Software Modeling I

Season 2024-III

Project advance programming models
APP Mimo

Report From:

APP
Application and functionalities
User Stories
CRC Cards
Class diagram
Squence diagram
Flow chart
Deployment diagram

By: Kevin Estiven Lozano Duarte
Code: 20221020152

Eng. Carlos Andrés Sierra, M.Sc
Computer Engineering
Universidad Francisco José De Caldas

Index

1). APP

Mimo: application to learn how to program languages such as Python, JavaScript, SQL, HTML and Swift.

Functionalities:
Mimo works as a personalized teacher in various fields of programming, first collecting information from its users to generate a more direct approach.

- Mimo generates a library of various languages.
- User experiences to create courses and tests.
- Access to the system from any device.
- M-Learning
- An evaluation system and and be able to see your progress.

Its approach:

Mimo uses a modular approach in its programming:

1). Interactive lessons
2). Practical exercises
3). Immediate feedback
4). Gemification
5). Projects
6). Diversity of contents

We can also get to identify certain characteristics of this application is well true that there are many models similar to these applications one of the most famous examples or one of the pioneers of these applications is Duolingo because many of these DM-Learning applications store information about the courses in the following ways:

1). Course structure (databases)
2). API and Backend services
3). Cloud storage
4). Data security
5). Analysis and personalization

2). APLICATION AND FUNCTIONALATIES

This makes in applications similar to Duolingo or in this case in our application Mimo which are applications that have conquered the public because the services work as the backbone of the application to connect the Fronted in the Backend.

2.1). Microservices architecture

In this type of applications we can see a database structure with a relational model where there is a system of tables that organizes information including users, courses, lessons, progress and achievements:

2.1.1 Main services

2.1.2 APIs

2.1.3 Databases

We could also find a non-SQL model where there are documents or collections in a system like mongo must and data can be stored in document formats Jason a structure could be users courses.

2.2). Functionalities of the services

These applications probably use an-app that allows communication between the application and the server the API would handle requests like:
- get the light list of resources
- save user progress
- retrieve specific lessons

Endpoints: each functionality of the application may have a specific EndPoint. For example:
- Get/courses: retrieves all available courses.
- Post/progress: saves users' progress in the specific lesson
- Get/users/{id}: retrieves user profile information

Real-time services

These applications could employ technologies such as web sockets to manage real-time communication, which could be used to:
- Update progress in real time, without the need to reload the page or app.
- Send automatic notifications or reminders when the user reaches certain milestones or when new content is available

Authentication and session management

The system will probably use JWT tokens to manage user sessions those tokens are generated at the time of login and attached in the HTTP request headers to authenticate the user in case of request are also allowed to log in via Google Facebook or Apple, probably using OAuth 2.0 as authentication protocol.

2.3). Synchronization and scalability

These applications must be able to handle thousands of users simultaneously, especially during traffic peaks (e.g. during the promotion of a new course). This could be achieved by using autoscaling services such as those of AWS that allow the infrastructure to dynamically grow or shrink according to demand. To ensure the integrity and privacy of the data of the users of these apps, multiple layers of security such as encryption in transit, encryption at rest, and strict sex policies are likely to be employed.

To guarantee the continuous availability and performance of the services, these apps could use monitoring tools such as Prometheus Grafana or dataDog, these tools allow:
- Monitor the status of the microservices and the infrastructure
- Detect bottlenecks or service failures
- Adjust server capacity in real time according to user traffic.

2.4). Design patterns

2.4.1). Microservice patterns:

Each of the functionalities such as authentication user progress course management etc. are encapsulated in their own microservice what does this mean that each service is independent and takes care of a specific function and this architecture facilitates scalability, allows faster upgrades and can reduce the impact of possible failures. For example:

- authentication services: handle registration, login, and token verification.
- Course service: manages the creation, modification, and deletion of resources.
- Progress service: handles logging and querying of user progress
- Notification services: sends notifications about new courses or reminders.

This means that these microservices have some key characteristics, which are the decompensation of services, communication with pencil and an independent deployment between them.

2.4.2). MVC design pattern (model-View-controller)

It is likely that the Fronted of these applications are structured following the model-view-controller pattern. This creates an independence of operation which means that it facilitates a more efficient maintenance in case of errors and offers simpler ways to test the correct operation of the system allowing the scaling of the application in case of being required.

2.4.3). Repository pattern

In the management of data persistence, such as user courses or progress, the repository pattern can be seen reflected in that it separates the logic that retrieves the data and assigns them to an entity model from the business logic that acts on the model, this allows our business logic to be independent of the type of touch comprises the data source layer in short a repository mediates between the domain and the data mapping layers.

2.4.4). CQRS pattern (Command query Responsibility Segregation)

It is used to separate the responsibility of reading operations, it helps us to separate the queries (data retrieval) from the commands, that is to say, from the insertion, update and deletion of data.

