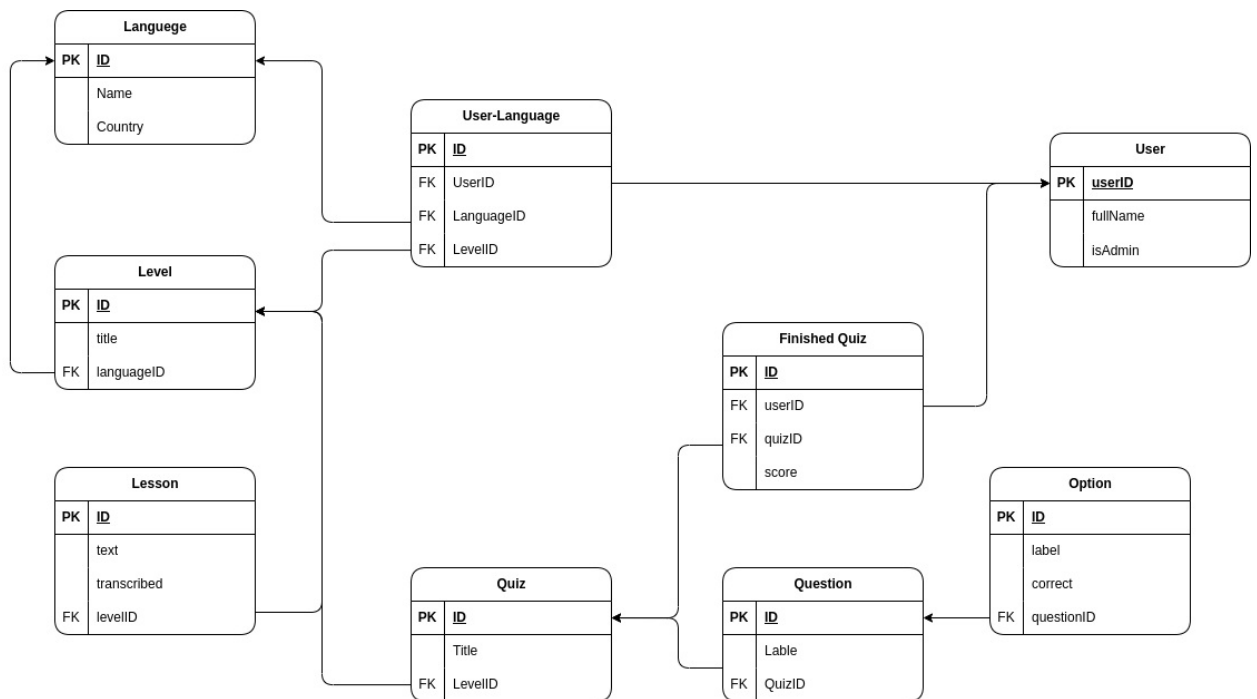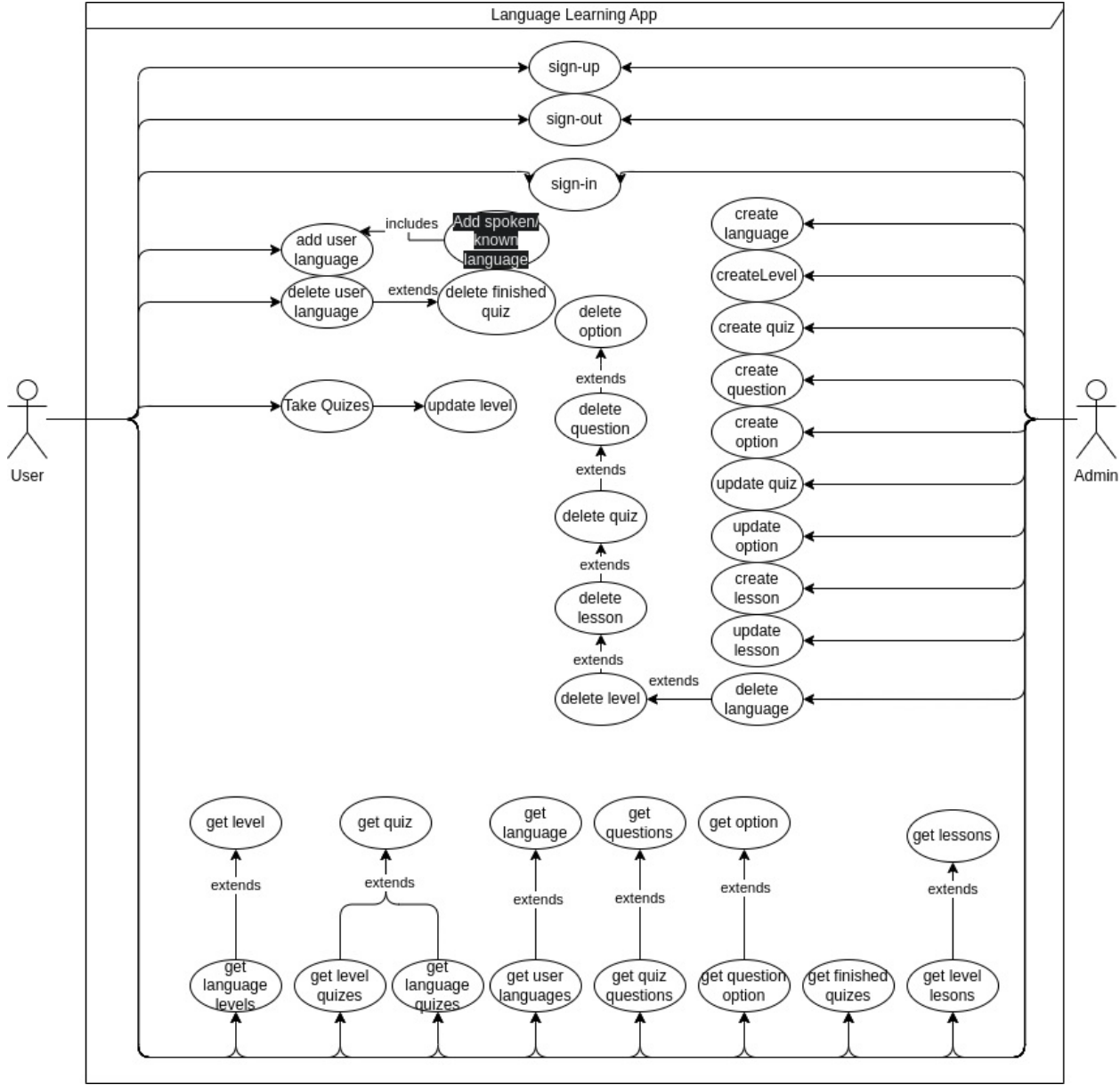# LINGER PROJECT

