Milestone 2 Report

20.05.2022

CMPE352 Practice App: ArtShare

Ali Can Milani, Atilla Türkmen, Aziza Mankenova, Cahid Arda Öz,
Can Atakan Uğur, Demet Yayla, Erim Erkin Doğan, Güney İzol,
Hasan Bingölbali, Mehmet Emin İpekdal, Musa Şimşek
Rafet Oğuz Pançuk

Group 7

List of Contents

1. Executive Summary

This report showcases our work done for the second milestone of the CMPE352, Introduction to Software Engineering course. Work done prior to the practice app can be found in *Milestone Report 1.* For milestone two, we have developed a straightforward art platform called *ArtShare* which offers users basic functionality such as login/signup, discussion forum, event announcements, digital art items. All of the features and the team members responsible for these features can be seen in section *1.2, Features of ArtShare.*

Following is a list of links to access our application:

- ArtShare
- ArtShare Rest API
- ArtShare Rest API Documentation on our wiki and on the cloud
- Release Tag

1.1. Stages of Development

We planned our work in the weekly meetings we held. In these meetings, we talked about the status and discussed the changes made since the last meeting, listed and assigned tasks to be completed until the next meeting. Reports from our meetings can be found in our wiki.

We started working on the practice app with some research. We identified a list of topics to research and assigned these. Then, each assignee prepared a report and shared it on the wiki. The topics we identified can be found in the meeting 10 report and the research reports can be found under the practice app section in our wiki sidebar.

After completing the research, we were ready to start implementing our application. We created teams to handle front-end, back-end and database design. The teams and the discussions can be found in the meeting 11 report. We also made drafts of feature and external API lists for future use. These drafts were useful especially for the database team since they designed their models accordingly.

Once the foundations of the application were created by the teams, we had another meeting and made the final list of features to include on our platform. We tried to keep the features related to an art platform application so that we would implement a similar platform to what we will create in CMPE451. The list of the features and the discussions are available in meeting 12 report. List of features are also available in *section 1.2, Features of ArtShare Rest API*.

Once we had the list of features and the assignees, we started working on the implementations of our features. Each team member handled the requirements, use case diagrams, class diagrams, sequence diagrams, front-end and back-end implementation, testing and documentation for their own feature.

While implementing features, we also worked on different improvements to our design and workflow:

- We implemented session system in the frontend to handle tokens
- We added CI to our GitHub repository which made it easier to track the status of the application.

1.2. Features of ArtShare

The features we implemented in ArtShare are like a subset of the features we planned in Milestone 1 for the art platform application. In the following table, you can see these features with the name of the member who handled the feature and a brief description of the feature:

Feature	Description	Team Member
Viewing and creating an art item	Only artists can create an art item but any user can view existing art items from the home page. Artist provides a name, description and content URI for the art item while creating it.	*Can Atakan Uğur*
Viewing and creating an event	Artists can create an event and users can view existing events.	*Cahid Arda Öz*
Posting to and viewing the discussion forum	Any user can view the discussion forum with existing forum posts in the database but only logged-in users can post to the forum. Logged-in-user provides title, content URI (URI for an image if user wants to prepend to the textual body) and main textual body. System assigns date, time and author info automatically.	*Demet Yayla*
Signup, login and logout	Users can sign up to the system by providing email, password, first name and last name along with selecting whether they are an artist by clicking a checkbox button.	Güney İzol
Viewing and adding/removing participants of an event	Only logged in users can do these three actions. If a user is already a participant, no double participant is provided. A participation can only be canceled by the event's creator or	Erim Erkin Doğan

	the participant itself	
Searching for a word from Wikipedia	Users can search for a Wikipedia page with a given input word as title	*Atilla Türkmen*
Viewing user profile	Any user can view their profile page.	*Atilla Türkmen*
Creating personalized share links	With the help of an external API. Our backend creates a share link for the event. This feature can be only used by logged in users.	Erim Erkin Doğan
Viewing a forum post details with the comments and creating new comments	Any user can view discussion forum posts' details with comments on a separate page but only logged-in users can comment on a discussion post.	*Aziza Mankenova*
Reporting copyright infringement, viewing copyright infringement reports, removing reported art items	Only the logged-in users can report copyright infringement, view all existing reports from the home page and delete the reported art items.	*Ali Can Milani*
Viewing content on the platform (home page) and searching for content (content being art items and events) in the system	Users can go to the home page and view the content on the page. Content includes art items and events. Users can also search for content with a keyword using the search bar.	*Cahid Arda Öz*
Request verification, view verification requests, approve/reject verification requests	There will be a button next to the logout button at top of the page. Logged-in users can request verification by merely clicking on it. The request then will be visible on home page to all users of the application with a button to approve/reject verification. If a user evaluates the request and clicks on approve/reject button for a verification request, the request will be removed from the database and will disappear from the homepage.	*Musa Şimşek*
Reporting copyright infringement, viewing copyright infringement reports, removing reported art items	Only the logged-in users can report copyright infringement, view all existing reports from the home page and delete the reported art items.	Ali Can Milani

As explained in section 1.1, each member handled the following for their features:

- Requirements
- Related part of the Use Case Diagram
- Related part of the Class Diagram
- Sequence diagrams
- Front-end and back-end implementation
- Testing
- Documentation

1.3. App Components

Our app consists of a client front-end used by users, a Rest API and a SQLite database. A Flask backend serves both the front-end and the Rest API. This may look like a *red flag* for the milestone 2 since we are required to write a pure Rest API and a client front-end that interacts with the Rest API. An application that doesn't exactly match the Rest API principles can easily be built with Flask.

We took great care in *not violating Rest API principles* when designing our application. Our Flask app serves HTML but these HTML files include `<script>` sections that make requests to the Rest API served by the same Flask application. After data is fetched from the Rest API, it is processed and inserted into the same HTML file which is rendered in the user browser.

Any functionality we have in our front-end client can be implemented using our Rest API by another front-end client or a mobile application. Our API is a pure Rest API.

1.4. Status Of Deliverables

Deliverable	Status	Update Frequency	Description
Milestone Report 2	Delivered	With individual contributions	Report describing the overall status and progress of our work
Requirements	Delivered	With individual contributions	Specifications on our practice-app
Design	Delivered	With individual contributions	Use Case, Class Diagrams and Sequence Diagrams

Source Code	Delivered	Frequently (and continuously)	Source code of our application
App (Deployed on AWS EC2)	Delivered	During tests and in the final deployment	Our app deployed on the cloud. It is accessible with the URL provided in the beginning of the report.
Documentation	Delivered	Frequently (and continuously)	Implementation of our ArtShare platform.
Release Tag	Delivered	Once	A version release on GitHub. Contains the source code and information about the release.

1.5. Challenges We Met as a Group

The main challenge was the learning curve distribution. Forming a communication to integrate this knowledge needed planning. We mostly managed this with proper documentation through issues, research documents put to wiki, small group calls to ask for help etc. Deciding on a common framework and tools was another challenging matter since knowledge did not intersect on any so there was learning new things for everyone. This forced us to learn how to learn. Also merging our codes to the main branch was a minor challenge.

2. Personal Contributions

Cahid Arda Öz	● Acted as **moderator** in the meetings. ● Created **initial files** for practice app. (Details in #137, PR #138) ● Prepared **AWS Research** for Practice App. Tried to answer questions relevant to our project (#142) ● Frequently asked **questions on Slack** to the Professor and TAs to clarify points about the practice app. ● Worked on creating the initial **front-end structure** for the practice app (with Demet Yayla and Atilla Türkmen, see #147). Designed application layout in base.html. Created examples with create_event.html and view_event.html. This task involved considerable learning about HTML, Javascript and templates as explained in the issue. ● **Designed, implemented, tested and documented my features**: ○ Handled view/create event features (see #158 for details) ○ Handled home feature (see #166 for details) ● Handled **session/token management in the frontend and backend** (with Güney İzol and Erim Erkin Doğan): ○ Improved the token creation (#179) ○ Designed and implemented authentication in frontend (see jwt.py) and backend (see api/jwt.py). (#179) ○ Handled session and token expiration (#208) ● Prepared notes for meeting #13 (#164)

	• Created an AWS account and tested **deploying the app on an EC2** instance Reported my progress in #171 after I was deploying the app. • Reviewing and writing sections of the **executive summary** (#184)

Demet Yayla	• Research on unittests in Python.(#145)
	• Template frontend file [file1|file2]preparation (foundation of frontend).(#147)
	• Review project plan deliverable (#182)
	• Creating Executive Summary for Milestone 2 report (#184)
	• Notetaker for Meeting #14 and Meeting #15
	• RESTAPI and frontend for Forum Post, unittests (#161).I learned unittest library, JS, HTML, CSS for this task.
	• Review merge request for practice-app/feature/wikipedia (#175)
	• Prepare Use Case Diagram, Sequence Diagram, Class Diagram and Requirements for discussion forum related features (#185)
	• Fill personal contributions for the Milestone Report 2 (#186)
	• Finalizing Milestone Report 2, practice-app and deliverables
	  ○ Compiling report
	  ○ Deploying the application
	  ○ Put the group report on the repository deliverable folder.
	  ○ Merge practice_app into master.
	  ○ Finalize the design documents (#228)
	• Review backend and frontend of all members to document into executive summary how the final version of the practice-app will functionate.
	• Documenting unittest for “Tool Evaluation” part in Milestone Report 2. (#204)
	• Created pull request (#209)
	• Created swagger documentation for my endpoints (#229).
	• Checked my endpoints with Postman (#161).