2.4.5). Observer pattern

It is probably used to improve events in real time or identifications in the application that allows us that any object that is implemented in the subscriber interface can subscribe to the notifications of the events.

2.4.6). Checking pattern

To improve performance and reduce latency

2.4.7). Factory pattern

It could be in use for the creation of complex objects, such as the enrollment process in a course or the configurations of a new user for example: a user registers in the application and a Factory is in charge of creating the object.

2.4.8). Adapter pattern

Could be present to integrate external systems such as third party APIs for authentication, this allows a structural design that helps with collaboration between objects with incompatible interfaces.

2.4.9). Singleton Pattern

It could be used to manage global instances that are needed in multiple parts of the system this helps us to make sure that a class has a single instance and at the same time provides a global access point to that instance.

2.4.10). Builder Pattern

It would be used when we need to create complex objects for example: in the construction of multiple lessons it helps us to create these objects step by step and the pattern allows us to produce different types and representations of an object using the same building code.

3). USER STORIES

- User stories one: new student-registration and first course

As a new user, I want to easily register and start a course to learn how to program from scratch.

- User stories 2: advanced student-continue process

As an advanced user, I want to learn to pick up my courses from where I left off, to continue my progress without having to manually search for the lesson.

- User stories 3: user wanting a reminder

As a user who studies intermittently, I want to receive daily reminders to continue my lessons and not lose the habit of learning.

- User stories 4: User who wants to check his statistics

As a user who monitors his progress, I want to see detailed statistics of my progress and study time to measure my performance.

- User Stories 5: User who wants to reinforce a concept

As a user who needs to review difficult concepts, I want to get quick access to additional exercises and detailed explanations on the topics I am having difficulty with.

- User stories 6: content manager

As a content administrator, I want to be able to easily add or update courses and lessons to keep the content up to date and relevant to users.

- User stories 7: User wants to change course

With a user dissatisfied with a course, I want to be able to easily switch to another course so I can find one that better suits my interests and skill level.

- User stories 8: user wants offline mode

With a user who has no Internet connection, I want to be able to download lessons for offline, so that I am not dependent on being connected while traveling or in an area without coverage.

- User stories 9: user who wants certificate

As a user who has completed a course, I want to get a digital certificate that shows my knowledge so I can share it with potential employers or in professional networks.

- User stories 10: user with a disability

As a visually impaired user, I want the application to be accessible through text-to-speech options, so that I can follow the lessons and content without any problems.

- User Stories 11: user who wants a personalized study plan

As a competitive user, I want to be able to compare my progress and scores with my friends to motivate me to learn faster and with more dedication.

- User story 12: user who wants to learn with micro lessons

As a user with little time available, I want to have access to quick micro lessons that I can complete in a few minutes to keep learning even if I have little time.

- User stories 13: user who wants integration with other apps

As a user who uses multiple learning applications I want my progress on the platform to be synchronized with other educational apps like Duolingo to have a centralized tracking of my learning.

- User story 14: user who wants to learn by playing

As a user who is motivated by gamification, I want lessons to include game elements (scores, badges, achievements) to make my learning more fun and competitive.

