VOICE COACH Application Design Document

Client: Roni Stern

Academic Adviser: Ehud Sivan

Team members:

Eli ben Shimol

Yonatan Shnits

Ziv Cohen Gvura

Nave Avraham

Table of Contents

VOICE COACH Application Design Document 1

1. Use cases 3

1.1 - User Profiles - The Actors........ 3

1.1.1 - The Actors: 3

1.1.2 – glossary of terms: 3

1.1.3 – main pages of the application 4

1.2 - Use Cases 6

2. System Architecture 26

2.1 system components........26

2.2 Deployment Diagram 27

3. Data Model........28

3.1 Description of data objects........28

3.1.1 Data Objects Diagram........28

3.1.2 Data Object Descriptions........28

3.2 Data Objects Relationship........32

3.3 Databases 32

4.Behavioral Analysis 35

4.1 Sequence Diagram........35

4.2 Events 44

4.3 states 48

4.3.1 explanation of machine states 48

4.3.2 state machine diagrams 48

5. Object oriented Analysis 53

5.1 class diagrams........53

5.2 Class Description........57

5.3 Packages 69

5.4 Unit testing 69

6. User Interface Draft 71

7. Testing 77

1. Use cases

1.1 - User Profiles - The Actors

1.1.1 - The Actors:

1. **User (Individual Practitioner):**

   One of the primary users of the system employs the voice coaching app for personal skill improvement. This can include individuals seeking to enhance public speaking, language pronunciation, singing abilities, or social communication skills. the user will have options to join classes and perform the assigned tasks given by the teacher/rabbi.

2. **Teacher/Rabbi:**

   In scenarios involving teaching, a teacher or rabbi may act as a primary user. They use the app to guide and assess students, providing feedback and personalized exercises for improvement.

1.1.2 – glossary of terms:

3. **Class**:

   Created by a single teacher and identified by a class ID, to join the class students will need to enter the class ID and be authorized by the teacher. The class contains tasks created by the teacher. In the class view students will see all the tasks that they have and will be able to interact with them. In the teacher view he will see for each task how each student preformed.

4. **Project**:

   Created by a single user, needs a sample recording. Will allow the user to record himself and get feedback according to the sample recording. The project will have analysis and will be able to track progress over time and different user recordings.

5. **Task**:

   Part of a class and created by a single teacher. Will include a sample recording and notes to students preforming the task. For the students the task will behave as a project but will also have place for feedback from the teacher. For the teacher a task will show the performance of each student and will give the option for additional feedback.

1.1.3 – main pages of the application

6. **Register page:**

   will require a email, password and will create a new user with those credentials.

7. **Login page:**
   Will require email, password. If a user with those credentials exists, the app will go to the main page otherwise it will show an error message.
8. **Main Page:**
   Will have a button for each of the following functionalities:
   - Viewing the projects
   - Viewing classes you attend to.
   - Viewing classes, you created
   - Settings
   - View profile
9. **Projects Page:**
   Will show a list of all the projects.

10. **Project Page:**

Will show progress in the given project, will show the sample recording and previous analysis. Will also allow to add a new recording.

11. Session Page:

Will visualize the recording of the user in real time and will show feedback according to the sample.

12. Analysis Page:

Will show the recording analysis. And for each comment will provide the mistake in the recording for the user to hear.

13. Attended classes Page:

Will show all the classes that you belong to.

14. **Student class Page:**

Will show all the tasks in the class and the teacher's general comments and info for the class.

15. **Student Task Page:**

Same as projects page but with the teachers comments and info.

16. **Created classes Page:**

Will show the classes you created.

17. **Teacher class Page:**

Will show the students in the class and all the tasks you created with an option to add more or delete.

18. **Teacher Task Page:**

Will show how each student preformed the task and will allow the teacher to add his feedback in addition to the app's feedback.

1.2 - Use Cases

1. **Use Case: Recording and Comparing Voice Samples**

    - **Primary Actor:** User (Student)
    - **Description:** The user, a student or individual, utilizes the voice coaching app to record a voice sample for practice and receives feedback by comparing it with their own attempt.
    - **Stakeholders and Interests:**
        - *User:* Aims to improve vocal skills by recording and comparing voice samples for targeted practice.
        - *Teacher/Rabbi:* Provides guidance and assesses student progress using the app.
    - **Preconditions:**
        - The user has the voice coaching app installed on a compatible device and is logged in.
        - The user's device has a functioning microphone and is connected to the internet.
        - The user already created a project with a sample.
    - **Postconditions:**
        - The user receives feedback on their recorded voice sample and personalized insights for improvement.

Main Success Scenario:

1. The user accesses the app and selects the "Project Page" feature.
2. The app prompts the user to upload a voice sample of a specific text or song.

3. After recording, the user receives the option to attempt the same text or song.
4. The app records the user's attempt and compares it with the original voice sample.
5. The app provides feedback on areas such as pitch accuracy, tone, pacing, and pronunciation.
6. The user receives personalized recommendations to address identified areas for improvement.

Alternative Flows:

- *Recording Failure:*
  - If there is an issue with the recording, the app prompts the user to re-record or troubleshoot the microphone.
- *Comparison Glitch:*
  - If there is a glitch in the comparison process, the app informs the user and suggests reattempting the exercise.
- *Teacher/Rabbi Review:*
  - See use case 10

2. **Use Case: Reviewing students Task**

   - **Primary Actor:** Teacher/Rabbi
   - **Description:** The teacher utilizes the class and task options to see the students voice sample with comments and will add feedback if needed.
   - **Stakeholders and Interests:**
     - *Teacher:* Aims to provide effective vocal coaching and assess student progress.
     - *Student:* Receives personalized feedback to improve singing skills.
   - **Preconditions:**
     - The teacher has the voice coaching app installed on a compatible device and is logged in to the "Teacher task Page".
     - Students have the app installed and have submitted their recorded voice samples.
   - **Postconditions:**
     - Students receive constructive feedback from the app and the teacher, and the teacher can track their progress.

Main Success Scenario:

1. The teacher accesses the app and navigates to the "Teacher task page" feature.
2. The app displays a list of student submissions with recorded voice samples.
3. The teacher selects a student's session to review.
4. The app compares the student's recording with the original voice sample, highlighting areas for improvement.

5. The teacher provides personalized feedback, suggestions, and additional exercises for the student.
6. The app updates the student's progress and stores the feedback for future reference.

Alternative Flows:

- *Technical Issues in Reviewing:*
    - If there are technical issues in reviewing student sessions, the teacher or rabbi may request students to resubmit or provide feedback through alternative means.
- *Student Feedback Loop:*
    - see use case number 11.

3. Use Case: Reviewing Song Performance Feedback

- **Primary Actor:** User
- **Description:** The user, a singer or musician, utilizes the voice coaching app to get feedback on their song performance. The app provides detailed insights, including pinpointing mistakes in the tune, graphical representations of recordings, and identification of mispronounced words.
- **Stakeholders and Interests:**
  - *User:* Aims to assess and improve their song performance based on detailed feedback.
- **Preconditions:**
  - The user has the voice coaching app installed on a compatible device.
  - The user has previously recorded a performance using the app.
- **Postconditions:**
  - The user gains a comprehensive understanding of their song performance, including areas for improvement and specific feedback on tune and pronunciation.

Main Success Scenario:

1. The user accesses the app and enters a project.
2. The user navigates to the "Analysis Page".
3. The app displays a list of previously recorded song performances.
4. The user selects a specific song performance for review.

5. The app generates detailed feedback, including graphical representations of the recording, highlighting mistakes in the tune.
6. The user receives feedback on mispronounced words, pitch accuracy, and overall pacing.
7. Graphs and visual aids illustrate specific points of improvement within the song.

Alternative Flows:

- *Unavailable Song Recordings:*
  - If there are no recorded song performances available, the app informs the user and prompts them to record a new performance for assessment.
- *Technical Glitch in Feedback Display:*
  - If there is a glitch in displaying feedback, the app notifies the user and suggests troubleshooting steps or contacting support.

4.Use Case: User Login

• **Primary Actor**: User/Teacher

• **Description**: The user enters the voice coaching app to access their personalized account by logging in, allowing them to utilize app features and track their progress.

• **Stakeholders and Interests**:

User: Aims to access the app's functionalities, including recording, feedback, and personalized coaching.

• **Preconditions**:

1. The user has successfully installed the voice coaching app on their compatible device.
2. The user has a valid account registered with the app.
3. The network is working and satisfy the app demands.
4. The server is up and running on the server computer.

• **Postconditions**:

1. The user successfully logged in to the system.
2. The main app page is displayed with all the options of the user.

Main Success Scenario:

1. The user launches the app on his device.

2. The app presents a login screen requesting the user's credentials (username and password).

3. The user enters his valid username and password.

4. The app verifies the entered credentials against the stored user data.

5. Upon successful verification, the app grants access to the user's account.

6. The user gains entry to the app's main page, where they can access various features, including recording and feedback functionalities.

Alternative Flows:

• Incorrect Credentials:

> If the user enters incorrect credentials, the app prompts an informative error message notifying the user what’s went wrong and allows the user to retry.

• Account Lockout:

> After a certain number of unsuccessful login attempts, the app locks the account temporarily for security reasons. The user receives instructions on unlocking their account or resetting the password.

5.Use Case: User Registration

• **Primary Actor**: User

• **Description**: The user initiates the registration process to create a new account within the voice coach app, enabling personalized access to the application's features.

• **Stakeholders and Interests**:

User: Aims to create a new account to utilize the app's recording, feedback, and coaching functionalities.

• **Preconditions**:

The user has downloaded and installed the voice coaching app on their compatible device.

• **Postconditions**:

The user successfully completes the registration process and gains access to their newly created account.

Main Success Scenario:

1. The user launches the app on their device.
2. The app presents a registration screen requesting essential information, such as username, email address and password.
3. The user enters the required information, ensuring compliance with any specified password complexity criteria.
4. The app validates the entered information, checking for unique usernames and valid email formats.
5. Upon successful validation, the app creates a new user account, associating it with the provided username and email address.
6. The app automatically logs in the newly registered user, granting immediate access to the app's main Page.

Alternative Flows:

- Existing Email or Username:

  If the entered email or username already exists in the system, the app prompts the user to choose a different and unique combination.

- Password Strength:

  If the entered password does not meet the specified strength criteria, the app prompts the user to create a more secure password.

6.Use Case: Handling Incoming Call During App Usage

• **Primary Actor**: User

• **Secondary Actor**: Incoming Call

• **Description**: The user is actively using the voice coaching app when an incoming call is received on their mobile device. The app must gracefully handle the call without disrupting the current state or losing user data.

• **Stakeholders and Interests**:

User: Aims to seamlessly handle an incoming call without losing progress or experiencing disruptions in the app.

App Developer: Ensures the app maintains a user-friendly and reliable experience even during external interruptions like incoming calls.

• Preconditions:

1. The user has the voice coaching app open and is actively engaged in an ongoing session or task.

• Postconditions:

2. The user successfully handles the incoming call without losing progress in the app.

Main Success Scenario:

1. While using the voice coaching app, the user receives an incoming call on their mobile device.

2. The app detects the incoming call and temporarily pauses the ongoing activity, saving the current state.

3. The app displays a notification or overlay indicating the incoming call and providing options to answer or decline.

4. The user chooses to answer the call or declines it.
If the call is answered, the app remains in the background while the user engages in the call.

Alternative Flows:

- **Call Declined**:

  If the user declines the incoming call, the app continues uninterrupted, and the user remains in the current state.

- **App Termination**:

  If the user chooses to exit the app during the call, the app can gracefully terminate, ensuring a smooth transition back to the app when the call ends.

- **App Paused State**:

  The app may save the current state periodically during active sessions to minimize potential data loss in the event of unexpected interruptions.

7. Use Case: Creating a class

- **Primary Actor:** Teacher
- **Description:** The teacher uses the voice coaching app to create and manage classes. Users can join via the class id that will be provided by the teacher. The teacher decides whether to accept or reject the request.
- **Stakeholders and Interests:**
    - *Teacher:* Aims to teach, manage class requests efficiently, and ensure a positive learning experience.
    - *Students:* seek to join a class with their teacher for optimized learning experience.
- **Preconditions:**
    - The teacher has the voice coaching app installed and a registered account.
    - The teacher has created a class within the app.
- **Postconditions:**
    - Class is created and the teacher receives the class id to share with other users.

Main Success Scenario:

1. The teacher accesses the app and navigates to the "Created classes page" section.
2. The teacher creates a new class and receives a class id.
3. The teacher adds details such as class name, description, and any specific requirements.
4. The teacher saves the class configuration.
5. Users (students) access the app and view the available classes in the "Attended classes Page" section.

Alternative Flows:

None.

Exceptional Flows:

- *Technical Issues:*
    - If there are technical issues with the request and approval system, the teacher and users are informed, and technical support can be notified.
- *Class Removal:*
    - The teacher can remove a class if needed, which automatically rejects pending requests.

8. Use Case: joining a class with given id.

- **Primary Actor:** Student
- **Description:** Users can search for specific classes by entering a unique class ID code.
- **Stakeholders and Interests:**
    - *User:* Aims to quickly find and join a specific class his teacher created.
- **Preconditions:**
    - The user has the voice coaching app installed and a registered account and is on the Attended classes Page.
- **Postconditions:**
    - The user successfully locates and accesses the desired class by its ID code.

Main Success Scenario:

1. The user accesses the app and navigates to the "Attended Classes Page".
2. Searching by ID code, the user enters the unique code associated with the desired class.
3. The app retrieves and displays relevant class based on the entered id.
4. The user selects the class to view additional details and, if interested, sends a request to join.

Alternative Flows:

- *Invalid ID Code:*
    - If the user enters an invalid or non-existent ID code, the app informs the user and prompts for a correct code.

Exceptional Flows:

- *Technical Issues:*
    - If there are technical issues with the search functionality, the user is informed, and technical support can be sought.

9. Use Case: Handling Student Join Request

- **Primary Actor**: Teacher
- **Description**: The teacher receives and manages a join request from a student who wants to enrol in their class. The teacher has the option to accept or reject the request.
- **Stakeholders and Interests:**
    - ***Teacher:*** Aims to maintain a working class containing only students related to the class.
    - ***Student:*** Seeks approval to join the class.
- **Preconditions:**
    - The teacher has the voice coaching app installed and a registered account.
    - The teacher has created a class, and a student has sent a request to join.
- **Postconditions:**
    - The teacher has either accepted or rejected the student's join request.

Main Success Scenario:

1. The teacher accesses the app and navigates to the "Created classes Page".
2. The teacher sees a notification indicating a pending join request for their class.

3. The teacher selects the class with the pending request to review the details.
4. The app displays information about the student attempting to join the class.
5. The teacher evaluates the student's request based on their suitability for the class.
6. The teacher decides either:
    - Accept the request, allowing the student to join the class.
    - Reject the request.
7. If accepted, the student receives a notification of approval and gains access to the class tasks.
8. If rejected, the student receives a notification of the decision.

Alternative Flows:

- ***Delay in Review:***
    - The teacher may take some time to review the request. The student is notified once a decision is made.

Exceptional Flows:

- ***Technical Issues:***
    - If there are technical issues with the request handling system, the teacher is informed, and technical support can be sought.
- ***Class Removal:***
    - If the teacher decides to remove the class, all pending requests are automatically rejected.

10. Use Case: Adding a Task in Class

•**Primary Actor:** Teacher

•**Description:** The teacher utilizes the voice coaching app to create and manage tasks within a class, providing students with specific assignments and sample recordings. Students interact with these tasks as projects, recording themselves and receiving feedback based on the provided sample. The teacher monitors student performance and offers additional feedback as necessary.

• **Stakeholders and Interests:**

1. Teacher: Aims to create tasks related to the class objectives, monitor student progress, and provide targeted feedback.
2. Students: Seek clear instructions and guidance from the teacher, aiming to complete assigned tasks effectively and improve their skills.

• **Preconditions:**

1. The teacher has the voice coaching app installed and a registered account and is on the Teacher class Page.
2. The teacher has created a class within the app.

• **Postconditions**:

1. The task is successfully added to the class, and visible to all students enrolled in the class.

Main Success Scenario:

1. The teacher accesses the app and navigates to the "Teacher Class Page".
2. The teacher selects the desired class where the task will be added.
3. The teacher clicks on the "Add Task" button.
4. The teacher uploads a sample recording for the task and provides any accompanying notes or instructions for students.
5. The teacher saves the task configuration within the class.

6. Students enrolled in the class access the app and view the newly added task in the class view.

Alternative Flows:

None.

Exceptional Flows:

- **Technical Issues:**
  If there are technical issues with uploading the sample recording or saving the task configuration, the teacher is notified, and technical support can be contacted for assistance.

11. Use Case: Answering Teacher's Feedback

•**Primary Actor:** User

•**Description:** The user, after receiving feedback from the teacher on their performance in a task, interacts with the voice coaching app to address the feedback and seek clarification or guidance. The user could view the teacher's feedback on a sample recording within the task and send questions or responses accordingly.

•**Stakeholders and Interests:**

1. User (Student): Aims to understand and improve upon the feedback provided by the teacher, seeking clarity and guidance to enhance their performance.
2. Teacher: Aims to provide constructive feedback to students, aiding their improvement and understanding of the task requirements.

•**Preconditions:**

1. The user has completed a task within a class and received feedback from the teacher.

2. The user has access to the voice coaching app is logged into their account and is on the "Student task Page".

•**Postconditions:**

1. The user successfully interacts with the teacher's feedback, seeking clarification or providing responses as needed.

Main Success Scenario:

1. The user accesses the voice coaching app and navigates to the "Student task Page".
2. The user selects the task for which they received feedback from the teacher.
3. The user reviews the feedback provided by the teacher on their sample recording.
4. If the user has questions or requires clarification on specific aspects of the feedback, they compose a message within the app.
5. The user sends the message to the teacher, seeking further explanation or guidance.

Alternative Flows:

None.

Exceptional Flows:

- Teacher Unavailability:
- If the teacher is unavailable to respond immediately, the user is informed and encouraged to check back later for a response.

2. System Architecture

2.1 system components

All the following components will reside on the client side using unity c#. All the audio files will be temporarily saved on the client side until it will synchronize with the server and then will be saved on the server for future usage.

