well as a team over extending the project.

The basic app and general features will be assessed based on diverse aspects of software design. The following rubric lists the aspects of consideration based on the judgement of the markers.

A full mark will only be granted if there are no bugs or design issues for each implemented feature. Partial marks may be given depending on the quality and progress of implementation.

Criteria	Excellent	Very Good	Satisfactory	Unsatisfactory
Data Structures	Chosen data structures well suited to subsequent usage, leading to efficient code in terms of both programmer and computer time. The chosen data structure can grow as the input grows without concern for memory (scalability).	Data structures are well suited for subsequent usage with minimal conversion or transformation required.	Satisfactory choice of data structures leading to inefficiency or repeated need to convert data structures or values to solve tasks.	Unsuitable data structures were chosen leading to significant inefficiency or inability to solve tasks. Data structures loaded in each task or different data structures used for each task.
Code Quality and Organization	Excellent documentation, naming and style, following conventions and making the code easy to read. Code is well commented and highly understandable. Code is very well organized, using generics and inheritance as appropriate. Construction for reuse. Easy to extend. The overall program structure makes the code easy to follow. The program is modular and easily expandable.	Good Documentation, naming and style are complete, consistent and appropriate. Code is well commented and partially understandable. Good code organization with appropriate use of inheritance. (mostly) Construction for reuse. Easy to extend. The overall program structure makes the code easy to follow. The program is modular and easily expandable.	A reasonable attempt at documentation, naming and consistent style, but it could be improved in places. Code is partially commented on and partially understandable. Alternatively, more significant shortcomings in one aspect. Inconsistent organization and/or repeated code. Construction mostly with Reuse. Code can be partially extendable.	Missing or poor documentation, poor naming, poor coding style. Significant unnecessary code. Lack of comments in code / code is not understable. Poor code organization, or Code is very difficult to follow. Construction with low reusability. Difficult to extend. Missing APK. Lacks @author annotations and @feature in the code files.
Report	Report is well organized, well presented, clear and concise. Project decisions are well detailed with examples and discussion. It also presents code and analysis. UML diagrams are presented. A clear and detailed testing summary is provided. References are provided.	Report is well organized and presented. Project decisions are detailed with suitable examples and discussion. It partly presents code and analysis. Some UML diagrams are presented. A clear testing summary is provided. References are provided.	Report includes necessary required content, but lacks organization and/or presentation (e.g., template text from the sample were left uncleaned) Project decisions are not clear but present some examples and discussion. It does not present code and analysis. References are partially provided.	Limited discussion or understanding of issues. Poor decisions made and choice of examples. Poor organization, or lack of clarity makes the report hard to follow. References are not provided. Reported information contradicts the submission (e.g., a reported feature was not truly attempted).