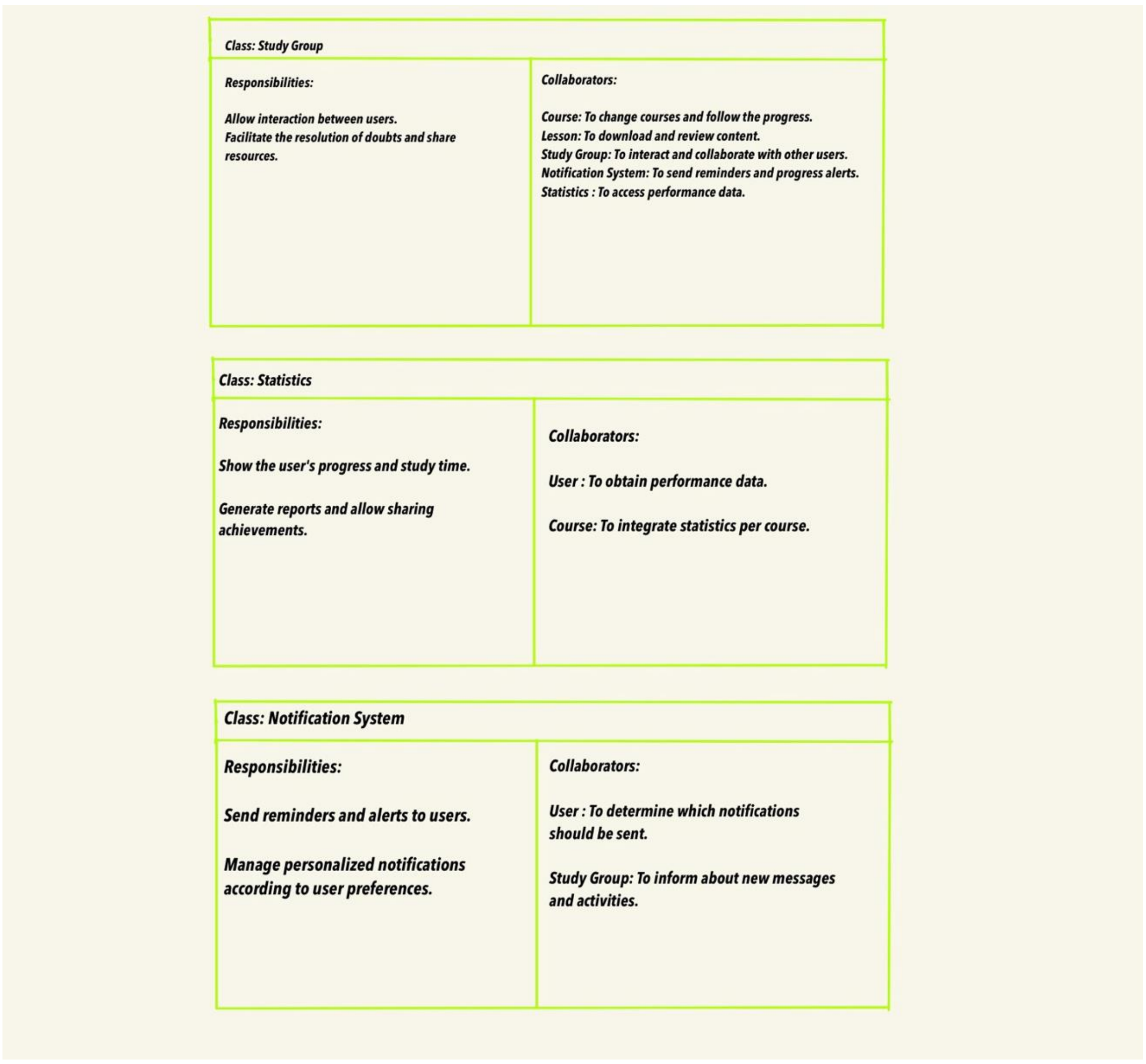
- User stories 15: user who wants personalized assessments

As a user who wants to measure my progress, I want periodic personalized assessments that allow me to see how far I have progressed and what areas I need to reinforce.

- User stories 16: user who wants to customize his experience

As a user who values customization, I want to be able to adjust the design and interface of applications to suit my visual and navigational preferences.

4). CRC CARDS



Class: Study Group	
Responsibilities: *Allow interaction between users.* *Facilitate the resolution of doubts and share resources.*	*Collaborators:* *Course: To change courses and follow the progress.* *Lesson: To download and review content.* *Study Group: To interact and collaborate with other users.* *Notification System: To send reminders and progress alerts.* *Statistics : To access performance data.*

Class: Statistics	
Responsibilities: *Show the user's progress and study time.* *Generate reports and allow sharing achievements.*	*Collaborators:* *User : To obtain performance data.* *Course: To integrate statistics per course.*

Class: Notification System	
Responsibilities: *Send reminders and alerts to users.* *Manage personalized notifications according to user preferences.*	*Collaborators:* *User : To determine which notifications should be sent.* *Study Group: To inform about new messages and activities.*

