CSC 431

UTutor Me

System Architecture Specification (SAS)

Team 01

Rayan N Excellent	Software Designer/ Developer
Dido Franceschi	Developer
Pavan Gudoor	Requirements Engineer/ Developer
Christopher Wu	Scrum Master/ Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	3/28/22	CW, RE, PG, DF	First write-up
2.0	3/31/22	CW, RE, PG, DF	First submission
2.1	4/19/22	CW, RE, PG, DF	Changes to Class Diagram (constraints to Account Class, new getter and setter functions)

Table of Contents

Table of Tables

41. System Analysis

41.1 System Overview

A mobile application will serve as a platform to enable our entire system.

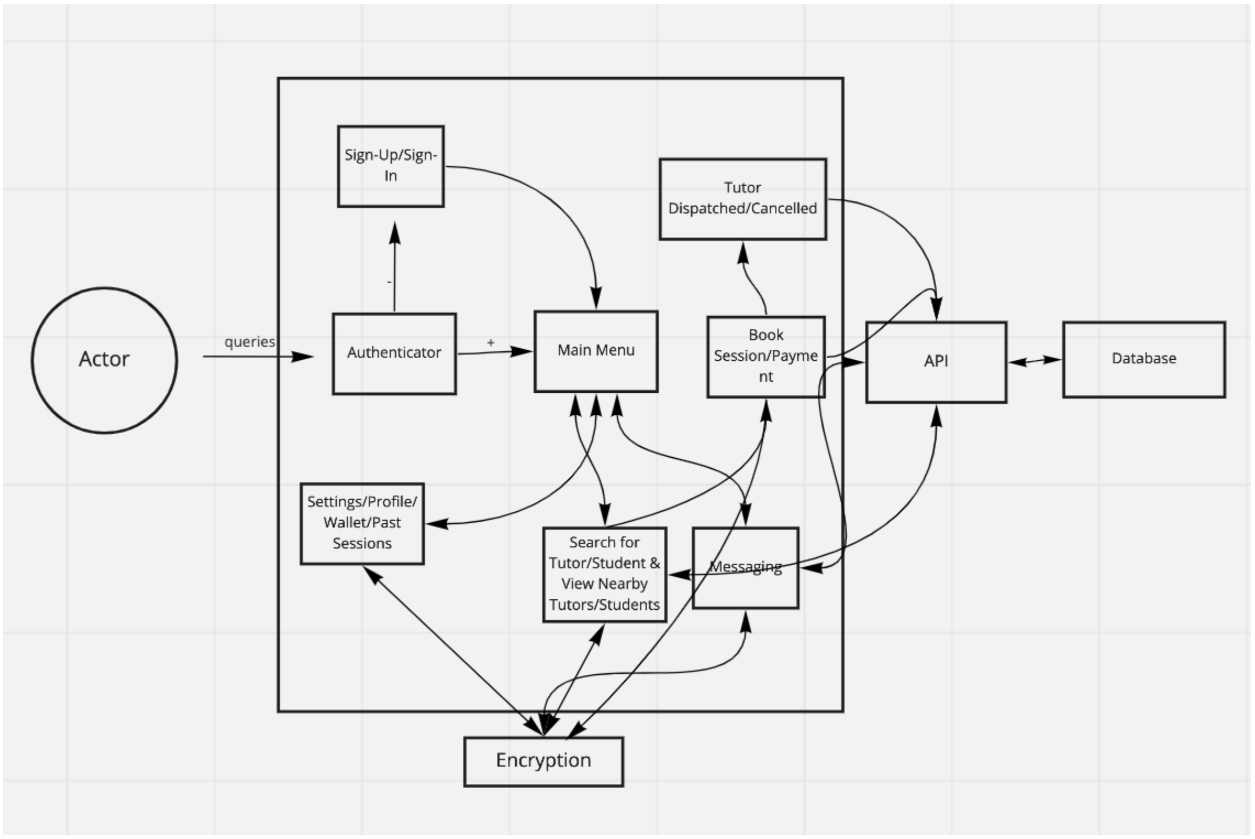
To use the app, a student first opens the app, entering details such as course selection and material and nearby tutors. The student is then matched with a tutor, either accepting or rejecting the request. If accepted, the session and payment is verified, and the student and tutor will meet at a certain location. Once the session is over, students are able to leave reviews for other students to view.