Ali Can Milani	• Research on Pull Request and Review Tools. [#144]
	• Reading the research team members made
	• Researching an external API to use in copyright infringement reports [#180], [Image Similarity API]
	• Implementations on Copyright Infringement [#180]
	  ○ Creating endpoint to report copyright infringements
	  ○ Creating endpoint to view the reports
	  ○ Creating endpoint to delete the reported art items
	  ○ Creating relevant templates and connecting with api endpoints
	• Creating diagrams for Copyright Infringement related features [#198]
	  ○ Use Case Diagram
	  ○ Class Diagram
	  ○ Sequence diagrams
	  ■ View Copyright Reports
	  ■ Create Copyright Reports
	  ■ Remove Reported Art Item
	• Working on the Copyright Infringement Requirements
	• Preparing Swagger documentation for Copyright Reports [#210]
	• Testing Copyright Reports endpoints using Postman [#211], [Postman Results Page]
	• Preparing unit tests for Copyright Report endpoints [#212]
	• Reviewing the art items implementations and merging them [#173], [#206]
	• Creating pull request for merging copyright infringement implementations [#201]
	• Creating new Release to deploy the changes to the app [ArtShare v1.0.3]

Musa Şimşek	• Research on Docker.
	• Reading the research team members made
	• Researching an external API to use in copyright infringement reports : Wikimedi-OnThisDay
	• Implementations on Verification Requests #162
	  ○ Creating endpoint to request verification
	  ○ Creating endpoint to view the requests
	  ○ Creating endpoint to review requests
	  ○ Creating relevant templates and connecting with api endpoints
	  ○ Preparing Swagger documentation for Copyright Reports
	  ○ Preparing unit tests for Verification Requests endpoints
	  ○ Testing verification.py endpoints using Postman
	• Creating diagrams for Verification Request related features
	  ○ Use Case Diagram
	  ○ Class Diagram
	• Integration of Bootstrap #207
	• Working on the Verification Requirements
	• Reviewing the several branches and merging them #216
	• Creating pull request for merging verification requesting implementations #214, #227

Erim Erkin Doğan	• Actively participated in all meetings • Created **initial files** for practice app. (Details in #137, PR #138) • Researched about flask framework. Documented the results on the wiki page with *Atilla Türkmen*. (#143) • Prepared Docker configuration with *Can Atakan Uğur* (#170, PR #178) • Created and handled API documentation wiki page. (#174) • Set up backend foundations for our app, researched about authorization methods. (#152) **Created requirements and design documents for the participation feature I implemented (#165, #217):** • Created requirements for the feature. (Wiki) • Updated the use case diagram with use cases about my feature. (Relevant issue link, Use case diagram) • Updated the class diagram inline with the feature. (Relevant Issue Comment, Class Diagram)

	• Created 4 sequence diagrams for sequences: View Participants, Add Participant, Remove Participant(s) and Get Share Link. (Issue reference #1, issue reference #2, Sequence Diagrams) **Created CI/CD pipelines for the app: (#187)** • For pipelines, I have gathered the keys and contacted the TA for their addition to repository. • Created CI pipeline for testing before pull requests.(#187, Commit) • Created CD pipeline and first test release and deployment for app. (#226) **Practice App Implementation:** (#165, PR #216, PR #232) • Implemented a feature about participation on events. This feature includes adding a participant, removing participant(s), viewing participants and creating a shortened share url redirecting to event page. • Used an external API named **Cutt.ly** to create a shortened link pointing to event page, so that participants can share the event with their friends. (External API docs, Relevant issue comment) • Implemented a frontend for the feature. (participate.html) • Implemented API endpoints: Made 1 `GET` external API call, implemented 3 `POST` and 1 `GET` request endpoints. (participants.py) • Implemented unit tests (test_participants.py) • Created Swagger documentation for participants feature. (Commit) • Updated configuration of the app. (commit, PR #208) **Reviewed Issues and Pull Requests**: • Reviewed first pull request for `practice-app`, the `.gitignore` file (PR #137) • Reviewed Cahid Arda's AWS EC2 deployment (#171) • Reviewed code, made some suggestions about testing coverage: View user profile feature (#172, PR #195, PR #222) • Reviewed and suggested changes in PR: Frontend token design (PR #179) • Reviewed, suggested and made changes on the PR: A fix about tokens expiring and sessions not expiring (PR #208) • Reviewed PR: Updated unittests for home and event features. (PR #188) • Reviewed PR: Making unittests use another database (PR #191) • Fixed the conflicts in the pull request: `practice-app/feature/copyright` (Commit, PR #201) • Reviewed PR of *Aziza Mankenova*: Merge practice-app/feature/discussion-comments into practice_app (PR #234) **Milestone Report** (#223): • Evaluated AWS and Github Actions • Filled personal contribution table • Added details about external API I have used (Cutt.ly)
Mehmet Emin İpekdal	
Güney İzol	• Implement sign up and login (#177) • Implement art_galleries endpoint (commit) • Write swagger docs for these endpoints (commit) • Write unit tests for these implemented endpoints • Write the relevant requirements for these endpoints • Create sequence diagram for sign up • Create sequence diagram for login

	• Create sequence diagram for viewing art galleries • Make additions to use case diagram for signup, login, and art_galleries

Aziza Mankenova	• Made research on testing API using Postman for other team members' convenience. And wrote the results to a wiki page (#141)
	• Contributed to the initial files for practice app. (Details in #137, PR #138)
	• Took part in all of the meetings, took notes during the meeting #12(#156)
	• Spent some time on learning Flask, HTML, CSS, Javascript
	• Completed **functional and system requirements** related to discussion forum comments (#203) and reviewed the requirements, adjusted and fixed some of them with *Cahid Arda Öz* for the final submission.
	• Created the initial use-case, UML Class and sequence diagrams on lucid.app for effective collaboration
	• Completed the parts of the **Use Case Diagram** that are about Discussion Forum Comments (#203)
	• Completed the parts of the **Class Diagram** that are about Discussion Forum Comments (#203)
	• Created the **Sequence Diagram** for "Posting Comments to Discussion Post", "View Discussion Posts with Comments" (#203)
	• Worked on creating the fundamental **database structure** with *Can Atakan Uğur* and *Mehmet Emin İpekdal* (#148)
	• Worked on fundamental **database** implementation of "Events", "Participants", "Users", "Artist" models(#151)
	• Reviewed *Can Atakan Uğur's* implementation of *ArtItem*, *DiscussionPost*, and *Comment* database models (#150)
	• Reviewed *Can Atakan Uğur's* implementation *Admin*, and *CopyrightReport* database models (#169)
	• **Practice-App implementations**:
	  ○ Implemented a feature about viewing discussion forum post information with the comments, and posting new comments under the discussion forum posts(#163)
	  ○ Utilized a Deep Translate **external API** for automatic translation of newly posted comments into Turkish(#163)
	  ○ Implemented **frontend** page for discussion forum posts, where you can find a full details about the posts and all the comments written under these posts. You can get redirected to these pages of discussion forum posts from the discussion forum page.(#163)
	  ○ Created Swagger **documentation** for the implemented GET and POST requests(#163)
	  ○ Wrote unittests for testing my GET and POST requests regarding discussion forum post comments(#163)
	  ○ Reviewed and approved a pull request #155 by Cahid Arda Öz. Merged backend base into practice_app branch.
	  ○ Reviewed,made suggestions to a pull request #209 by Demet Yayla and merged discussion forum branch into practice_app branch
	  ○ Reviewed, made suggestions to a pull request #232 by Erim Erkin Doğan regarding the participants branch and merged into practice_app
	  ○ Opened a comprehensive pull request #215 upon finishing all the features of the comments, documentation and testing to merge forum post comments branch into practice_app branch. Have made changes according to the feedback
	  ○ Reviewed, approved the pull request #233 by *Musa Şimşek* to merge `practice-app/feature/verification1.2` to `practice_app`. Consequently, performed a merge with practice_app

	• Milestone Report 2:
	  ◦ Reviewed the whole report, fixed minor mistakes
	  ◦ Filled in personal contribution table
	  ◦ Completed evaluation of Postman as a tool used(#218)

Rafet Oğuz Pançuk	
Atilla Türkmen	• **Researched** about the flask framework. Documented the results on the wiki page with *Erim Erkin Doğan***.** (#143) • Participated in 14 of 15 **meetings**. • **Took notes** in meeting #9. • Worked on creating the initial **front-end structure** for the practice app with Demet Yayla and Cahid Arda Öz. Designed layout and added fetch calls in the **art item information page and wikipedia search page**. (#147) • Individually studied **javascript, html, css and flask**. • Created **use case diagrams, class diagrams, sequence diagrams and requirements** for users to view their profile for the practice app. (#197) • Created **use case diagrams, class diagrams, sequence diagrams and requirements** for users to search a word on wikipedia for the practice app. (#196) • Implemented the feature for users to view their profile with front-end template, a GET call and unit tests. (#172) • Implemented the feature for users to search for a word with an external GET call to wikipedia API, front-end template, a POST call and unit tests. (#175) • Documented the behavior of these endpoints.(issues: #219 and #220) (documentation link) • Regularly reviewed pull requests of other group members: #159 #205 #167 #193 • Regularly opened pull requests: #194 #195 #222
Can Atakan Uğur	**Meetings** • Actively participated in all 6 meetings after Milestone 1. This resulted in full participation in all team meetings (15/15) for me. • Took notes in Meeting #10 (#139, Meeting Notes) **Research** • Made individual study on Flask, Django, and .NET • Made research on Docker with *Musa Şimşek* and documented it (#140**,** Research Report) **Infrastructure** • Regularly monitored the Sidebar and Wiki • Created the `practice-app/database-base` branch to divide the work of creating initial project files (#149, Commit) • Integrated Swagger into the Practice App for API documentation (#192**,** Commit) **Requirements** • Created the initial Practice App Requirements template Wiki page (Revision) • Completed Art Item Requirements (#199) • Completed Non-Functional Requirements (#199) **Use Case Diagram**

- Created the initial Use Case Diagram example (#190)
- Completed the parts of the Use Case Diagram that are about Art Items (#189, Completed Part)
- Worked on finalizing the Use Case Diagram before submission with *Demet Yayla* (#228)

Class Diagram

- Created the initial Class Diagram example (#190)
- Completed the parts of the Class Diagram that are about Art Items (#189, Completed Part)
- Worked on finalizing the Class Diagram before submission with *Demet Yayla* (#228)

Sequence Diagrams

- Created the Sequence Diagram for "Viewing an art item" (#189, Diagram)
- Created the Sequence Diagram for "Creating an art item" (#189, Diagram)