Testing	JUnit coverage test achieves at least 70% of code (without UI). Clear evidence of testing to verify correctness and robustness. Exceptions and error cases are checked. Clear evidence of Unit and Integration tests.	JUnit coverage test achieves at least 60% of code (without UI). Some evidence of testing to verify correctness and robustness. Clear evidence of boundary and normal functioning tests. Some evidence of Unit and Integration tests.	Repeatable unit testing is performed on the majority of the project. Appropriate use of Test Suites and Parameterized tests.	Test are not well designed (random tests). Minimal or no test cases can be found.
User Interface	User interface is intuitive and can be easily used without guidance. Use of consistent theme and style. Feedback is provided to users based on interactions. UI is responsive (adapt to different screen sizes and orientations). Stylish and friendly look and feel.	User interface is mostly intuitive and can be used with little guidance. Use of consistent theme and style. Some feedback is provided to users based on interactions. UI is mostly responsive (reasonably adapt to different screen sizes and orientations) Friendly look and feel.	Standard user interface and not very intuitive (needs some guidance). Use of theme and style are not consistent. Navigation is not clear. Lack of feedback to users based on interactions. UI is responsive and provides only screen orientation mode.	Standard user interface and unintuitive (needs guidance). Inconsistent use of theme and styles. Navigation is not clear. Lack of useful feedback to users based on interactions. No guidance to users. UI is not responsive and provides only one screen orientation mode.
Creativity / Uniqueness / Specific Theme	App is innovative and can be applied in a real world scenario. Problem statement is clearly defined and has the potential to impact multiple beneficiaries. Unsolved problem with high significance. High degree of creativity in the design and features.	App is innovative and can be applied in a real world scenario to a limited number of beneficiaries. Problem statement is clear and has the potential to impact a limited number of beneficiaries. Problem with high significance. Good combination of unique design and features.	The developed app can be applied in a real world scenario to a limited number of beneficiaries. Problem statement is clear but with low potential for impact. Some special features are incorporated.	Standard App created only for demonstration purposes. Lack of innovation, creativity and special features. Low or no potential for impact.
Teamwork	A clear decision-making procedure is formally established by the group, a document formalizes the roles and contributions/ideas given by each member of the group. Goals are well established, and priorities are well documented and organized. All members respect each other, and conflicts are resolved with open dialogue and compromise (well documented). A conflict resolution approach is documented and well defined. Disputes are described along with their outcomes.	Communication has worked well within the group and you have been able to adapt to a situation that has arisen. A procedure for making decisions is informally established by the group. Goals are clear and achievable. Priorities are clearly documented. All members respect each other, and some conflicts are resolved with open dialogue and documented. Interactions between group members are documented. A conflict resolution approach is documented. Disputes are described along with their outcomes.	Tasks have been well divided, with each member completing a significant part of the project (Git history and minutes). A procedure for making decisions exists, but it is not clear. Goals are unclear, too general or unachievable. Some members did not feel free to contribute, ask questions, or share ideas. Interaction between group members is limited and not documented. A superficial conflict resolution approach is documented. Disputes are described along with their outcomes.	Cannot find (at least 3) meeting minutes. One-person team (by Git history). Decisions are made by individuals. The group atmosphere is competitive and individualistic. Conflicts cannot be resolved between group members. Low interaction between group members. No conflict resolution approach is documented.

*If you want to have your assignment remarked (appeal), you need to explain with reference to the rubrics why your original mark was wrong.

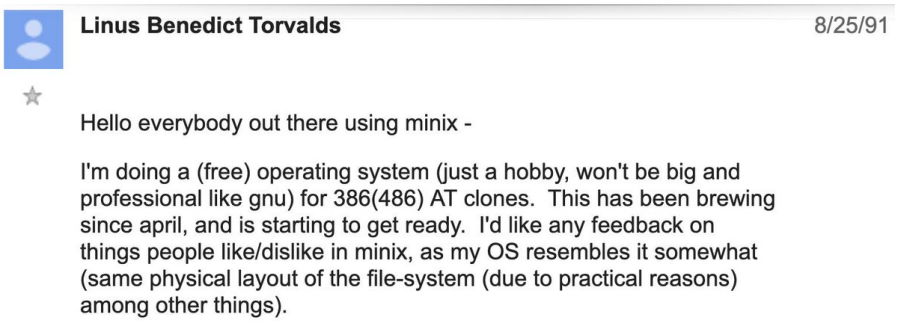
Other relevant information

You may use any version of Android Studio and Android SDK. But make sure your app code can be compiled by the tutors. It is recommended that you develop your app on Android Studio emulator so that you can demo your app via a sharing computer screen. The app must be developed using Java.

There is flexibility for the server part *(if you want to implement it)* using different options, for example, PHP, Google Firebase, or third-party services. Only for the project, you can use any version of JUnit.

There will be no restriction on external libraries. Make sure that the external libraries are clearly referenced in your documentation and report. Only remember that we will evaluate what your group did, this is very important to understand!

Last but not least, here is what Linus Torvalds said about his project in 1991:



Linus Benedict Torvalds 8/25/91

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

It is time to code! Have fun!