Android Project: Online Tutoring Appointment Management System

SEG2105 – Introduction to software engineering

2025

School of Electrical Engineering and Computer Science

University of Ottawa

Professor: Dr. Hussein Al Osman

Group 2

Student 1: Kevin Huang, 300417431

Student 2: Reda TALA-IGHIL 300455925

Student 3: Alexandre Turgeon, 300401347

Student 4: Colton Ventry, 300430827
Student 5: Yassine Yandouzi, 300448905

Submission date: December 1, 2025

Introduction

For this project we built an online tutoring appointment management system (OTAMS). The purpose of this project was to create an android application allowing the University of Ottawa students to book tutoring session for a desired course in an effective manner using agile principals. Our application was designed for three different types of users. The first type of user is an administrator whose main role is to manage the system by accepting or declining a registration coming from new students or tutors. The second type of user is a student who can search for courses, book and cancel tutoring sessions and rate their tutor after a completing a tutoring session. The third type of user is a tutor who can set new availability for a course, cancel tutoring session and availability and manage incoming requests from students.

Developing requirements

Administrator:

- As an administrator, I want to view a list of pending registration requests so that I can process new users' registration.
- As an administrator, I want to accept or reject new user registration so that I can control who has access to the application.
- As an administrator, I want to view a user important information so that I can make an informed decision about a user registration.
- As an administrator, I want to view a list of approved users so that I can see all the users of this application.
- As an administrator, I want to view a list of rejected users so that I can reconsider my decision if necessary.

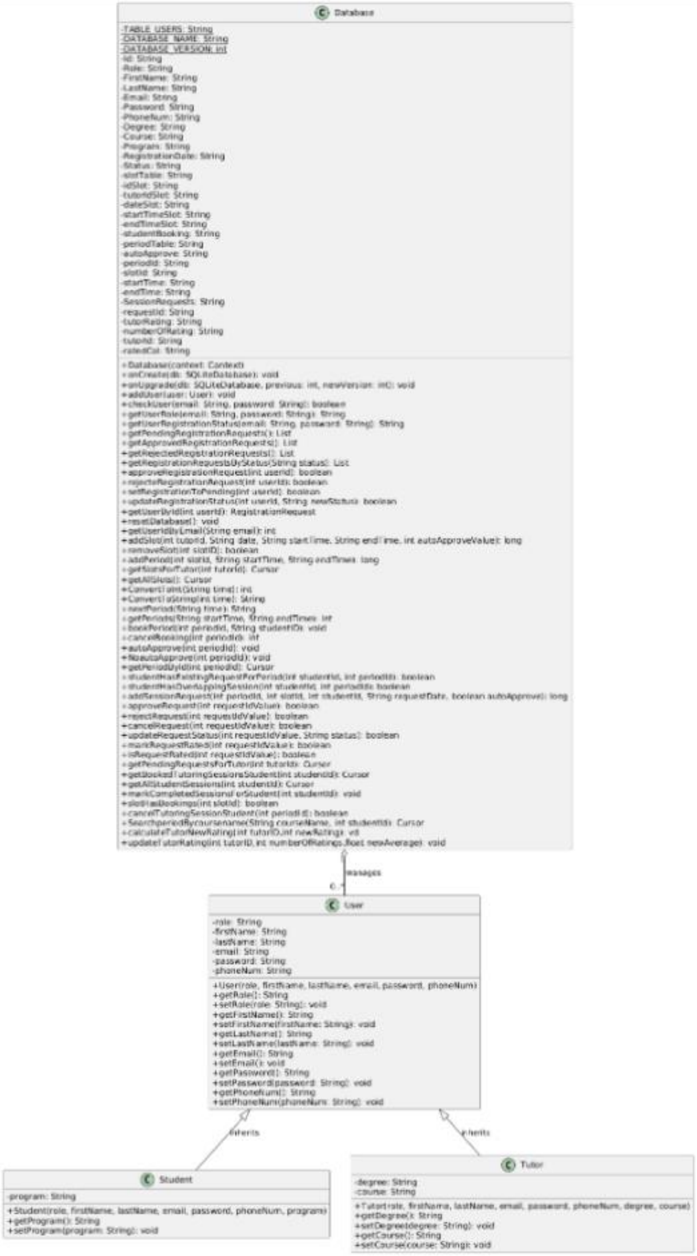
Tutor:

- As a tutor, I want to register for a new account as a tutor with the courses I can offer so that I can become a tutor in the system.
- As a tutor, I want to create new availability time slots with 30 minutes increments so that student can see when I am available.
- As a tutor, I want to accept or refuse a student tutoring session request so that I can plan my next tutoring session.

Student:

- As a student, I want to register for a new account so that I can become a student to access the management system.
- As a student, I want to search available tutoring time slots for a desired class so that I can find a tutor for a desired course.
- As a student, I want to view a tutor name, rating and course offered so that I can choose the best tutor for my needs.
- As a student, I want to make a tutoring session request for an available tutor time slot so that I can reserve a session for a desired course.
- As a student, I want to cancel an accepted tutoring session if it is less than 24 hours in advance so that I can free a time slot if I am unable to attend.
- As a student, I want to rate a tutor following my tutoring session so that I can provide feedback about a tutor.

