SEG2105 – Introduction to Software Engineering
Fall 2025
Android Project: OTAMS (20%)

Instructions

1. You will work in the same team throughout the semester
2. You will submit each deliverable as a release on GitHub
3. At the end of the semester, the team must demonstrate a completed application. For instance, one team member demonstrating a screen with one functionality while another member showing another screen with another functionality (running on a different phone), **WILL NOT** be accepted. The team must produce a single application with all the required features.

Academic Honesty

Using someone else's work and presenting it as your own is considered academic fraud. You are responsible for writing all the code for the application. However, as a software developer, it is natural to refer to online resources such as code examples or online forums. This is an important aspect of the software development process. Nevertheless, it is expected that you will drastically limit the practice of directly copying code that you have not developed.

If you do need to copy a significant portion of code, defined here as more than five lines, it is crucial to acknowledge its source by including a comment above the code snippet. Additionally, it is essential that you thoroughly understand any code you incorporate from external sources. The instructor and/or teaching assistants may inquire about these code snippets, and you should be prepared to answer questions related to them.

Note about Generative Language Models – Please Read Carefully

The course instructor acknowledges that ChatGPT and other generative language models, when used appropriately, can provide significant value to software developers. However, excessive reliance on these tools can hinder the learning process. Therefore, there are two specific scenarios in which you are permitted to use these tools:

1. To search for information about error messages or exceptions.
2. To find guidance on how to perform generic operations that involve only a few lines of code (e.g., connecting to a database).

Any usage of these models beyond these two situations is strictly prohibited. The instructor will employ experimental tools to scan your code for any signs of AI-based generation.

While we understand that generative language models have quickly become a key part of the software development process, we believe that at this stage of your education, it is essential to hone your core development skills. By doing so, you will be better prepared to later become an effective “human in the loop” who is able to interpret, refine, and debug generated code. Without the foundational, hands-on experience of developing software with minimal or no AI assistance, it is unlikely that you will become an effective software developer in this era of generative AI.

Project Description

In this project, you will implement the Online Tutoring Appointment Management System (OTAMS). OTAMS is a mobile application designed to streamline the process of tutoring appointment scheduling at the University of Ottawa Help Centre. OTAMS supports three types of users: **Student**, **Tutor**, and **Administrator**.

- The **Student** browses available tutoring slots so they can book or cancel sessions and rate Tutors.
- The **Tutor** manages their availability and responds to session requests.
- The **Administrator** approves the account-registration requests of **Students** and **Tutors**.

In the next sections, we describe the application in detail from the perspective of each user.

Student

To become a **Student**, a user submits a registration request that must be approved by the **Administrator**. Once registered, a **Student** can:

- View their upcoming sessions.
- Cancel an existing session (only if it starts in > 60 minutes).
- View their past sessions.
- Rate a Tutor following a completed session.
- Book a session by selecting an available 30-minute slot for a desired course.

Tutor

To become a **Tutor**, a user submits a registration request that must be approved by the **Administrator**. Once registered, a **Tutor** can:

- Specify and modify the 30-minute availability slots they would like to offer.
- View their upcoming sessions.
- View their past sessions.
- Cancel sessions.
- Approve or reject session requests.

The **Tutor** may also elect to auto-approve all session requests, so they do not have to act on each one individually. Requests are simply accepted automatically.

Administrator

The **Administrator** is a pre-registered user (their username and password are seeded in the database when the system is first launched). The **Administrator** can approve or reject registration requests from Students and Tutors.

User Interface Design

This course does not focus on user interface design. However, students are encouraged to produce aesthetic user interfaces that are easy to use. Consider the Android Design Guidelines when designing your application: https://developer.android.com/design.

Deliverables

The project is divided into four incremental deliverables. Each deliverable must be submitted by the posted due date as a GitHub release. Late submissions will incur penalties (see below). All work must be maintained in your Git repository throughout the term.

Deliverable	Due date
1 – GitHub and User Accounts (3%)	October 13
2 – Mostly Administrator Features (4%)	October 27
3 – Mostly Tutor Features (4%)	November 10
4 – Mostly Student Features (and Integration) (9%)	December 1

Late submissions penalties:

- Less than 24 hours ➔ -5%
- 24 hours or more but less than 48 hours ➔ -10%
- 48 hours or more ➔ -100%

Deliverable 1 – GitHub and User Accounts

(Read the "Project Description" section carefully before you start working on this deliverable.)

Join the course GitHub Classroom, create your team repository and accept this project (see "Steps to Create your Team on GitHub").

Implement the **Student** and **Tutor** registration forms. For now**,** we assume that registrations do not yet require Administrator approval; that workflow will be added in **Deliverable 2**.