The database, employed through Spring Boot, will serve to store the necessary information such as user data, sessions, reviews, and bookings. By also incorporating the Google Maps API to track and display nearby students/tutors, each component will interact with each other in a way that provides all the necessary functionalities of the application, that is, searching for nearby tutors, requesting tutor service, viewing tutor ratings, and payment methods.

Handling tutor requests and student selection requires both information to be stored on the database. The request is processed via the student's geolocation, and lastly, the information will return the tutors present nearby in the database.

To determine student/ tutor estimated time of arrival (ETA), an algorithm to determine the shortest path will be implemented.

41.2 System Diagram


Actor
queries
Authenticator
-
+
Sign-Up/Sign-In
Main Menu
Tutor Dispatched/Cancelled
Book Session/Payment
API
Database
Settings/Profile/Wallet/Past Sessions
Search for Tutor/Student & View Nearby Tutors/Students
Messaging
Encryption

41.3 Actor Identification

There are four actors in our system:

- New Users: Users (including both students and tutors) that are new to UTutorMe or haven't created an account.
- Registered Students: Users that have registered a student account.
- Registered Tutors: Users that have registered a tutor account.
- The database itself and the api that accesses our database that contains all our information/user

41.4 Design Rationale

1.4.1 Architectural Style

UTutorMe will employ a 3-tier architecture. The 3-tiers consist of the UI layer, business logic layer, and the service layer.

1. For the UI layer, we will be using React Native, as it easily provides support for different operating systems.
2. For the business logic layer, this will enable students to book a tutor as well as allow tutors to accept student requests, which will all be handled by a panel that will receive both requests and manage the operations accordingly.
3. For the service layer, we will use Spring Boot for storing tutor/student information, descriptions, and sessions.

1.4.2 Design Pattern(s)

UTutorMe will employ the “facade” structural design pattern and the “strategy” behavioral design pattern.

“Facade” hides the complexities of our system to the user by providing an interface to the user that enables the user to easily access the complexities of our system. We will have numerous algorithms and functions running in the backend that contribute to a potentially complicated service, when we want to emphasize efficiency and accessibility. Thus, “facade” was a natural choice for our structural design.