UML:


Database
-TABLE_USERS: String
-DATABASE_NAME: String
-DATABASE_VERSION: int
-Id: String
-Role: String
-FirstName: String
-LastName: String
-Email: String
-Password: String
-PhoneNum: String
-Degree: String
-Course: String
-Program: String
-RegistrationDate: String
-Status: String
-slotTable: String
-idSlot: String
-tutorIdSlot: String
-dateSlot: String
-startTimeSlot: String
-endTimeSlot: String
-studentBooking: String
-periodTable: String
-autoApprove: String
-periodId: String
-slotId: String
-startTime: String
-endTime: String
-SessionRequests: String
-requestId: String
-tutorRating: String
-numberOfRating: String
-tutorId: String
-ratedCol: String
+Database(context: Context)
+onCreate(db: SQLiteDatabase): void
+onUpgrade(db: SQLiteDatabase, previous: int, newVersion: int): void
+addUser(user: User): void
+checkUser(email: String, password: String): boolean
+getUserRole(email: String, password: String): String
+getUserRegistrationStatus(email: String, password: String): String
+getPendingRegistrationRequests(): List
+getApprovedRegistrationRequests(): List
+getRejectedRegistrationRequests(): List
+getRegistrationRequestsByStatus(String status): List
+approveRegistrationRequest(int userId): boolean
+rejectRegistrationRequest(int userId): boolean
+setRegistrationToPending(int userId): boolean
+updateRegistrationStatus(int userId, String newStatus): boolean
+getUserById(int userId): RegistrationRequest
+resetDatabase(): void
+getUserIdByEmail(String email): int
+addSlot(int tutorId, String date, String startTime, String endTime, int autoApproveValue): long
+removeSlot(int slotID): boolean
+addPeriod(int slotId, String startTime, String endTime): long
+getSlotsForTutor(int tutorId): Cursor
+getAllSlots(): Cursor
+ConvertToInt(String time): int
+ConvertToString(int time): String
+newPeriod(String time): String
+getPeriods(String startTime, String endTime): int
+bookPeriod(int periodId, String studentID): void
+cancelBooking(int periodId): int
+autoApprove(int periodId): void
+NoautoApprove(int periodId): void
+getPeriodById(int periodId): Cursor
+studentHasExistingRequestForPeriod(int studentId, int periodId): boolean
+studentHasOverlappingSession(int studentId, int periodId): boolean
+addSessionRequest(int periodId, int slotId, int studentId, String requestDate, boolean autoApprove): long
+approveRequest(int requestIdValue): boolean
+rejectRequest(int requestIdValue): boolean
+cancelRequest(int requestIdValue): boolean
+updateRequestStatus(int requestIdValue, String status): boolean
+markRequestRated(int requestIdValue): boolean
+isRequestRated(int requestIdValue): boolean
+getPendingRequestsForTutor(int tutorId): Cursor
+getBookedTutoringSessionsStudent(int studentId): Cursor
+getAllStudentSessions(int studentId): Cursor
+markCompletedSessionsForStudent(int studentId): void
+slotHasBookings(int slotId): boolean
+cancelTutoringSessionStudent(int periodId): boolean
+SearchperiodBycoursename(String courseName, int studentId): Cursor
+calculateTutorNewRating(int tutorID,int newRating): vd
+updateTutorRating(int tutorID,int numberOfRatings,float newAverage): void
manages
0..*
User
-role: String
-firstName: String
-lastName: String
-email: String
-password: String
-phoneNum: String
+User(role, firstName, lastName, email, password, phoneNum)
+getRole(): String
+setRole(role: String): void
+getFirstName(): String
+setFirstName(firstName: String): void
+getLastName(): String
+setLastName(lastName: String): void
+getEmail(): String
+setEmail(): void
+getPassword(): String
+setPassword(password: String): void
+getPhoneNum(): String
+setPhoneNum(phoneNum: String): void
inherits
inherits
Student
-program: String
+Student(role, firstName, lastName, email, password, phoneNum, program)
+getProgram(): String
+setProgram(program: String): void
Tutor
-degree: String
-course: String
+Tutor(role, firstName, lastName, email, password, phoneNum, degree, course)
+getDegree(): String
+setDegree(degree: String): void
+getCourse(): String
+setCourse(course: String): void

Contributions

	Deliverable 1	Deliverable 2	Deliverable 3	Deliverable 4
Colton Ventry	Added assortment of input verifications including email. Added error messages and fixed placement of messages depending on the problem field. Created demo video.	Unsuccessfully attempted email notifications.	Began implementing features to time slot creation including back and auto-accept. Implemented time slot overlap checks and the associated error messages.	Fixed search system to find time slots as a student.
Alexandre Turgeon	Add UML domain diagram	Fix admin filtering and add login feedback	Add AvailabilitySlot model with validation helpers (D3 backend, deprecated on branch) Each tutor now only sees and manages their own availability slots. The DB enforces uniqueness per tutor but same date/time can exist for different tutors. Adds a	Student cannot book overlapping sessions same day, Student overlap/duplicate checks, search hides requested slots, 24h cancel rule, rating storage/average update, completed-tagging, slot delete guard, “No upcoming sessions” empty state, Rating System for tutor.

			tutor specific availability slots	3 XML pages
Reda TALA-IGHIL	Made the student and tutor registering page UI, and the buttons to switch between the two pages put the admin credentials into the database	The administrator can approve an already rejected request and added comments all over the code	Added a period table, which has intervals of 30min added functions in the database class for later use implemented the auto approve slider into the database added a session requests table into the database made the function to add a session request made the UI page to show student requests and accept or reject , and implement that in the backend (database)	Added a ui for searching a session by coursename (and the backend too), added a book button by the side of each result, made the session information showup too added tutor rating (a columns on the database and functions to calculate it and set a new value to it) improved the ui style on alot of pages, and added alot of utility functions in the database also fixed some issues from the previous commit to make it work with my new implementation