Registration data to capture:

Student	Tutor
First name	First name
Last name	Last name
Email address (username)	Email address (username)
Account password	Account password
Phone number	Phone number
Program of study	Highest degree
	Courses offered (one or more)

The **Administrator** is pre-registered and therefore does not complete a form.

After registering, the user can log in. A second screen must display:

Welcome! You are logged in as <role>,

where <role> is **Student**, **Tutor**, or **Administrator**. The user can also log off. No other functionality is required at this stage.

You can optionally use Firebase or SQLite for DB support. If you do not use a DB at this point, you can store the information in memory (but it would be lost when you terminate the app) or on a file.

Submission Requirements

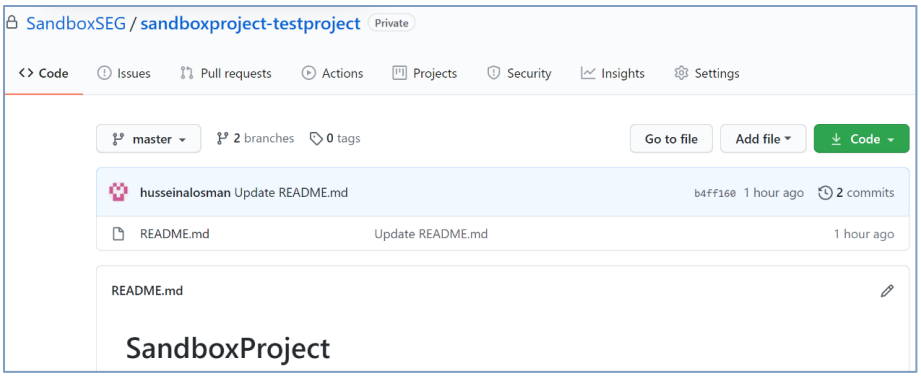
In addition to completing the implementation, please prepare:

1. A PDF document containing a UML class diagram of your system domain model. Such model should not include Activity classes. It should only include classes that represent the core entities, their attributes, and the relationships between them, such as associations, generalizations, and multiplicities relevant to the system's business logic. For example, in Deliverable 1, the UML class diagram may include classes such as Student, Tutor, and Administrator, each with relevant attributes like firstName, lastName, email, password.
2. A README file with the credentials needed to sign in as Administrator.
3. A short demo video, no longer than 5 minutes, that shows the app's functionality as outlined in the marking scheme.

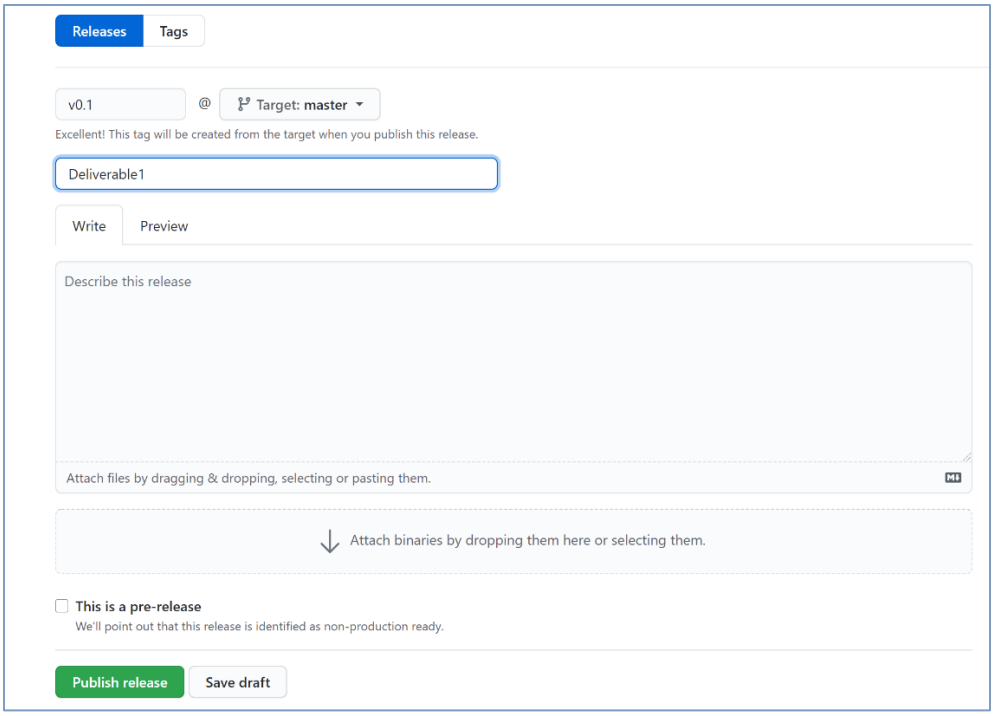
Submission Instructions

To submit, you will create a release from your repository as follows:

4. In your repository, click on "tag."



5. Click on the "Create new release" button and fill the form as follows:
    a. For "Tag version," enter v0.1
    b. For "Release title," enter Deliverable1
    c. From the section of the form where you can attach binaries, upload the **APK of your app**.
    d. From the section of the form where you can attach files, **upload the demo video, UML diagram, and README files**.