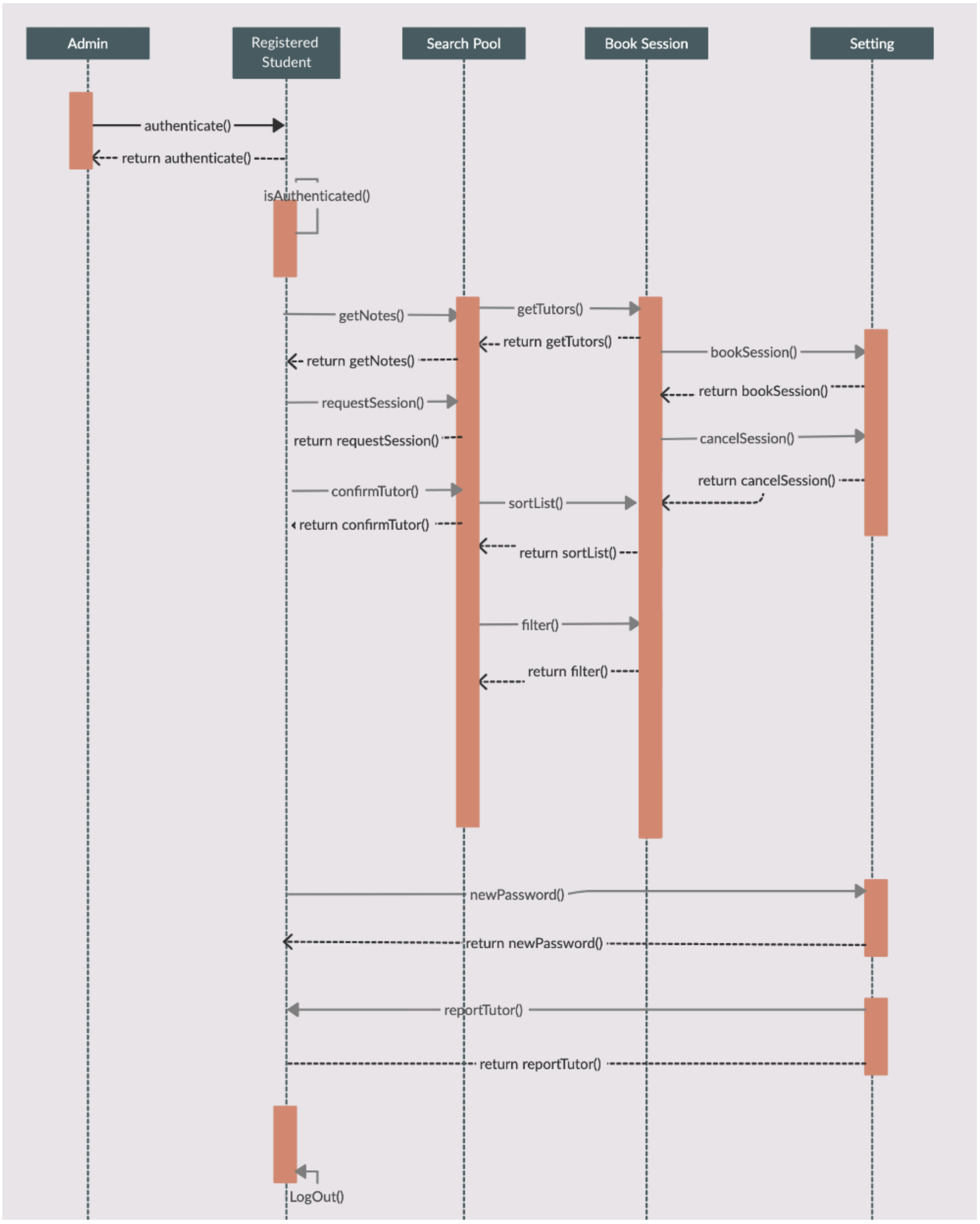
“Strategy” enables the code behind UTutorMe to respond to run-time instructions on using one of a family of algorithms rather than a single, pre-set algorithm that will limit the functionality of our program. A useful example of this will be ETA regarding a tutor meeting a student or vice versa, which could depend on the tutor having a car, walking, public transportation, etc. “Strategy” allows us to adapt to the user’s run-time needs, and allows us to have software that is loosely coupled.