Practice App - Implementation

- Worked on creating the fundamental **database structure** with *Aziza Mankenova* and *Mehmet Emin İpekdal* (#148)
    - Implemented initial *ArtItem*, *DiscussionPost*, and *Comment* database models (#150)
    - Implemented initial *Admin*, and *CopyrightReport* database models (#169)
    - **Reviewed** Aziza Mankenova's implementation of *Event*, *User*, *Artist*, and *Participant* database models (#151)
    - `practice-app/database-base` branch is merged to the main branch during a meeting (PR #153)
    - Worked on `__init.py__` file to integrate database creation (Commit)
- **Implemented Art Item related features (#173, PR #206)**
    - Implemented **two API endpoints**, one with **GET** and one with **POST**. For viewing an art item and creating an art item features, respectively.
    - I used an **external API endpoint (with GET request)** from the provider Imagga, which takes the URL of an image, and returns the dominant colors of it.
    - Mainly worked on `art_item.py` and necessary html files. The main implementations can be found in this issue comment.
    - Implemented front-end for Art Item related features. HTML files can be found for viewing, and creating Art Items.
    - Updated `home.html` to show Art Items (Commit)
- Prepared **Docker configuration** with *Erim Erkin Doğan* (#170, PR #178)

Practice App - Testing & Documentation

- Tested my API endpoints with Postman (#200)
- Implemented unit tests for my API endpoints (#183)

	• Worked on `__init.py__` for Swagger initialization and settings. Integrated authentication to the documentation, too. (Commits: 9ceb974, 24386cb) • Prepared my `YAML` files for Swagger API documentation (#192, `art_item_GET.yml`, `art_item_POST.yml`) • Helped *Musa Şimşek* for resolving errors in his Swagger documentation **Practice App - Pull Request Review** • Actively reviewed codes written by other team members • Requested changes when necessary, approved & merged otherwise • Here are all pull requests that I participated in the review process: PR #154, PR #201, PR #202, PR #205, PR #209, PR #215 • In addition to those, I followed the issue tracking of other team members, and wrote reviews for them. You can find my comments and feedbacks on: Bootstrap, Event, Unit Tests **Planning** • Completed the Project Plan deliverable for Milestone 2 (#182, Practice App Project Plan) • Updated the old Project Plan in the repository (Revision) **Milestone 2 Group Report** • Reviewed all parts of the report, fixed typos, and improved the English. Also contributed to formatting the report. • Added *Docker*, *Swagger*, and *ProjectLibre* evaluations under *Evaluation of Tools* (#224) • Filled personal contribution table (#225) • Participated in the process of preparing final deliverables (#228)
Hasan Bingölbali	

3. Evaluation of Tools

3.1. HTML and Javascript

HTML and Javascript are relatively easy tools to build front-end templates in the sense that they don't require the programmer to set-up and learn vast frameworks. It still took some time for us to design front-end templates that interacted with a Rest API since none of the members were familiar with such a task prior to the practice app project. We eventually came up with the idea of

embedding Javascript inside the HTML templates and making requests to the Rest API there which worked out pretty nicely.

Only problem with this approach was that we weren't able to learn styling our templates in time and our website looks as if it was created before the Dot-com bubble.

3.2. Flask

Our app is served by a Flask back-end. Templates in Flask were quite useful. We were able to create rather useful templates with the jinja engine and use them effectively. Route, Blueprint, Session and Templates were all intuitive structures and they made it easy to build a Rest API for our application.

Only problem we faced was handling tokens in the front-end and back-end. We used `flask_jwt_extended` and spent considerable time on managing tokens in the front-end and back-end.

3.3. Unittest

We chose unittest unit testing framework for our app which is a standard module which you don't need to install if you have python installed. It was easy to use and verbose enough for tracing errors. setUp and tearDown methods eased the process and the initial search with documentation helped with the learning curve.

Challenges were relative imports and running directories, they are hard to arrange in Python. Also at first there was confusion as to how we were asked to test, so we tried mocking the jwt authentication but failed. Later, consulting the professor, we switched to adding a fake user to a test_database.db.

3.4. Postman

Postman is an API platform that makes it easy for developers to create, share, test, and document APIs. We used Postman for testing the external and our own RESTful APIs. Postman allows us to make a collection of requests and save them for later use. It is a very useful feature which saves our time. Instead of writing the body, headers, endpoints, and type of request every time, you can save the API requests and use them again in the future. Also, it allows us to share the requests with others for convenience. The user interface is convenient and it does not require much effore to learn to use it. In general, it is a very useful tool for testing APIs, which made our work easier. I suppose there were no challenges in using Postman.

3.5. GitHub Actions

We used Github Actions to create pipelines for our app to ensure it is not broken after a merge. So we set up a CI workflow to test the code everytime a PR or push to “practice_app” repository. It was an easy task with already included templates on the platform which served our usecases. We have also used the Github actions to automatically deploy our app to AWS on an EC2 instance. We again used a template, so our task was again easy with 1-2 changes on the workflow. At the end CI/CD pipelines helped us spend less time on bug fixing or deploying. Our only complaint about the actions is that for it to work properly it should be located at the master branch, otherwise we can’t manually run it. If we place it on the master branch, then we can’t manually run it to test or deploy other branches' source code. So, a change which will help in this direction would make an already good tool better.

3.6. AWS

We used AWS and its various tools to deploy our application. The tools we used were ECS(Elastic Container Service), ECR(Elastic Container Registry), and EC2 instances. We have opted to use an automated(Continuous Delivery) approach, so that our application can be deployed just with a push. Setting up the AWS to handle automatic deployment with Github Actions was hard as the tools are very advanced and capable. AWS with its vast number of tools had multiple ways to implement a CD workflow, which is always nice to have variation which a simple platform doesn’t . This situation creates a learning curve but the documentation on the web and on Github’s site were detailed and helpful. Maybe a less detailed and user friendly cloud platform like Digitalocean, AWS LightSail, Vultr could have been chosen as it would be much easier to launch and deploy while having less freedom.

3.7. ProjectLibre

ProjectLibre was used for project planning in the Practice App, too. Again, while preparing the plan for the Practice App, too, we benefited from the extra features that ProjectLibre provides us with. Adding predecessors, assigning resources, being able to fix the duration of the event and seeing the beginning and ending dates in detail were some of these advantages of using ProjectLibre. Apart from those, it could create the Gantt Chart automatically, given the dates and assigned resources.

The problem with using ProjectLibre was that the application did not run so smoothly sometimes, it even crashed occasionally. However, after getting used to it, it was even easier to handle all these cases, and work more conveniently.

3.8. Docker

We used Docker in our application to automate the deployment of it as a portable, and self-sufficient container so that it could run on the cloud servers. Docker made it much easier to create, deploy, and run our application, ArtShare, with the use of those. Thanks to the isolated infrastructure provided by Docker, we did not experience inconsistencies. Also, Docker made our deployment process much quicker. Apart from those, our application was much more portable thanks to it.

We did not experience any notable problem with Docker, except the learning process for every one of us to get used to it.

3.9. Swagger (with flasgger)

For the API documentation, we used a Flask friendly module called “flasgger” which simply integrates most of the features of Swagger, an API documentation library. Thanks to Swagger, we were able to extract the API documentations of our endpoints automatically. It was just sufficient to fill the necessary YAML files and connect those to our endpoints.

The most difficult part with the Swagger integration was probably adding the authentication. Since flasgger did support OpenAPI 3.0+ only in beta, we used the default versions that it provided. However, in that case, Bearer Token Authentication was not supported internally, so we needed to define it manually during the configuration. Nevertheless, everyone was able to learn the use of Swagger quickly, and they created their API documentations conveniently.

Requirements

1. Functional Requirements

1.1. User Requirements

1.1.1. Account Functions

- 1.1.1.1 A visitor shall be able to sign up by providing email, password, first name, last name and whether an artist account is desired or not.
- 1.1.1.2 A user shall be able to log in by providing email and password.
- 1.1.1.3 A user shall be able to see their account information which includes first name, last name, email and verified status.

1.1.2 Wikipedia Search

- 1.1.2.1 Users shall be able to search the definition of a word from Wikipedia.

1.1.3. Events

- 1.1.3.1. Users shall be able to view the events on the website
- 1.1.3.2. Users shall be able to view the weather forecast for the location of the event
- 1.1.3.3. Users shall be able to create events
- 1.1.3.4. Users shall be able to view participants for an event
- 1.1.3.5. Users shall be able to participate in events
- 1.1.3.6. Users shall be able to cancel their participation to an event
- 1.1.3.7. Users shall be able to create a personalized share link to event

1.1.4. Art Items

- 1.1.4.1. Users shall be able to view an art item in the platform.
- 1.1.4.2. Users shall be able to view dominant colors in an art item.
- 1.1.4.3. Artists shall be able to create an art item in the platform.
- 1.1.4.4. The art item shall include a name, a description, and an image.

1.1.5. Discussion Forums

- 1.1.5.1 Users shall be able to open Forum Get page to view all posts that has been sent.
- 1.1.5.2 Users shall be able to open Forum Post page.
    - 1.1.5.2.1 Logged-in user can send post to be seen in Forum page
    - 1.1.5.2.2 Non-logged-in user gets redirected to error page when attempting to post to forum.
- 1.1.5.3 Users shall be able to view a discussion post's details, such as title, description, image, creator, date, and the comments below it.
- 1.1.5.4 Logged-in users shall be able to add new comments to the discussion post, providing the body of the comment and a link to the content URI.

1.1.6 Home

- 1.1.6.1. Users shall be able to view the content on the platform (art items, events).
- 1.1.6.2. Users shall be able to search for content on the platform.

1.1.7. Copyright Reports

- 1.1.7.1. Users shall be able to report copyright infringement
- 1.1.7.2. Users shall be able to view copyright infringement reports
- 1.1.7.3. Users shall be able to remove an reported art item

1.1.8. Verification

- 1.1.8.1. Users shall be able to request verification
- 1.1.8.2. Users shall be able to see pending verification requests in the home page.
- 1.1.8.3. Users shall be able to accept/reject any unreviewed verification request.

1.1.9 Art Galleries

- 1.1.9.1 Users shall be able to view a list of 5 available art galleries worldwide.

1.2. System Requirements

1.2.1. Account Functions

- 1.2.1.1 System shall return an access token following a successful sign up or login event.

1.2.2. Art Galleries

- 1.2.2.1 System shall return a list of 5 available art galleries worldwide to a user when demanded.

1.2.3. Events

- 1.2.3.1. System shall be able to create personalized share links for an event.
- 1.2.3.2. Systems shall limit the participation cancellation to participating user or event's creator.
- 1.2.3.3. System shall limit the viewing participation to logged in users.
- 1.2.3.4. System shall block the participation of not logged in and already participating users.