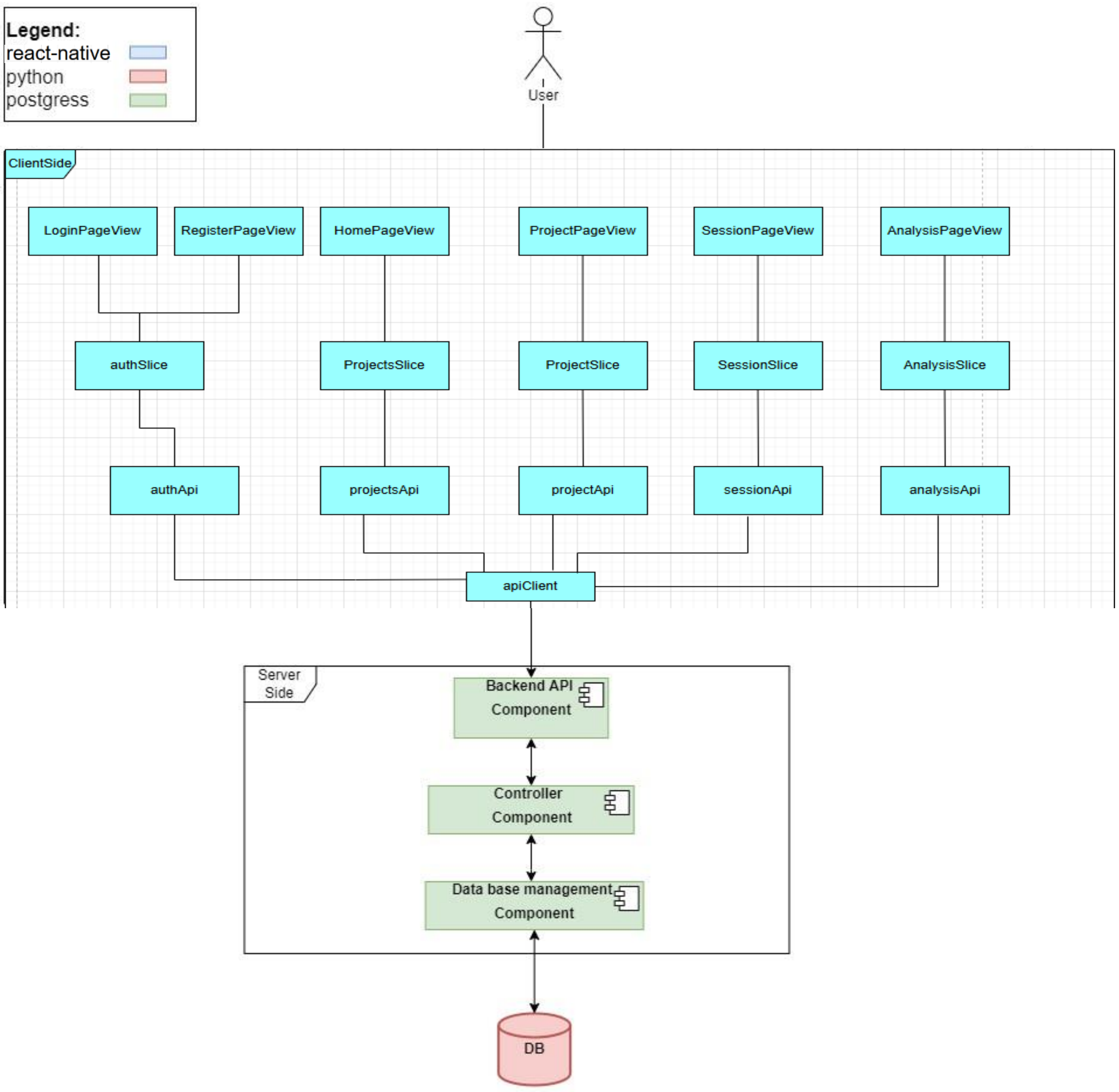
- **UI component –** The unity engine which is responsible for the UI visualization and event handling like buttons, page transitions, etc.
- **Audio analyzer component – a** component that will include the in-depth analysis of our app that will be on the client side.
- **Visualization component – a** component that will be responsible for visualizing the user input sound and the sample sound during the user recording session. And will visualize the instant feedback.
- **Real-time analyzer component –** A component that will be responsible for the basic analysis feedback during the recording like tempo, notes, and rhythm.
- **Environment handler component –** this component is responsible for running the environment needed for the audio analyzer component.
- **Data handler component –** This component is responsible for managing and parsing the data objects on the client side.
- **API client component –** This component is responsible for connecting to the server and requesting and retrieving data that is needed for the client side using endpoints.

The following components reside on the server side:

- **Database management component –** This component is responsible for handling the data of all the clients and saving it in the database.
- **Controller component –** will be responsible for handling the logic and data classes of the database.

- **Backend API component –** This component will include all the endpoints of our server for client communication.

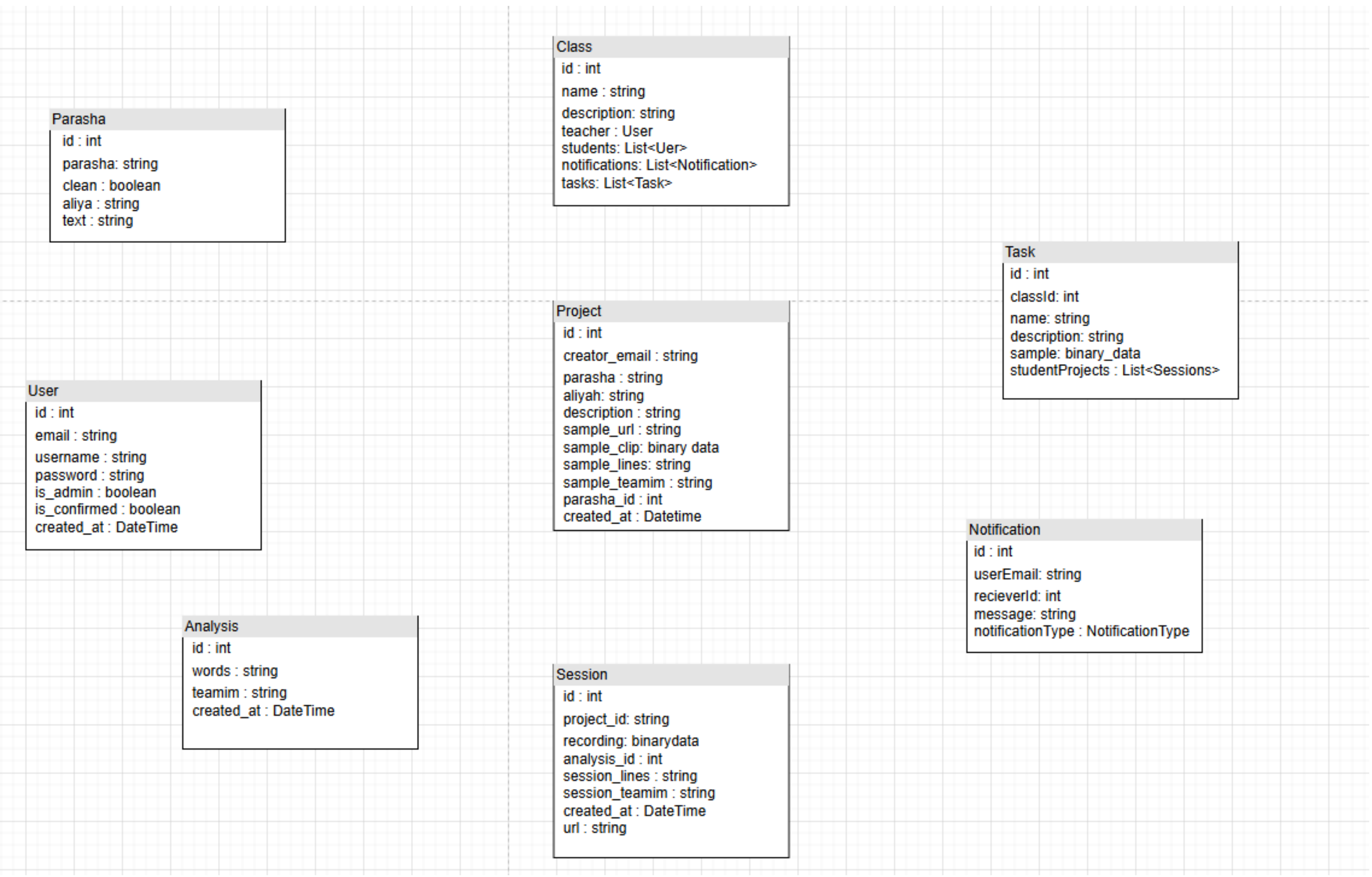
2.2 Deployment Diagram



3. Data Model

3.1 Description of data objects

3.1.1 Data Objects Diagram



3.1.2 Data Object Descriptions

- User

  Contains information about the user in the system.
  - Id – a unique ID for each user.
  - Email – the user's email address
  - username – a name that the user chooses and will be displayed to other users.

    - password – the password of the user that will be checked at the authentication.
    - Is_admin – whether this user is admin or not.
    - Is_comfirmed – whether the user's email address has been confirmed
    - Created_at – the account creation time.

- Project
  Contains information about a project that the user created.
    - Id – the project's unique identifier.
    - Creator_email – the email of the user that created the project.
    - Prasha – the name of the Parasha this project is made for.
    - aliya – the name of the Aliyha this project is made for.
    - description –short text to describe the project.
    - Sample_url – the address of the Rabbi's sample.
    - Sample_clip – the audio data in binary format.
    - Sample_lines – the sample's text.
    - Sample_teamim – the sample's Teamim.
    - Parasha_id – the key to get the Parsha from our DB.
    - Created_at – the time and date for this project creation.

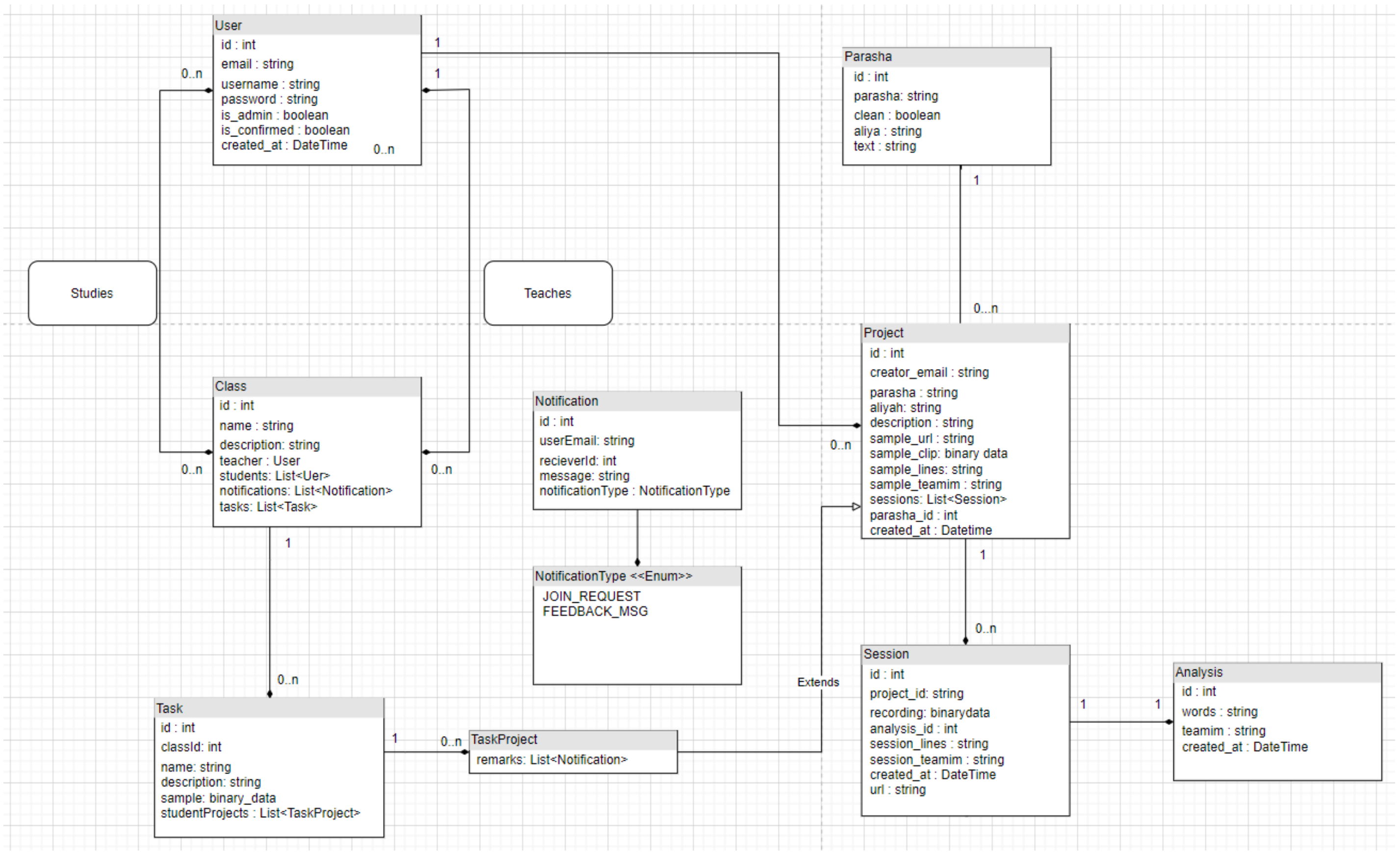
- Session
  Contains information about the performance of the user.
    - Id – part of the session unique identifier.
    - ProjectId – the ID of the project the session belongs to and part of the Recording identifier.
    - recording – the audio clip of the user performing the recording.
    - Analysis_id – analysis of data of the user performing the recording.
    - Session_lines – the session's text (from the user's recording).
    - Session_teamim – the session's Teamim.

  - Created_at
  - url – the session's address.

- Analysis
  Contains information about the in-depth analysis of the user performing the recording.
  - Id – unique identifier of the object.
  - words – the words that the user got wrong.
  - Teamim – the Teamim that the user got wrong
  - Created_at
- Class
  Contains the information about a class that was created by the user.
  - Id – unique identifier of the class.
  - Name – a class name that was given by the user.
  - Description – a class description that was given by the user.
  - Teacher – the user who created the class.
  - Students – list of users that joined the class.
  - Notifications – list of requests from users to join the class.
  - Tasks – list of tasks created by the teacher for the students to attempt.
- Notification
  Contains the information about a notification and its type.
  - Id – unique identifier of the notification.
  - User email – the email address of the user that sent the notification.
  - Receiver ID – the destination of the notification depends on its type.
  - Message – the contents of the notification sent by the user.
  - notificationType – the type of the notification.
- Notification type
  Enum with values of the notification type.

    - JOIN_REQUEST – indicates that a user wants to join one of the receiver classes.
    - FEEDBACK_MSG – a chat between the teacher and student about a specific task.
- Task
  Contains information about a task in a class.
    - Id – unique identifier of the Task.
    - classId – the class that the task belongs to.
    - Name – task name that was given by the teacher.
    - Description – task description that was given by the teacher.
    - Sample - audio clip chosen by the teacher of the task in creation to be an example of how to perform.
    - studentProjects – a list of TaskProjects submitted by the students for the task.
- TaskProject
  Extends project and contains remarks from the teacher.
    - Remarks – list of notifications that are from the teacher to the student telling him how to improve.
- Parasha
  Contains the relevant text from the bible.
    - Id
    - Parasha – the name of the Parasha
    - Clean – whether the text contains Teamim or not
    - Aliya – the Aliya (could be first, second etc.)
    - Text – the bible’s text for this Aliya

3.2 Data Objects Relationship



3.3 Databases

Since most usage scenarios are carried out on the user side of the system, on the backend side the tables exist to save the progress of each user and the relationship between user-class-teacher and the tasks each teacher assigns to his students.

<u>User</u>:

Represents users of the application. They have email, username, and password attributes. Users can create projects (projects), be assigned to classes (assigned_classes), and create classes (created_classes).

Recording:

Represents audio recordings associated with projects. It has a foreign key reference to the Project model and an analysis associated with it.

Analysis:

Represents data analysis associated with a recording. It has a one-to-one relationship with the Recording model.

Project:

Represents projects created by users. Each project belongs to a user (creator_email) and can have multiple recordings (recordings).

Class:

Represents classes in the application. It has a teacher (teacher_id) and a list of students (students). Classes can have notifications (notifications) and tasks (tasks).

Notification:

Represents notifications sent to users. It contains information about the user, receiver, message, and notification type.

Task:

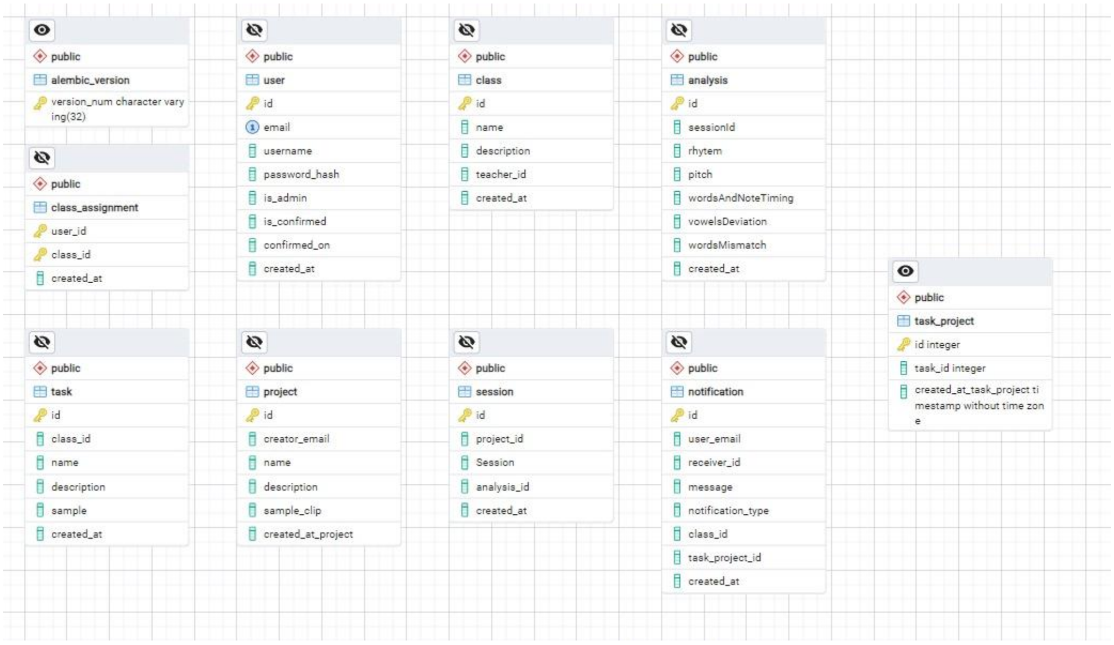
Represents tasks associated with classes. Each task belongs to a class and can have multiple task projects (task_projects).

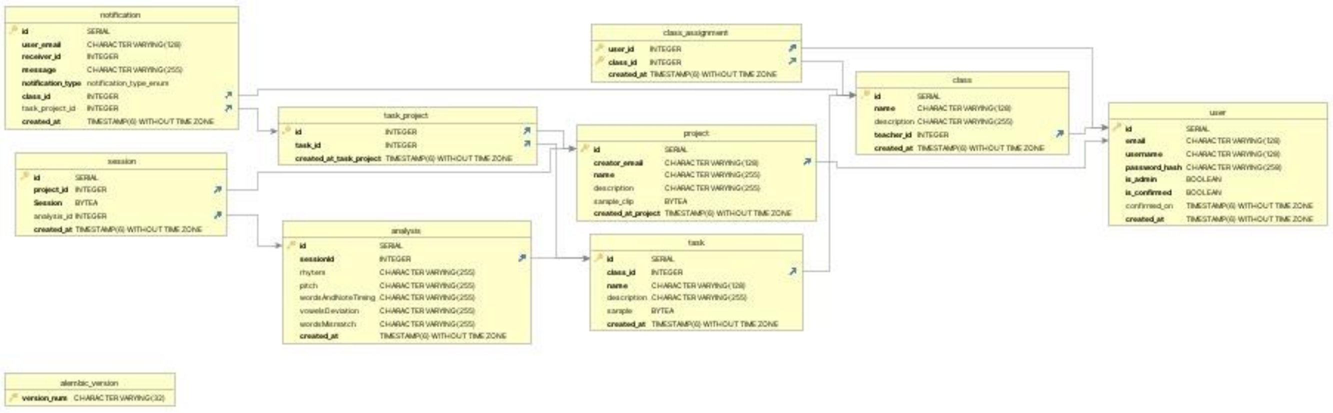
<u>TaskProject</u>:

Extends the Project model and represents task-specific projects. It contains feedback (feedback) associated with the task.