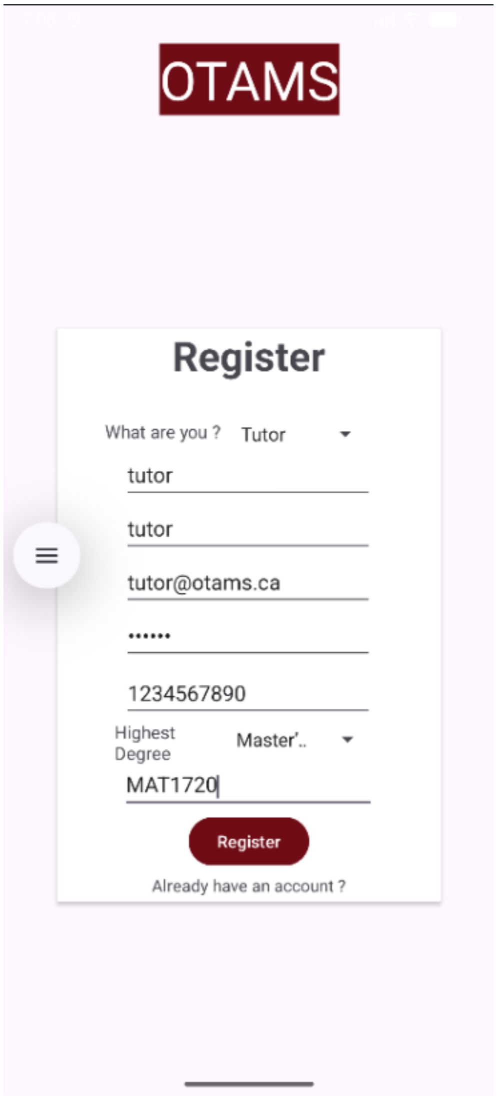
Kevin Huang	Creating the database in SQLite, and variable class of User, Student, and Tutor. Implementing proper register and login function in database where the system will remeber the user's inputted info, along with email and password in a unique account, so that the user can login back in without need to create a new account.	Proof-tested project and created the demo video of showcasing all functions	Created the ui for Tutor Console, where the user can click on Create button, which will open a time slot creating page, in which the user can choose a date using Calendar function, and input start and end time. Both have multiple error check on formatting and logics, such as that the time interval has to be in 30min. Inputted data will be stored as a column in a database table to be implemented onto the Tutor Console Linear Layout section as a single time slot box	Testing functionality of the project and created the demo video
Yassine Yandouzi	Made the welcome page screen following a registration waiting	UML, Implemented the AdminActivity, admin can approve or reject a registration, following the	UML, added a log off button for the tutor console, updated the ReadMe.md file	UML, created a student console, the student can see a list of future and past tutoring

	for the admin approval.	approval of a registration a user can login their account, the admin has access to a list of all the accepted and declined user's registrations and updated part of the database		session, added a log off button on the Student Console, a student can cancel a request if the session is in less than 24 hours, updated the database for canceled tutoring sessions and the whole final report

*Every team member has reviewed and agreed upon the contribution as detailed in the contribution table.

Signature: Yassine Yandouzi, Reda TALA IGHIL, Alexandre Turgeon, Colton Ventry, Kevin Huang

Screenshots of the Application



Register

What are you ? Tutor

tutor

tutor

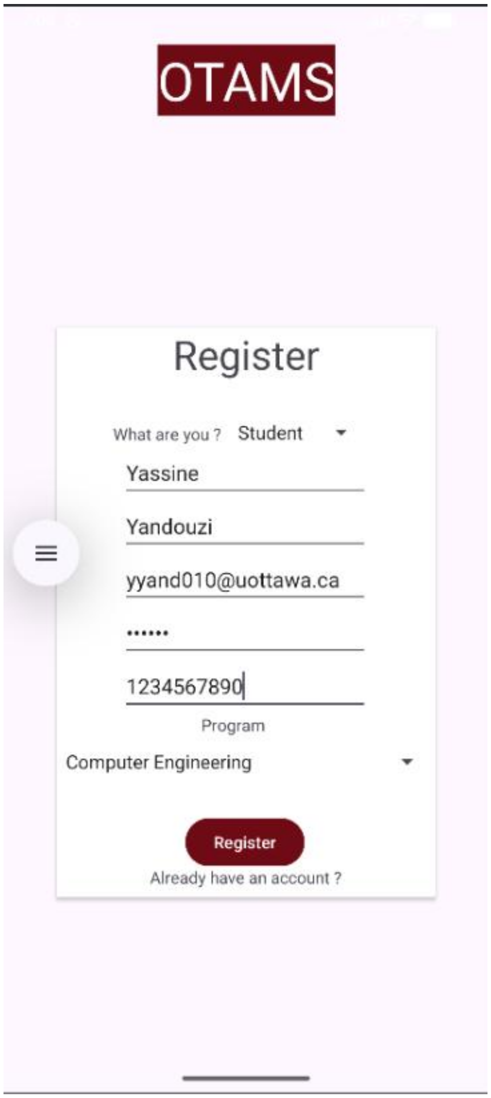
tutor@otams.ca

••••••

1234567890

Highest Degree Master'..