1.2.4. Art Items

- 1.2.4.1. System shall be able to showcase an art item in its Art Item page.
- 1.2.4.2. System shall be able to find and show the dominant color in an Art Item.
- 1.2.4.3. System shall be able to create a new art item object in the platform via the Art Item creation page.

1.2.5. Discussion Forums

- 1.2.5.1 System shall enable user to view all the forum posts existing in database.
- 1.2.5.2 System shall not give permission to non-logged-in users to add new post to the database.
- 1.2.5.3 System shall enable logged-in users to add a new post to the database.
- 1.2.5.4 System shall censor the main body of the posts being sent for bad words.
- 1.2.5.5 System shall be able to showcase the information about each of the discussion posts with the comments on a separate page
- 1.2.5.6 System shall not give permission to non-logged-in users to add new comments under the discussion posts.

- 1.2.5.7 System shall enable logged-in users to add a new comment under the discussion posts.
- 1.2.5.7 System shall enable provide the translation of each comment into Turkish.

2. Non-Functional Requirements

2.1. Security

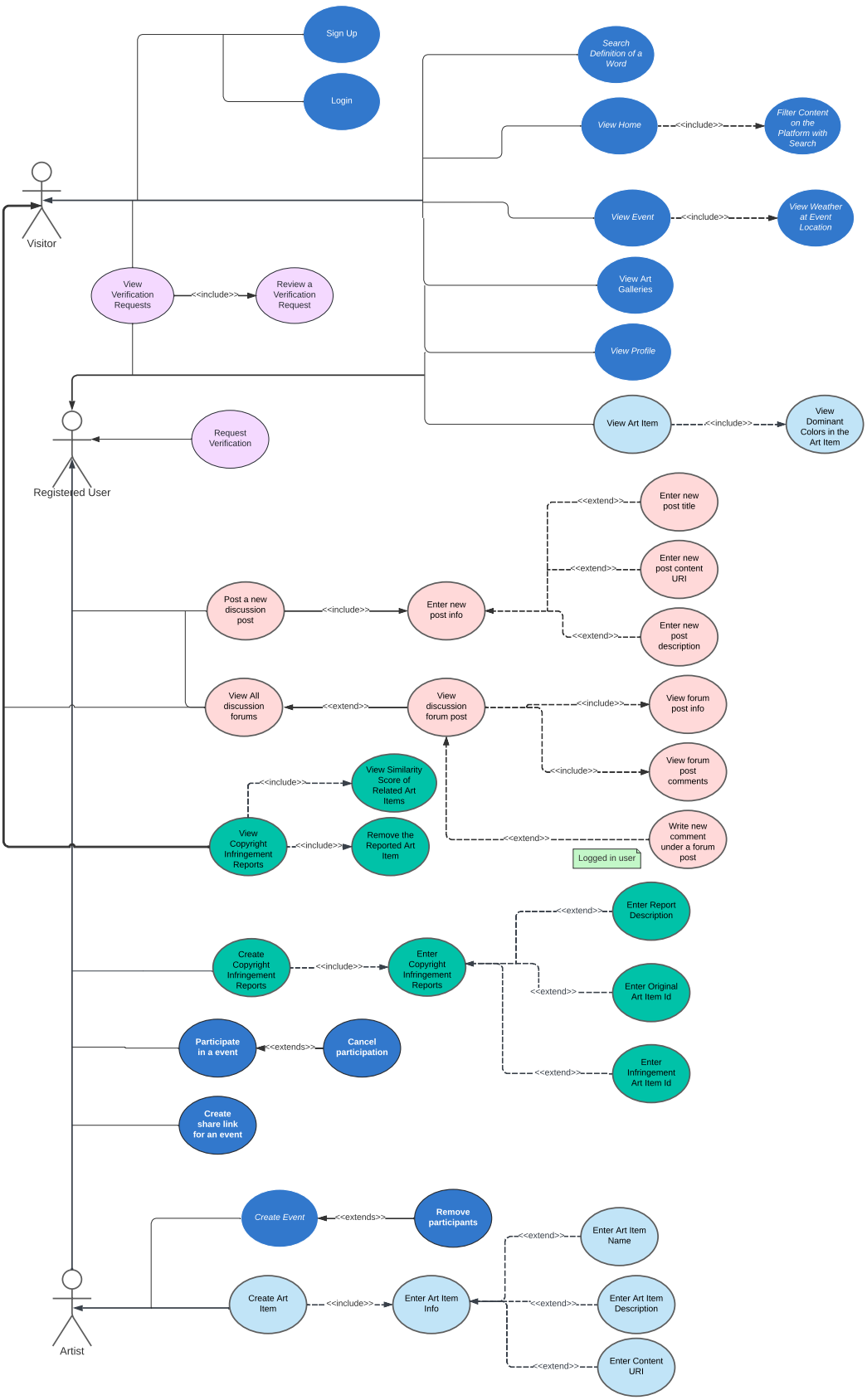
- 2.1.1. The passwords of users shall be encrypted in the database

2.2. Accessibility

- 2.2.1. Platform shall support English language.
- 2.2.2 Platform shall be accessible via modern web browsers (Chrome, Opera, Safari, Firefox, Edge)

