Levin GmbH

Your E-Care
Software Architecture Document

Version 1.2

Revision History

Date	Version	Description	Author
05/12/22	1.0	Add section 1 and 2	Nguyen Minh Van
09/12/22	1.1	Add section 3 and 4	Everyone
25/12/22	1.2	Add section 5 and 6	Nguyen Minh Van, Ha Tuan Lam

Table of Contents

Software Architecture Document

1. Introduction

The purpose of this document is to provide a detailed architecture design of the Your E-care System by focusing on the model of the program and all its packages. These attributes were chosen based on their importance in the design and construction of the application.

This document will address the background for this project, and the architecturally significant functional requirements. Each package mentioned will be described through a comprehensive set of class diagrams followed by an architectural overview, which includes a bird's eye view and a complete description of patterns that will be used to address the core quality attributes. This will be followed up by a look at a couple of views into the system.

This document intends to help the development team determine how the system will be structured at the highest level. It is also intended for the project sponsors to sign off on the tall level structure before the team shifts into detailed design. Finally, the project coach can use this document to validate that the development team is meeting the agreed upon requirements during his evaluation of the team's efforts.

Term list:

- Stakeholder: any person involved or affected, directly or indirectly, by this product.
- Javascript: (originally) web-browser interpreted programming language for enhancing websites in a dynamic way.
- React: a free and open-source front-end JavaScript library for building user interfaces based on UI components. Meta and a community of individual developers and companies maintain it.
- React Native: an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web and Windows by enabling developers to use the React framework along with native platform capabilities.

Acronym list:

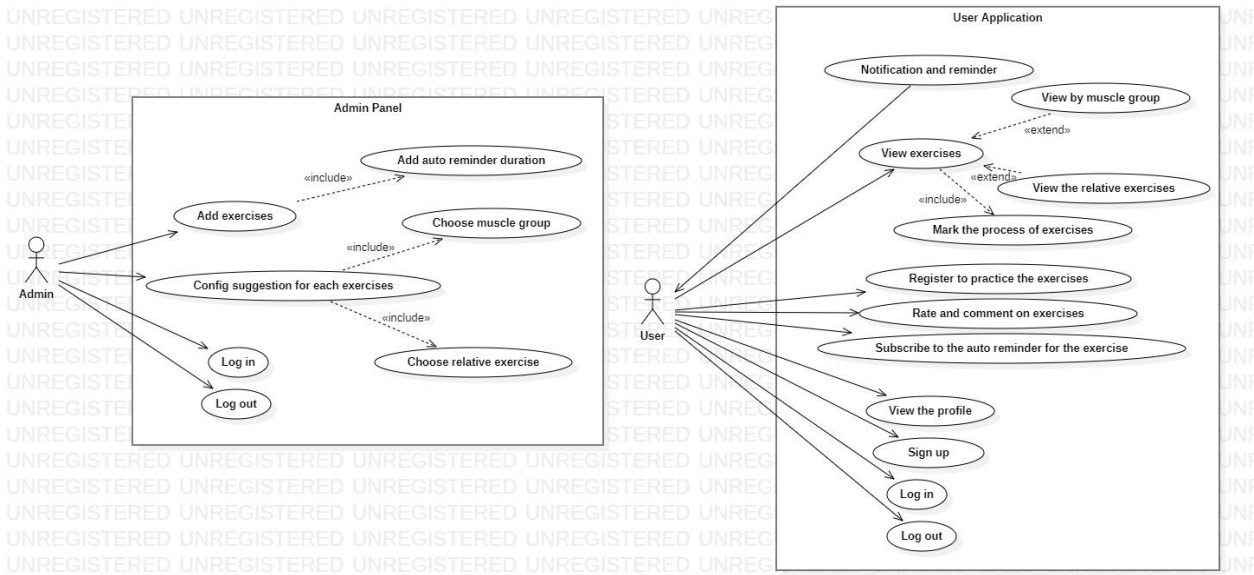
- MVC: Model - View - Controller model
- SAD: Software Architecture Document
- UI: User Interface
- API: Application Programming Interface, a protocol used as an interface to allow communication between different components.
- REST: Representational State Transfer, web API featuring a stateless client-server infrastructure.