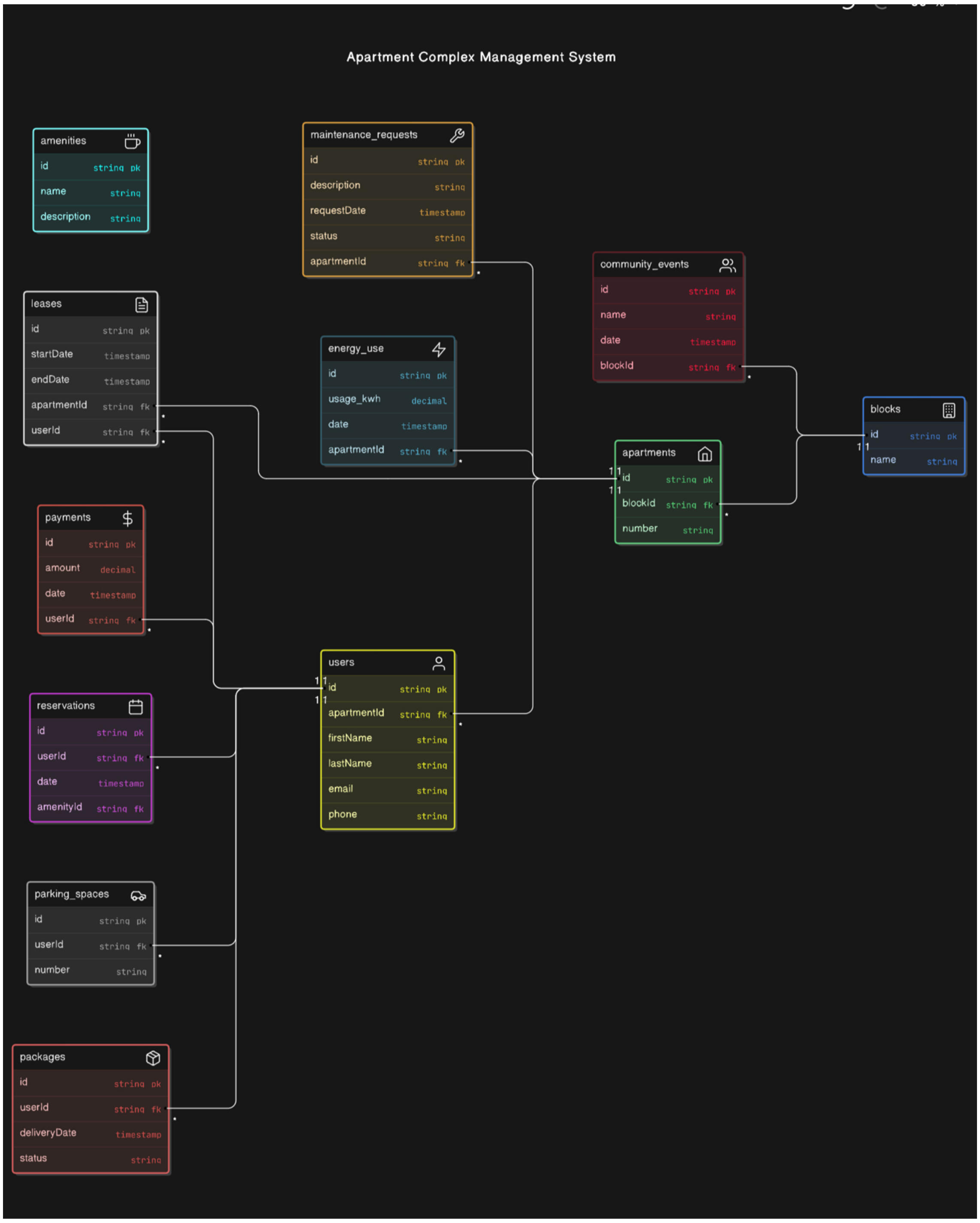
Class user	
Responsibilities: Register and log in. Change course. Download lessons to study offline. Receive study reminders. Consult progress statistics. Access personalized courses and participate in study groups.	**Collaborators:** Course: To change courses and follow the progress. Lesson: To download and review content. Study Group: To interact and collaborate with other users. Notification System: To send reminders and progress alerts. Statistics : To access performance data.

Class: Lesson	
Responsibilities: Show lesson content. Allow download to access offline. Provide additional exercises for review.	**Collaborators:** Course: To belong to a specific course. Evaluation: To associate evaluations with the content. User: To mark lessons as "difficult" and recommend exercises.