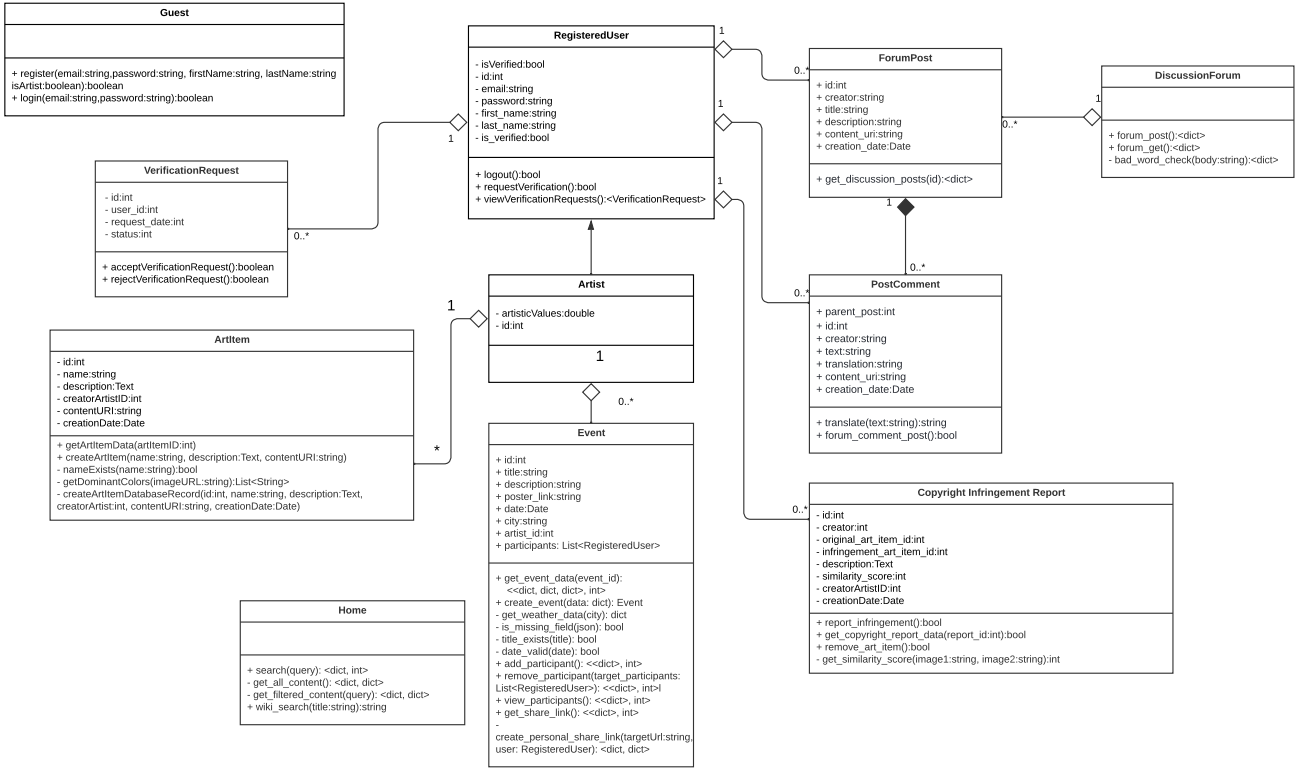
Use Case Diagram

USE CASE DIAGRAM



UML Class Diagram

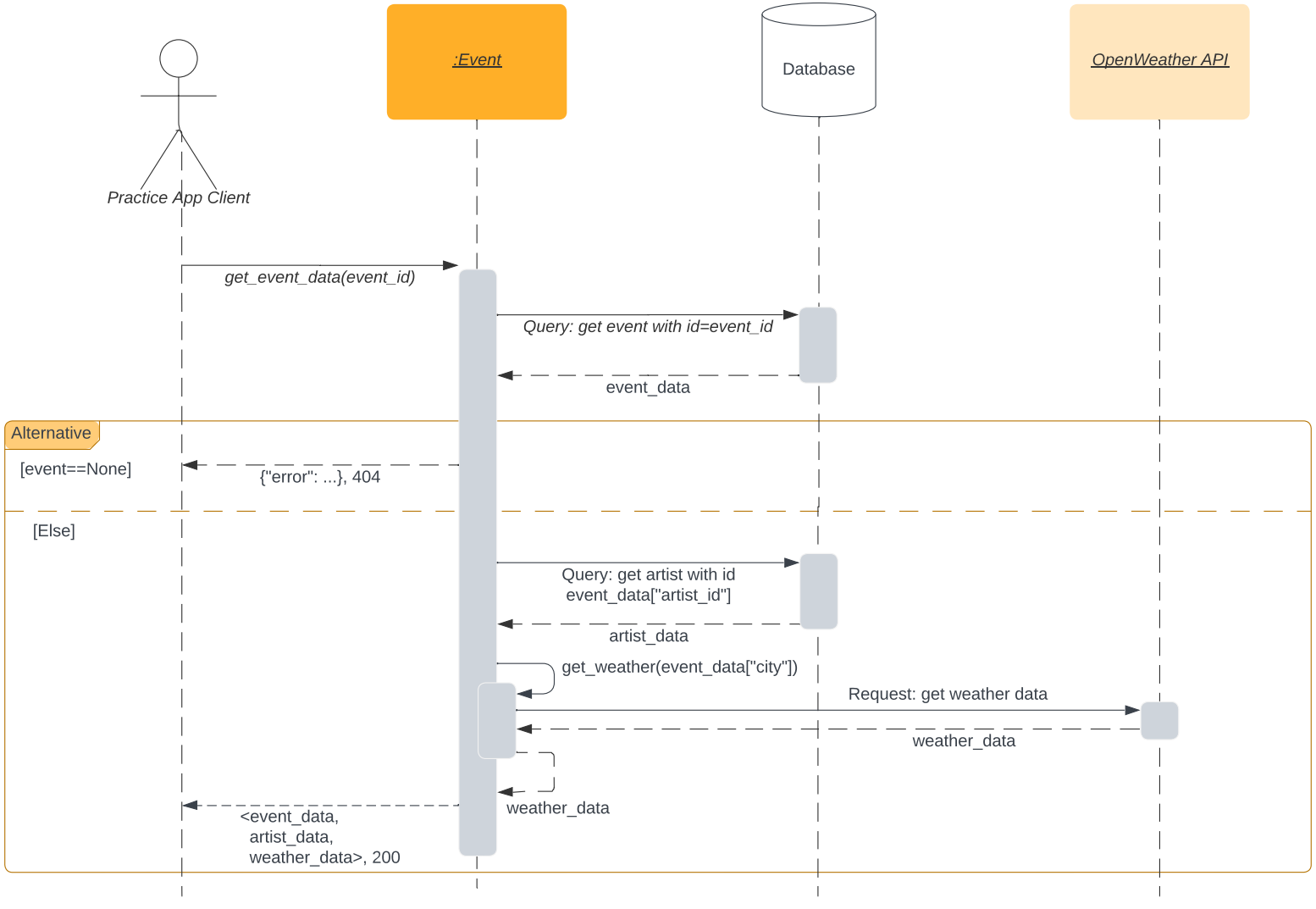
UML CLASS DIAGRAM



Sequence Diagrams

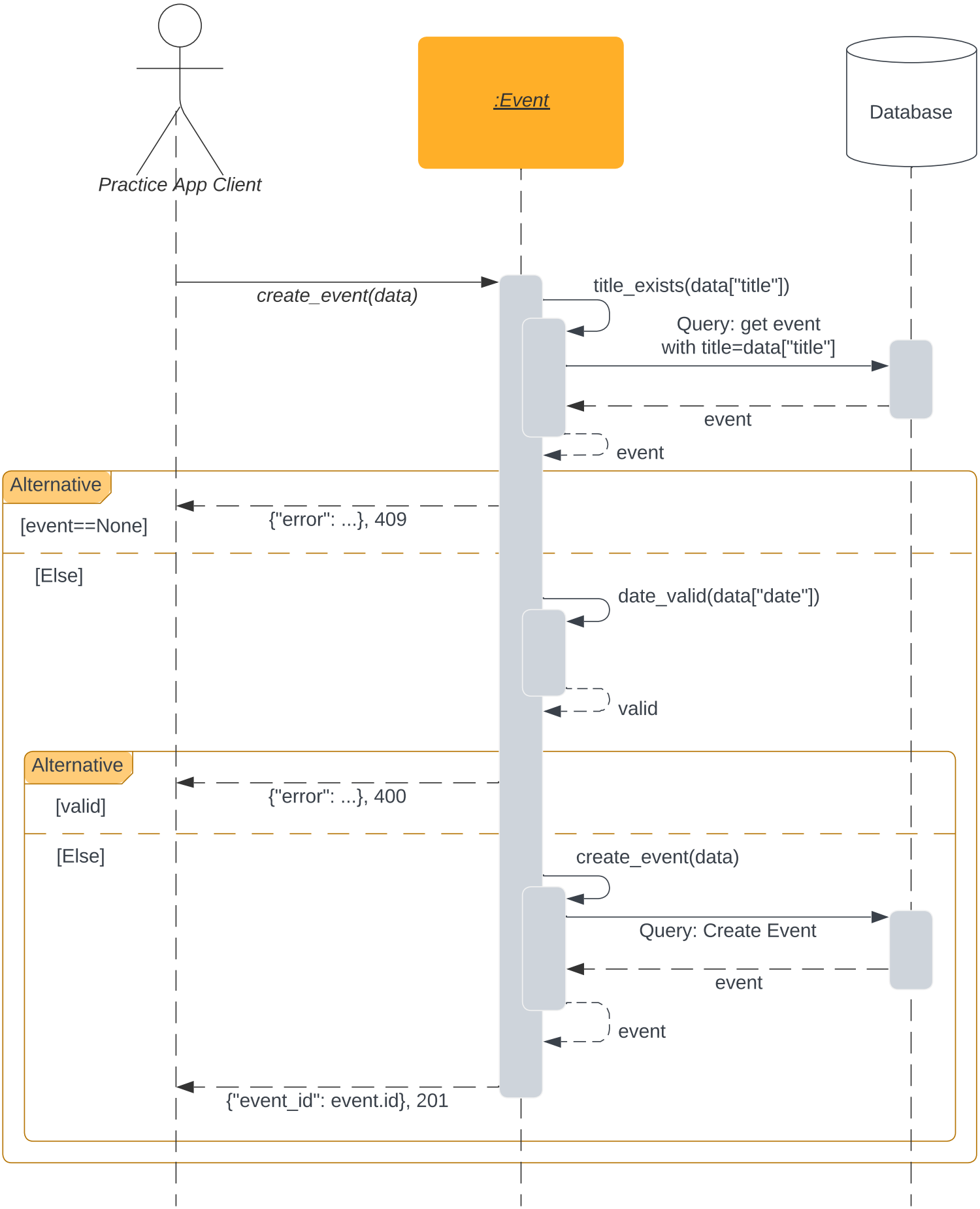
View Event Sequence Diagram

Cahid Arda Öz | May 20, 2022



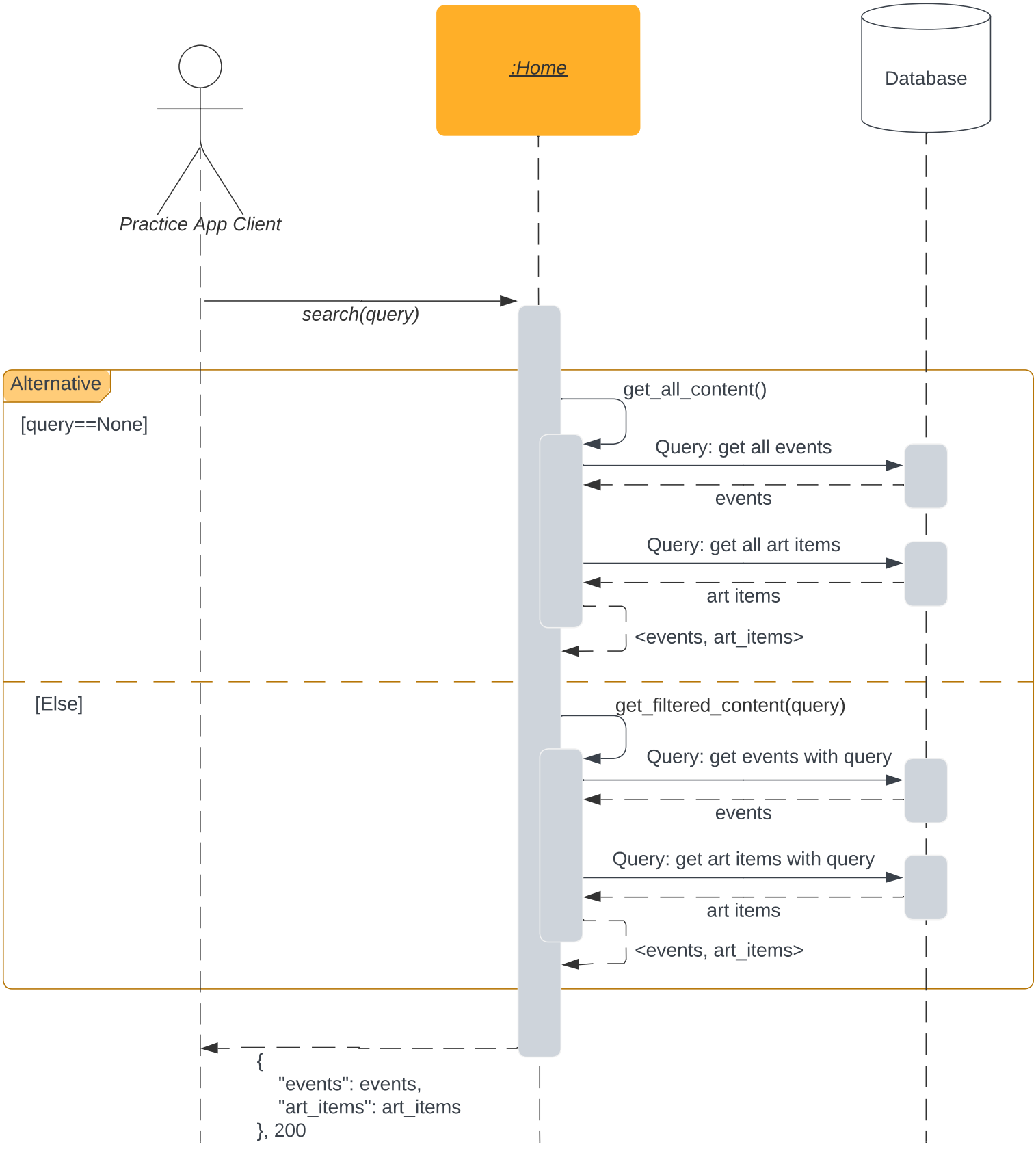
Create Event Sequence Diagram

Cahid Arda Öz | May 20, 2022



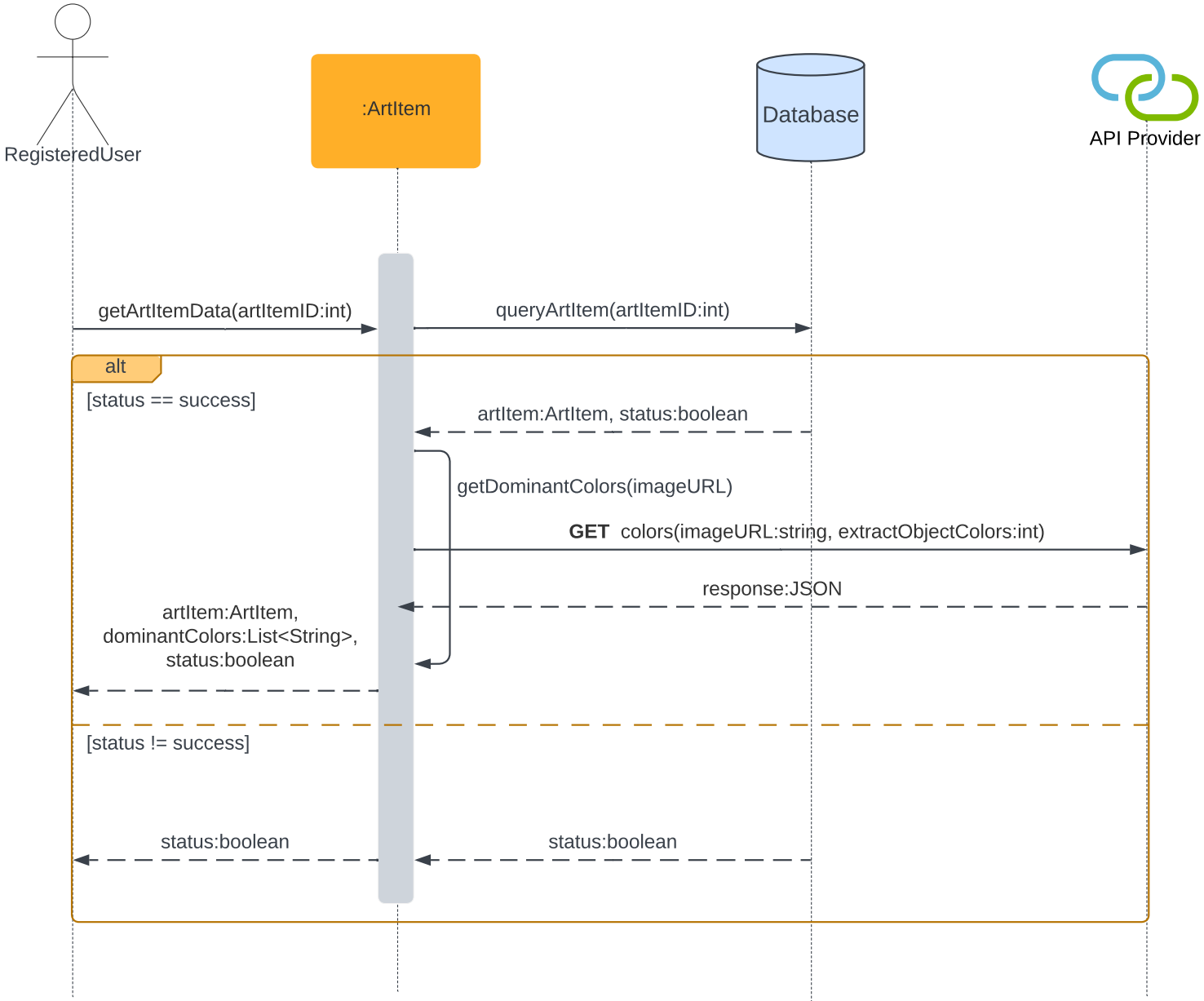
View Home Sequence Diagram

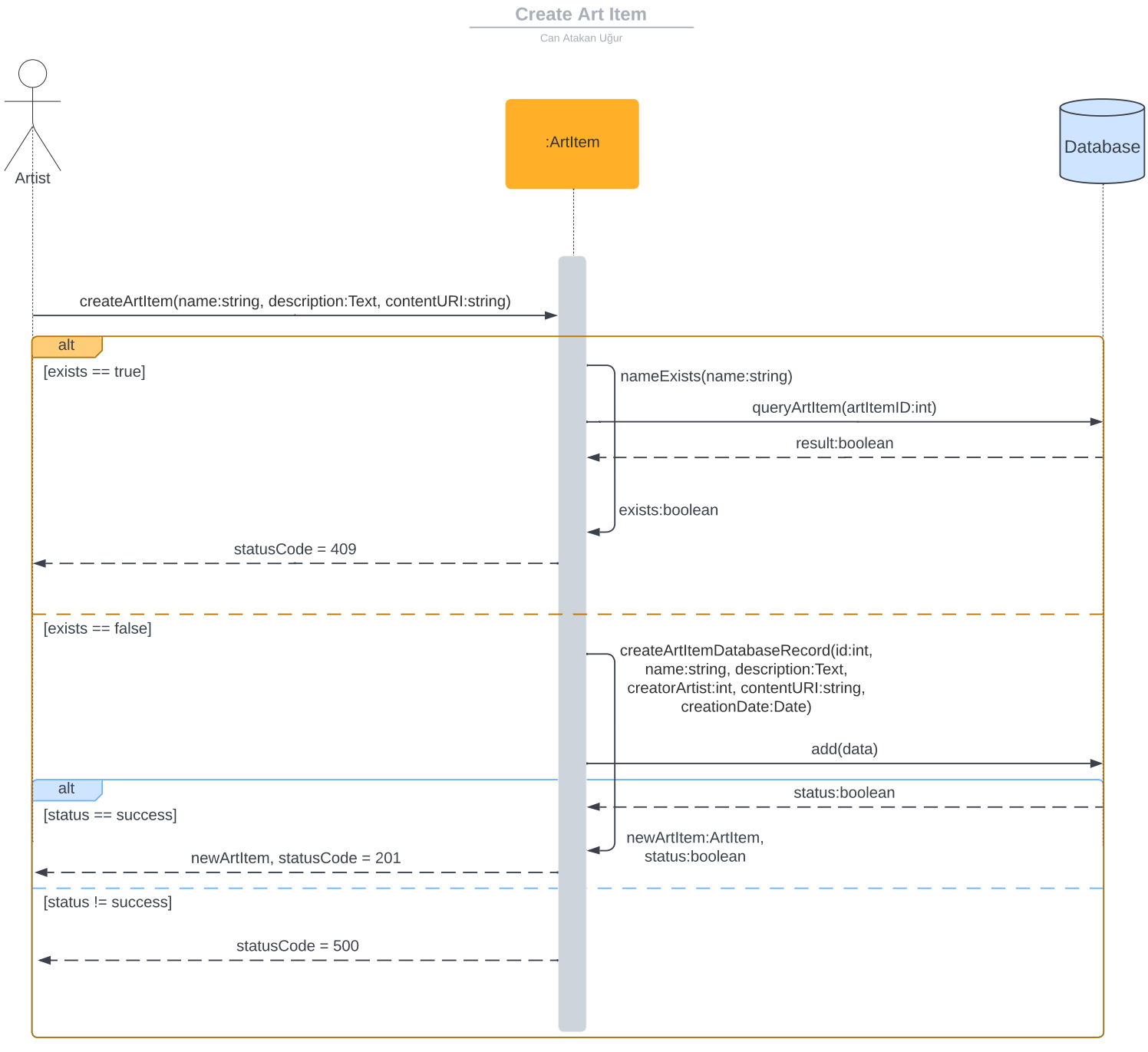
Cahid Arda Öz | May 20, 2022



View Art Item

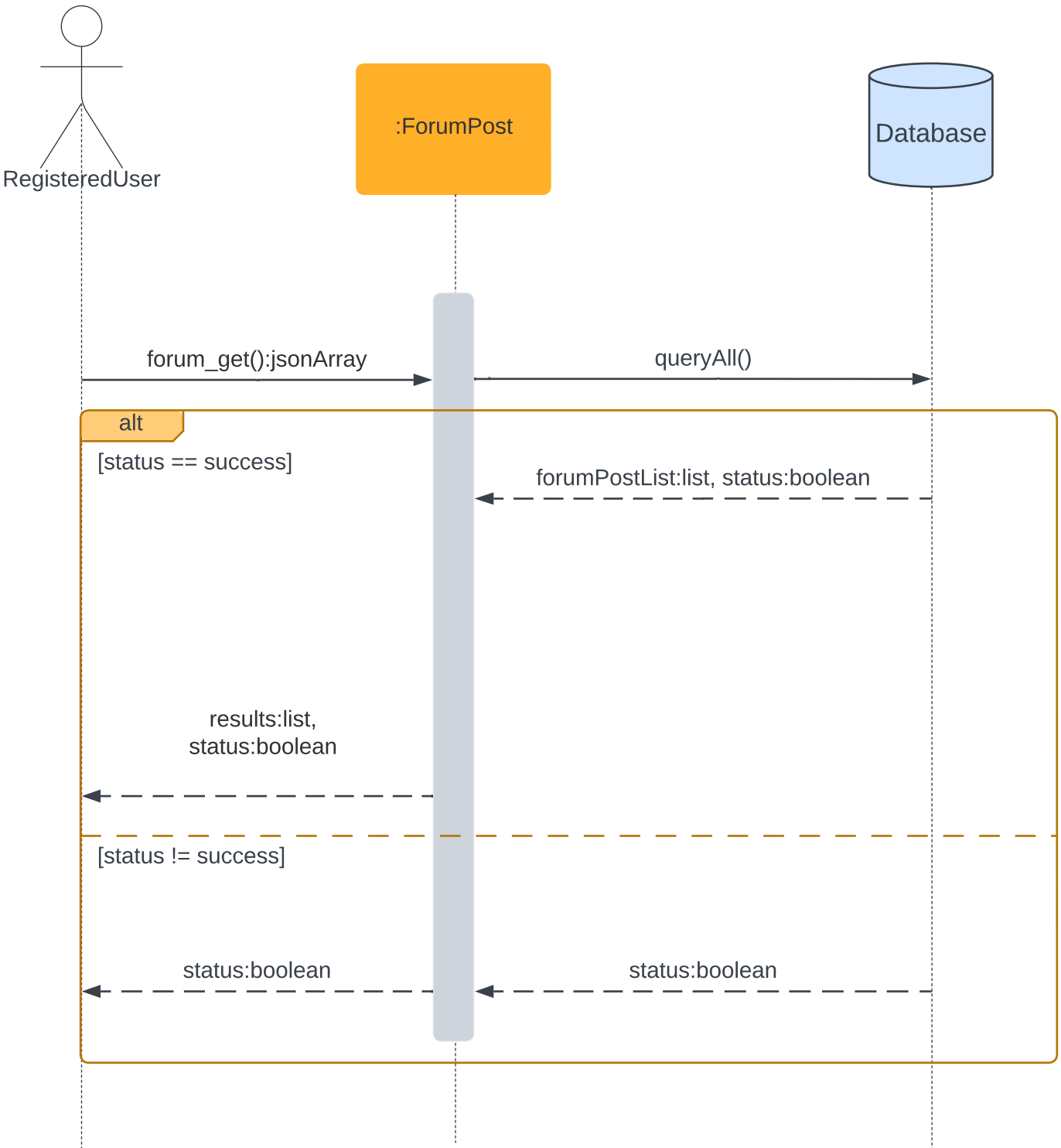
Can Atakan Uğur




Create Art Item
Can Atakan Uğur
Artist
:ArtItem
Database
createArtItem(name:string, description:Text, contentURI:string)
alt
[exists == true]
nameExists(name:string)
queryArtItem(artItemID:int)
result:boolean
exists:boolean
statusCode = 409
[exists == false]
createArtItemDatabaseRecord(id:int, name:string, description:Text, creatorArtist:int, contentURI:string, creationDate:Date)
add(data)
alt
[status == success]
status:boolean
newArtItem:ArtItem, status:boolean
newArtItem, statusCode = 201
[status != success]
statusCode = 500