References:

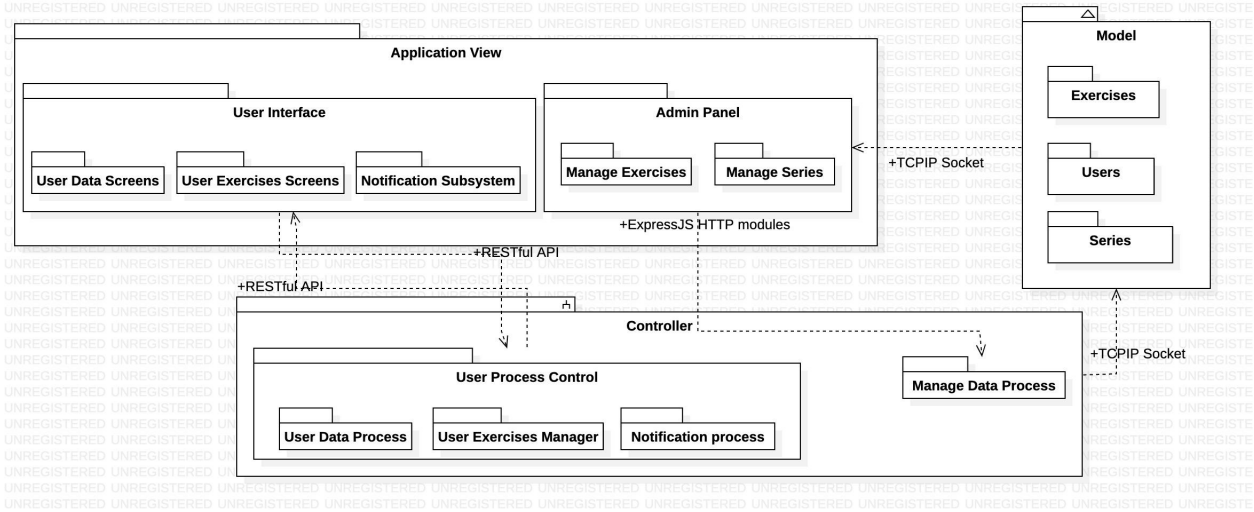
2. Architectural Goals and Constraints

- Application Programming Language: React Native (to run for both Android and iOS)
- Server Programming Language: NodeJS (for real-time running and easy to handle RESTful API)
- Database Environment: MongoDB (this is NoSQL Database, for large scale and real-time applications)
- Emulator System Requirements:
    - Android: 10.0 or newer.
    - iOS: 14.0 or newer
- Security Requirements: Use hash for password so that the user profile is protected.

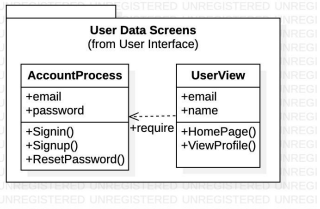
3. Use-Case Model



4. Logical View



4.1 Package: User Data Screens



Class: AccountProcess

Attributes in this class:

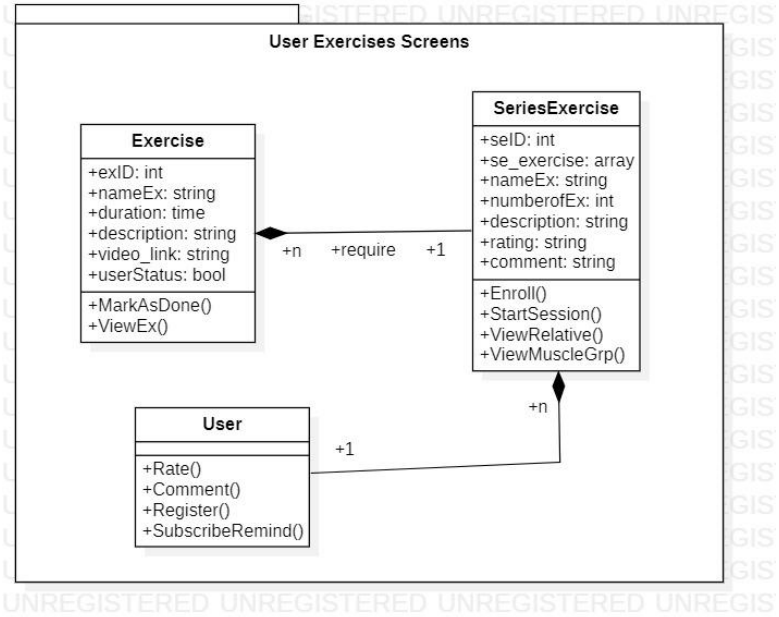
+ email: the email that the user typed in
+ password: the password that the user typed in