<u>ClassAssignment</u>:

Represents the many-to-many relationship between users and classes. It associates users with the classes they are assigned to



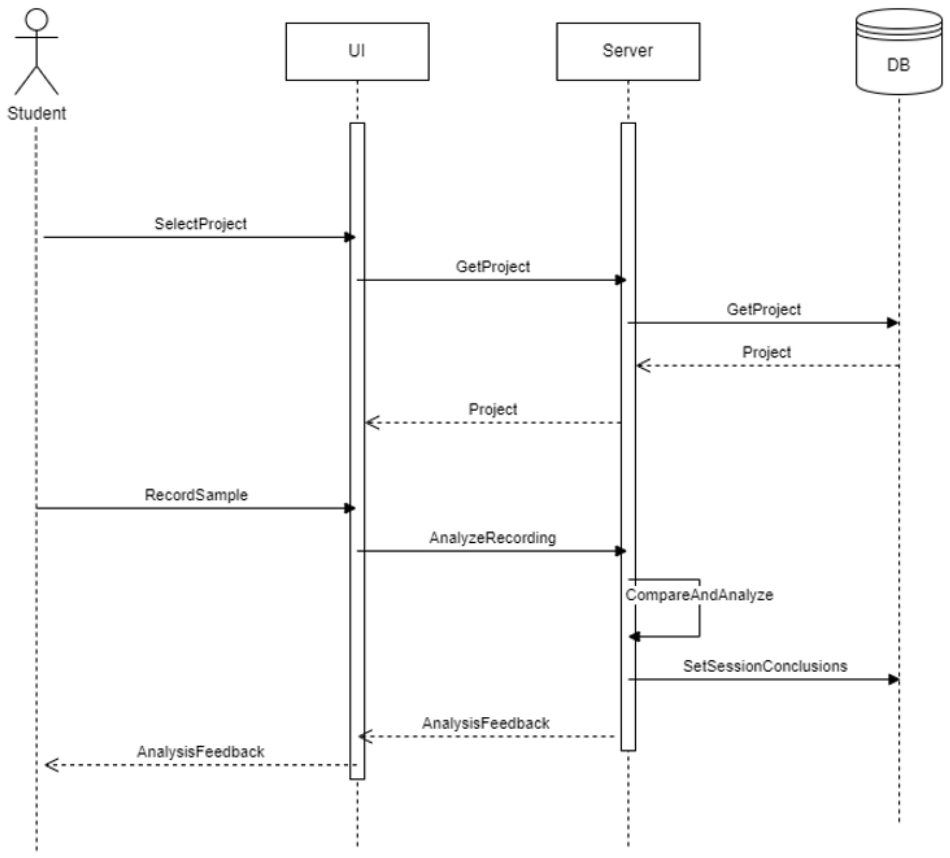


notification
id SERIAL
user_email CHARACTER VARYING(128)
receiver_id INTEGER
message CHARACTER VARYING(255)
notification_type notification_type_enum
class_id INTEGER
task_project_id INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
class_assignment
user_id INTEGER
class_id INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
class
id SERIAL
name CHARACTER VARYING(128)
description CHARACTER VARYING(255)
teacher_id INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
task_project
id INTEGER
task_id INTEGER
created_at_task_project TIMESTAMP(6) WITHOUT TIME ZONE
project
id SERIAL
creator_email CHARACTER VARYING(128)
name CHARACTER VARYING(255)
description CHARACTER VARYING(255)
sample_clip BYTEA
created_at_project TIMESTAMP(6) WITHOUT TIME ZONE
user
id SERIAL
email CHARACTER VARYING(128)
username CHARACTER VARYING(128)
password_hash CHARACTER VARYING(256)
is_admin BOOLEAN
is_confirmed BOOLEAN
confirmed_on TIMESTAMP(6) WITHOUT TIME ZONE
created_at TIMESTAMP(6) WITHOUT TIME ZONE
session
id SERIAL
project_id INTEGER
Session BYTEA
analysis_id INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
analysis
id SERIAL
sessionId INTEGER
rhytem CHARACTER VARYING(255)
pitch CHARACTER VARYING(255)
wordsAndNoteTiming CHARACTER VARYING(255)
vowelsDeviation CHARACTER VARYING(255)
wordsMismatch CHARACTER VARYING(255)
created_at TIMESTAMP(6) WITHOUT TIME ZONE
task
id SERIAL
class_id INTEGER
name CHARACTER VARYING(128)
description CHARACTER VARYING(255)
sample BYTEA
created_at TIMESTAMP(6) WITHOUT TIME ZONE
alembic_version
version_num CHARACTER VARYING(32)

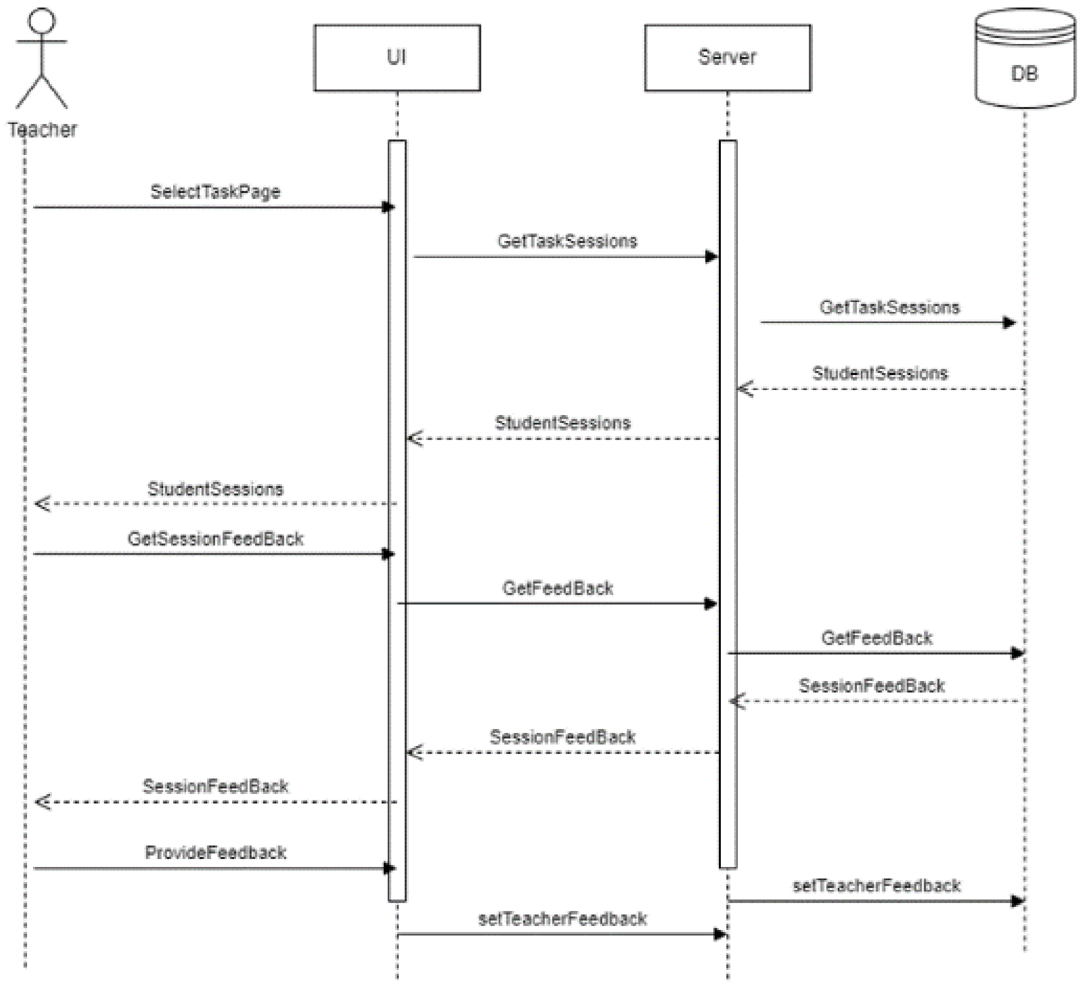
4.Behavioral Analysis

4.1 Sequence Diagram

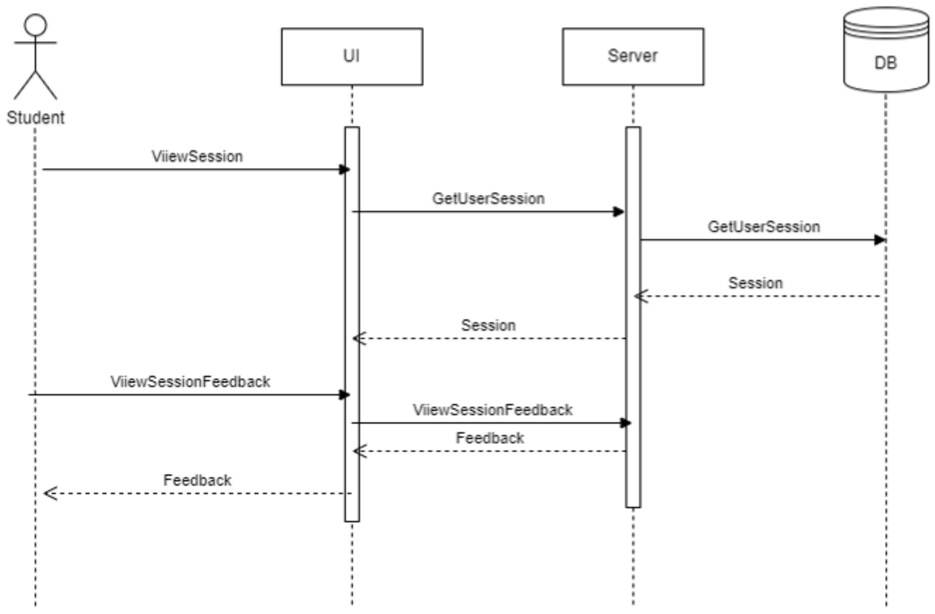
1) Recording and comparing voice samples:



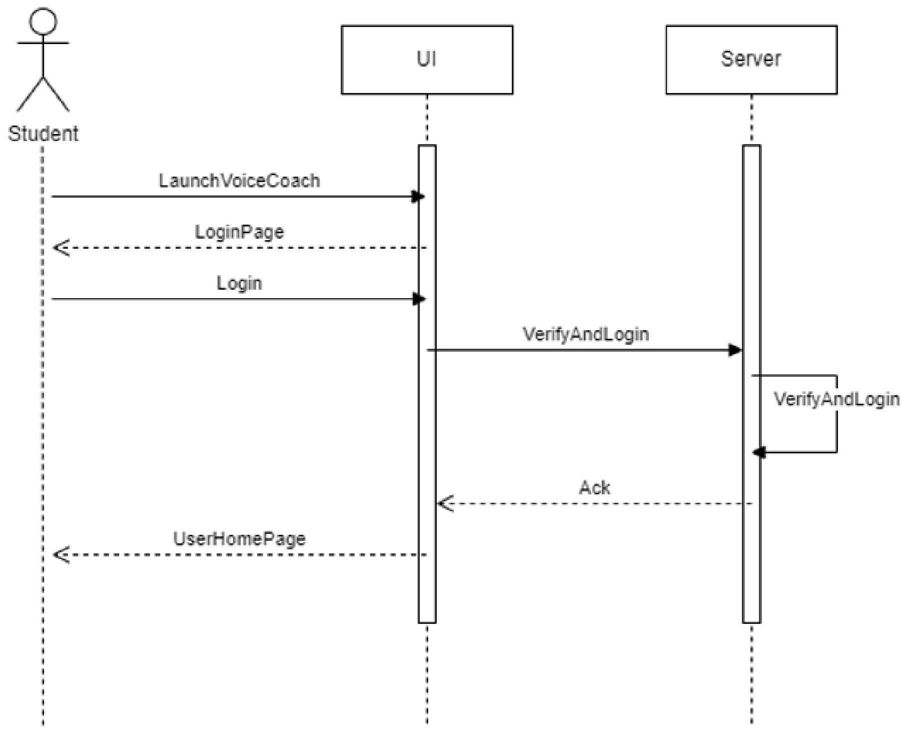
2) Reviewing students Task:



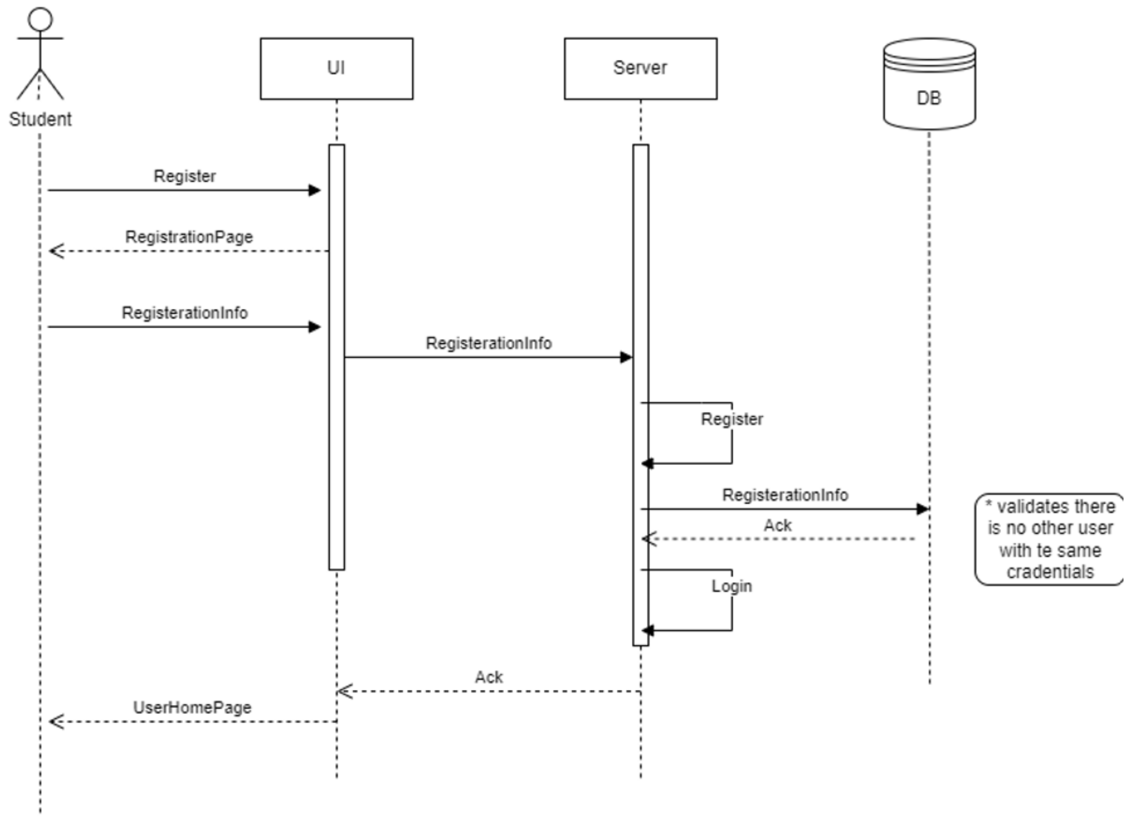
3) Reviewing song performance feedback:



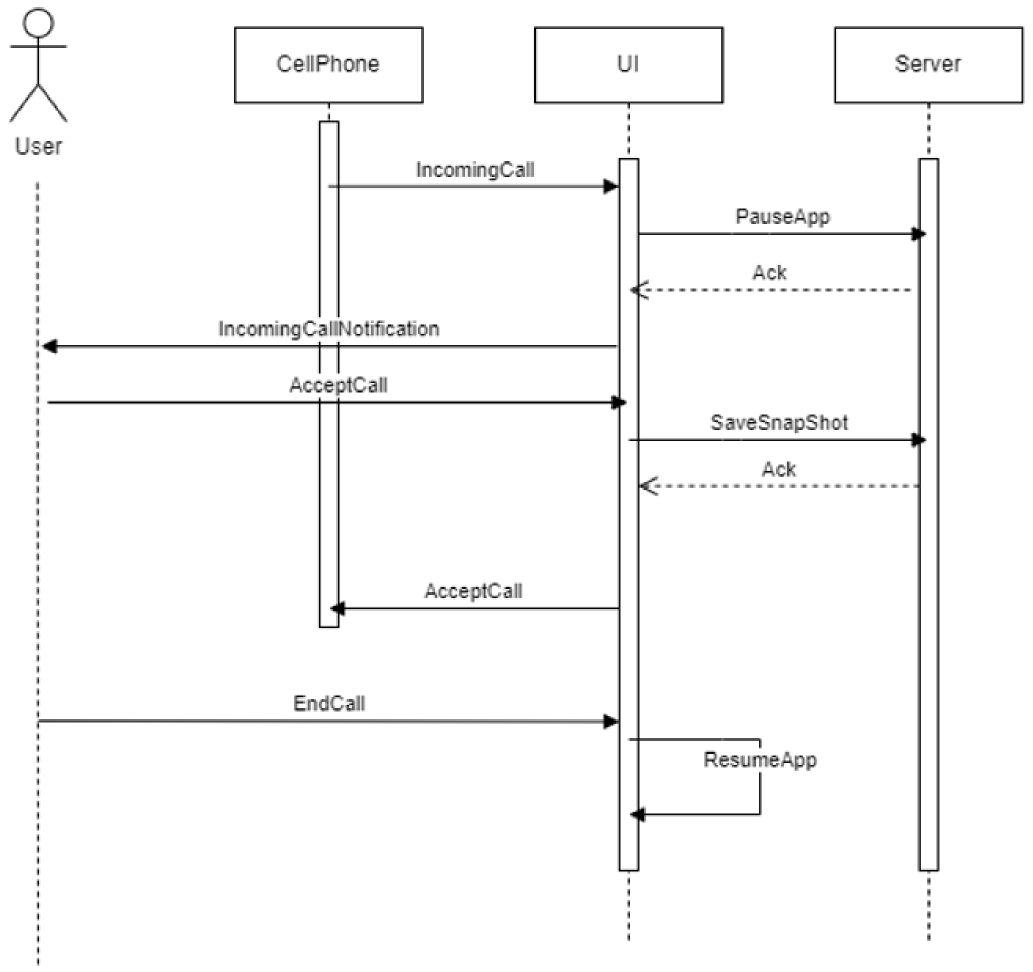
4) User Login:



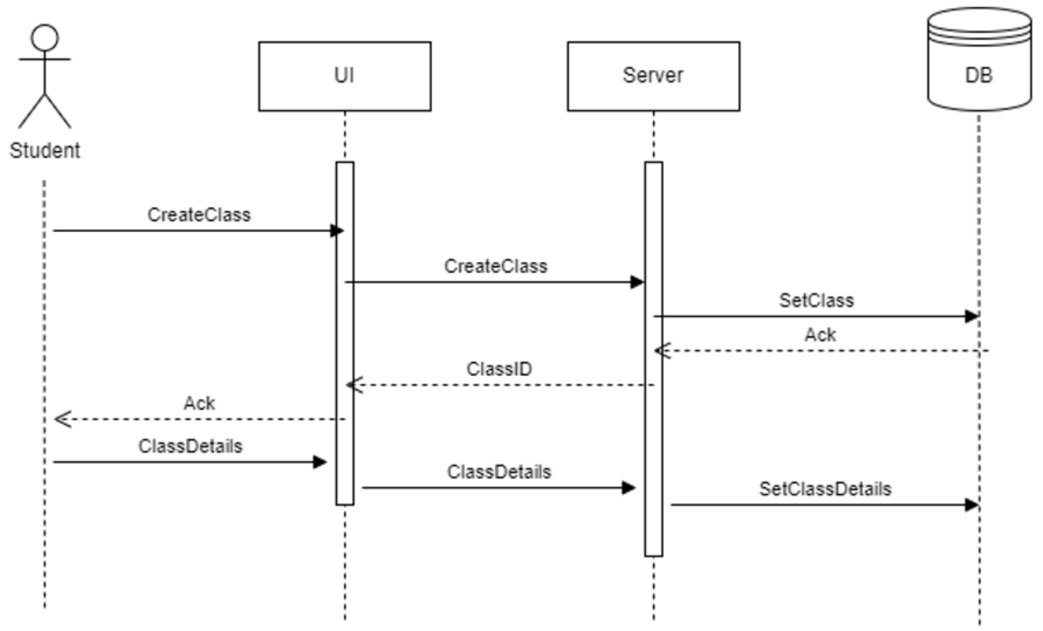
5) User Registration:



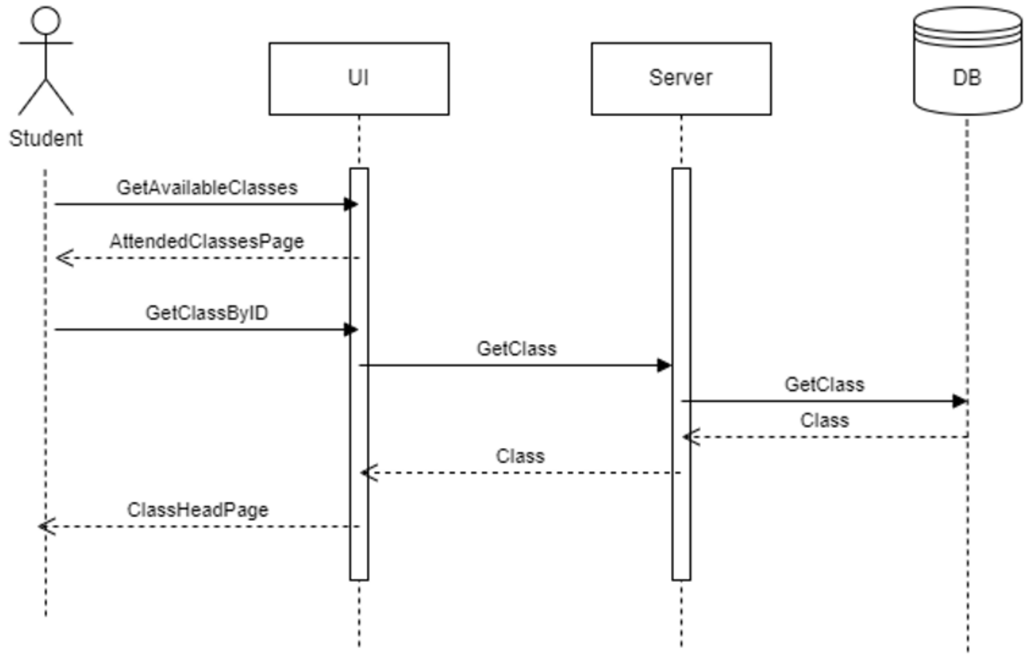
6) Handling incoming call During App usage:



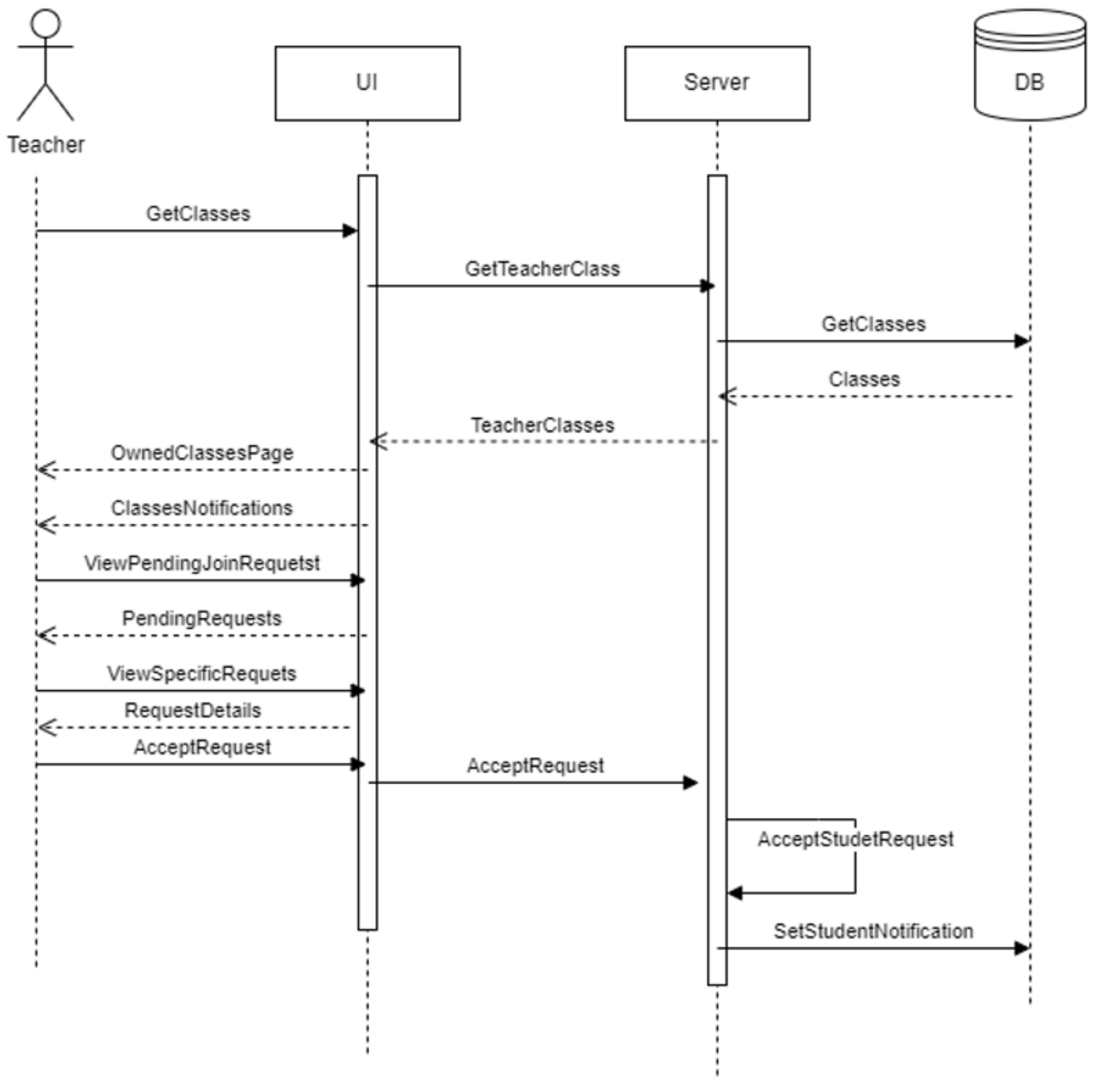
7) Creating a Class:



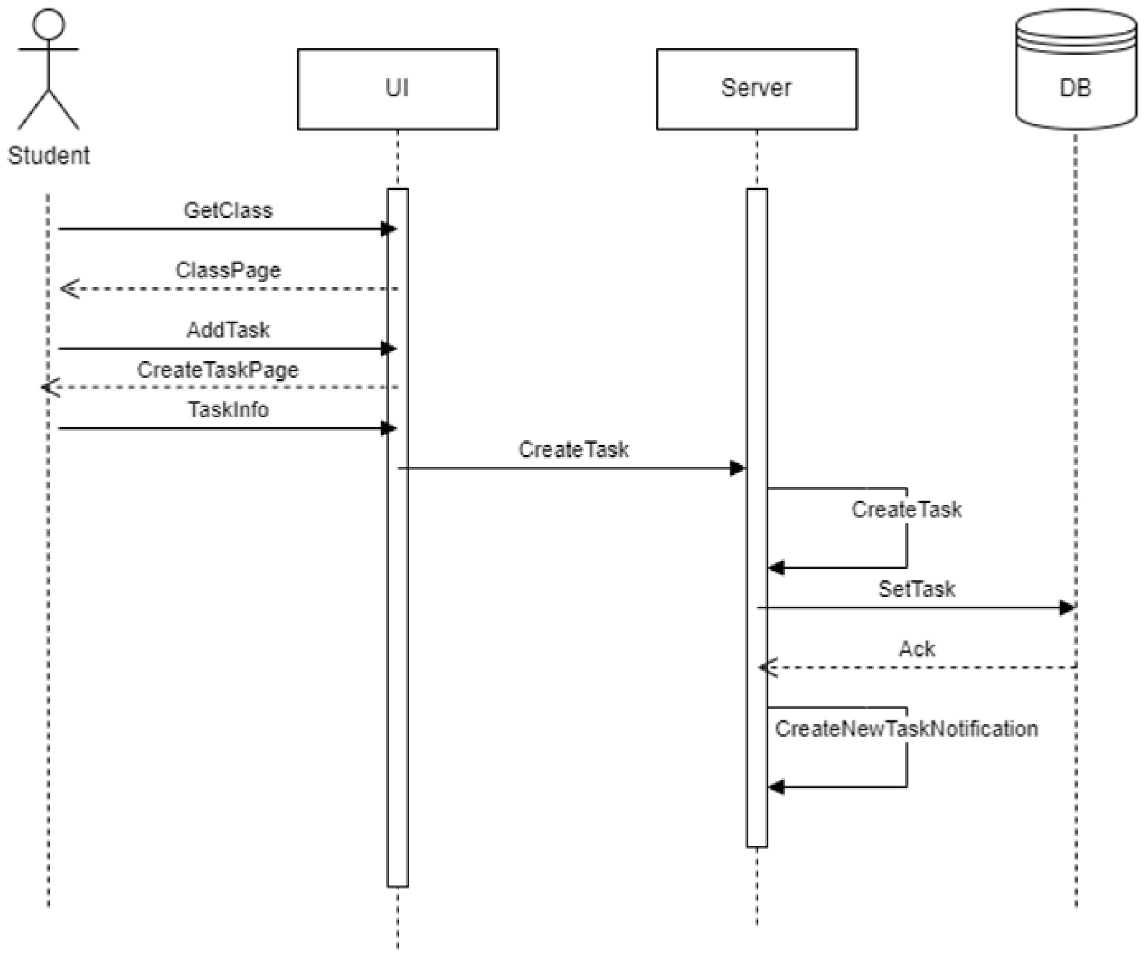
8) Joining Class with given id:



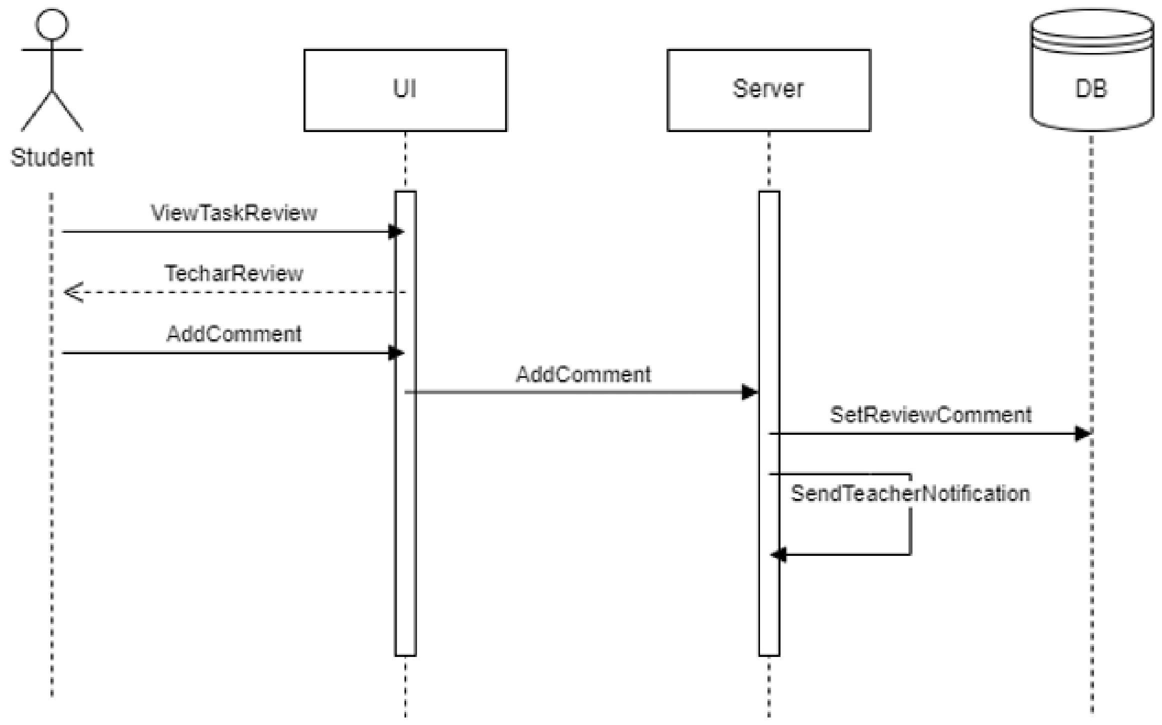
9) Handling Student Join Request:



10) Adding Task in Class:



11) Answering Teacher's Feedback:



4.2 Events

Event	Description	Action
Register	The system registers a new user.	The user chooses to register to the system, he enters his email and password, if all the information is valid the user is saved in the DB and the user is redirected to the main menu
Login	The system performs user login procedure.	The user performs the login with the information he gave and registration, the system checks if a user with this information exists within the system and if so, the user is redirected to the main menu.
Logout	The system performs the user logout procedure, saving all his actions and discarding any temporary data from the phone	The user chooses to logout from the app, the system then discards all the files that were temporarily saved on the phone (after they have been transferred to the DB) and logs the user out of the system.
Create project	The system adds a new project to the user projects.	The user creates a new project for his own use. The system adds the project to the list of the projects of the user.
Add sample	The system adds a sample to a project of the user.	The user chooses a new project he wants to work on and selects a sample to add to the project to be the goal of the project. The system adds the sample to the project in the DB.
Record a session	The system records a user performing the sample and adds the session to the project sessions.	The user chooses to record a new session of him performing the sample, the system starts the record session and once the user chooses to end the session, the system adds the session to the project in the DB and starts the analysis process.

Create session analysis	The system creates an analysis for a session that has ended	After the user chose to end a session, the system starts the analysis process for the session and after the it ends the system adds the analysis to the session DB and shows the user the analysis report.
Show session analysis	The system presents the user an analysis on a previous session he had in the project	The user chooses a project he created and then chooses one of his previous sessions in the project, the system will then present the analysis of the session for the user to view.
Delete project	The system deletes the project from the projects list of the user.	the user chooses a project he created that he wants to delete, the system then asks the user if he is certain if he wants to delete the project, if so, the project is removed from the user DB and the project will no longer be visible
Delete session	The system deletes the session from the sessions list of a project of the user	The user chooses a project he created and then he can choose a session he had in the project. the user selects to delete the session, the system makes sure if the user wants to delete the session and if so the session is removed from the project DB and will no longer be visible.
Create class	The system adds a new class to the created classes of the user	The user creates a new class that he will teach. The system adds the class to the list of created classes of the user and gives the user the class ID to share with his students.
Join a class	The system sends a join request to the teacher from the user requesting to join	The user goes to the attended classes page and enters the class ID in the search class segment, a notification is then sent to the teacher informing him

		that a user wants to join his class and asks the teacher if he wants to allow the user to join. The system saves the request notification in the teacher DB.
Delete a class	The system deletes the class and informs all the students	The teacher chooses a class he created and then has the option to delete it. The system makes sure the teacher wants to delete his class and if so the class and all its tasks will be removed from the DB. Also, a notification will be sent to all the students who were in the class.
Leave a class	The system removes the user from a class and informs the teacher that the user has left.	The user chooses a class he attends and can choose to leave the class. The system makes sure the user wants to leave and if so the user will be removed from the class DB, all of his performed tasks will be removed and the teacher will be informed.
Perform a task	The system allows a student of a class to perform tasks related to the class records them performing the task and adds the session to the task sessions of the user in the class.	The user chooses a class he is a part of and then chooses a task within this class, the system then allows the user to perform the task (the same way as performing a project) and saves the session in the task DB.
Write remarks to a student	The system adds the remarks to the user's performed task for him to view	The teacher chooses a class he teaches and within the class, he chooses to see all the students that performed the task and when he chooses a user, he can see his sessions in the task. The teacher can then leave a remark for the student to view the next time he performs the task.

		The system also saves the remarks in the tasks DB.
Question remarks	The system adds questions from the user to the teacher about remarks given to him/questions about the task in general for the teacher to view.	The user chooses a class he attends and then can choose a task he wants to work on, in the task the student can see the remarks a teacher has left for him and can send questions to the teacher about the remarks. The teacher can view the questions next time he enters the task page. The system also saves the questions in the task DB.
delete a task	The system deletes the task from the selected class and all the students are informed	The teacher chooses a class he created, and he can choose a task to delete. The system makes sure the teacher wants to delete the task and if so the task and all the students' sessions are deleted from the DB. The students will also be informed.
Exit the app	The system saves the current state of the system and informs the user that any projects that are in progress will not be saved	The user chooses to close the app, if he is in the progress of a new session the system informs the user that it will not be saved unless he finishes the session, and then the system continuous according to the user's response (either discarding the session or continuing it if the user chose so).

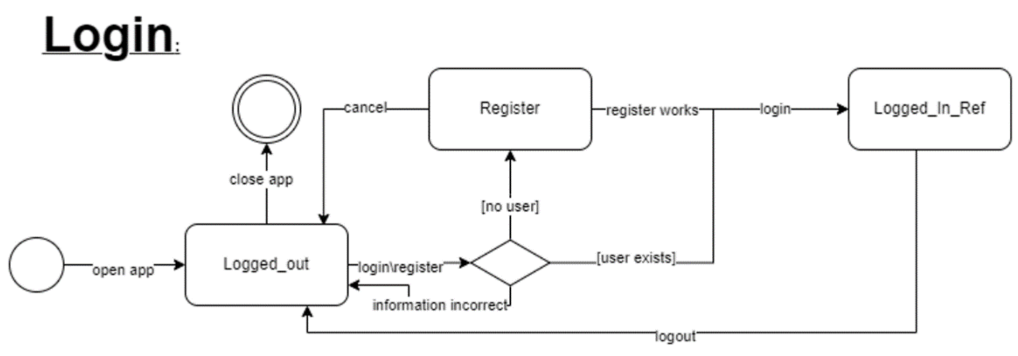
4.3 states

4.3.1 explanation of machine states

In our system, data is saved for the user (meaning is entered to the database) when the user has clicked on a button (for example chose to end a recording). Data that was not saved still exists in the system but only temporarily, meaning that if a user exits the system and closes it without finishing a task, all data will be saved except for the last task because the user didn’t choose to save it, if the user exists the system but does not close it, the data will still exist but only for a while, if the user will not return to the system in a short time, its authentication token will be expired meaning he will not be able to use the system until he logs in again, so it that case he will also lose progress.

In any other case, all the data will be saved, and the user will be able to continue from where he stopped.