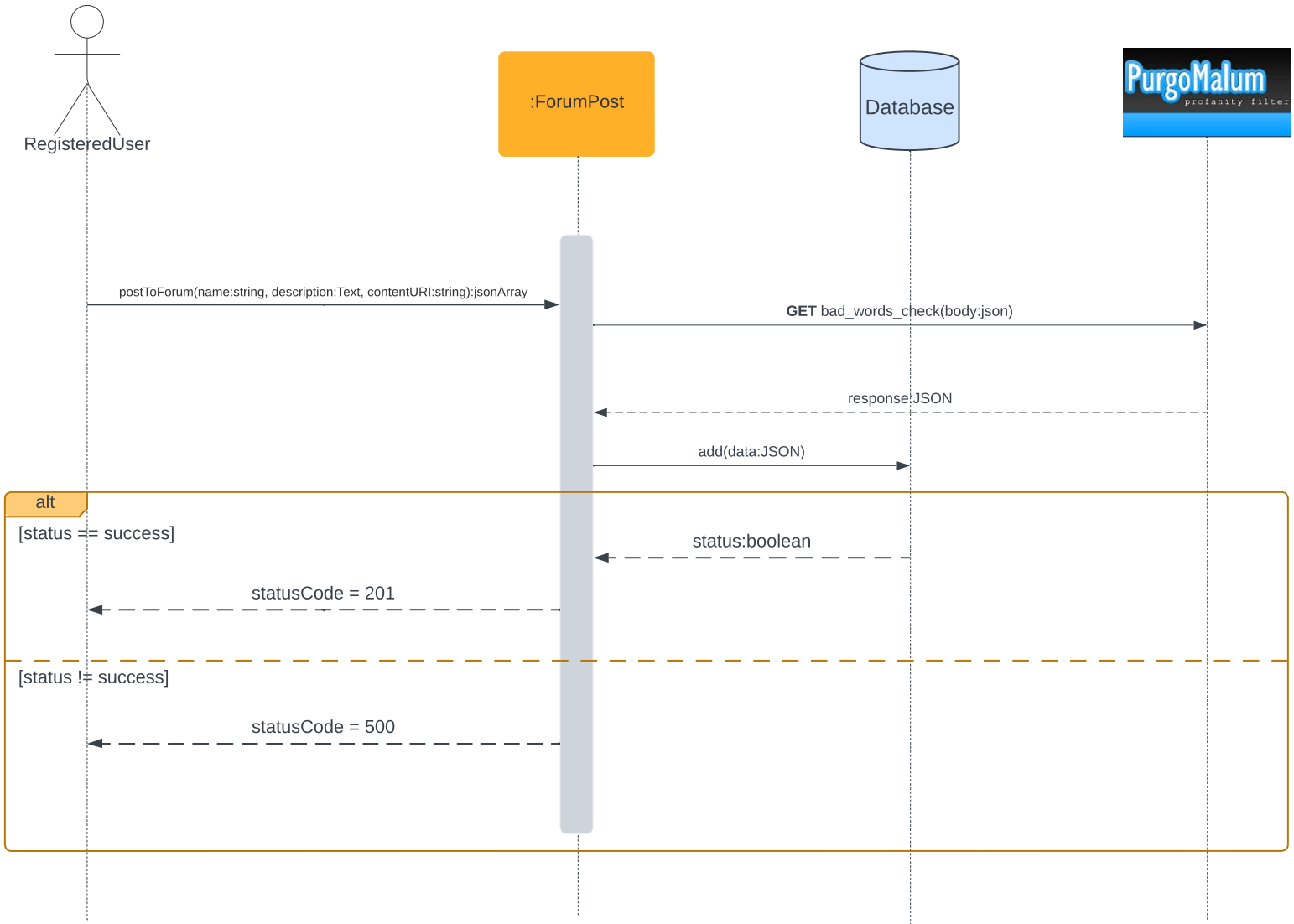
View Discussion Forum

Demet Yayla



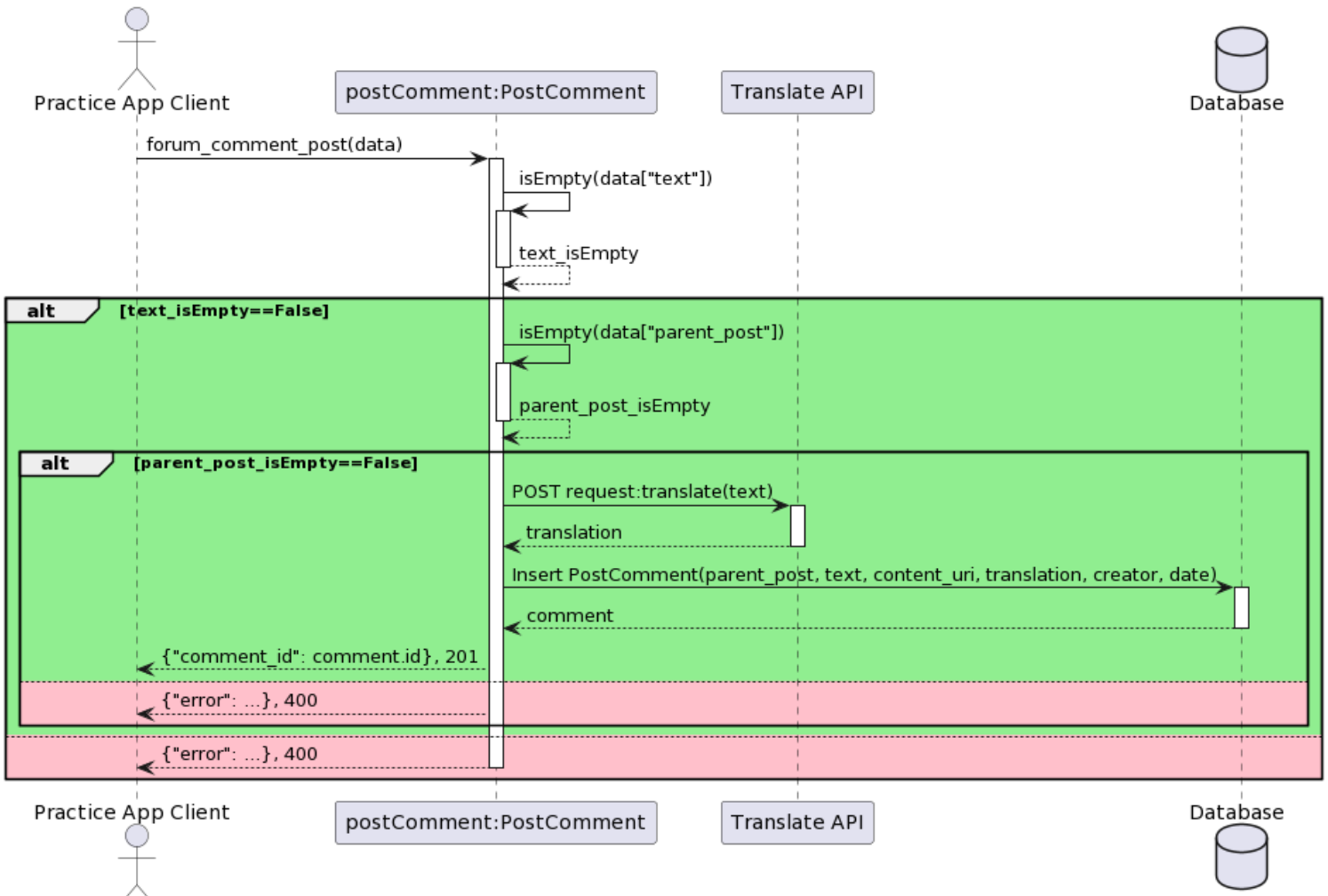
Post to Discussion Forum

Demet Yayla



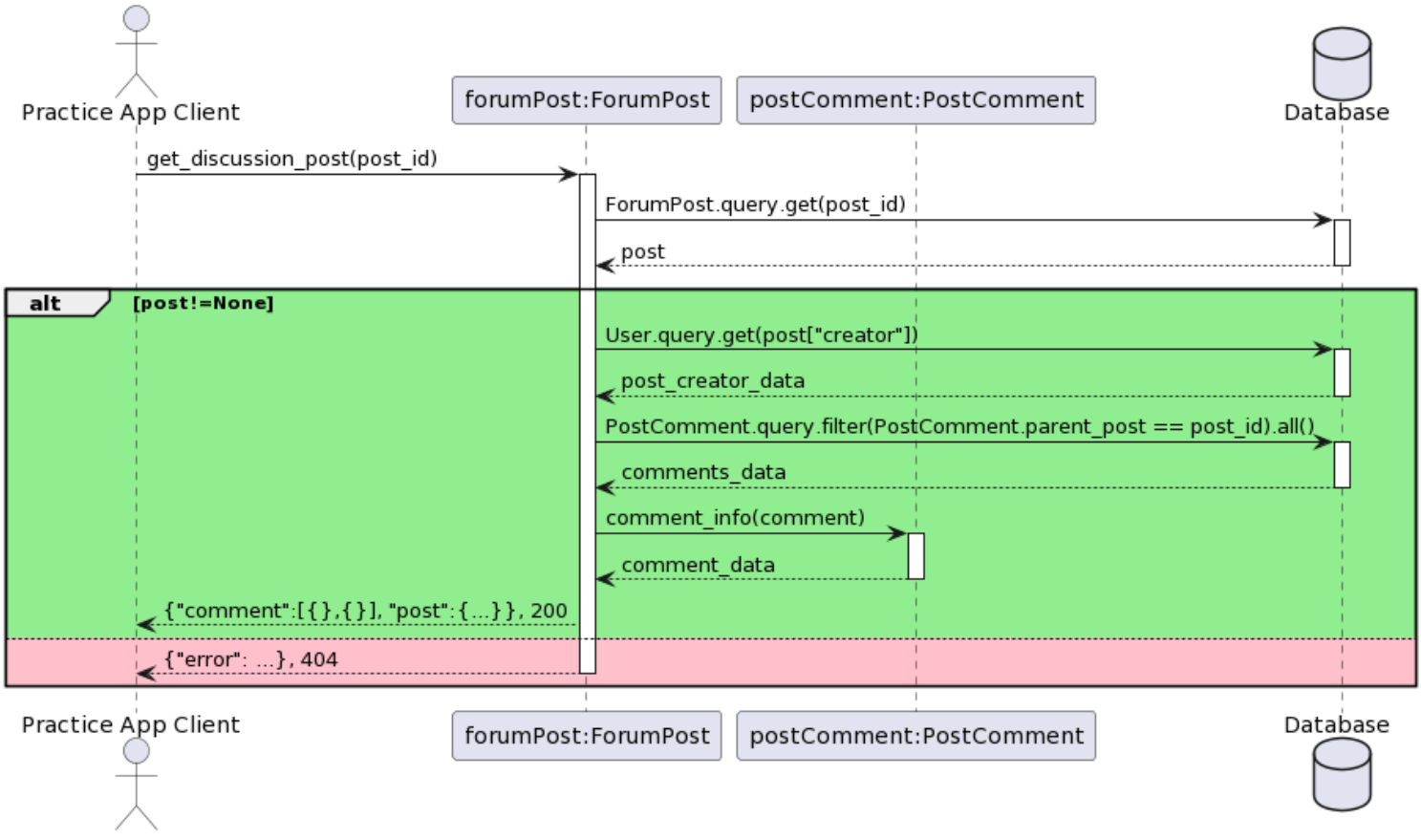
Post Comment to Discussion Post