Operations in this class:

- Signin(): Process the login step for the user, require email and password typed correctly.
- Signup(): Create a new account for a new user.
- ResetPassword(): Reset the new password for the user, require email correctly to an account in the

database.

4.2 Package: User Exercises Screens



Class: Exercise

Attributes in this class:

+ exID: int
+ nameEx: string
+ duration: time
+ description: string
+ video link: string

Operations in this class:

+ MarkAsDone(nameEx, userStatus): Process which users mark the exercise done.
+ ViewEx(nameEx, duration, description, video_link, userStatus): Process which users can view the exercises.

Class: User

Operations in this class:

+ Rate(seID): Process in which users rate the exercise.
+ Comment(seID): Process in which users comment on the exercise.
+ Register(seID): Process in which users register the exercise.
+ SubscribeRemind(seID): Process in which users subscribe to remind them to take exercise.

Class: SeriesExercise

Attributes in this class:

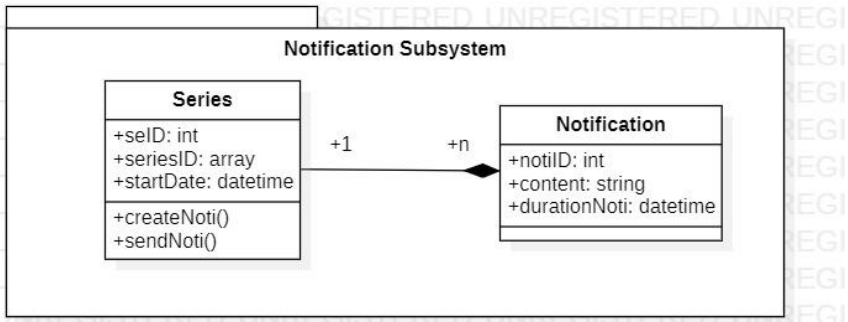
+ seID: int
+ se_exercise: array
+ nameEx: string
+ numberofEx: int
+ description: string
+ rating: string
+ comment: string

Operations in this class:

+ Enroll(seID): Process which users enroll to the course.

+ StartSession(seID, userStatus): Process which users start the session.
+ ViewRelative(seID): Process which users view the relative exercises.
+ ViewMuscleGrp(seID): Process which users view the muscle group exercises.

4.3 Package: Notification SubSystem



Class: Series

Attributes in this class:

+ seID: int
+ seriesID: array
+ startDate: datetime

Operations in this class:

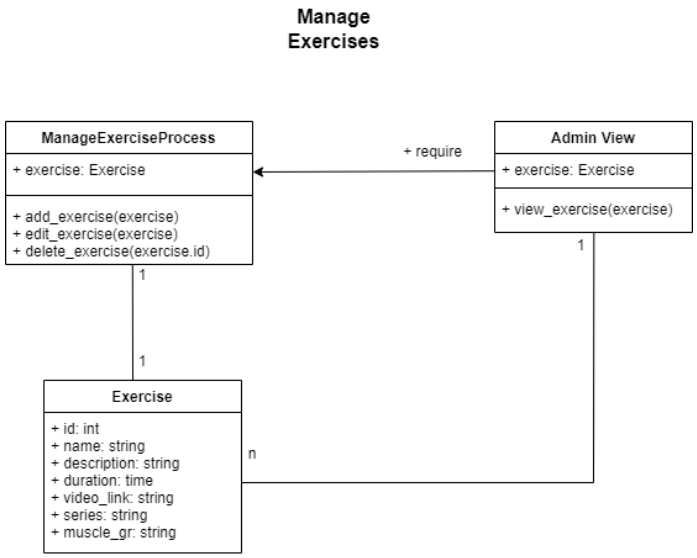
+ createNoti(notiID, seriesID): Process which system creates the notification.
+ sendNoti(notiID): Process which system sends the notification to the users.