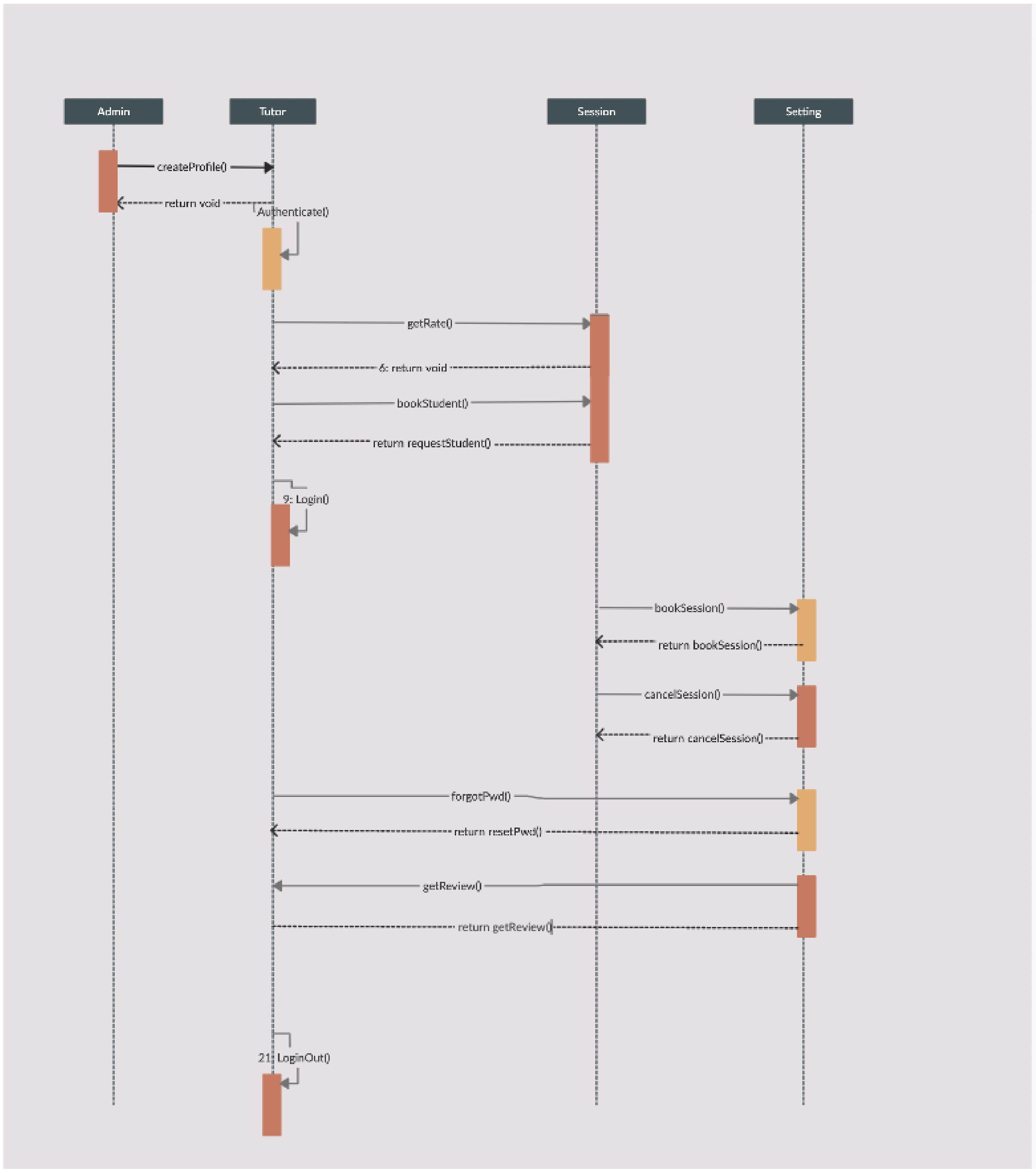
1.4.3 Framework

UTutorMe will use the following stack/framework: Spring (composed of Java, Hibernate, and MySQL/PostgreSQL). UTutorMe will be primarily a mobile application upon which our company will employ a majority of our services, but we will also have a website with some functionality as well. Spring is therefore a useful selection in that its java application has extensions for building both web applications and allowing usage in android environments. It also promotes inversion of control and dependency injection, so that code becomes easier to test and maintain. Spring is lightweight, easty to test, enables powerful abstraction which will facilitate the “facade” design pattern, allows for loose coupling, and contributes to fast development, which will be very useful considering our algorithms run in real-time and quick changes will inevitably have to be made if anything suddenly ever arises.