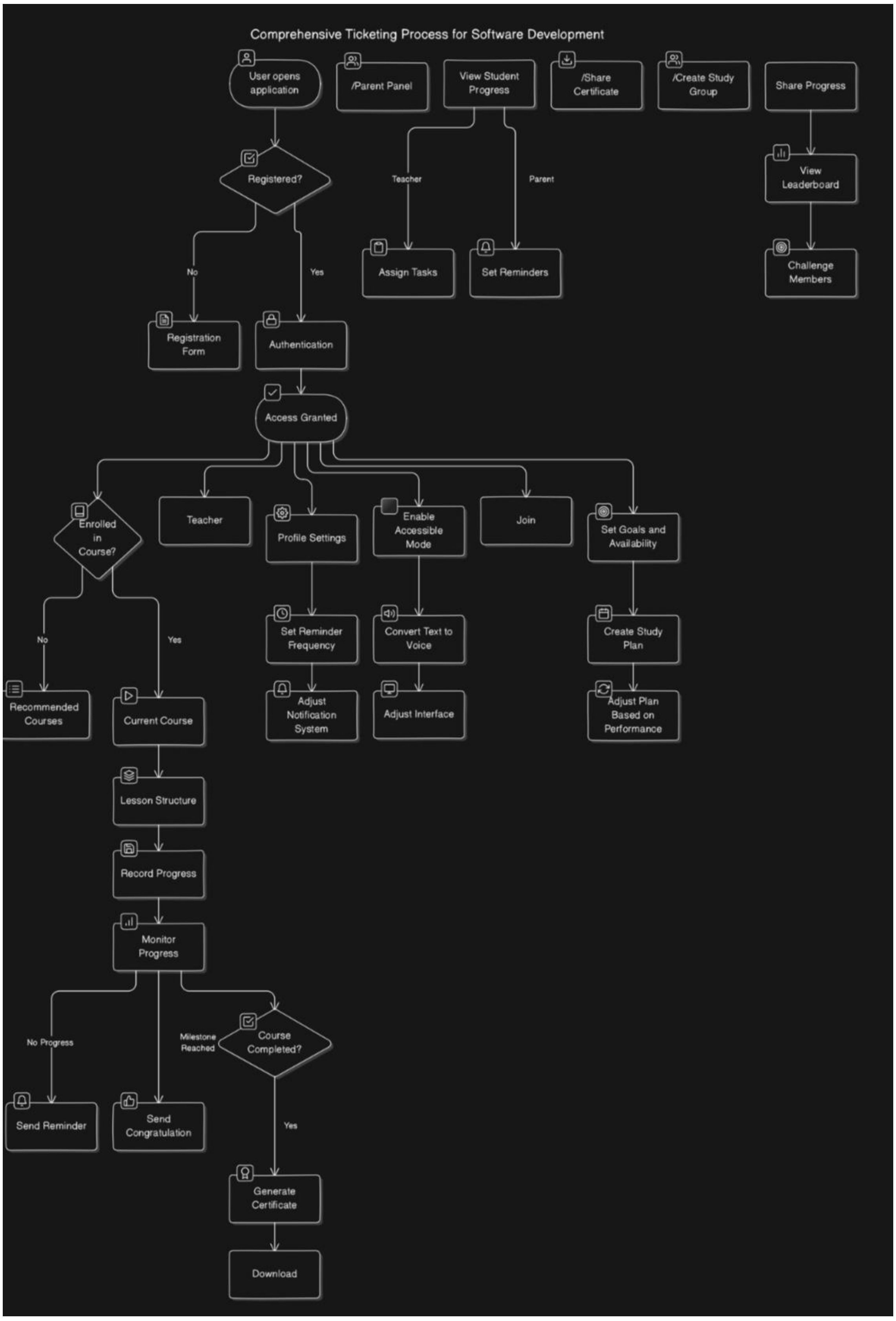
Class: Evaluation	
Responsibilities: Evaluate user responses. Provide feedback on performance.	**Collaborators:** Course: To associate evaluations with a specific course. User: To get answers and calculate scores.

Class: Content Manager	
Responsibilities: Manage courses and lessons. Keep the content updated.	**Collaborators:** Course: To add and modify courses. Lesson: To manage the specific content of each course.