6. Make sure to rename the APK after the name of your team (e.g., "Project_Group_1_debug_apk").

Deliverable 1 Marking Scheme

Feature or Task	% Weight (out of 100)
The team created in GitHub classroom contains all members of the group.	10%
Each member of the group has made at least one commit to the repository.	10%
The team has submitted a demo video that shows the app's functionality.	10%
The UML Class diagram of your system domain model is valid.	5%
The APK file is submitted correctly with the release.	5%
A user can create a **Student** or **Tutor** account with the required fields.	15%
The **Administrator**, **Tutor**, or **Student** can see the "welcome screen" after successful authentication, which specifies the user's role.	15%
The user can log off after logging in.	10%
All fields are validated, with appropriate error messages displayed for incorrect inputs.	20%
Optional Bonus - The group uses a database (e.g., Firebase, SQLite, or another similar technology) to store user data.	+5% (bonus)

Deliverable 2 – Mostly Administrator Features

(Read the "Project Description" section carefully before you start working on this deliverable.)

You should implement the Administrator functionality. Moreover, if you have not done so, you should start using a DB (e.g., Firebase, SQLite, or another similar technology).

The **Administrator** has an inbox to receive registration requests from **Students** and **Tutors**. Each request contains all the information the user entered in the registration form except for the password. The **Administrator** can approve or reject a registration request.

The **Administrator** can view the list of registrations they previously rejected. They can select any request from this list and change their decision to approve the registration. However, once approved, the decision can no longer be reversed.

When a **Student** or **Tutor** registers their account, they cannot gain access to the system before obtaining the approval of the **Administrator**. When a **Student** or **Tutor** attempts to log in, one of the following occurs:

- If their registration request was approved by the **Administrator**, they reach the welcome screen.
- If their registration request was rejected, they receive a message informing them of the rejection and providing a (fake) phone number for contacting the administration if they wish to resolve the matter.
- If their registration request has not yet been processed, they receive a message stating that approval is pending.

In addition to completing the implementation, please prepare:

1. A PDF document containing a UML class diagram of your system domain model (Deliverables 1 and 2 only). Do not include Activity classes.
2. A README file with the credentials needed to sign in as Administrator.
3. A short demo video, no longer than 5 minutes, that demonstrates the app's functionality as outlined in the marking scheme.

To submit, please follow the instructions specified under the **Deliverable 1** description.

Deliverable 2 Marking Scheme

Feature or Task	% Weight (out of 100)
The updated UML Class diagram of your system domain model is valid and includes relevant classes for Deliverables 1 and 2.	10%
The APK file is submitted correctly with the release.	5%
The team has submitted a demo video that shows the app's functionality.	10%
The **Administrator** can view the list of registration requests.	5%
The **Administrator** can view the user's information associated with each request (excluding passwords).	5%
The **Administrator** can approve or reject a registration request.	5%
If approved, the request disappears from the list of pending requests.	5%
If rejected, the request appears in the rejected requests list.	5%
The **Administrator** can view the rejected requests list.	5%

The **Administrator** can approve a previously rejected request (removes it from rejected list).	5%
If the user request was approved, they are directed to the welcome screen.	5%
If the user request was rejected, they receive a message informing them of the rejection and displaying a phone number to contact the administration.	5%
If the user request has not been processed yet, they receive a message informing them that approval is pending.	5%
All registration requests, along with their statuses, are stored in the DB.	25%
Optional Bonus - Users receive email or phone notification upon approval/rejection.	+5% (bonus)

Deliverable 3 – Mostly Tutor Features

(Read the "Project Description" section carefully before you start working on this deliverable.)

You should implement the majority of the **Tutor**'s functionality. A **Tutor** can create availability slots by specifying the date, start time, and end time. Start and end times must be in 30-minute increments (e.g., 8:00, 8:30, 9:00, etc.). A **Tutor** cannot select a date that has already passed or define overlapping time slots.

During slot creation, the **Tutor** can choose whether to manually approve session requests or allow them to be automatically approved.

The **Tutor** has a list of upcoming sessions and past sessions. If they tap on an upcoming session, they can view the **Student**'s information. They can view and then approve or reject pending session requests from **Students** or cancel previously approved sessions.