Class: Notification

Attributes in this class:

+ seID: int
+ seriesID: array
+ startDate: datetime

4.4 Package: Manage Exercises



Class: Exercise

Attributes in this class:

+ name: string
+ description: string
+ duration: time
+ video: object
+ series: string
+ muscle_gr: string

Class: ManageExerciseProcess

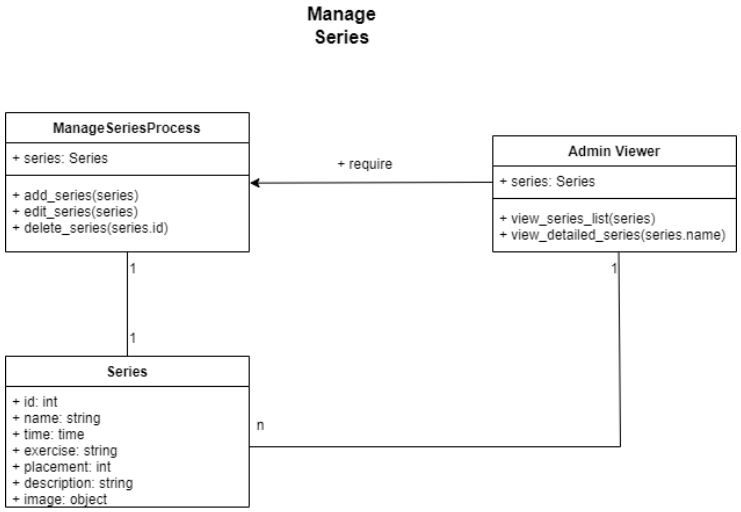
Attributes in this class:

+ exercise: Exercise

Operations in this class:

+ add_exercise(exercise): Process which admin adds the exercise with name, description, duration, video, series and muscle_gr type.
+ edit_exercise(exercise): The admin edits the exercise's properties such as name, description, duration, video, series and muscle_gr type.
+ delete_exercise(exercise.name): Process in which admin removes the whole exercise or properties in an exercise such as name, description, duration, video, series and muscle_gr type.

4.5 Package: Manage Series



Class: Series

Attributes in this class:

+ name: string
+ time: time
+ exercise: string
+ placement: string
+ description: string
+ image: object