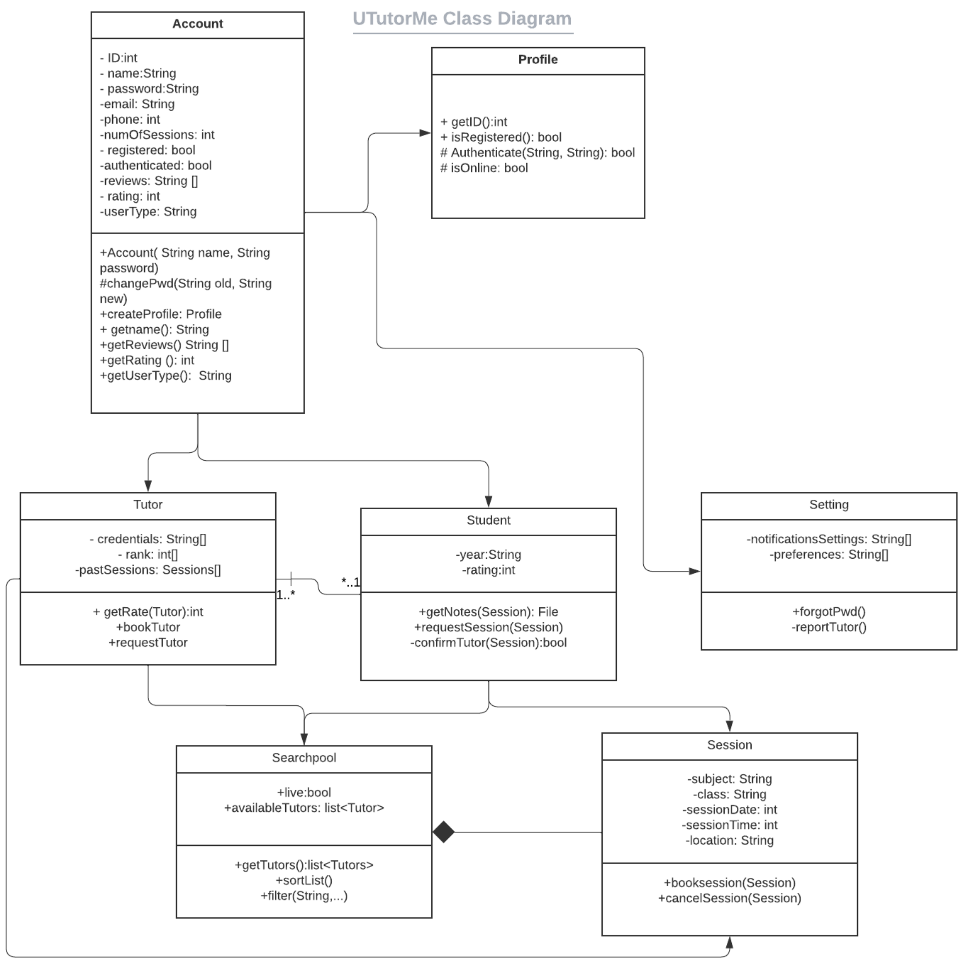
In the future, UTutorMe hopes to grow our tech stack as we scale up and incorporate other enterprises/utilties including Elasticsearch, Paypal, etc.

42. Functional Design




Admin
Tutor
Session
Setting
createProfile()
return void
Authenticate()
getRate()
6: return void
bookStudent()
return requestStudent()
9: Login()
bookSession()
return bookSession()
cancelSession()
return cancelSession()
forgotPwd()
return resetPwd()
getReview()
return getReview()
21: LoginOut()

43. Structural Design


UTutorMe Class Diagram
Account
- ID:int
- name:String
- password:String
-email: String
-phone: int
-numOfSessions: int
- registered: bool
-authenticated: bool
-reviews: String []
- rating: int
-userType: String
+Account(String name, String password)
#changePwd(String old, String new)
+createProfile: Profile
+ getname(): String
+getReviews() String []
+getRating (): int
+getUserType(): String
Profile
+ getID():int
+ isRegistered(): bool
# Authenticate(String, String): bool
# isOnline: bool
Tutor
- credentials: String[]
- rank: int[]
-pastSessions: Sessions[]
+ getRate(Tutor):int
+bookTutor
+requestTutor
1..*
*..1
Student
-year:String
-rating:int
+getNotes(Session): File
+requestSession(Session)
-confirmTutor(Session):bool
Setting
-notificationsSettings: String[]
-preferences: String[]
+forgotPwd()
-reportTutor()
Searchpool
+live:bool
+availableTutors: list<Tutor>
+getTutors():list<Tutors>
+sortList()
+filter(String,...)
Session
-subject: String
-class: String
-sessionDate: int
-sessionTime: int
-location: String
+booksession(Session)
+cancelSession(Session)