For now, the **Tutor** can delete any availability slot they previously created. However, in **Deliverable 4**, you will implement a restriction preventing deletion of slots tied to booked sessions.

In addition to completing the implementation, please prepare:

1. A PDF document containing a UML class diagram of your system domain model (Deliverables 1, 2, and 3 only). Do not include Activity classes.
2. A README file with the credentials needed to sign in as Administrator.
3. A short demo video, no longer than 5 minutes, that demonstrates the app's functionality as outlined in the marking scheme.

To submit, please follow the instructions specified under the **Deliverable 1** description.

Deliverable 3 Marking Scheme

Feature or Task	% Weight (out of 100)
The updated UML Class diagram of your system domain model is valid and includes relevant classes for Deliverables 1, 2, and 3.	10%
The APK file is submitted correctly with the release.	5%
The team has submitted a demo video that shows the app's functionality.	10%
The **Tutor** can create new availability slots (date, start time, end time).	10%

The **Tutor** cannot select a past date or overlapping slot.	5%
The **Tutor** can choose whether requests require manual approval or not.	10%
The **Tutor** can view a list of upcoming sessions.	5%
The **Tutor** can view a list of past sessions.	5%
The **Tutor** can view a list pending session requests from **Students**.	5%
The **Tutor** can see the information of a **Student** that made a session request.	5%
The **Tutor** can approve, reject, or cancel sessions.	10%
The **Tutor** can delete availability slots they created.	10%
All fields are validated, with clear error messages.	10%
Optional Bonus - The group integrates with CircleCI to run automated builds and test cases.	+5% (bonus)

Deliverable 4 – Mostly Student Features (and Integration)

(Read the "Project Description" section carefully before you start working on this deliverable.)

You should implement the **Student**'s core functionality and finalize integration. **Students** can view a list of sessions they have booked or requested. The list should be sorted by date with the most recent sessions first. There must be an indicator showing whether their request was approved, rejected, or is still pending.

Students can cancel pending sessions. They can cancel approved sessions if they are not scheduled to start within the next 24 hours. **Students** can also view their past sessions and rate Tutors (1 to 5 stars) after a completed session.

Students can search for available sessions by course. They must enter a course code to view available slots. For each slot returned in the search results, the **Student** can see the **Tutor's** name and average rating. Once a suitable time slot is found, the **Student** can request a session. That time slot is then removed from the **Student's** search results. The system must prevent **Students** from booking sessions that conflict in time with their existing bookings or requests.

Unlike **Deliverable 3**, the system must now prevent **Tutors** from deleting availability slots that are associated with booked sessions. If a **Tutor** attempts to delete such a slot, an appropriate error message should be displayed.

Finally, you must implement at least four local unit test cases. There is no need to use Espresso or UI instrumentation testing.

In addition to completing the implementation, please prepare:

1. A PDF document containing your final report (see the marking scheme below for details).
2. A README file with credentials to sign in as Administrator.
3. A short demo video (max 10 minutes) showing all the app's functionality as described in the marking scheme.

To submit, please follow the instructions specified under the **Deliverable 1** description.

Deliverable 4 Marking Scheme

Feature or Task	% Weight (out of 100)
The UML Class diagram includes all relevant classes.	10%
The APK file is submitted correctly with the release.	5%

The team has submitted a video demonstrating the final app.	10%
Final Report:	
• Title page	2.5%
• Short Introduction	2.5%
• Updated UML class diagram	-
• Table specifying the contributions of team members for each deliverable	5%
• All the screenshots of your app	5%
• Lessons learned	5%
The **Student** can view a list of sessions they have booked or requested.	5%
The **Student** can see whether their session was approved, rejected, or is pending.	5%
The **Student** can cancel their request (less than 24 hours before approved session).	5%
The **Student** can search by course code and view available slots.	5%
The **Student** can see the name and average rating of each Tutor associated with a time slot returned in the search results.	5%
The **Student** can request to book a session which disappears from their search results.	5%
The **Student** cannot book overlapping sessions.	5%
The **Student** can rate a **Tutor** after completed sessions.	5%
The **Tutor** cannot delete slots with booked sessions.	5%
All fields are validated, with clear error messages.	5%
At least four local unit tests are implemented.	5%
Optional Bonus – The **Student** can export their approved sessions to their Google Calendar.	+5% (bonus)

Artefacts and Evaluation Summary

Deliverable	Weight	Artefacts
#1	3 %	GitHub Repo, APK, README File, UML v0.1
#2	4 %	APK, UML v0.2, README File, Demo Video
#3	4 %	APK, UML v0.3, README File, Demo Video
#4	9 %	APK, UML v0.4, README File, Unit Tests, Final Report, Demo Video