Class: ManageSeriesProcess

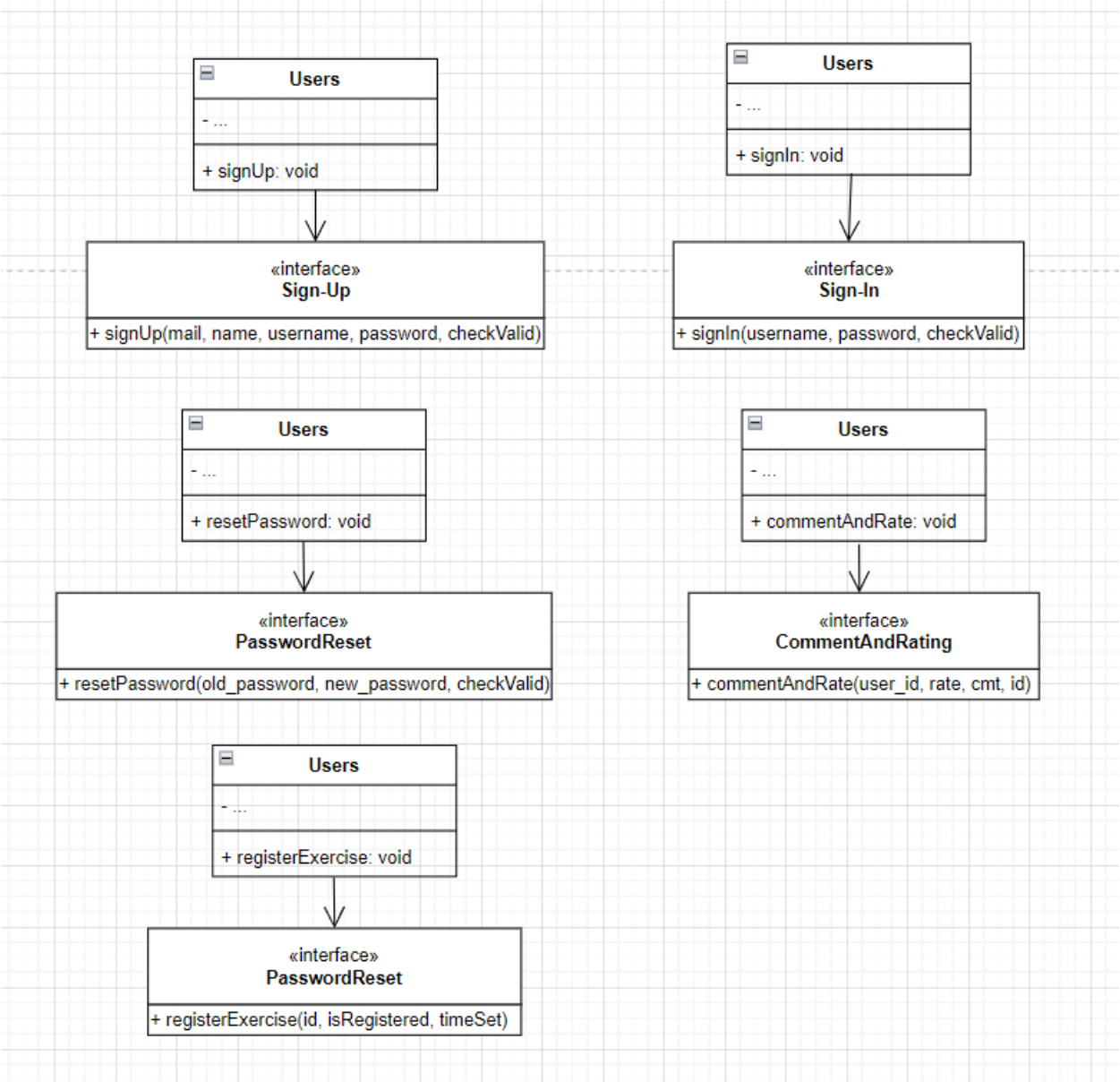
Attributes in this class:

+ series: Series

Operations in this class:

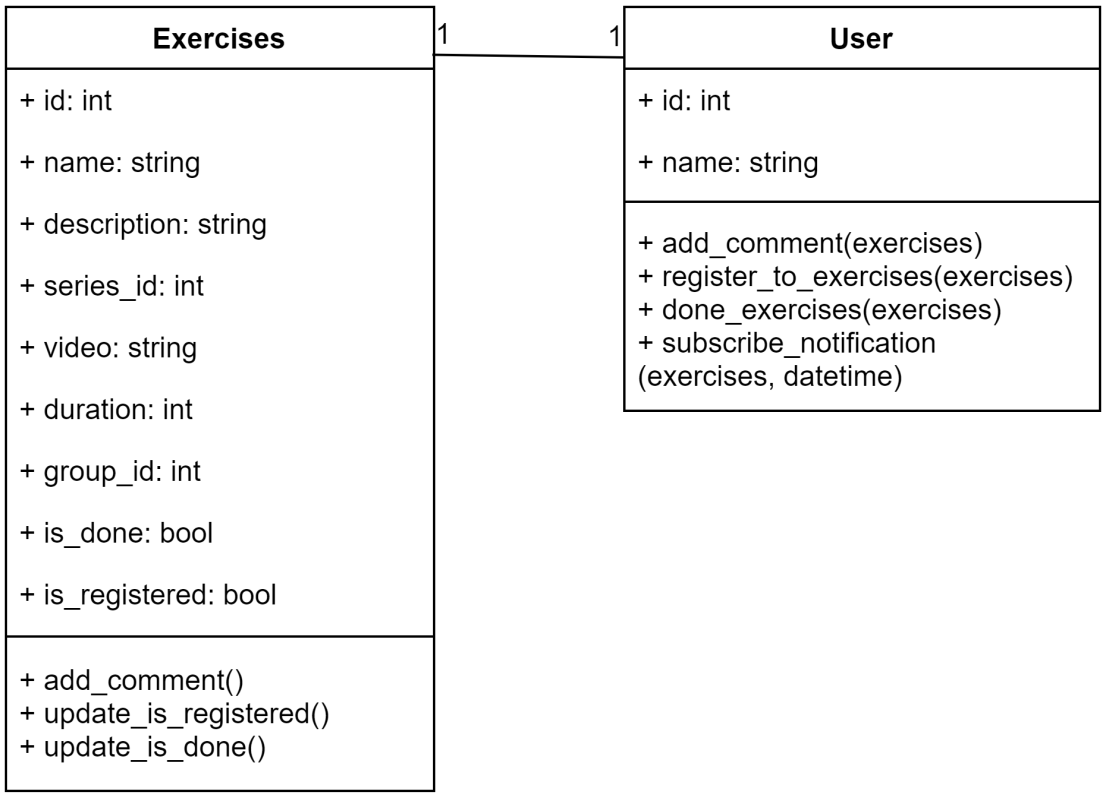
+ add_series(series): Process which admin creates a new series of exercises.
+ edit_series(series): Process which admin adds, removes, and edits exercise in series.
+ delete_series(series.name): Process which admin removes a series of exercises.

4.6 Package: User Data Process



- signUp(mail, name, username, password, checkValid): user signs up.
- signIn(username, password, checkValid): user signs in after signing up.
- resetPassword(old_password, new_password, checkValid): user can change their password.
- commentAndRate(user_id, rate, cmt, id): user can comment and rate the exercise.
- registerExercise(id, isRegistered, timeSet): user can register the exercise.

4.7 Package: User Exercises Manager



In this package:
Class: User
Operations in this class:
- + add_comment(exercises): user can add comments to the exercises when he has finished the exercises.
- + register_to_exercises(): user can register for the exercises.
- + done_exercises(exercises): user update the process of the registered exercises.
- + subscribe_notification(exercises, datetime): user can add a notification for the system to remind them to practice in those exercises.

Class: Exercises
Attributes in this class:
- + is_registered and is_done is a local variables for the specific user to store whether the user registered for the exercises and whether the user has finished the exercises.

Operation in this class:
- + add_comment(): add a comment to the model when the user adds a comment.
- + update_is_registered(): is call when the user registers to the exercises (update local storage)
- + update_is_done(): is call when the user finishes the exercises (update local storage)

4.8 Package: Notification Process

Class: Notification Process:

- + SendSeriesNotiInfo(): Send the data of duration and time to notify the user to the application from the database.

4.9 Package: Manage Data Process

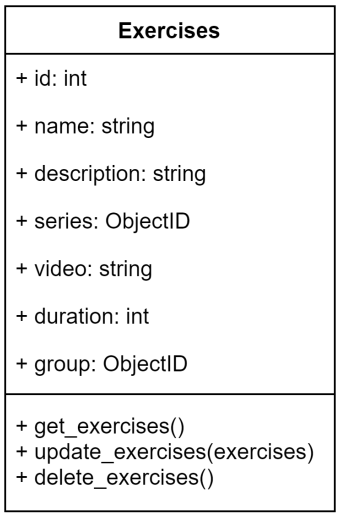
In this package:
Class: Exercises

- + Add(): the operation that adds a new exercise to the database received from the administrator.
- + Update(): the operation edits the information of an exercise.
- + Delete(): the operation that deletes an exercise.