BACKEND SYSTEM

This is a language Learining App. It works with the **Users** that subscribe to language to learn through **lesson** and **quizes** and their test score recorded in **finishedQuiz**. The User can subscribe to multiple languages and achieve different levels of proficiency recorded in **UserLanguage**. Obviously each quiz has different sets of questions and **answers**.
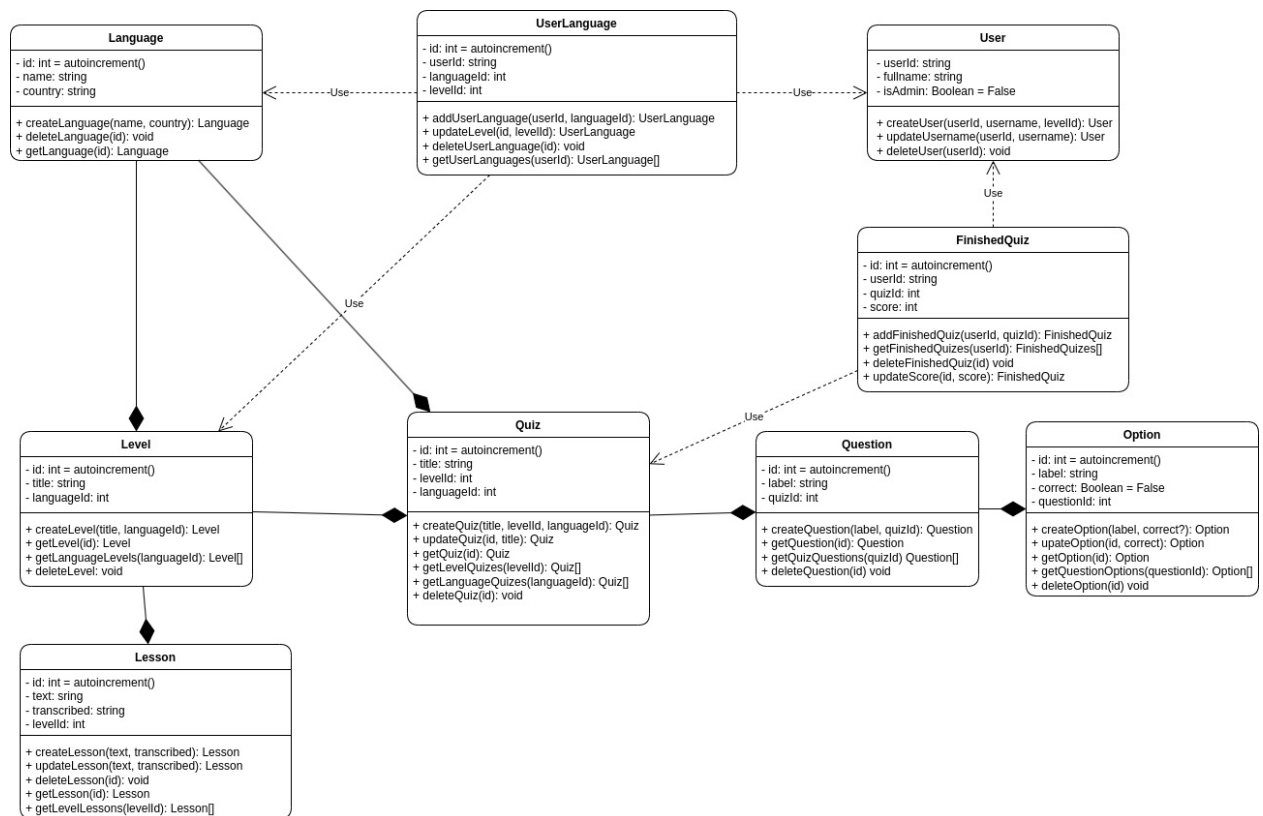
ER DIAGRAM