5). DIAGRAM CLASS

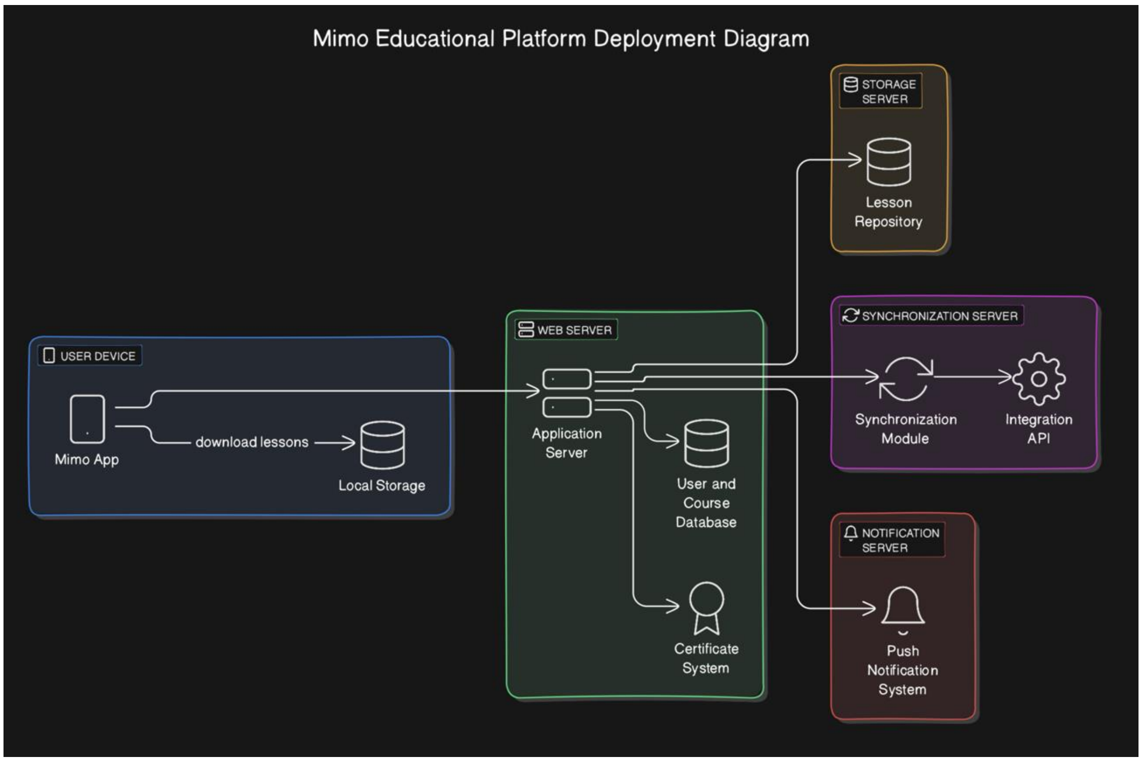


5). FLOW CHART



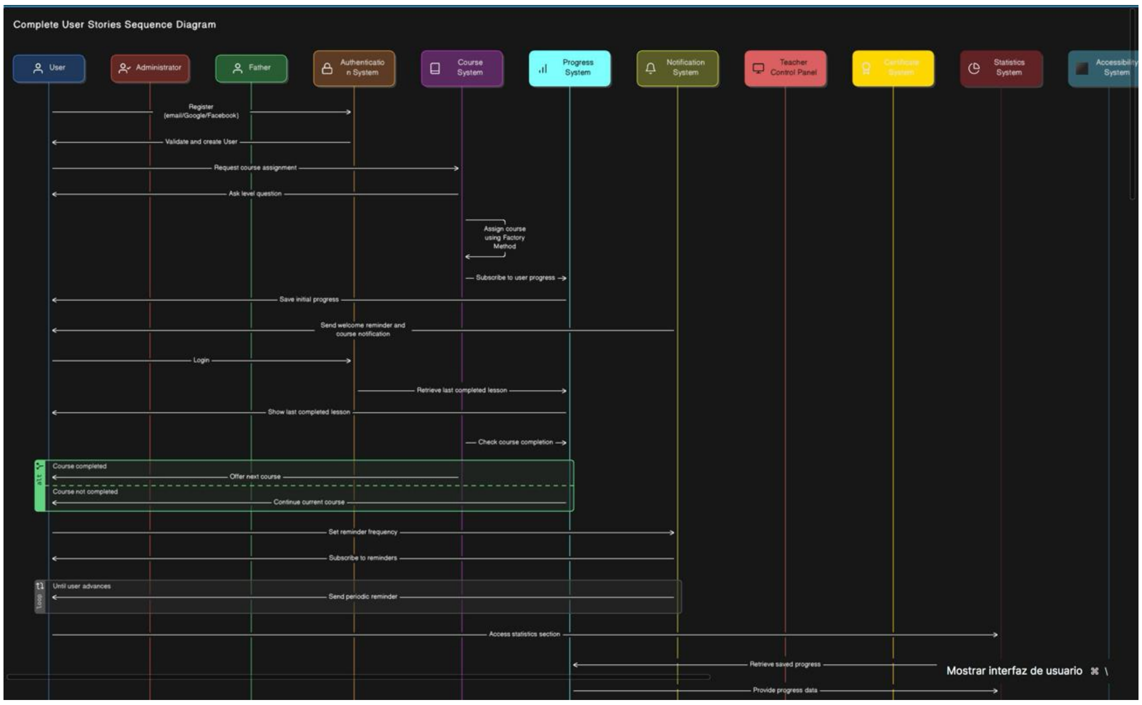
Comprehensive Ticketing Process for Software Development
User opens application
/Parent Panel
View Student Progress
/Share Certificate
/Create Study Group
Share Progress
Registered?
Teacher
Parent
View Leaderboard
No
Yes
Assign Tasks
Set Reminders
Challenge Members
Registration Form
Authentication
Access Granted
Enrolled in Course?
Teacher
Profile Settings
Enable Accessible Mode
Join
Set Goals and Availability
No
Yes
Set Reminder Frequency
Convert Text to Voice
Create Study Plan
Recommended Courses
Current Course
Adjust Notification System
Adjust Interface
Adjust Plan Based on Performance
Lesson Structure
Record Progress
Monitor Progress
No Progress
Milestone Reached
Course Completed?
Send Reminder
Send Congratulation
Yes
Generate Certificate
Download