MAT1720



Register

What are you ? Student

Yassine

Yandouzi

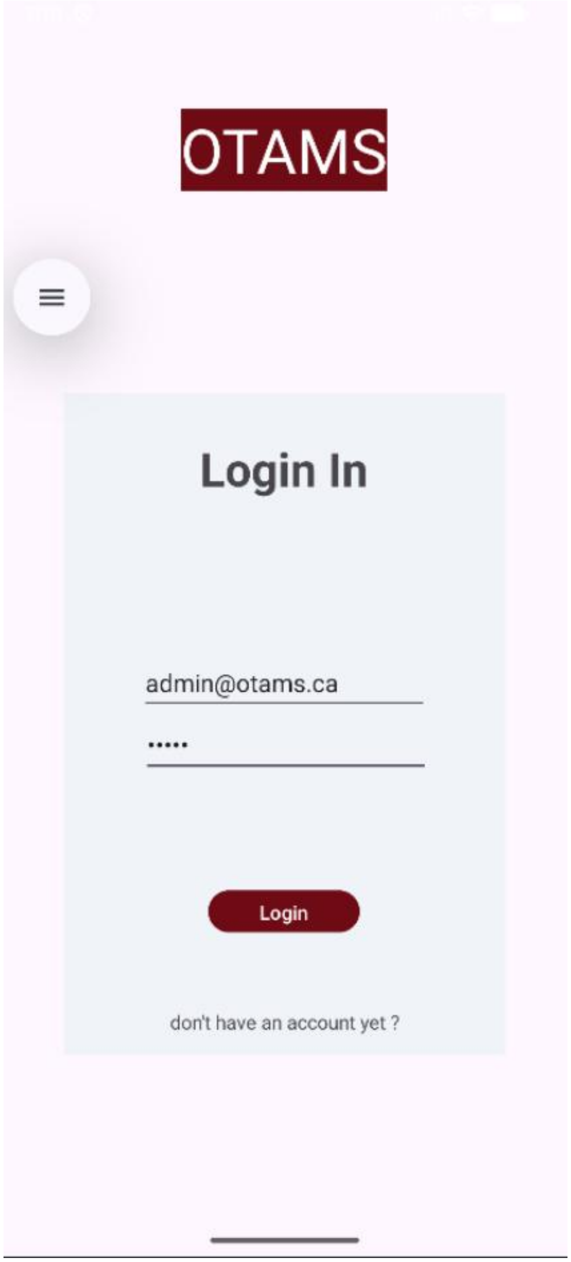
yyand010@uottawa.ca

••••••

1234567890

Program

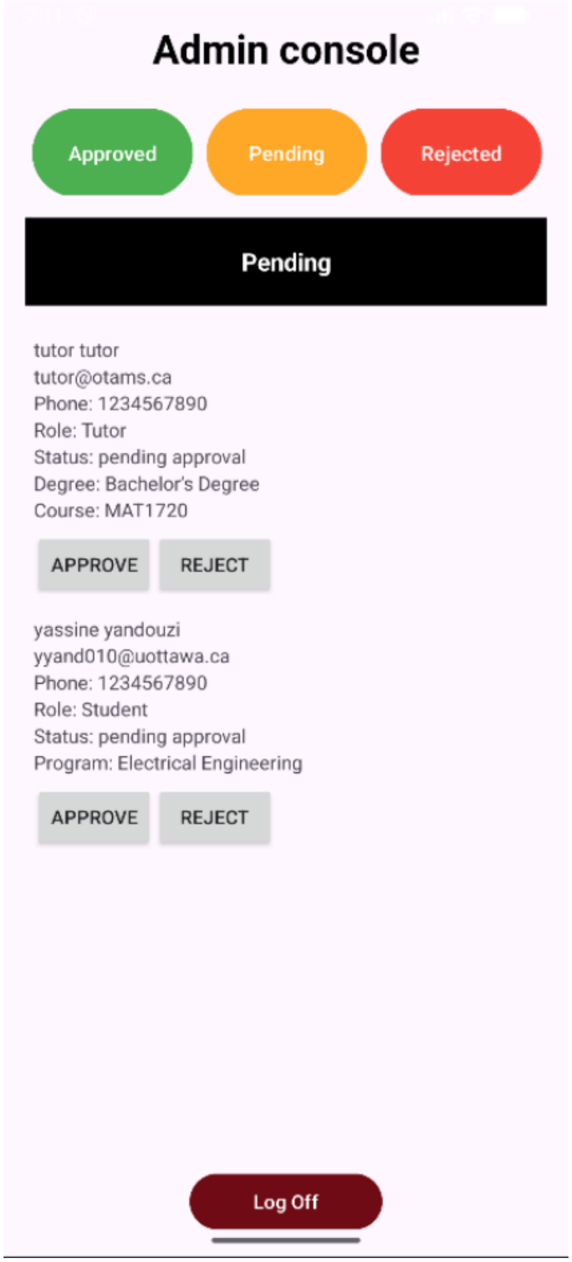
Computer Engineering



Login In

admin@otams.ca

•••••

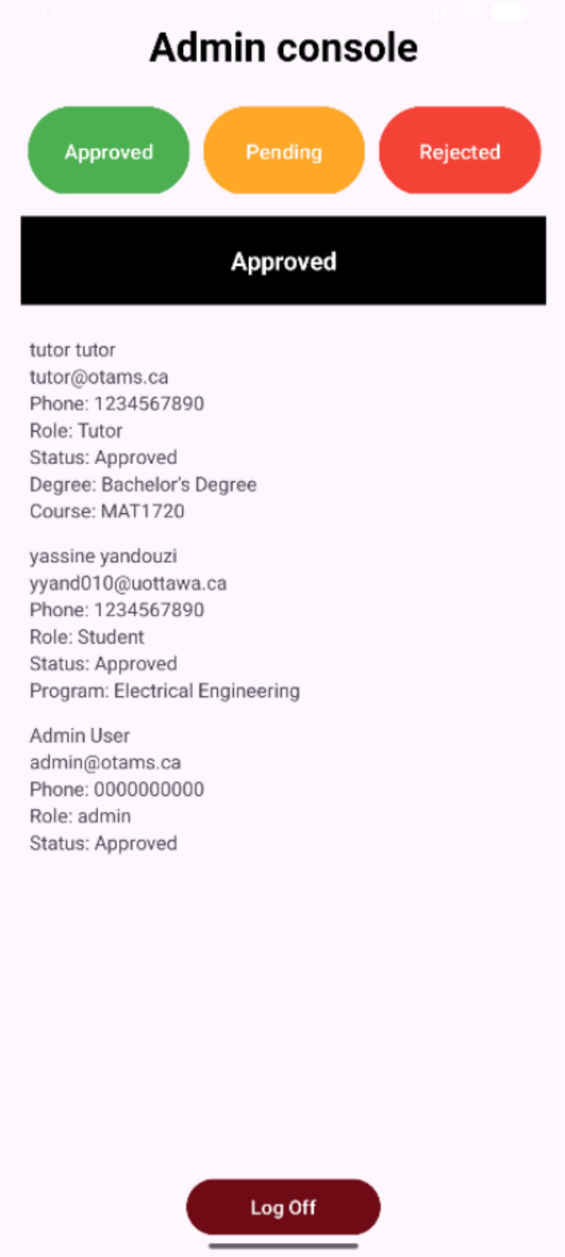


Admin console

Pending

tutor tutor
tutor@otams.ca
Phone: 1234567890
Role: Tutor
Status: pending approval
Degree: Bachelor's Degree
Course: MAT1720