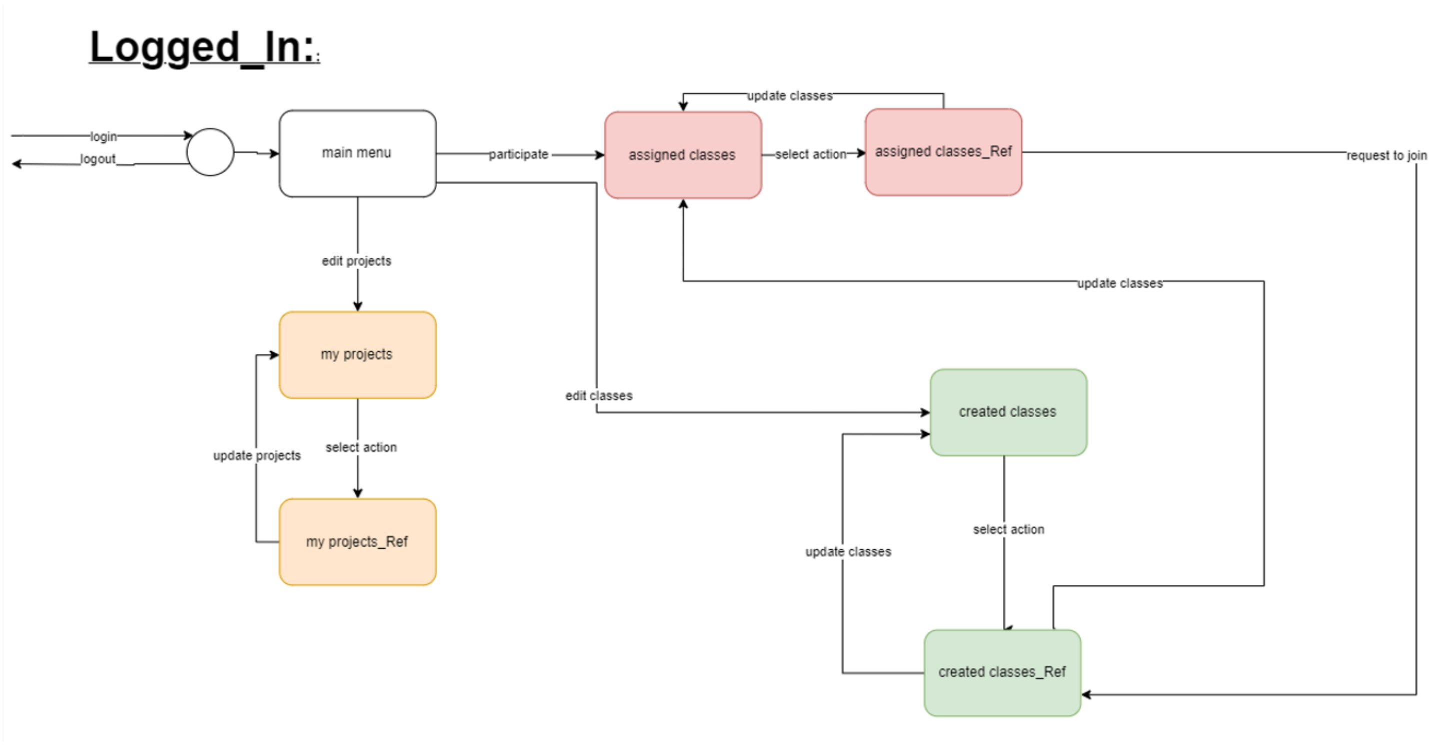
4.3.2 state machine diagrams



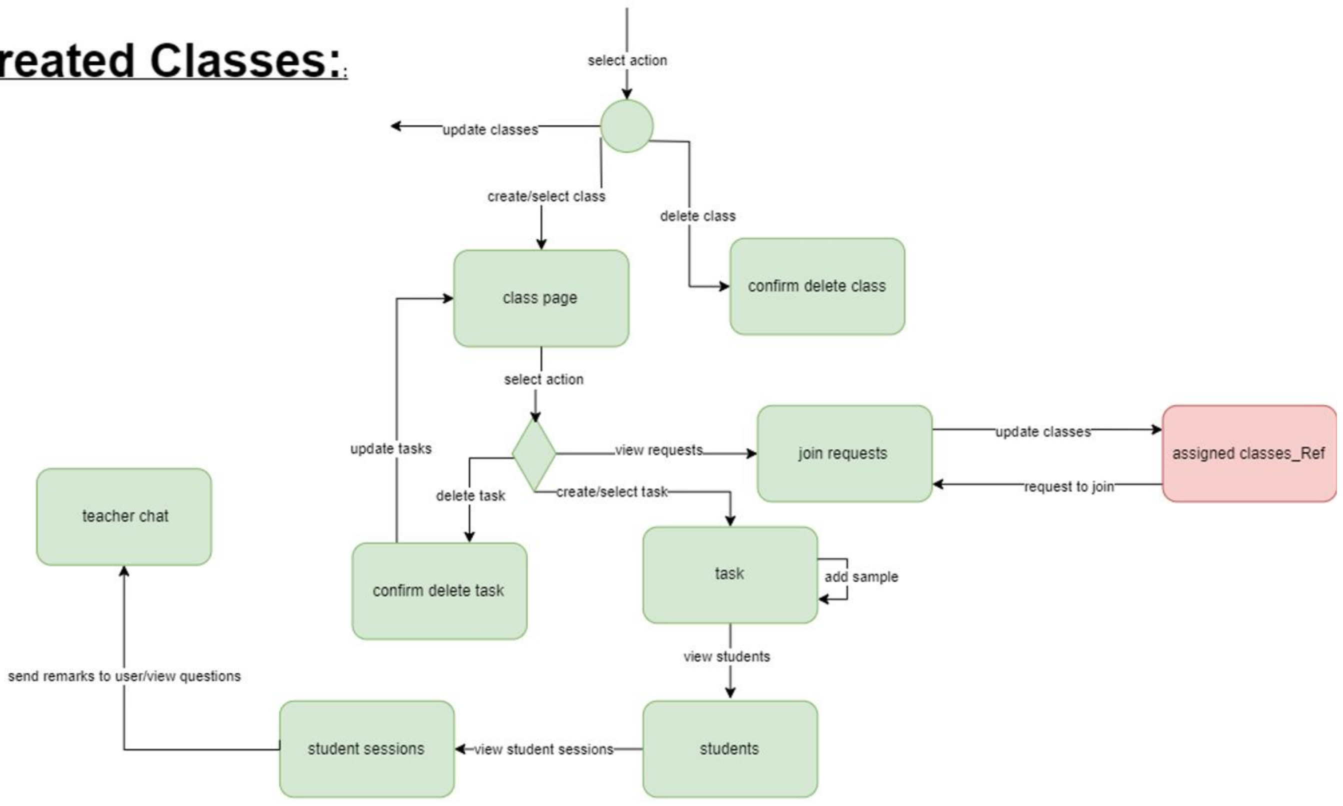
Initial State When the application is opened, it's in the "Logged_out" state
From here the user can try to log in or register. If the user exists and the information provided is correct, the system transitions to the "Logged_In_Ref" state. If the information is incorrect, the system loops back to the "Logged_out" state. If no user is found, the system transitions to the "Register" state for the user to create a new account. Once logged in, the system is in the "Logged_In_Ref" state,

and the user can log out to transition back to the "Logged_out" state. The user can also close the application from the "Logged_out" state.

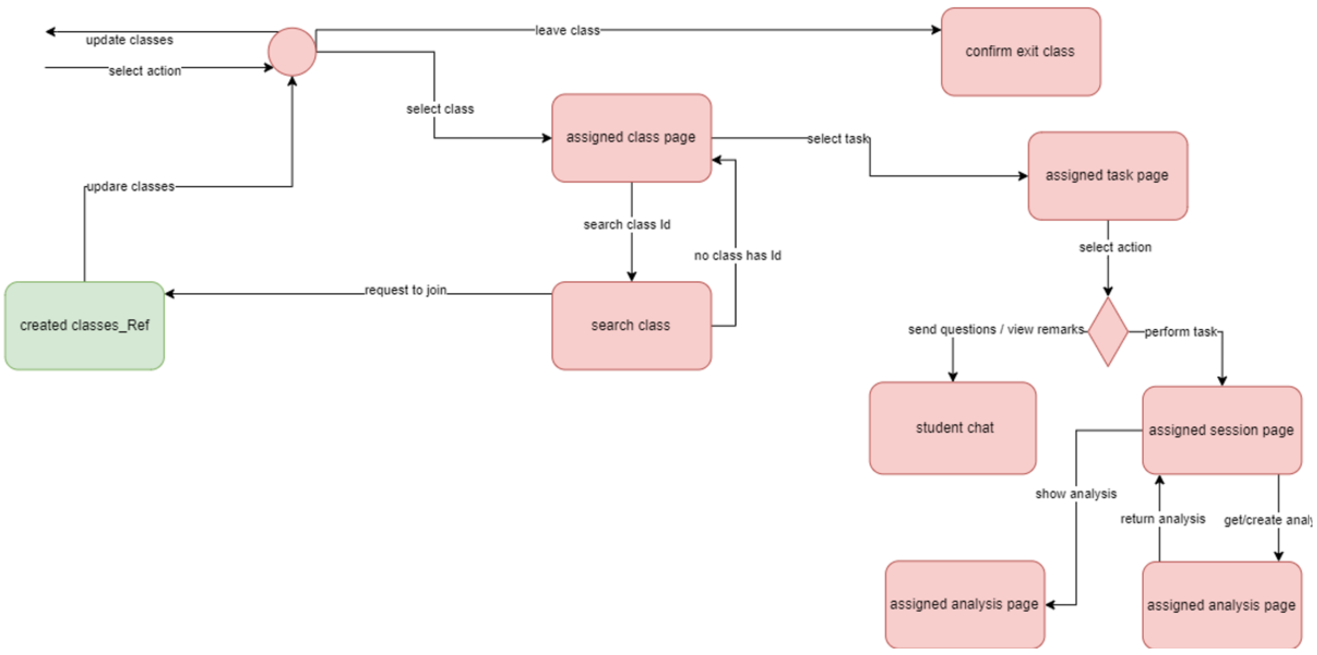
The states are saved at each transition point, especially during login attempts, registration, successful login, and logout actions. The application likely saves the state in a session or a similar persistent storage mechanism, such as local storage or a database. Login/Register Attempt: The system checks against a database or user repository.



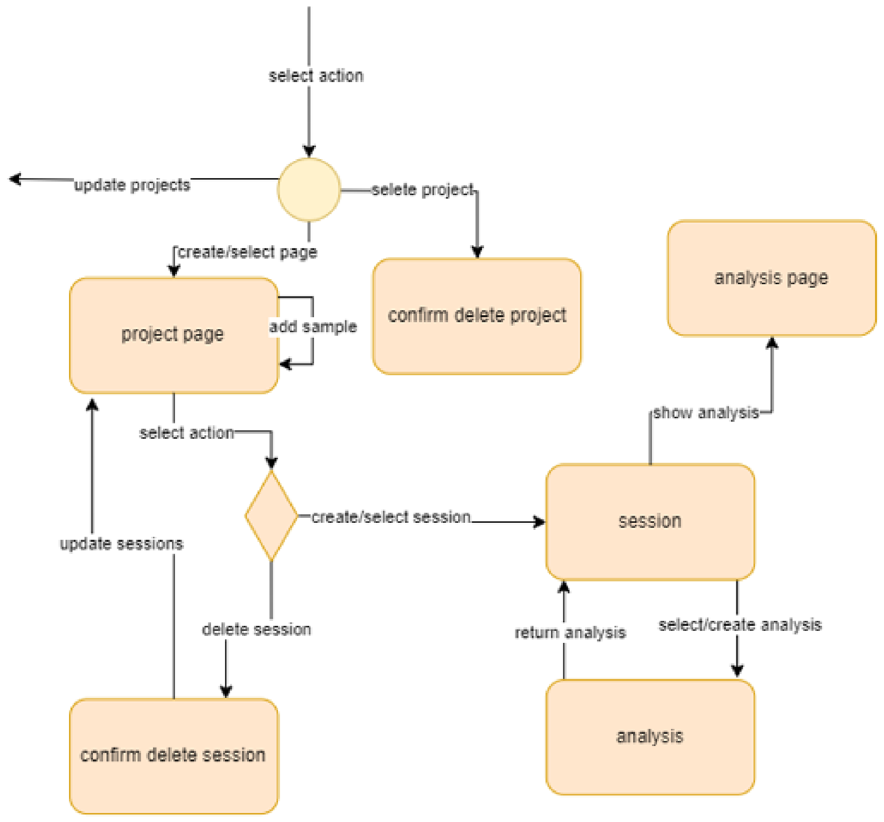
Created Classes:



Assigned Classes:



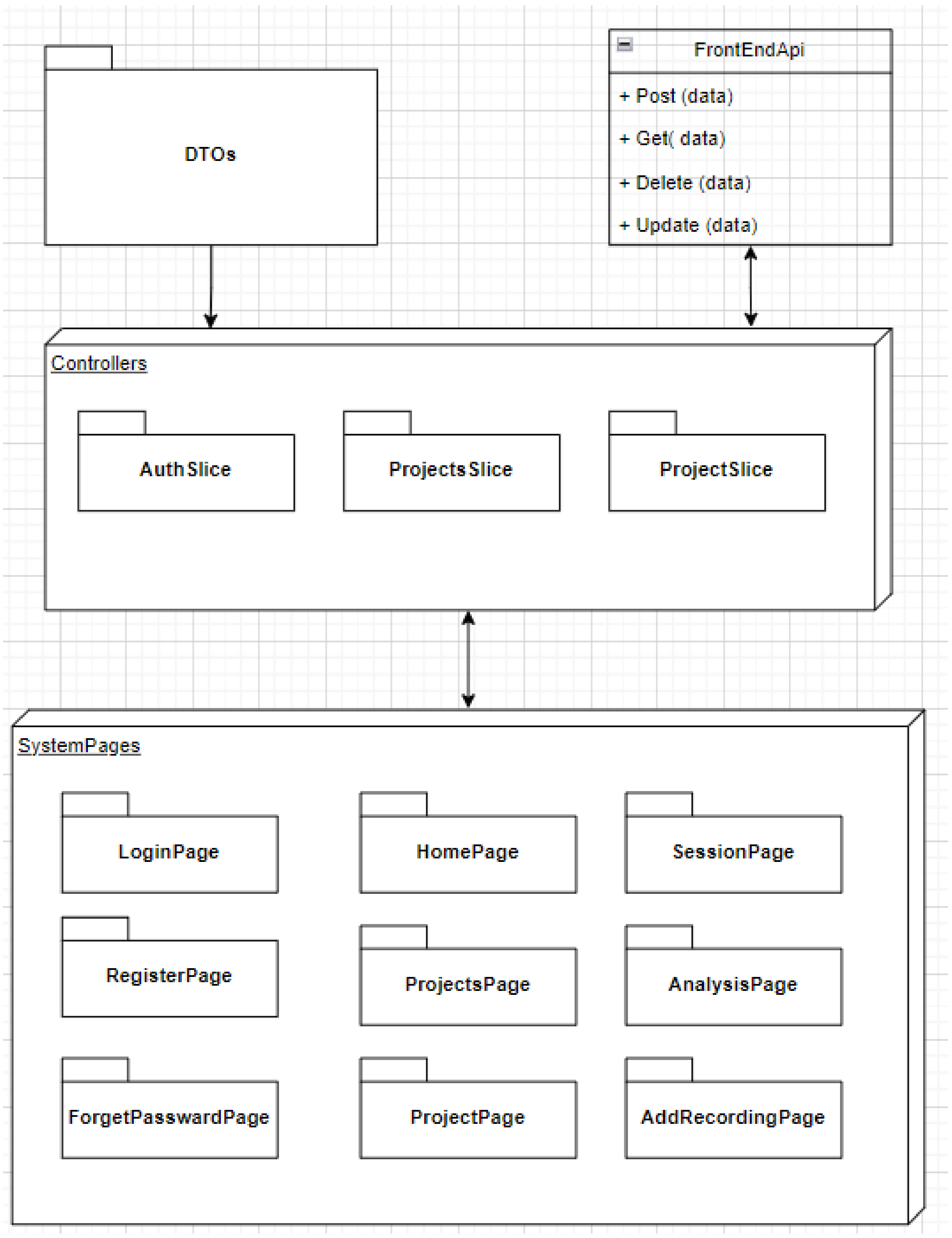
My_Projects:


select action
update projects
selete project
create/select page
project page
add sample
confirm delete project
analysis page
show analysis
select action
create/select session
session
update sessions
delete session
return analysis
select/create analysis
analysis
confirm delete session

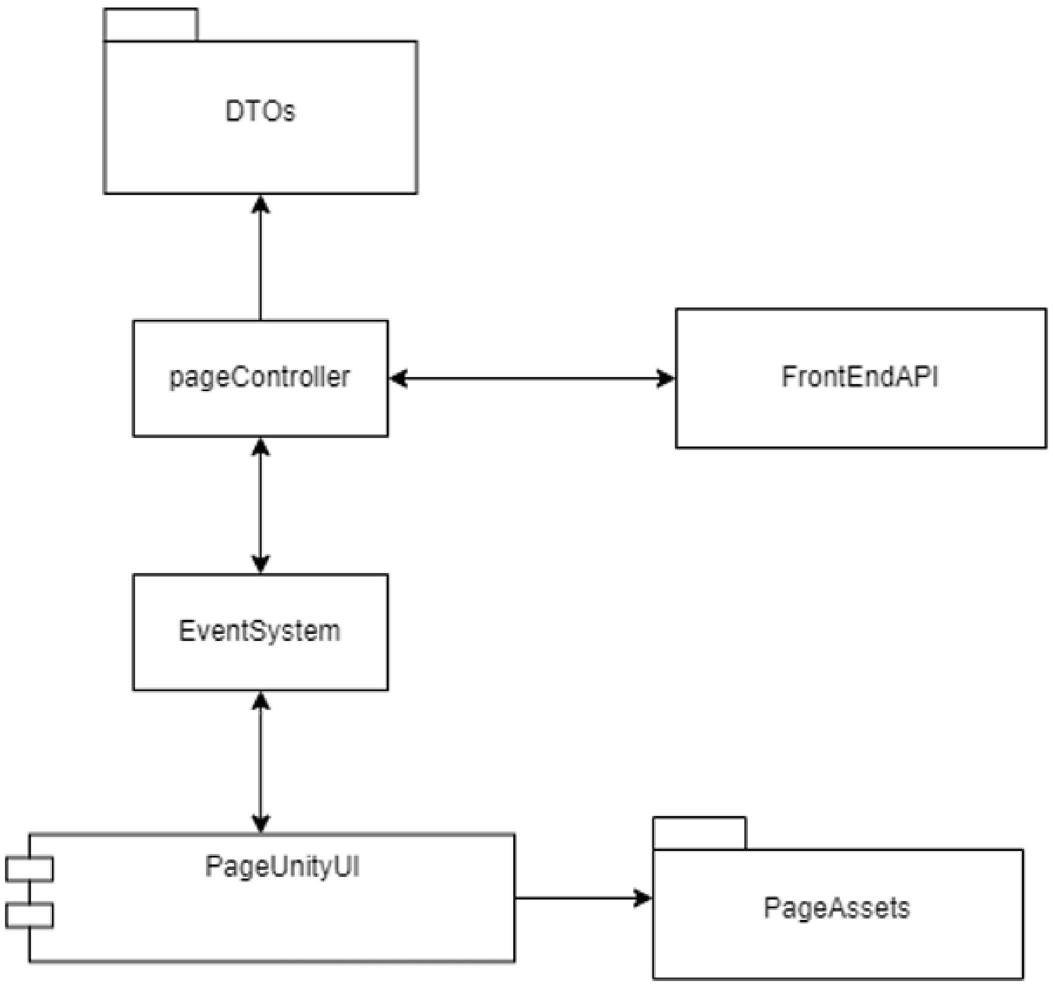
5. Object oriented Analysis

5.1 class diagrams

Overview of the client side:

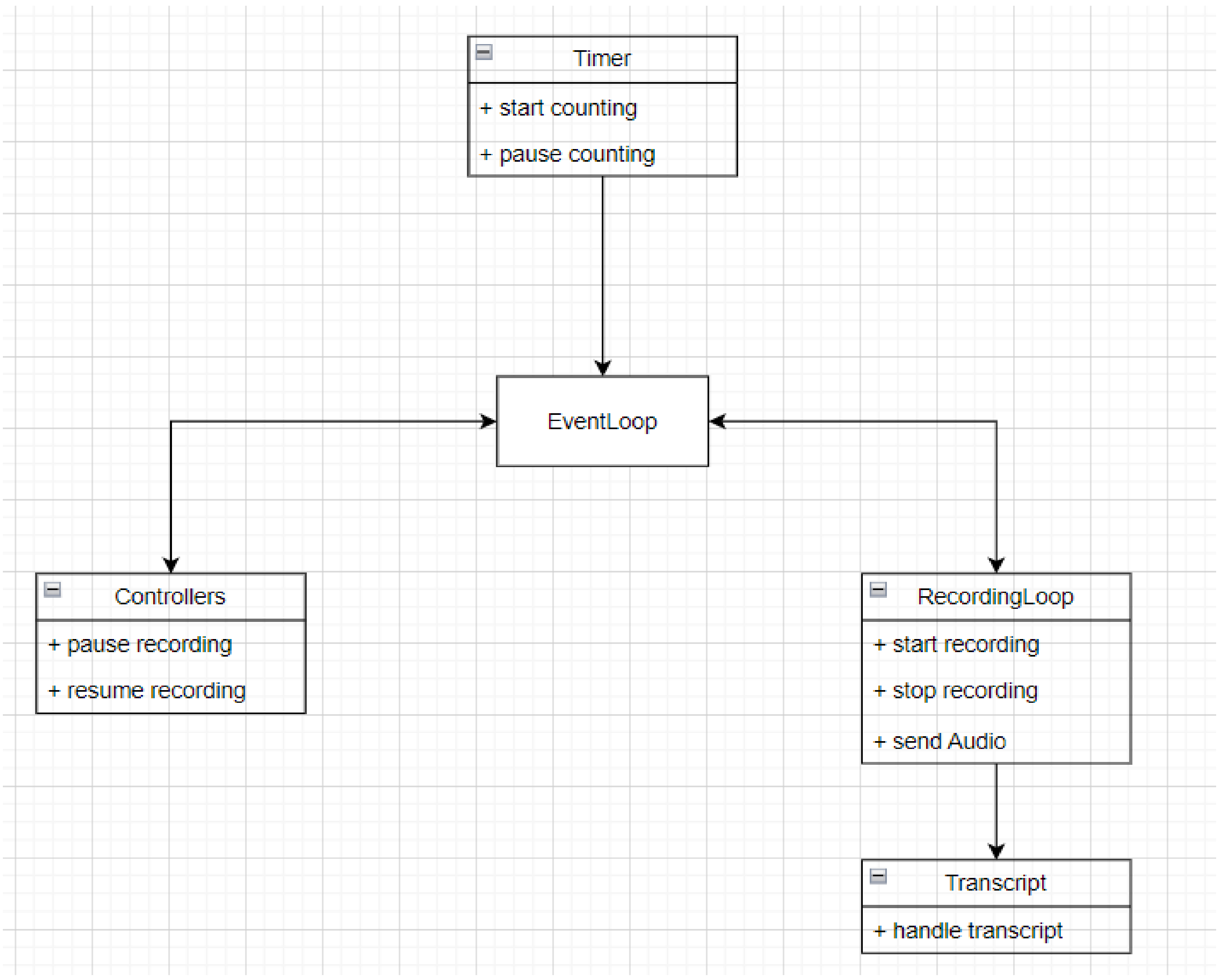


General UI page:

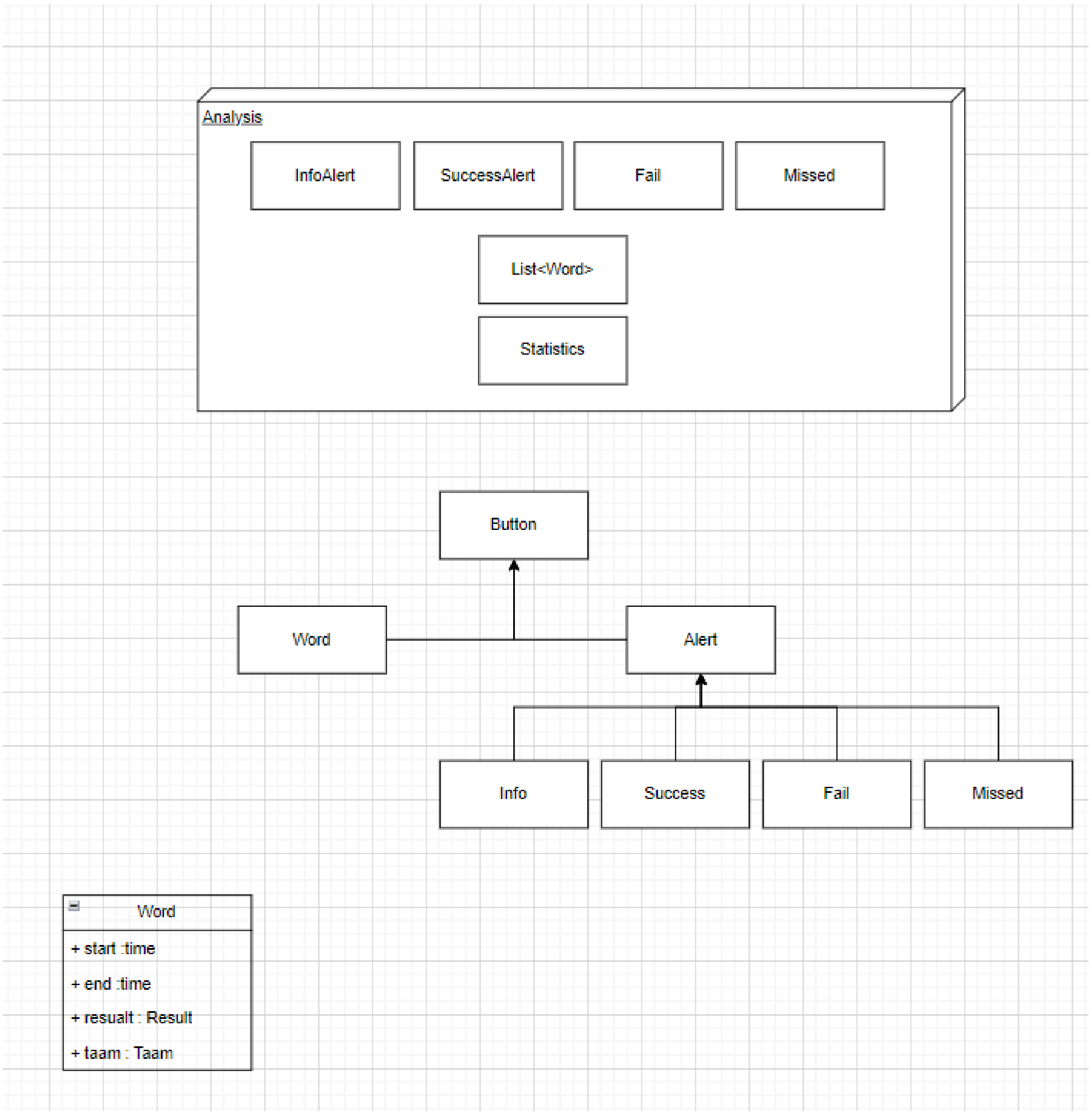


Specific info about each page will be provided in section 5.2.

Session page:



Analysis Page:

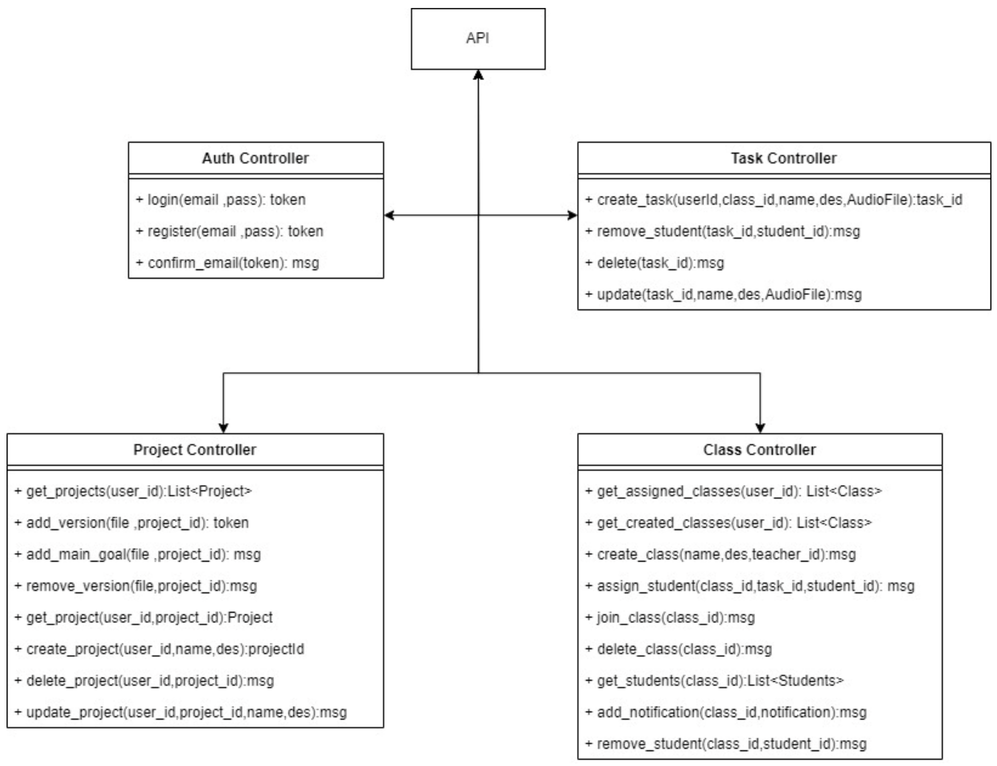


<u>Server-side Overview:</u>

5.2 Class Description

<u>General UI Page description:</u>

- <u>DTOs:</u>
  For every page, there will be dedicated DTOs to encapsulate the relevant data required for that specific page. These DTOs will be used for parsing requests and exchanging data between the controllers and APIs, they will be used during the serialization and deserialization of JSON responses and requests.
- <u>Controllers:</u>
  The controllers oversee the page's logic and handle communications with both APIs and UI tools, as well as manage the DTOs.
- <u>EventSystem:</u>
  A dedicated object on each page for connecting the UI elements with the code elements using the unity tools.


API
Auth Controller
+ login(email ,pass): token
+ register(email ,pass): token
+ confirm_email(token): msg
Task Controller
+ create_task(userId,class_id,name,des,AudioFile):task_id
+ remove_student(task_id,student_id):msg
+ delete(task_id):msg
+ update(task_id,name,des,AudioFile):msg
Project Controller
+ get_projects(user_id):List<Project>
+ add_version(file ,project_id): token
+ add_main_goal(file ,project_id): msg
+ remove_version(file,project_id):msg
+ get_project(user_id,project_id):Project
+ create_project(user_id,name,des):projectId
+ delete_project(user_id,project_id):msg
+ update_project(user_id,project_id,name,des):msg
Class Controller
+ get_assigned_classes(user_id): List<Class>
+ get_created_classes(user_id): List<Class>
+ create_class(name,des,teacher_id):msg
+ assign_student(class_id,task_id,student_id): msg
+ join_class(class_id):msg
+ delete_class(class_id):msg
+ get_students(class_id):List<Students>
+ add_notification(class_id,notification):msg
+ remove_student(class_id,student_id):msg

- PageUnityUI:
  The UI components are managed by the react native engine.
- Page Assets:
  Assets that are being used by a specific page like images and animations.
- FrontEndAPI:
  Handles the connection and synchronization with the server, managing interactions with the controllers through DTOs.

The functions of the UI Page will be handheld in the controllers and will contain the general crud operations for the specific data and DTO's of the Page.

Crud operations:

- Create Operation (POST):
  - Description: This operation creates a new entity within the system.
  - Pre-conditions: Valid input data is provided. The entity to be created does not already exist in the system.
  - Post-conditions: The new entity is successfully created and stored in the system. The system returns a successful response with the newly created entity's details.
- Read Operation (GET):
  - Description: This operation retrieves existing entity/ies from the system.
  - Pre-conditions: The entity to be retrieved exists in the system.
  - Post-conditions: The requested entity/ies are successfully retrieved from the system. The system returns a successful response with the retrieved entity/ies' details.
- Update Operation (PUT/PATCH):
  - Description: This operation updates existing entity/ies in the system.

    - Pre-conditions: Valid input data is provided. The entity to be updated exists in the system.
    - Post-conditions: The specified entity is successfully updated with the provided data. The system returns a successful response with the updated entity details.
- Delete Operation (DELETE):
    - Description: This operation deletes existing entity/ies from the system.
    - Pre-conditions: The entity/ to be deleted exists in the system.
    - Post-conditions: The specified entity is successfully deleted from the system. The system returns a successful response confirming the deletion.

<u>Session Page Description:</u>

- Voice Sample visualizer:
  responsible for the visualization of the sample audio clip that the users want to learn, using the Unity engine and the relevant data and functions.
    - loadClip – loads the clip that the user chooses from his device.
    - StartVix – initializing the page for the visualization.
    - FixedUpdate – visualizing the sample audio using Unity engine tools.
- VoiceRecordingVisualizer – responsible for visualization and management of the input audio clip that the user performs and gives instant feedback to the user.
    - startRecording – starts the recording of the user and initializing all the required components.
    - StartVix – initializing the page for the visualization.
    - FixedUpdate – visualizing the sample audio using Unity engine tools.

    - stopRecording – stops the session and saves the recording for future analysis.
    - TrimSilence – after the session is done the functions trim unnecessary parts of the user recording.
- ClockScript – responsible for a clock that is presented for the users to see.
- AudioRecorder – responsible for recording the audio from the user session and saving it in the relevant place.
    - save recording – saves the recording on the user's device until synchronization with the server.
    - writeWaveHeader – used for transforming the user recording to a WAV format.
- PitchEstimator – responsible for pitch estimation given the current user session in real-time.
    - Estimate – gets the user sound as input and calculates the pitch accordingly.
    - GetSpectrumAmplitude – calculates the amplitude for the estimator to use.
- AnalysisScript – responsible for analyzing in real-time the user's input sound.
    - Analyze – receives the input sound and analyzes the input in real-time using the estimator.
    - TrimSilence – trims the silent parts of the input sound for better analysis.

Analysis Page Description:

This is a special instance of UI Page. Note that the following functions use the librosa library.

- createFeedbackDict:
    - Functionality: Creates and initializes a feedback dictionary with predefined feedback messages for different types of deviations.
    - Preconditions: None.

- Postconditions: The feedback_dict dictionary is populated with predefined feedback messages.

- load_and_normalize_audio:
  - Functionality: Loads an audio file, trims silence, and normalizes it.
  - Preconditions: Requires a valid file path to the audio file.
  - Postconditions: Returns the normalized audio signal (y_normalized) and its sampling rate (sr).
- extract_pitch:
  - Functionality: Extracts the pitch of an audio signal.
  - Preconditions: Requires the audio time series (y), its sampling rate (sr), and optionally minimum and maximum frequencies for analysis.
  - Postconditions: Returns the extracted pitch values (pitch_values) and their corresponding times.
- extract_rhythm:
  - Functionality: Extracts rhythm and tempo from an audio signal.
  - Preconditions: Requires the audio time series (y) and its sampling rate (sr).
  - Postconditions: Returns the tempo, beats, times, and onset envelope of the audio signal.
- extract_note_durations_and_times:
  - Functionality: Extracts note durations and start times from an audio file.
  - Preconditions: Requires a valid file path to the audio file.
  - Postconditions: Returns the durations, start times, and sampling rate of the audio file.
- extract_vowel_formants:
  - Functionality: Extracts the first two formant frequencies of vowels over time from an audio signal.
  - Preconditions: Requires the audio signal, its sampling rate, and optional parameters for window length and time step.

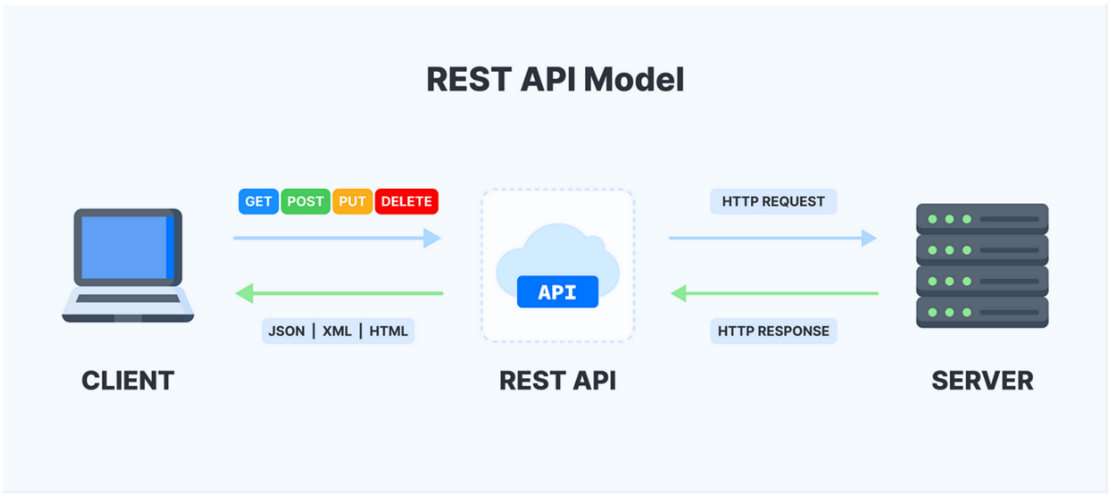
    - Postconditions: Returns the times, F1, and F2 formant frequencies.
- frequency_to_note_name:
    - Functionality: Converts frequencies to note names, applying a threshold to filter out less dominant notes and sampling to reduce the number of notes plotted.
    - Preconditions: Requires an array of frequencies and the sampling rate.
    - Postconditions: Returns sampled note names and their corresponding times.
- plot_combined_analysis:
    - Functionality: Plots combined analysis (waveform, spectrogram, pitch, rhythm/tempo) for two audio signals.
    - Preconditions: Requires the audio signals (y1, y2), their sampling rates, and optional titles and save path.
    - Postconditions: Generates and saves a combined analysis plot.
- generate_rhythm_tempo_feedback:
    - Functionality: Generates feedback based on rhythm and tempo comparison between two audio signals.
    - Preconditions: Requires tempo and beat information for both audio signals.
    - Postconditions: Returns feedback based on rhythm and tempo comparison.
- cluster_deviations:
    - Functionality: Clusters deviations that are close in time.
    - Preconditions: Requires a list of time points with deviations.
    - Postconditions: Returns a list of clusters, where each cluster is a list of deviations close in time.
- format_cluster_times:
    - Functionality: Formats clusters of deviations for feedback.

    - Preconditions: Requires a list of clusters with deviation times.
    - Postconditions: Returns a string representation of clustered deviation times.
- generate_pitch_feedback:
    - Functionality: Generates feedback based on pitch track comparison between two audio signals.
    - Preconditions: Requires pitch values and times for both audio signals.
    - Postconditions: Returns feedback based on pitch track comparison.
- generate_duration_feedback:
    - Functionality: Generates feedback based on the duration comparison between two audio signals.
    - Preconditions: Requires note durations, start times, sampling rate, and a threshold for considering duration mismatches significant.
    - Postconditions: Returns feedback based on duration comparison.
- generate_vowel_feedback:
    - Functionality: Generates feedback based on the vowel formant frequency comparison between two audio signals.
    - Preconditions: Requires formant frequencies and times for both audio signals.
    - Postconditions: Returns feedback based on vowel formant frequency comparison.
- analyze_and_compare:
    - Functionality: Analyzes and compares two audio files, generating various types of feedback.
    - Preconditions: Requires valid file paths to the two audio files.
    - Postconditions: Generates feedback based on the analysis and comparison of the audio files.

Server-Side Description:

Since most of the analysis happened on the front side, the system is built so the user can save their progress accept tasks, and share his achievements with others.

This is why we chose Rest API architecture.



Every action made by the user causes some web requests to the server (CRUD operations), and then the controllers at each route manipulate the data that is stored in the database.

The controllers are named after the model in the db. they're in charge of, and the request gets to them by the routes defined in the system.

Controllers:

Auth Controller:

This controller handles the auth actions: login, register … and manipulate the User model.

- Register: register new users to the system

  Pre-condition:

  - The user does not exist in the system.
  - The user entity is not confirmed as a valid user.

Post-condition:

  - The user entity created in the system.
  - The user entity is confirmed as a valid user.

- Login: accessing a user to the system

  Pre-condition:

  - the user exists in the system.
  - the user confirmed as a valid user.
  - the user is not connected to the system.

  Post-condition:

  - the user is connected to the system.

- confirm_email: send an email verification message to the user and confirm their account.

  Pre-condition:

  - the user exists in the system.
  - the user is not confirmed as a valid user.
  - the user is not connected to the system.

  Post-condition:

  - the user gets a verification msg.

Class Controller

This controller handles the crud operations for the Class model.

- create_class: creates a new class and assigns its creator as the teacher.

  Pre-condition:

  - the user exists & login to the system.

  Post-condition:

  - the class entity created by the parameters.
  - The creator it’s the class's teacher.

- Join_class: join a new student to the class.

  Pre-condition:

  - the class exists in the system.
  - the student is not a member of this class.

  Post-condition:

  - the student is a member of this class.
- remove_student: removes student from a class

  Pre-condition:

  - the student exists in the system.
  - the student is a member of the class.

  Post-condition:

  - the student is no longer a member of the class.
- get_assigned_classes: get the classes that a user is a member of.

  Pre-condition:

  - the user exists in the system.

  Post-condition:

  - None
- get_created_classes: get the classes that a user has created.

  Pre-condition:

  - the user exists in the system.

  Post-condition:

  - None
- delete_class: deleting the class entity.

  Pre-condition:

  - the class exists in the system.

  - the sender of the request it’s the teacher

  Post-condition:

  - the class no longer exists in the system.
  - The student is not assigned to this class anymore.
  - Any task in this class is not deleted.
- get_students: return the student that a member of this class.

  Pre-condition:

  - the class exists in the system.

  Post-condition:

  - None
- add_notification: send msg to the students.

  Pre-condition:

  - the class exists in the system.
  - there is at least one student in this class.

  Post-condition:

  - each member of this class gets a notification.
- assign_student: assign a task to a specific student.

  Pre-condition:

  - the student exists in the system.
  - the student belongs to the class of this teacher.

  Post-condition:

  - A new task has been created and assigned to this student.

Project Controller

This controller handles the crud operations for the Project model and Session model.

No need to specify the create, get, update, or delete methods.

- add_main_version: creates Session and assigns it to a project as the main goal.

  Pre-condition:

  - the project exists in the system.

  Post-condition:

  - A new Session has been created and assigned as the main goal.
- add_version: creates Session and assigns it to a project as a new version.

  Pre-condition:

  - the project exists in the system.

  Post-condition:

  - A new Session has been created the assigned as a new version.

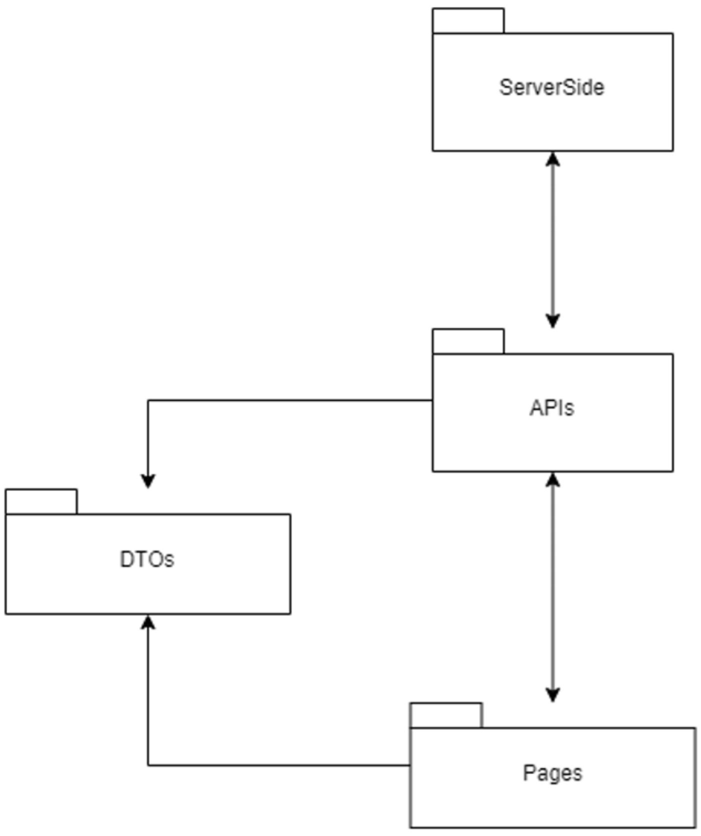
Task Controller

This controller handles the crud operations for the Task model.