REST API STRUCTURE

USE CASE DIAGRAM

CLASS DIAGRAM

Language
- id: int = autoincrement()
- name: string
- country: string

+ createLanguage(name, country): Language
+ deleteLanguage(id): void
+ getLanguage(id): Language

UserLanguage
- id: int = autoincrement()
- userId: string
- languageId: int
- levelId: int

+ addUserLanguage(userId, languageId): UserLanguage
+ updateLevel(id, levelId): UserLanguage
+ deleteUserLanguage(id): void
+ getUserLanguages(userId): UserLanguage[]

User
- userId: string
- fullname: string
- isAdmin: Boolean = False

+ createUser(userId, username, levelId): User
+ updateUsername(userId, username): User
+ deleteUser(userId): void

FinishedQuiz
- id: int = autoincrement()
- userId: string
- quizId: int
- score: int

+ addFinishedQuiz(userId, quizId): FinishedQuiz
+ getFinishedQuizes(userId): FinishedQuizes[]
+ deleteFinishedQuiz(id) void
+ updateScore(id, score): FinishedQuiz

Level
- id: int = autoincrement()
- title: string
- languageId: int

+ createLevel(title, languageId): Level
+ getLevel(id): Level
+ getLanguageLevels(languageId): Level[]
+ deleteLevel: void

Quiz
- id: int = autoincrement()
- title: string
- levelId: int
- languageId: int

+ createQuiz(title, levelId, languageId): Quiz
+ updateQuiz(id, title): Quiz
+ getQuiz(id): Quiz
+ getLevelQuizes(levelId): Quiz[]
+ getLanguageQuizes(languageId): Quiz[]
+ deleteQuiz(id): void

Question
- id: int = autoincrement()
- label: string
- quizId: int

+ createQuestion(label, quizId): Question
+ getQuestion(id): Question
+ getQuizQuestions(quizId) Question[]
+ deleteQuestion(id) void

Option
- id: int = autoincrement()
- label: string
- correct: Boolean = False
- questionId: int

+ createOption(label, correct?): Option
+ upateOption(id, correct): Option
+ getOption(id): Option
+ getQuestionOptions(questionId): Option[]
+ deleteOption(id) void

Lesson
- id: int = autoincrement()
- text: sring
- transcribed: string
- levelId: int

+ createLesson(text, transcribed): Lesson
+ updateLesson(text, transcribed): Lesson
+ deleteLesson(id): void
+ getLesson(id): Lesson
+ getLevelLessons(levelId): Lesson[]

TASK ALLOCATION

Member	Task 1	Task 2
Thomson	Create User controller and routes	
Emmanuel	Create Quiz controller and routes	Create FinishedQuiz controller and routes
Divad	Create UserLanguage controller and routes	
Delano	Create Level controller and routes	
Makia	Create Lesson controller and routes	
Glenn	Create Question controller and routes	Create Option controller and routes

You can easily perform your task by following the same process that has been done with the Language class. You can see that at

`server/src/controllers/languageController.ts` for the controller and `server/src/routes/language.ts` for the routes.