Class: Series

- + AddSeries(): the operation to add a new series to the database received from the administrator.
- + ModifySeries(): modify the name, exercises, duration, and description of a series.
- + DeleteSeries(): the operation to delete an existing series.

4.10 Package: Exercises



Class: Exercise

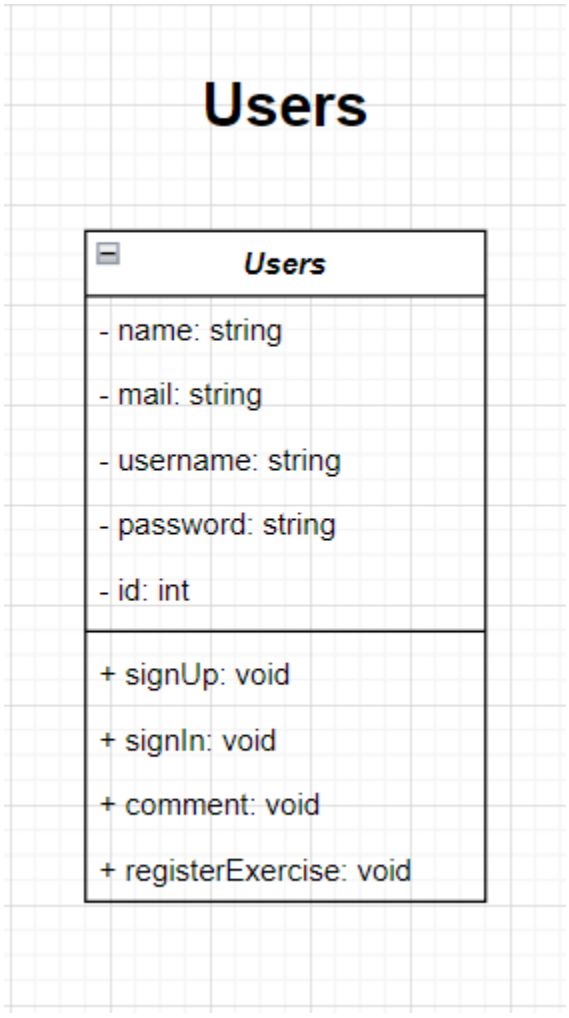
Attributes in this class:

+ id: int
+ name: string
+ description: string
+ duration: int
+ video: string
+ series: ObjectID
+ group: ObjectID

Operations in this class:

+ get_exercises(): return these exercises for the front-end to view information about these exercises.
+ update_exercises(exercises): receive the new information from the front-end and update it in models.
+ delete_exercises(): delete the exercises from the models.

4.11 Package: Users



Class: Users

Attributes in this class:

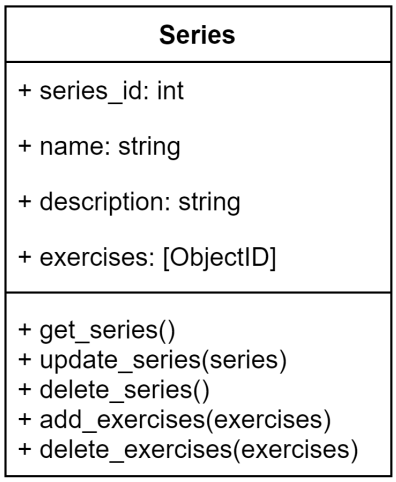
+ name: string

+ mail. string
+ username: string
+ password: string
+ id: int

Methods in this class:

+ signUp: void
+ signIn: void
+ comment: void
+ registerExercise: void

4.12 Package: Series



Class: Series

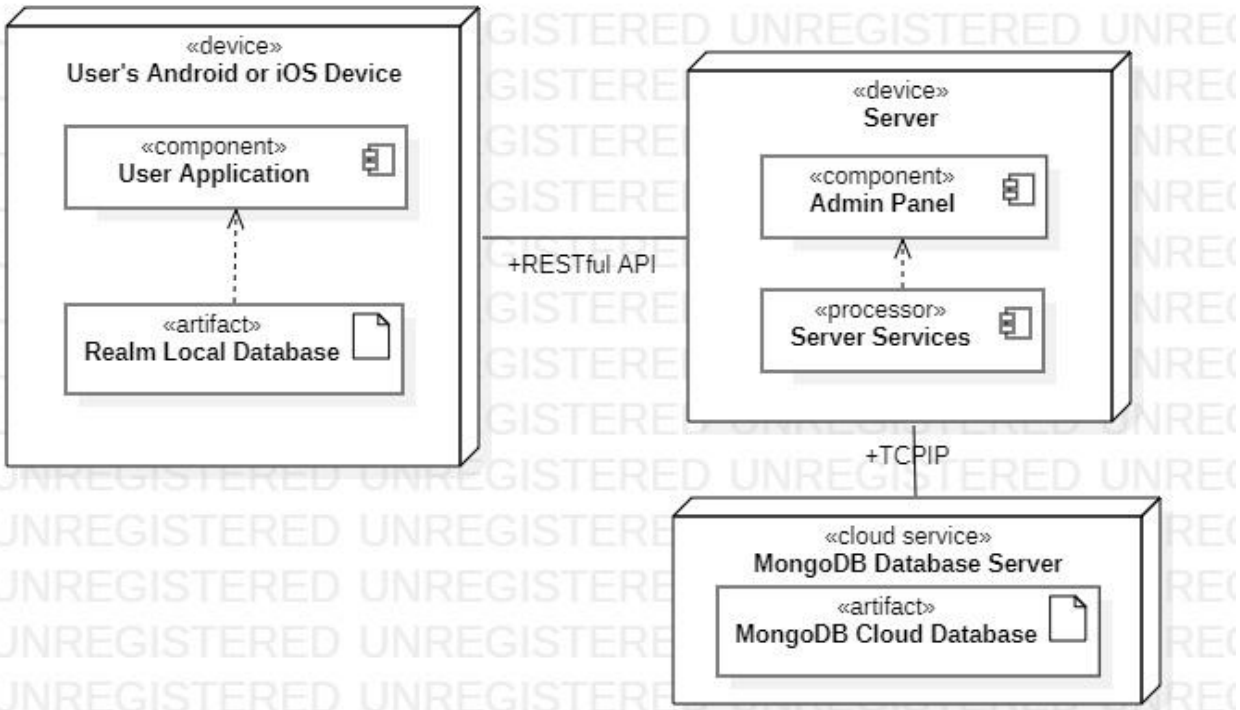
Attributes in this class:

+ series_id: int
+ name: string
+ description: string
+ exercises: [ObjectID]

Operations in this class:

+ get_series(): return this series for the front-end to view information about this series.
+ update_series(series): receive the new information from the front-end and update it in models.
+ delete_series(): delete the series from the models.
+ add_exercises(exercises): add new exercises to the array of exercises
+ delete_exercises(exercises): delete that exercises from the series.

5. Deployment



6. Implementation View

The source code included 2 parts: user application (in React Native) and server (in NodeJS).

User application source code view:

- **sources/**: storing source code of the front-end (including components, functions, hooks, screens).
  - **components/**: store the components to reuse for the screens.
  - **functions/**: store some specific functions to execute
  - **hooks/**: the hooks to connect all the screens in the app.
  - **screens/**: contain all the screens in the app.
- **constants/**: storing the declaration of some constant variable in the app.
- **assets/**: storing assets things like fonts, images, etc.
  - **fonts/**: storing fonts.
  - **images/**: storing images.
- **android/**: configuration for Android.
- **ios/**: configuration for iOS.
- **react-native.config.js**: custom configuration file of the app.
- **package.json**: package manager file.
- **yarn.lock**: package manager file.
- **App.js**: Main app file.
- **README.md**: the document of the application code.
- Some others configuration files and folders.

Server code view:

- **model/**: storing the model of the database.

- **routes/**: store the route of the restful API.
- **controllers/**: store the controller of the restful API.
- **app.js**: Main server file.
- Some configuration files of the back-end.