No need to specify the create, get, update, or delete methods.

5.3 Packages

Our app will be divided into the following packages:



Server-Side Package: contains all the server-side components like controllers and DB configurations.

APIs: contains all the APIs and their configurations on the client side.

DTOs: contains all the DTOs of the client side for the communications between the system components.

Pages: contains all the application pages with the relevant logic and assets.

5.4 Unit testing

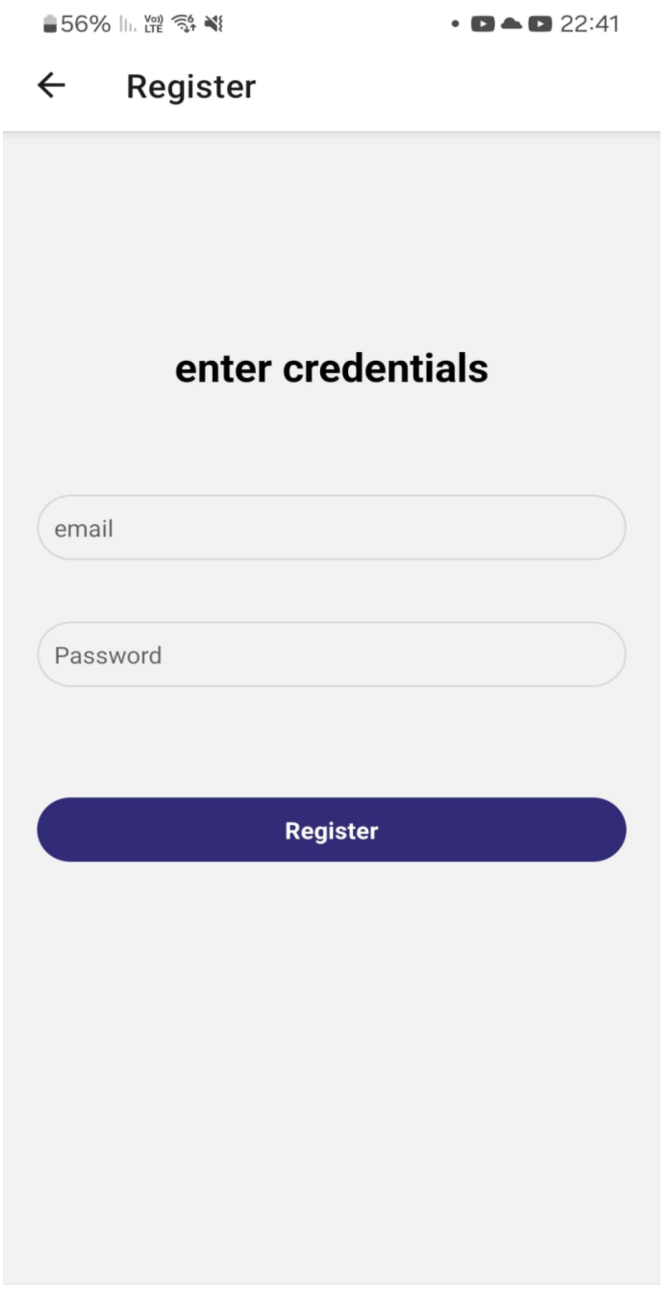
Our unit testing will primarily focus on the controller operations and checking that the system states changes according to the action that was performed.

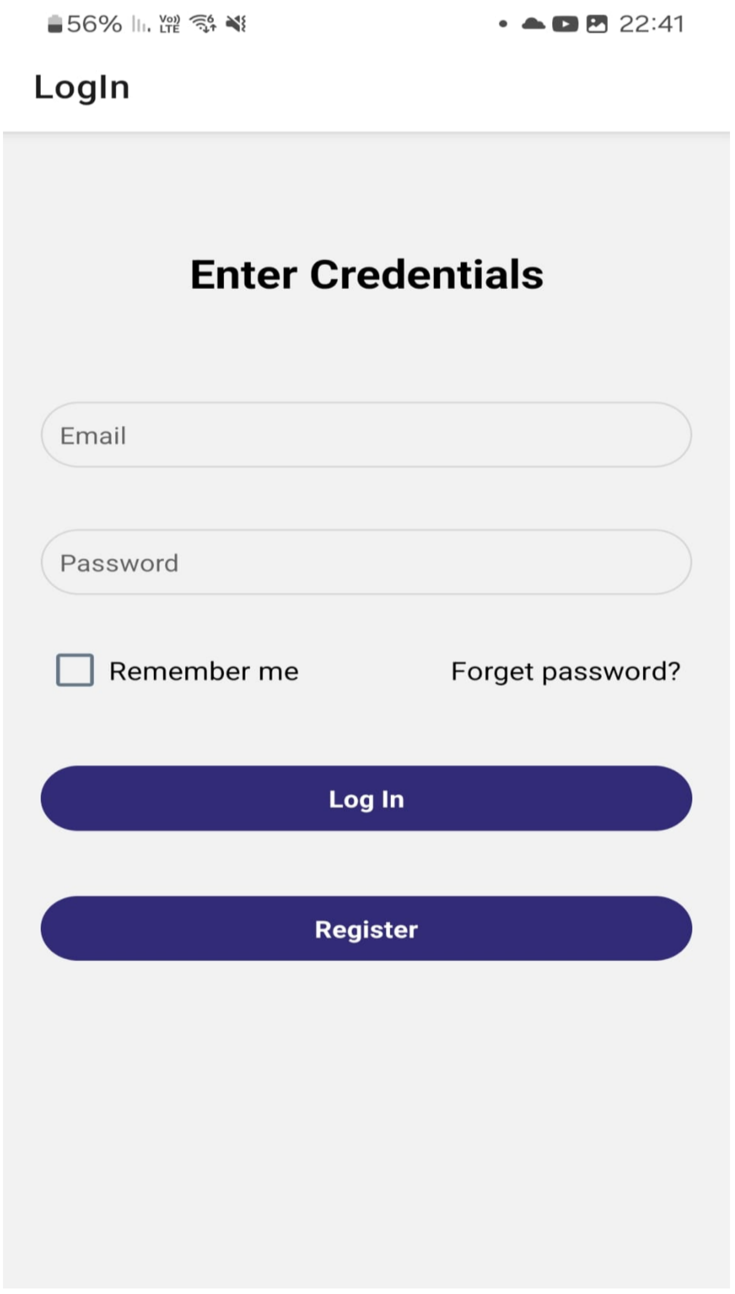
Function Name	Test description	Expected result
Create	Create with valid input	A new entity is successfully created and stored in the system, and a successful response is returned.
Create	Create with invalid input	The operation fails and returns an appropriate error response.
Read	Retrieve an existing entity	The requested entity is successfully retrieved from the system, and a successful response is returned.
Read	Retrieve a non–existing entity	The operation fails and returns an appropriate error response.
Update	Update with valid input data	The specified entity is successfully updated with the provided data, and a successful response is returned.
Update	Update with invalid input data	The operation fails and returns an appropriate error response.
Delete	Delete an existing entity	The specified entity is successfully deleted from the system, and a successful response confirms deletion.
Delete	Delete a non-existing entity	The operation fails and returns an

		appropriate error response.

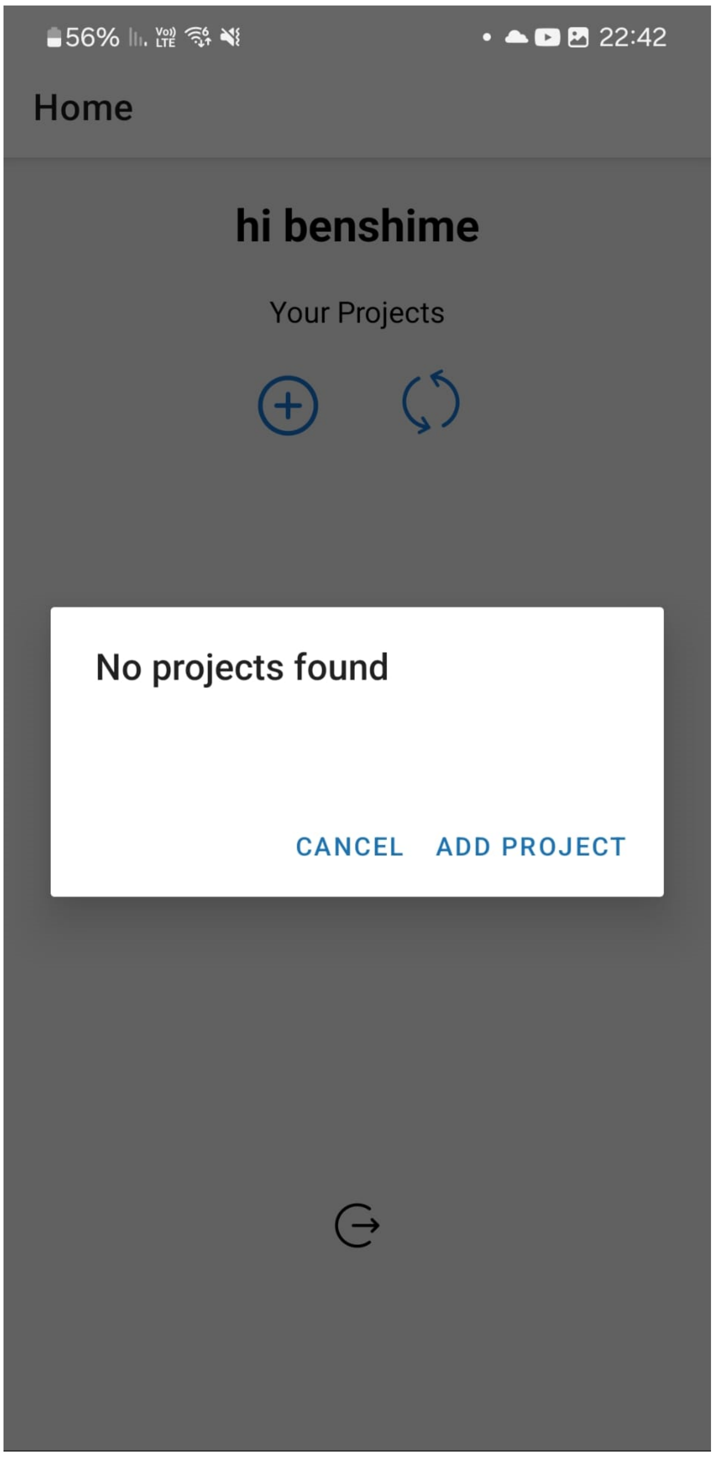
6. User Interface Draft

Login and Register Pages:





Projects Page

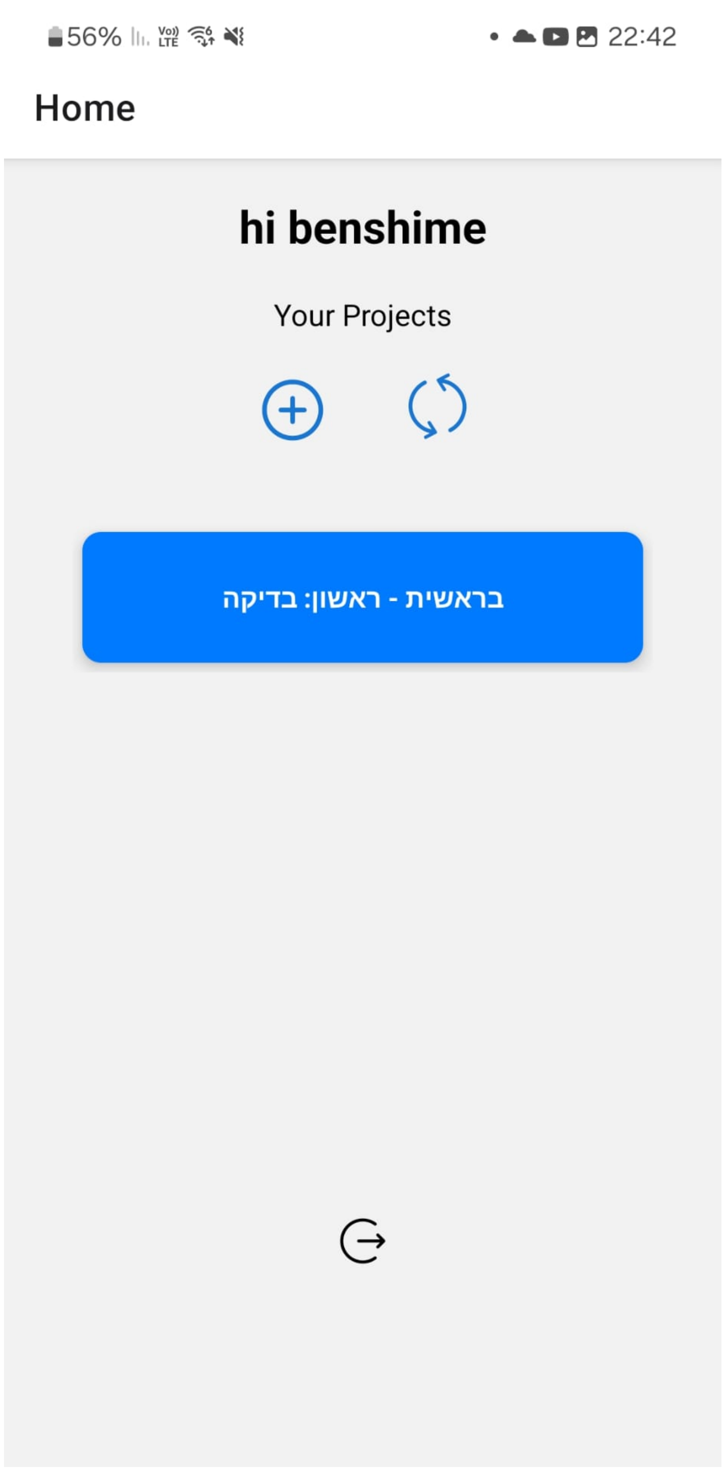

56% 22:42

Home

hi benshime

Your Projects

No projects found

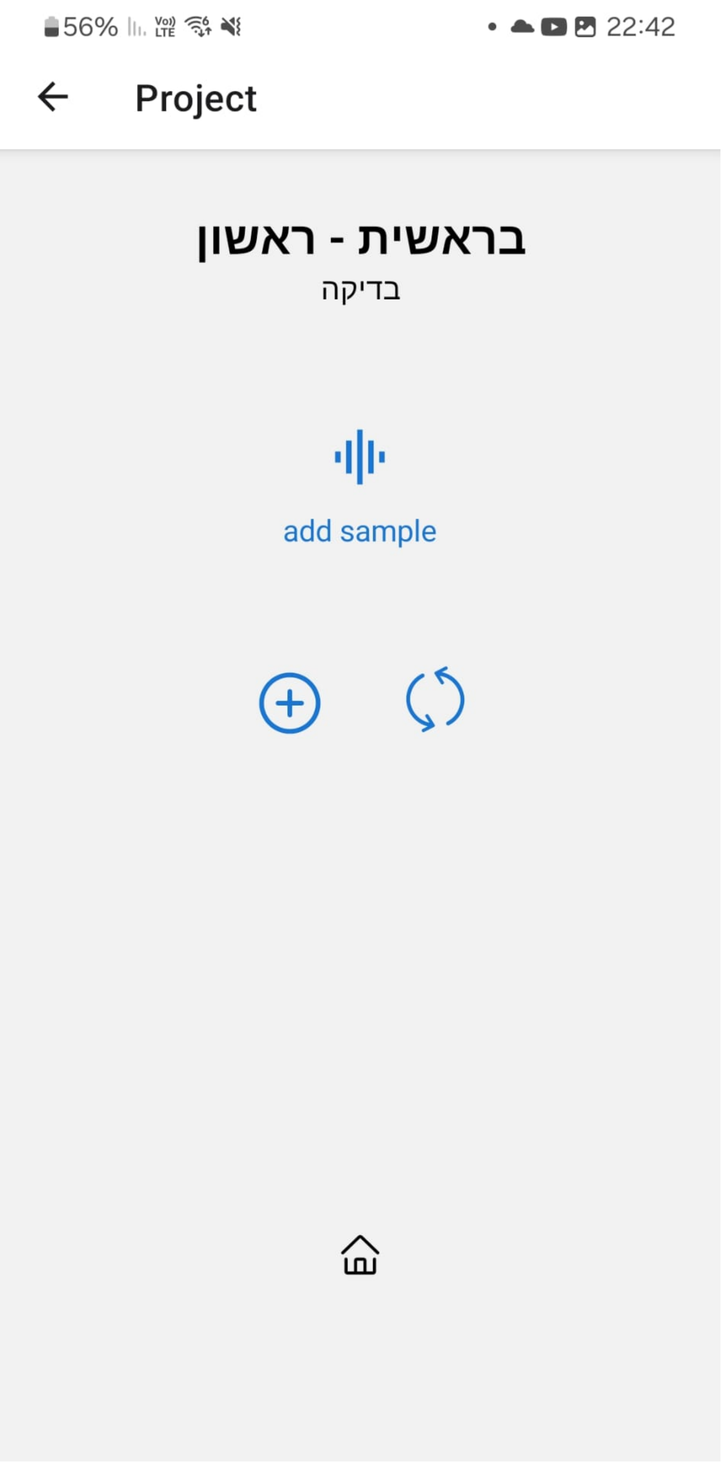

56% 22:42

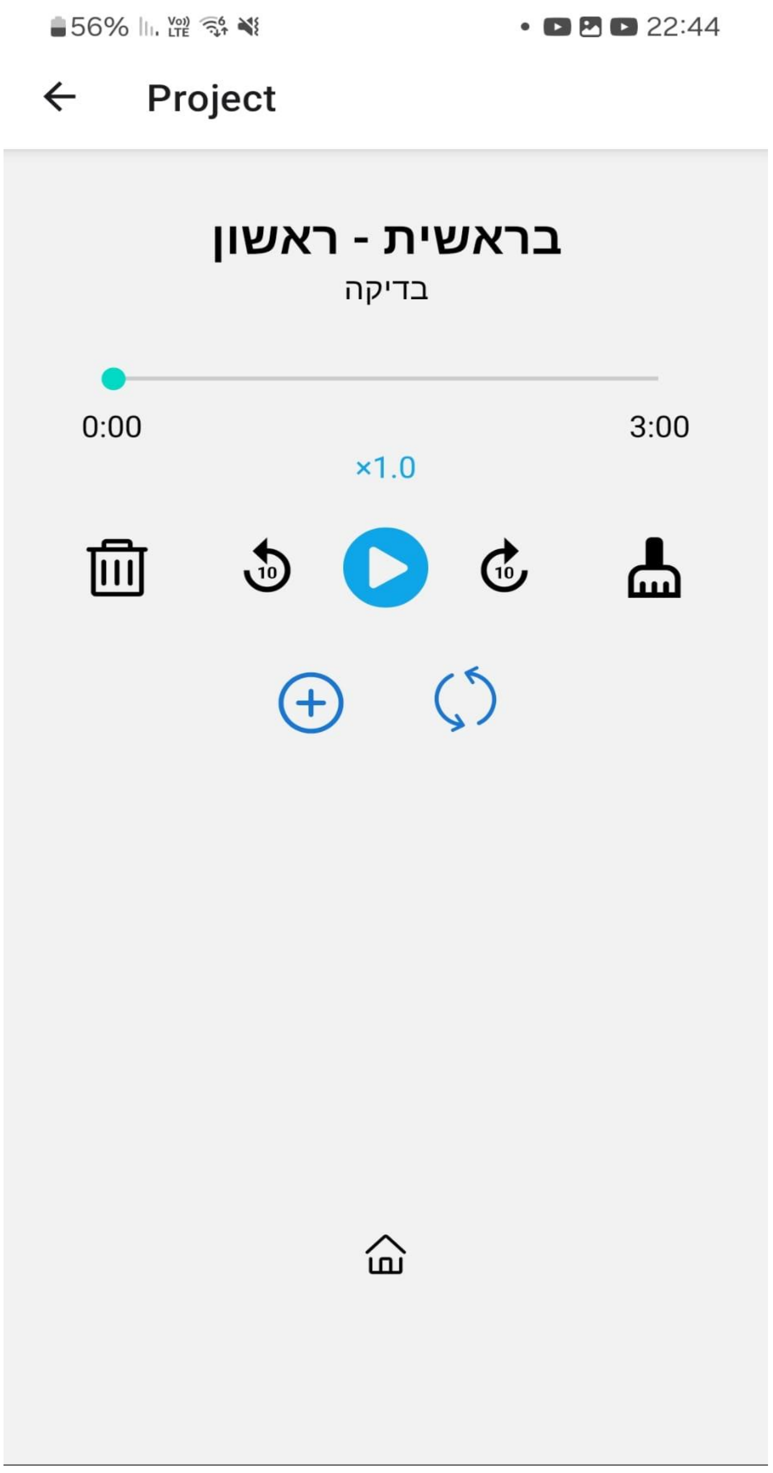
Home

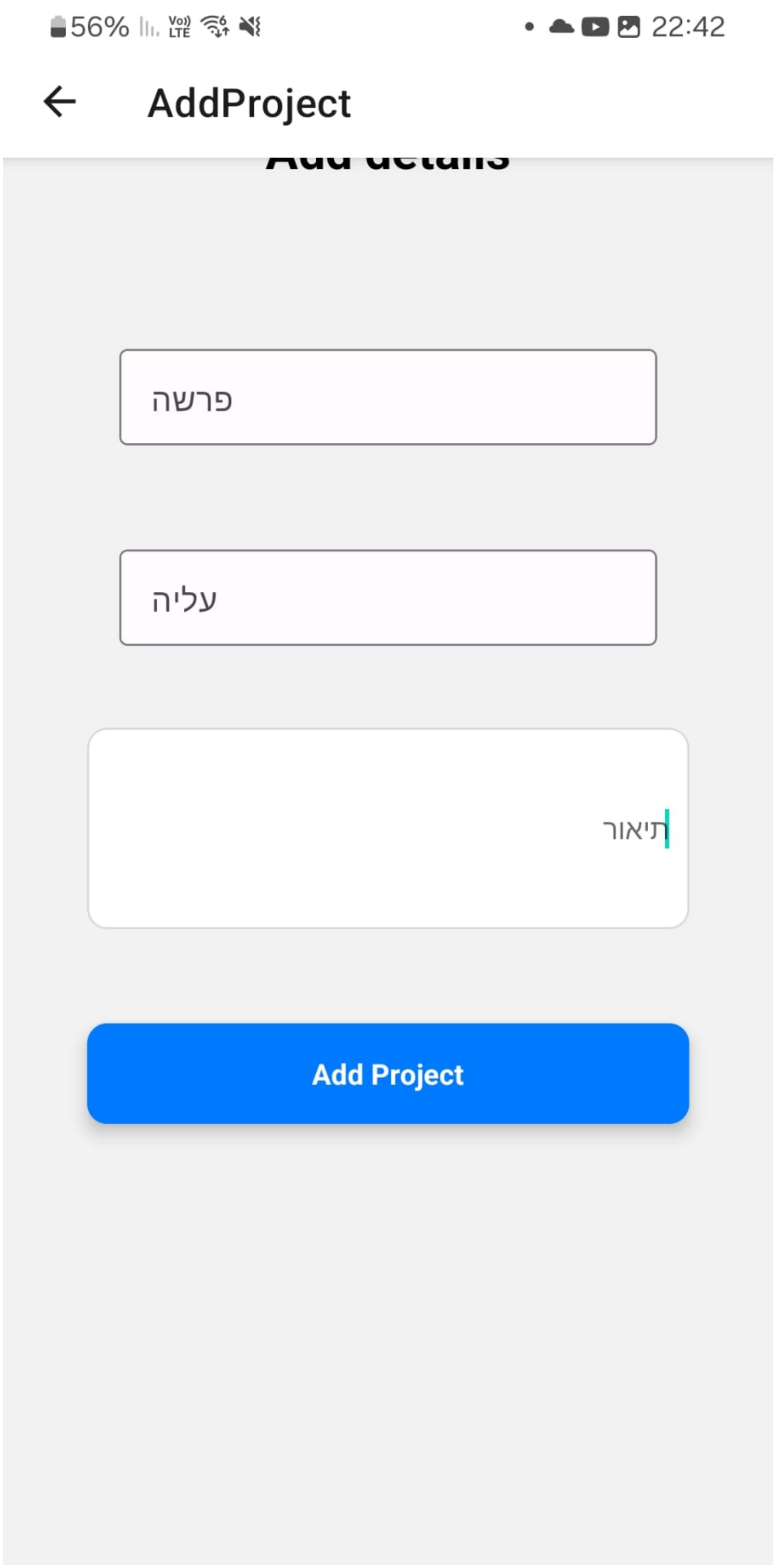
hi benshime

Your Projects

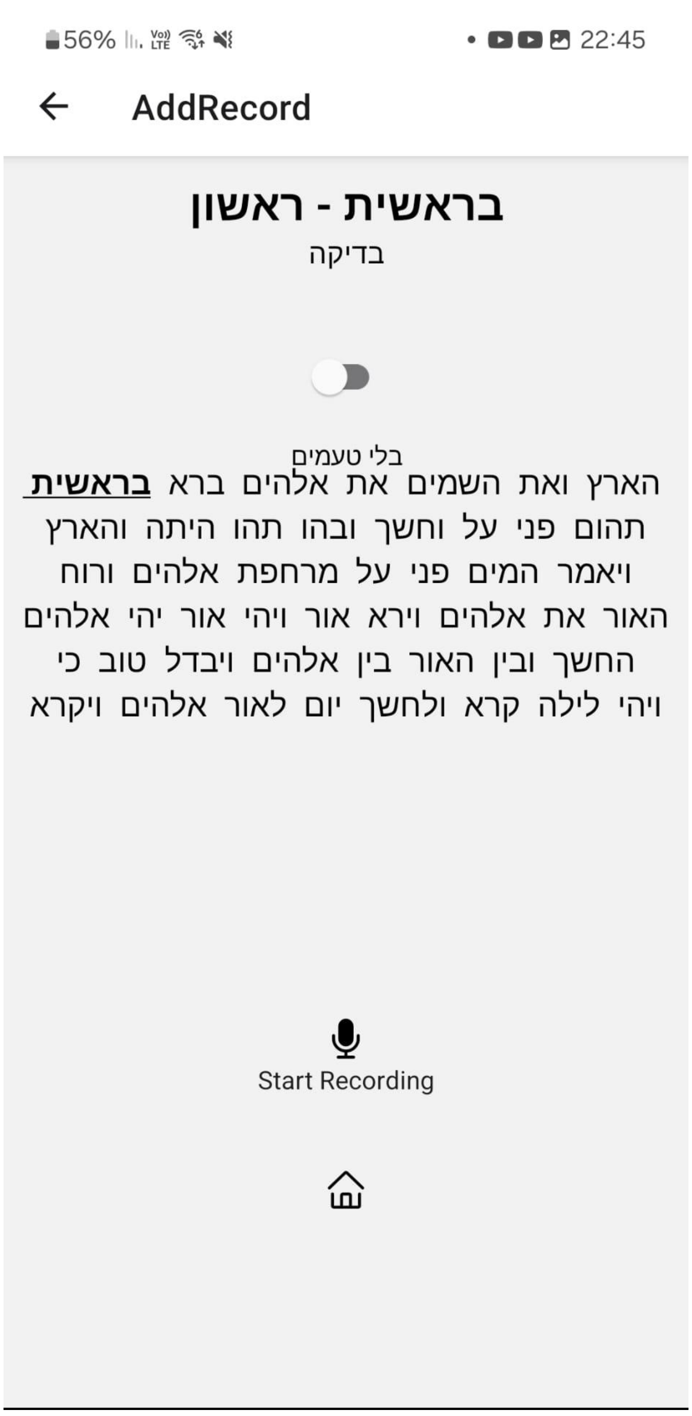
Project page:






56% 22:42
AddProject
Add details
פרשה
עליה
תיאור
Add Project

Session Page:



56% 22:45

AddRecord

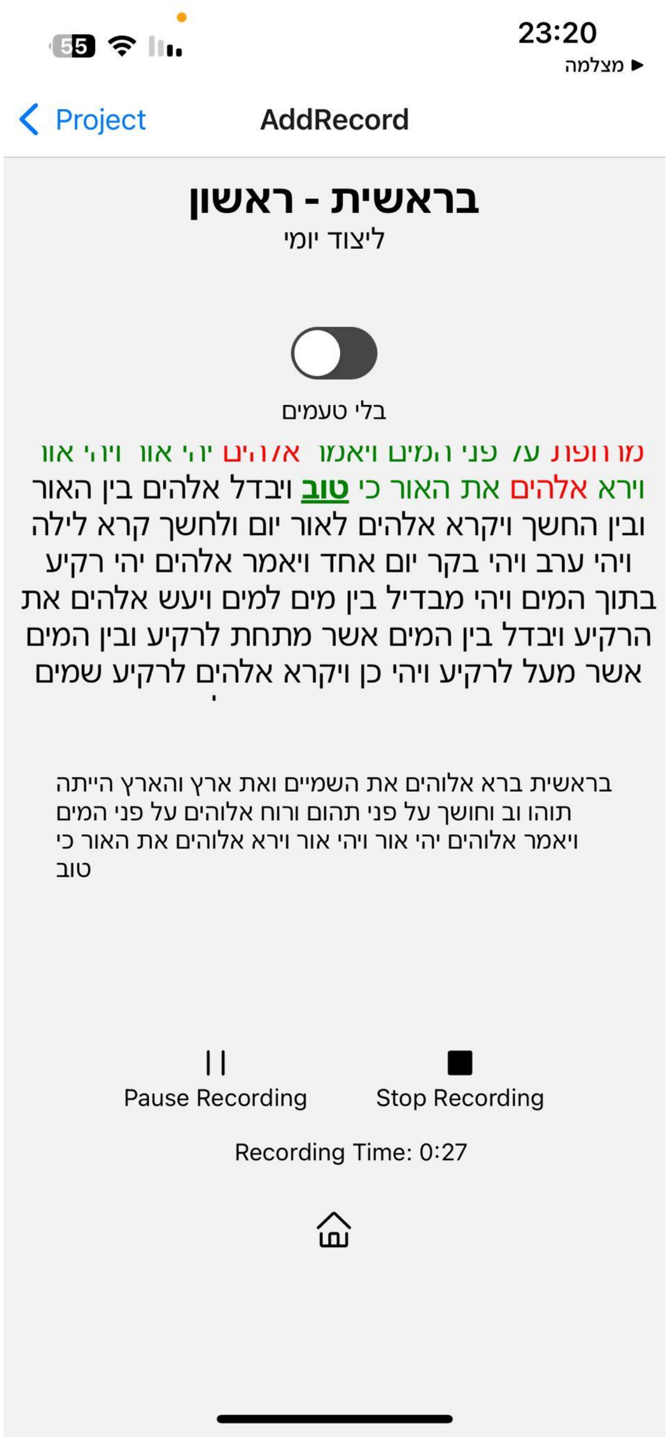
בראשית - ראשון

בדיקה

בלי טעמים

בראשית ברא אלהים את השמים ואת הארץ והארץ היתה תהו ובהו וחשך על פני תהום ורוח אלהים מרחפת על פני המים ויאמר אלהים יהי אור ויהי אור וירא אלהים את האור כי טוב ויבדל אלהים בין האור ובין החשך ויקרא אלהים לאור יום ולחשך קרא לילה ויהי

Start Recording



23:20

מצלמה ◀

Project

AddRecord

בראשית - ראשון

ליצור יומי

בלי טעמים

מרחפת על פני המים ויאמר אלהים יהי אור ויהי אור וירא אלהים את האור כי **טוב** ויבדל אלהים בין האור ובין החשך ויקרא אלהים לאור יום ולחשך קרא לילה ויהי ערב ויהי בקר יום אחד ויאמר אלהים יהי רקיע בתוך המים ויהי מבדיל בין מים למים ויעש אלהים את הרקיע ויבדל בין המים אשר מתחת לרקיע ובין המים אשר מעל לרקיע ויהי כן ויקרא אלהים לרקיע שמים

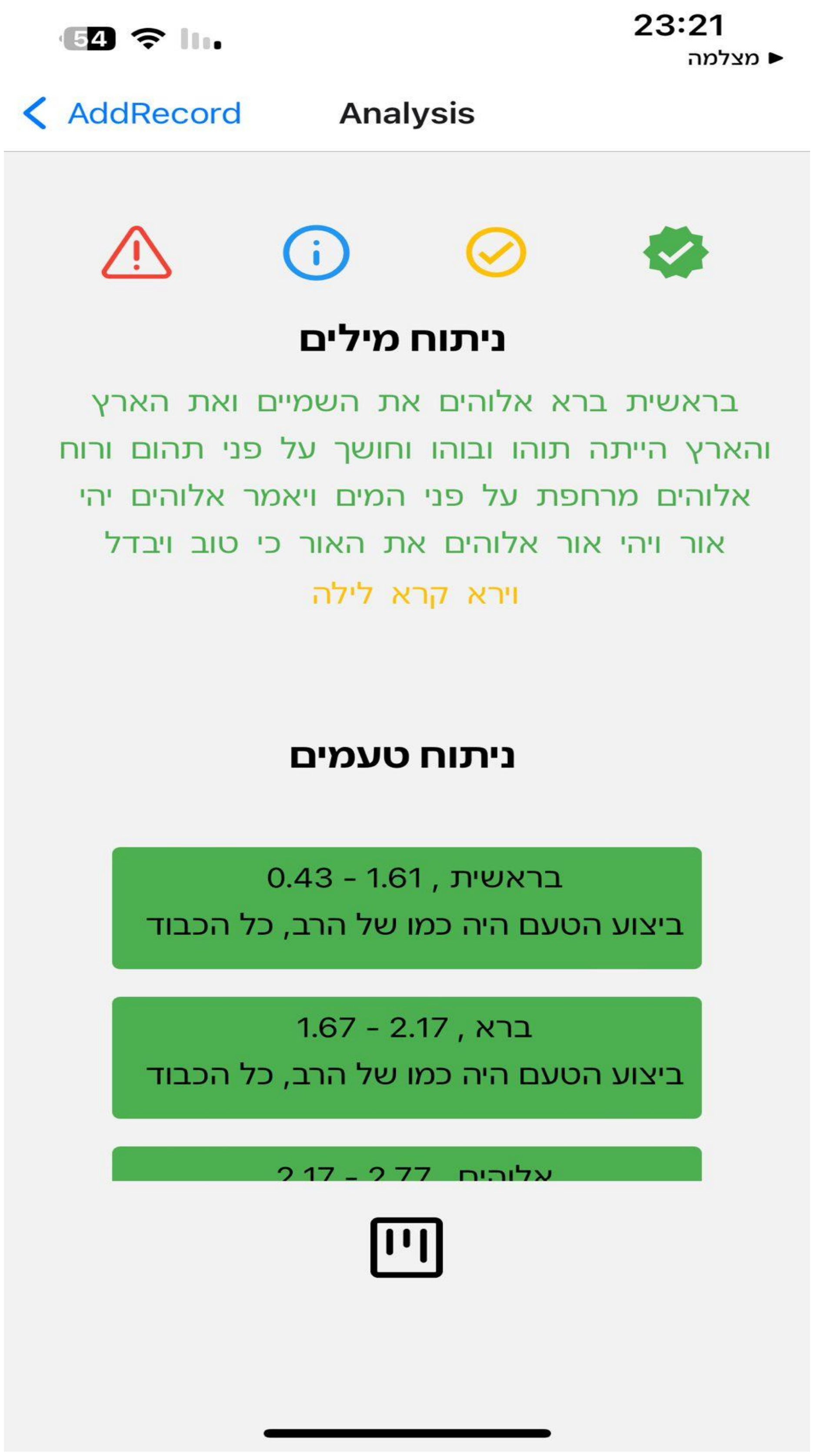
בראשית ברא אלוהים את השמיים ואת ארץ והארץ הייתה תוהו וב וחושך על פני תהום ורוח אלוהים על פני המים ויאמר אלוהים יהי אור ויהי אור וירא אלוהים את האור כי טוב

Pause Recording

Stop Recording

Recording Time: 0:27

Analysis Page:



23:21

מצלמה ◄

AddRecord Analysis

ניתוח מילים

בראשית ברא אלוהים את השמיים ואת הארץ והארץ הייתה תוהו ובוהו וחושך על פני תהום ורוח אלוהים מרחפת על פני המים ויאמר אלוהים יהי אור ויהי אור וירא אלוהים את האור כי טוב ויבדל

וירא קרא לילה

ניתוח טעמים

בראשית , 0.43 - 1.61
ביצוע הטעם היה כמו של הרב, כל הכבוד

ברא , 1.67 - 2.17
ביצוע הטעם היה כמו של הרב, כל הכבוד

7. Testing

NON-FUNCTIONAL

ID	TEST DESCRIPTION	EXPECTED RESULT
1.	Simultaneously log in and use the application with a group of 100 users.	The application will remain stable and function as designed.
2.	Conducting real-time analysis of user recordings and delivering feedback within the coaching session.	Successfully analyze the user's recordings in real time and provides immediate feedback based on the analysis.
3.	User inserts valid credentials for registration (email and password)	The system will transfer the user to the login screen and will save his details in the DB
4.	Evaluate the system's CPU usage to ensure it does not exceed 20% of the device's total CPU capacity.	The system consistently utilizes no more than 20% of the device's CPU, indicating optimized performance.
5.	Confirm that the application's file size does not exceed 5MB	The application's total file size is at or below 5MB.
6.	No internet connection	The app alerts the user with an appropriate error message
7.	Sudden shutdown	The app will save the latest data and will ensure to sync any change so the user work will not be thrown away
8.	Upload recording (session or goal recording) retrieve the recording via the app	The uploaded recording and the retrieved recording are the same
9.	Create a variety of errors due to different reasons	Check that the log messages and error messages indeed indicate the true fault
10.	Keep track of all the changes that happened in 24 24-hour cycles (based on the app backup time)	Check that the data saved in the backup process match the data expected.
11.	Perform some actions as a user and keep track of the actions performed.	Check the system's user log and ensure the actions you executed are listed in the log.
12.	The system should encrypt the user's sensitive data like passwords and email addresses.	The user records in the DB (password and email) are encrypted.
13.	Use the app via different platforms like Chrome, edge, android, and IOS.	The app remains stable, and all the functionality executes as expected.
14.	The user (which is already registered) logs in the system	The user successfully logged in and the app presented him with the home screen
15.	Logged-in user logging out from the app	The app successfully allows the user to log out and displays the login page
16.	Unauthorized call to the server- the server gets an unknown token with the request from the client	The system will not conduct the requested action and will ask the user to log in again

Use Case Tests

ID	TEST DESCRIPTION	EXPECTED RESULT
1.	Log in as a student and start a session based on a task	The app analyzes the user recording and by comparing it to the goal recording generates feedback for the user
2.	Log in as a teacher go to a student session recording and review the session if needed	The teacher review is saved in the db under the user session and the user when logged in can view the teacher review
3.	Login	The user logged in successfully and can view his user home page and execute tasks
4.	Reregistered existing user	The app shows an error message that informs there is already a user with those details
5.	Initiate incoming calls while the app is running	The app prompts the user whether to accept the call if rejected the app continues its run otherwise the app's current state is saved and the incoming call screen pops
6.	Log in as a teacher and create a class	The class is being created, the user gets a message the process completed successfully, and he can access the new class page
7.	The student enters a class ID and requests to join the class	The app shows the user the class he is looking for based on its ID and redirects him to the class homepage where he can view details about it and send a request to join the class
9.	Teacher accessing one of his classes which has a join request and accepts or declines the request	The app shows the teacher the class requests and whether one accepts the request or not the app notifies to the user
10.	Teacher adding a task to a non-active class	The app pops an error message that indicates the reason for the error (can’t add tasks to a class that is not active)
11.	Students viewing teacher review and comment on his response	The response is listed and the teacher can view it