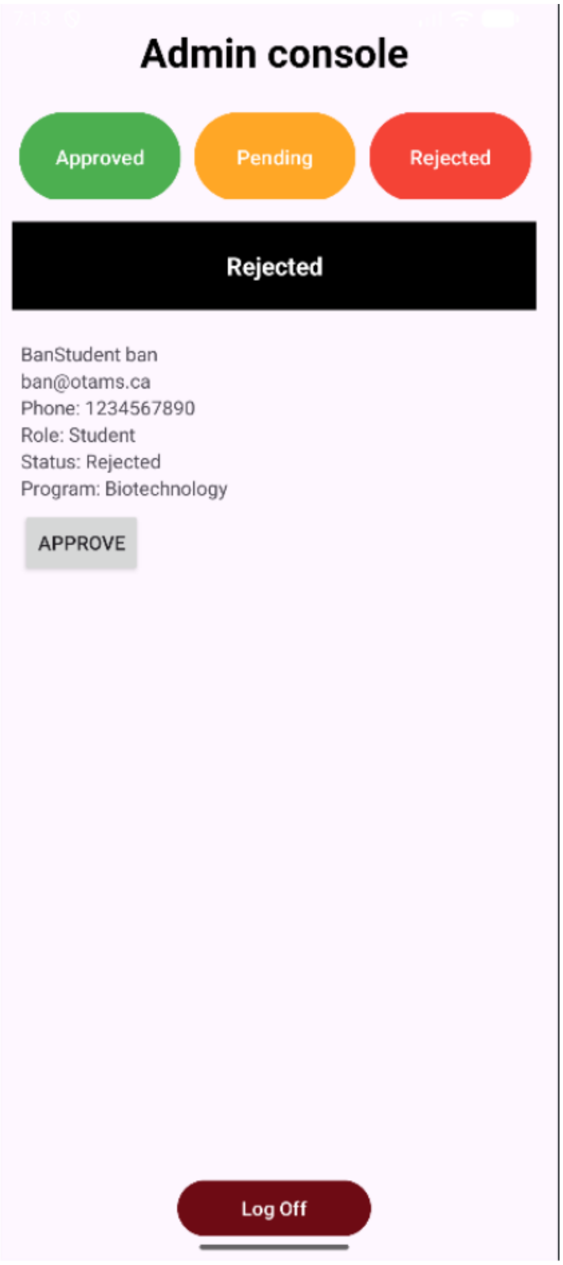
yassine yandouzi
yyand010@uottawa.ca
Phone: 1234567890
Role: Student
Status: pending approval
Program: Electrical Engineering



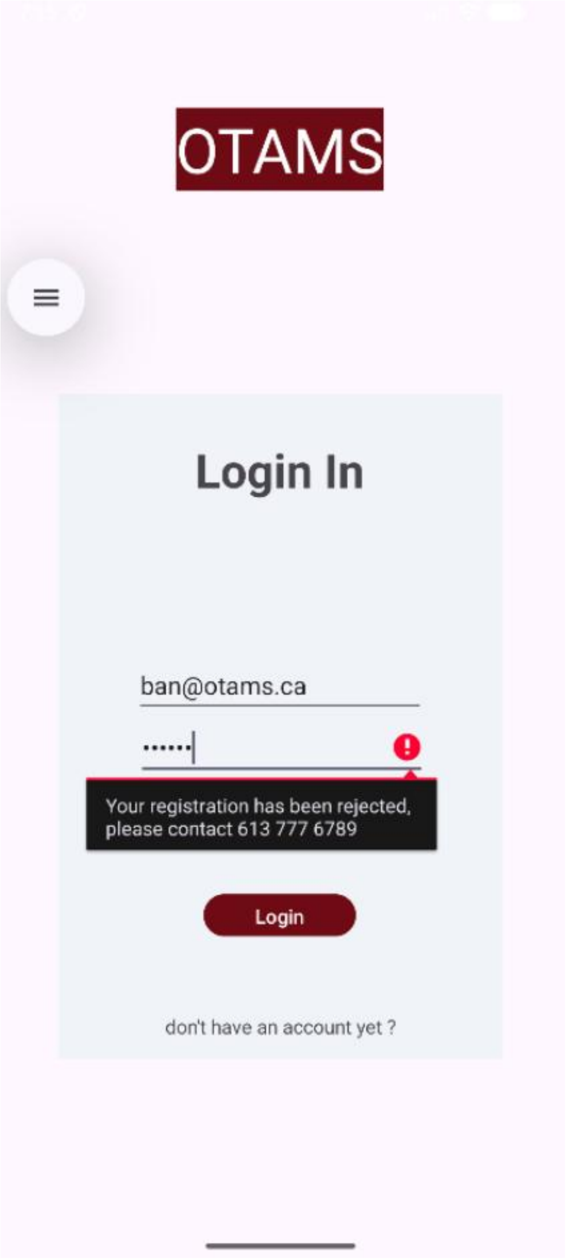
tutor tutor
tutor@otams.ca
Phone: 1234567890
Role: Tutor
Status: Approved
Degree: Bachelor's Degree
Course: MAT1720

yassine yandouzi
yyand010@uottawa.ca
Phone: 1234567890
Role: Student
Status: Approved
Program: Electrical Engineering

Admin User
admin@otams.ca
Phone: 0000000000
Role: admin
Status: Approved



BanStudent ban
ban@otams.ca
Phone: 1234567890
Role: Student
Status: Rejected
Program: Biotechnology