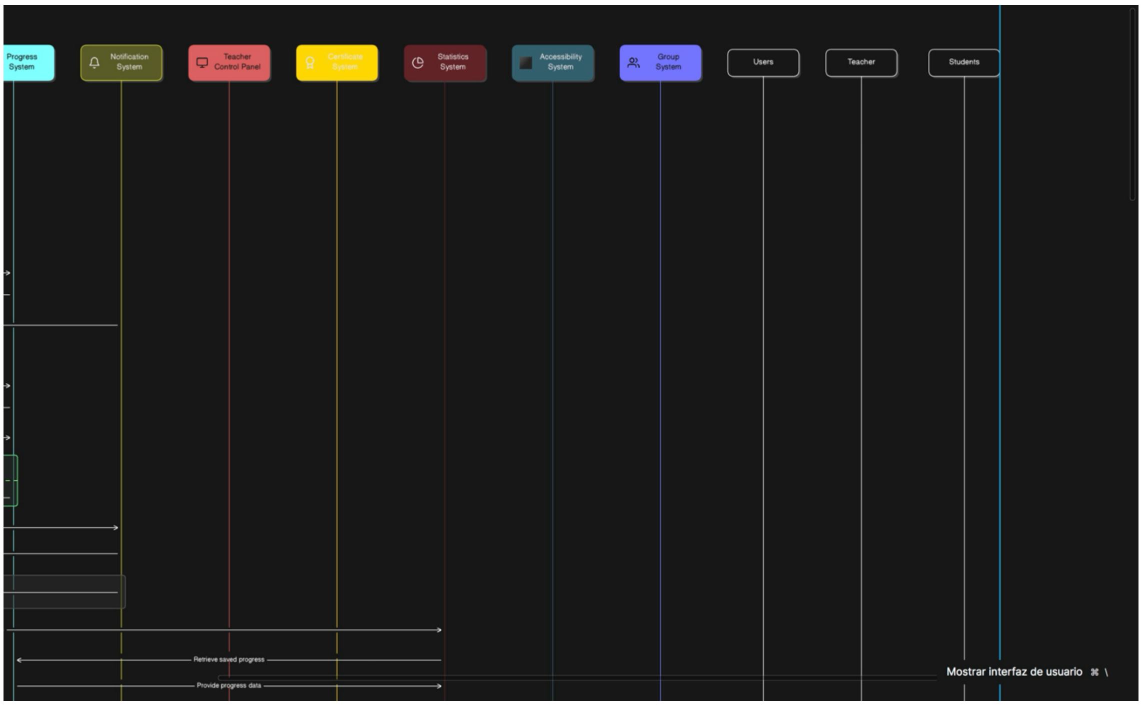
6). DEPLOYMENT DIAGRAM

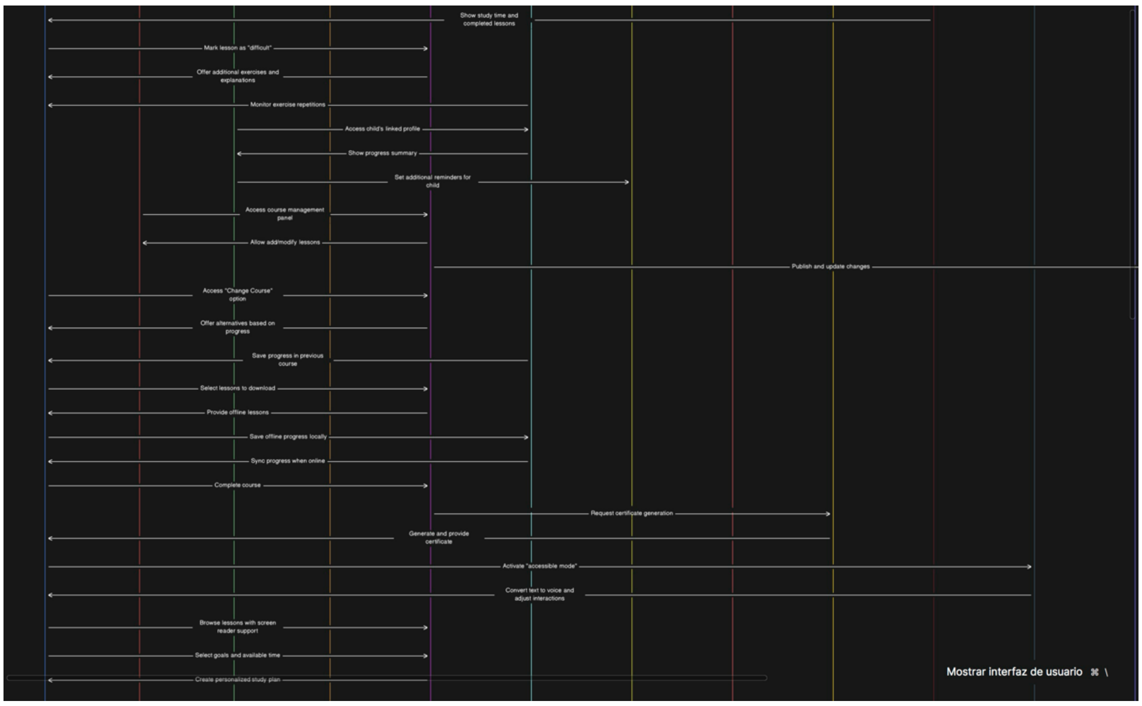


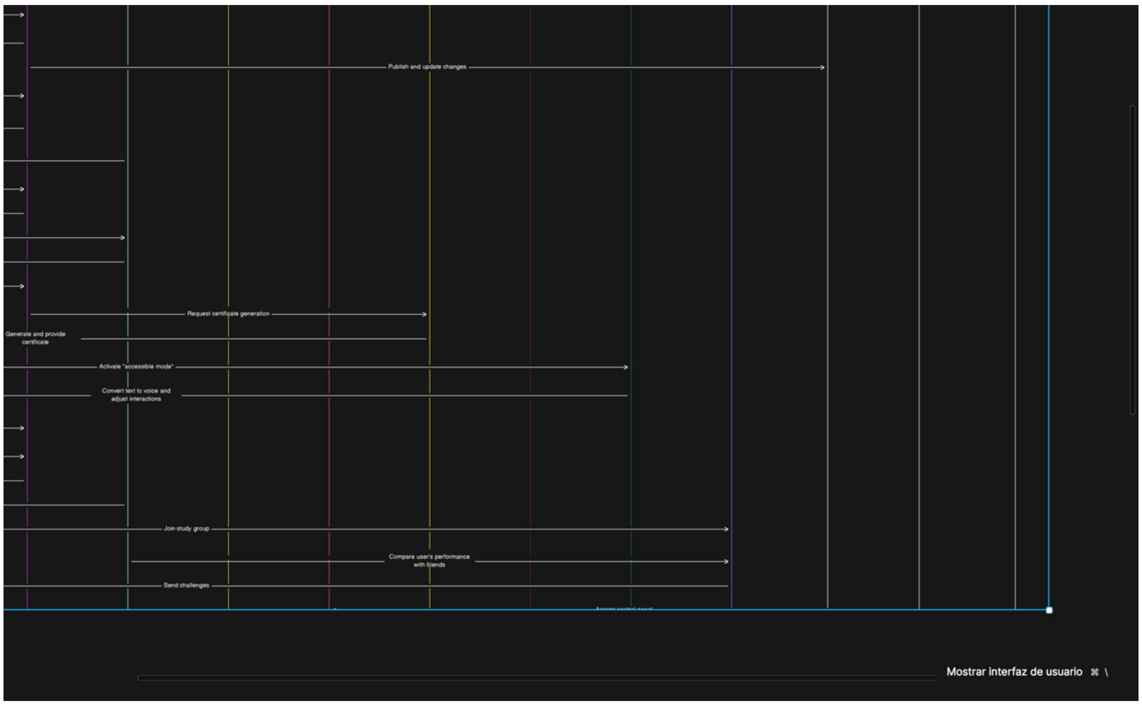
7). SEQUENCE DIAGRAM

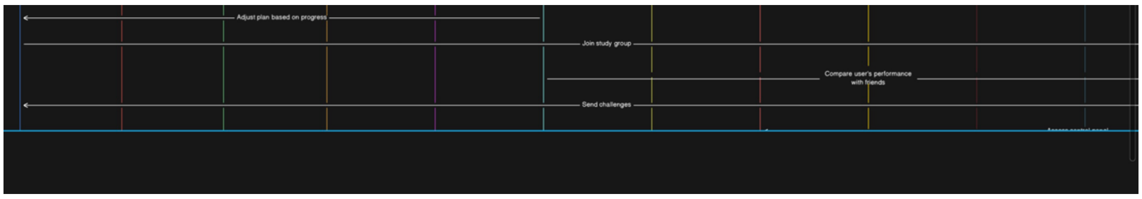
Note: the diagram was very large and it wouldn't let me export it so I divided it into organized parts






Show study time and completed lessons
Mark lesson as "difficult"
Offer additional exercises and explanations
Monitor exercise repetitions
Access child's linked profile
Show progress summary
Set additional reminders for child
Access course management panel
Allow add/modify lessons
Publish and update changes
Access "Change Course" option
Offer alternatives based on progress
Save progress in previous course
Select lessons to download
Provide offline lessons
Save offline progress locally
Sync progress when online
Complete course
Request certificate generation
Generate and provide certificate
Activate "accessible mode"
Convert text to voice and adjust interactions
Browse lessons with screen reader support
Select goals and available time
Create personalized study plan
Mostrar interfaz de usuario ⌘ \


Publish and update changes
Request certificate generation
Generate and provide certificate
Activate "accessible mode"
Convert text to voice and adjust interactions
Join study group
Compare user's performance with friends
Send challenges
Mostrar interfaz de usuario ⌘ \


Adjust plan based on progress
Join study group
Compare user's performance with friends
Send challenges