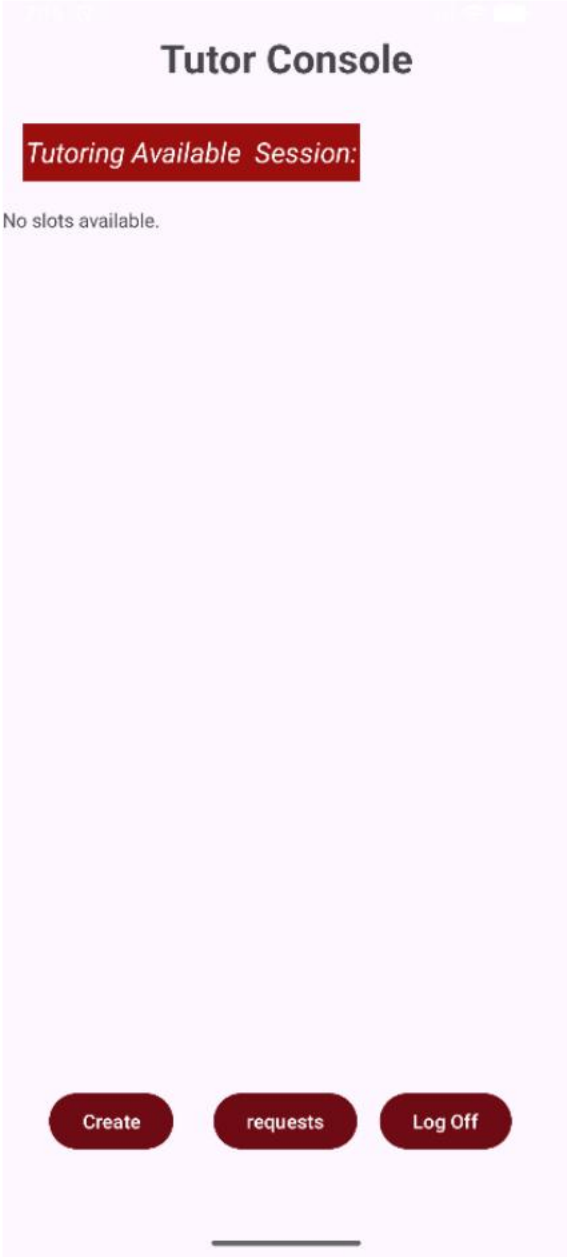
Login In

ban@otams.ca

••••••

Your registration has been rejected, please contact 613 777 6789

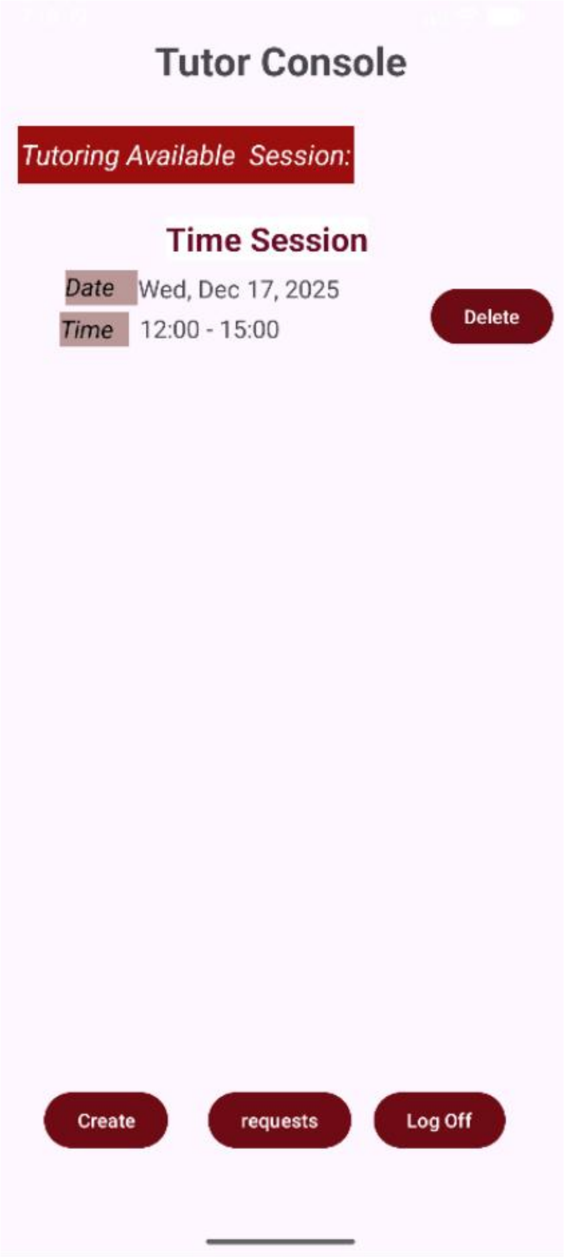
don't have an account yet ?



Tutor Console

Tutoring Available Session:

No slots available.



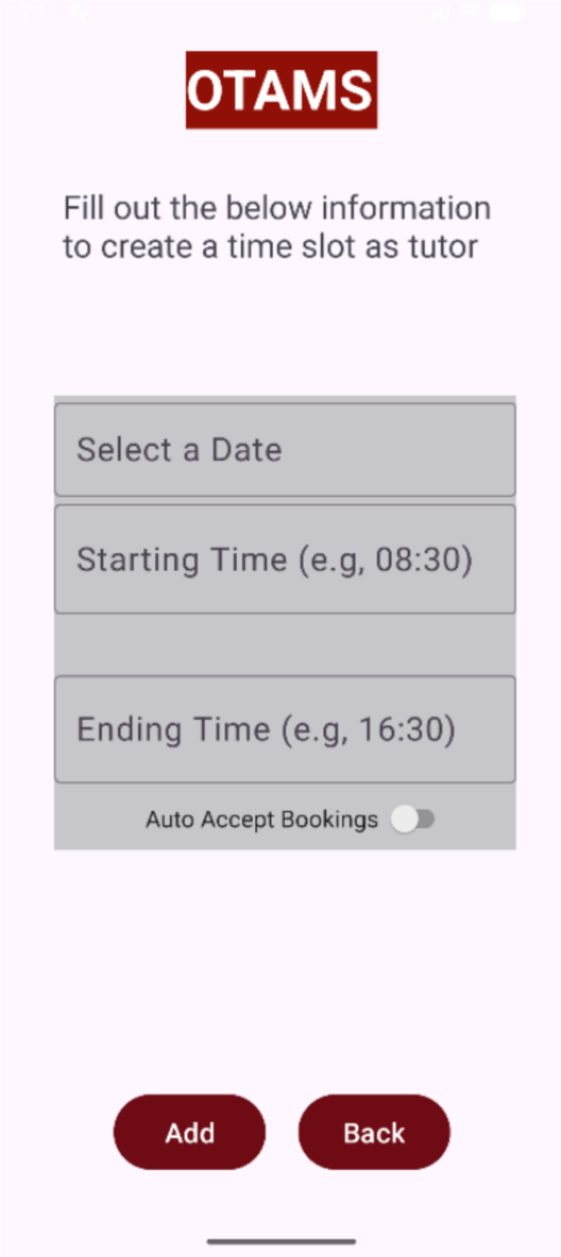
Tutor Console

Tutoring Available Session:

Time Session

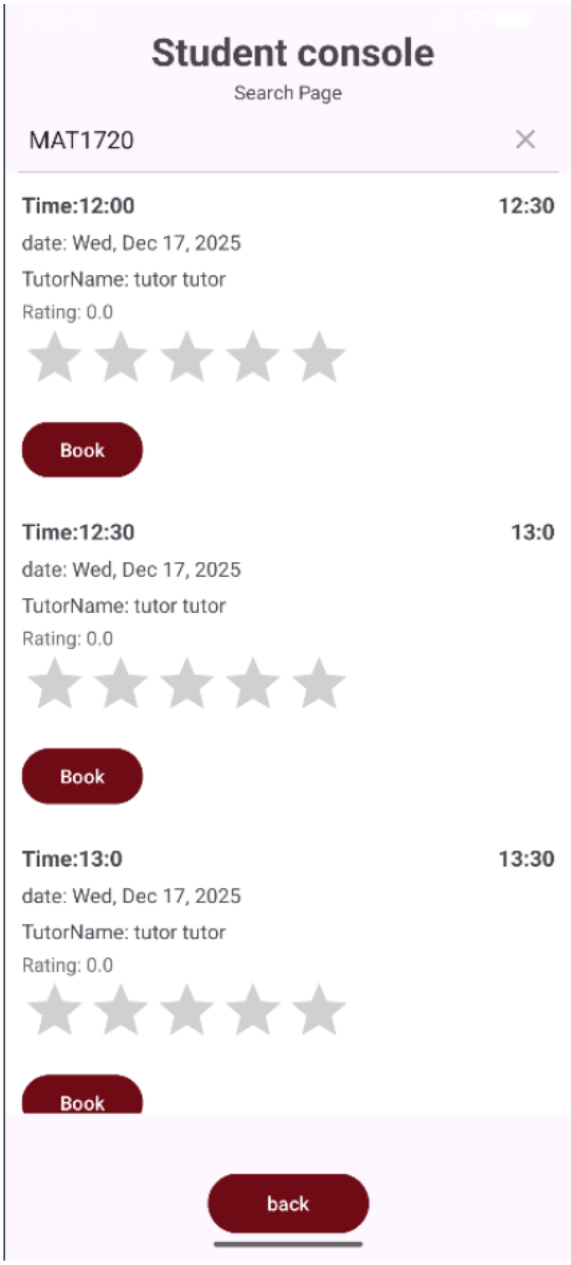
Date Wed, Dec 17, 2025

Time 12:00 - 15:00



OTAMS

Fill out the below information to create a time slot as tutor



Student console

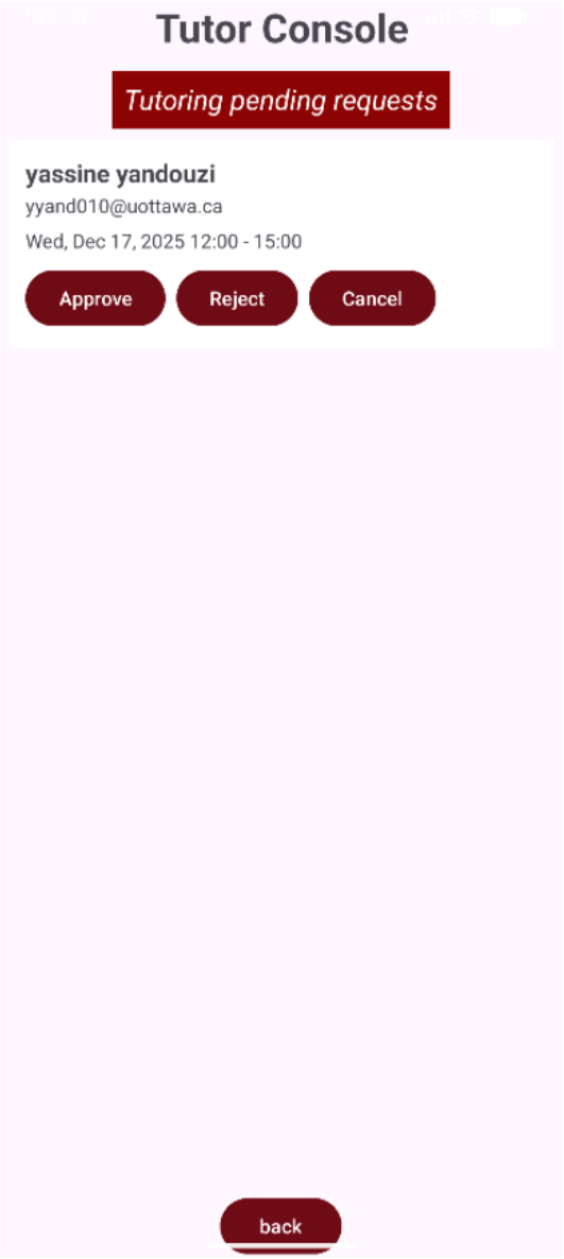
Search Page

MAT1720

Time:12:00 12:30
date: Wed, Dec 17, 2025
TutorName: tutor tutor
Rating: 0.0

Time:12:30 13:0
date: Wed, Dec 17, 2025
TutorName: tutor tutor
Rating: 0.0

Time:13:0 13:30
date: Wed, Dec 17, 2025
TutorName: tutor tutor
Rating: 0.0



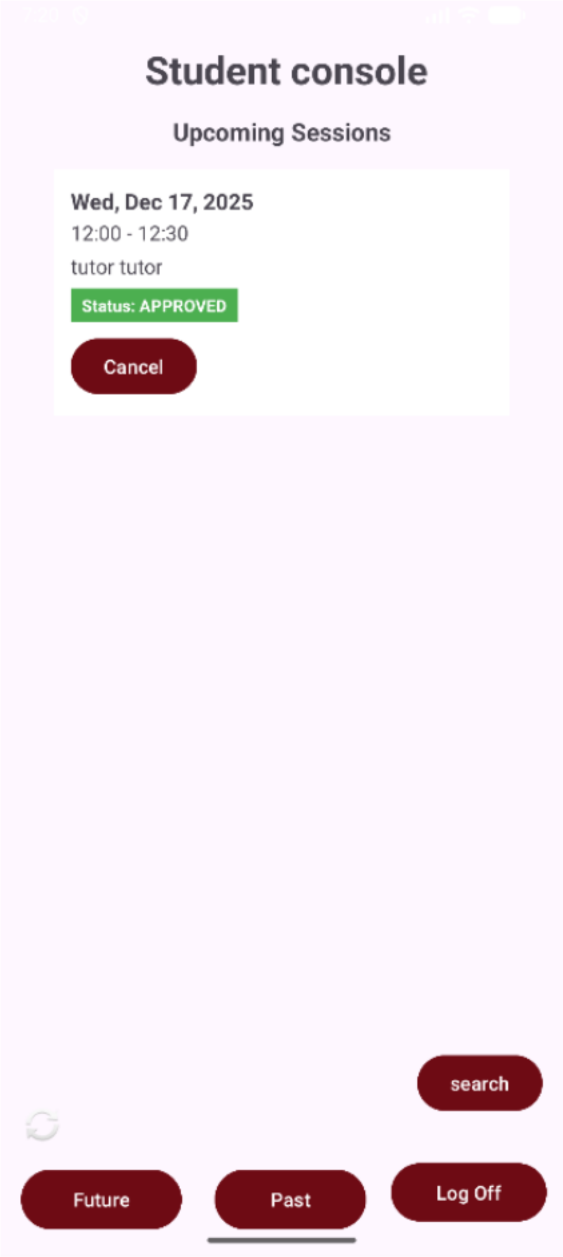
Tutor Console

Tutoring pending requests

yassine yandouzi

yyand010@uottawa.ca

Wed, Dec 17, 2025 12:00 - 15:00



Student console

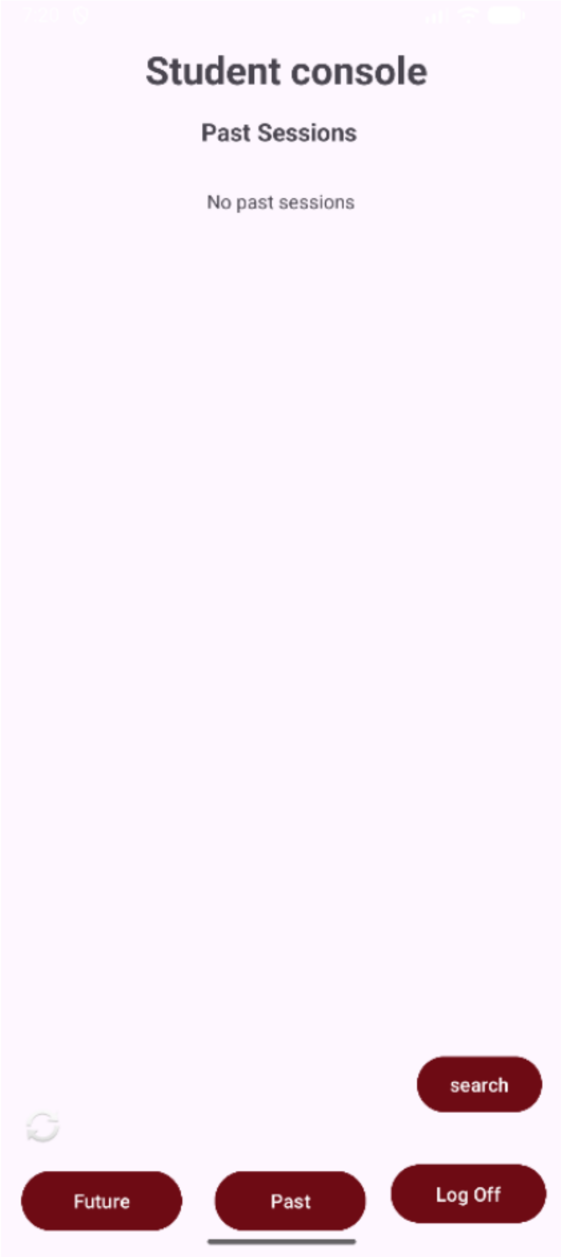
Upcoming Sessions

Wed, Dec 17, 2025

12:00 - 12:30

tutor tutor

Status: APPROVED



Student console

Past Sessions

No past sessions

Lesson Learned

- Using a shared GitHub taught us the critical importance of version control. We learned the importance of managing branches, solving merge issues and the importance of detailed commits.

- This project taught us the importance of time management and good team communication. Working on this project with 4 fixed deadlines, we learned the importance of properly planning our work in advance by prioritizing important feature to reach fixed deadlines.

- This project taught us how to analyse, understand and implement requirements into functional code. This required us to understand and break down into smaller piece a requirement for a better integration.

- By sectioning this project in 4 deliverables, we learned the importance of breaking down complex demands in smaller task to implement and test each feature effectively using agile requirements.

- Using a database for this project allowed us to learn and understand the functioning and implementation of a database for a real time application using SQLite. We learned how to implement a database managing different state (pending, approved, rejected) and its behaviour.

- This project taught us how to create an app from a user perspective. This was particularly important for appropriate error message and a clear user interface. This allows the user to have a better understanding of the app by receiving clear instruction and error messages.

Conclusion

In conclusion, the development of the Android Online Tutoring Appointment Management System (OTAMS) successfully demonstrated the practical use of a shared GitHub repository, Agile principal, UML diagram theory, java programming, database implementation, UI design and the creation of a functional android application. This application allows student to seek tutoring session to competent tutor for a desired class upon administrator approval to the application for a user. This project resulted in a fully functional management system for three different types of users and help gain the team essential skills for software development: time management, respecting deadlines, version control, database implementation, UI design, team collaboration, agile requirements and java programming. We are proud to deliver a functional, user solution that effectively addresses the tutoring needs of the University of Ottawa student community.