Aziza Mankenova



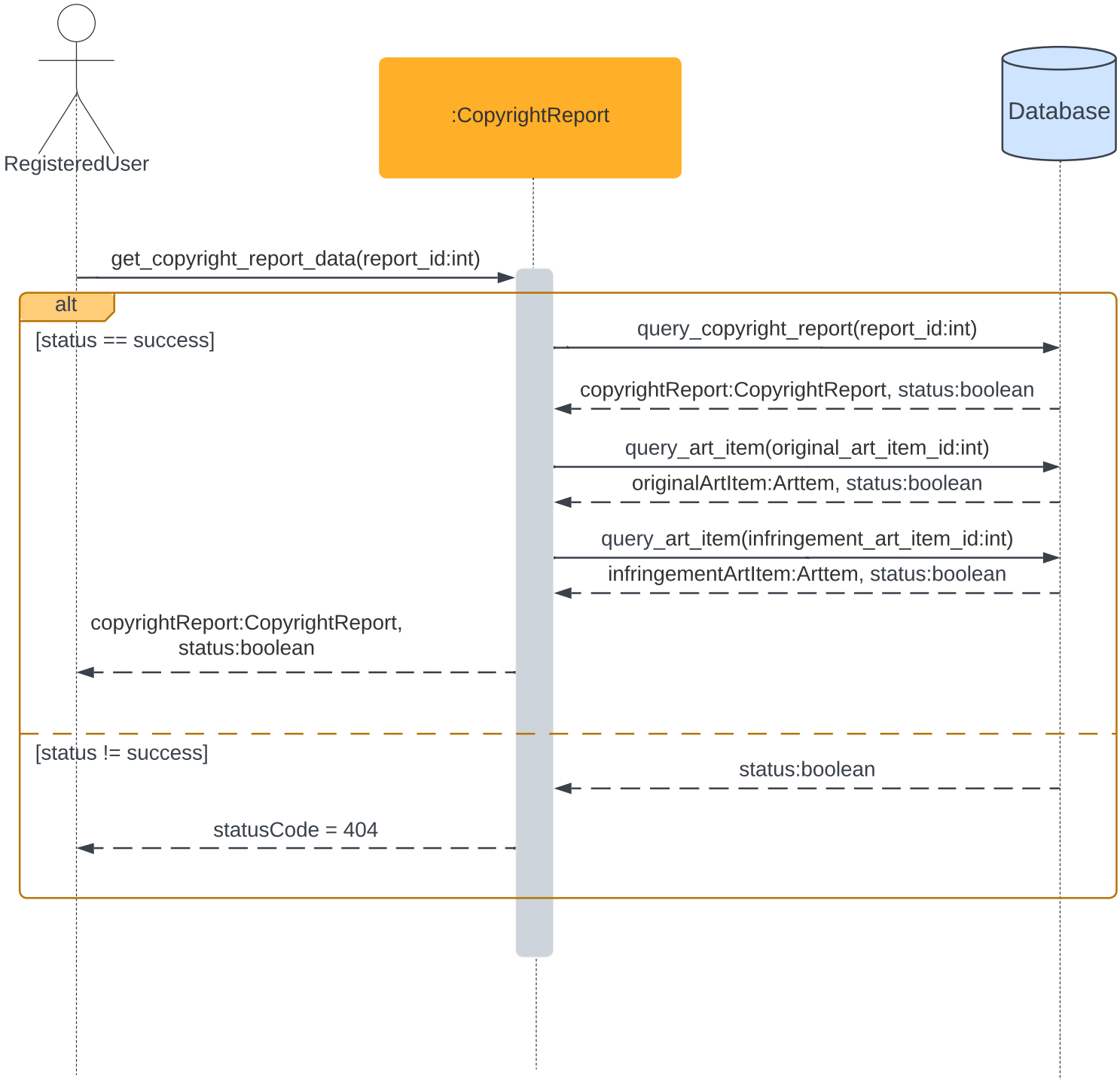
View Discussion Post with Comments

Aziza Mankenova



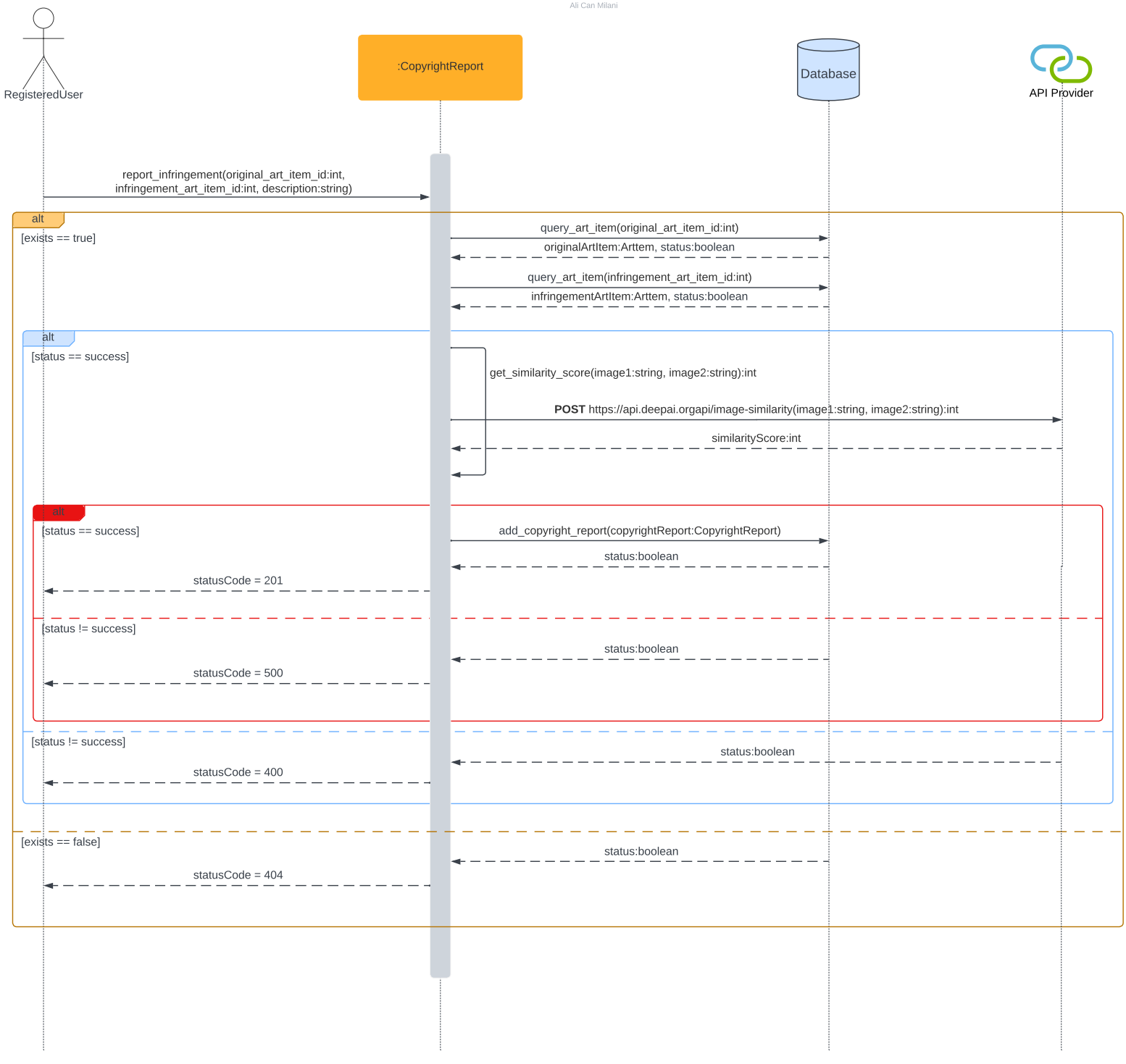
View Copyright Infringement Reports

Ali Can Milani



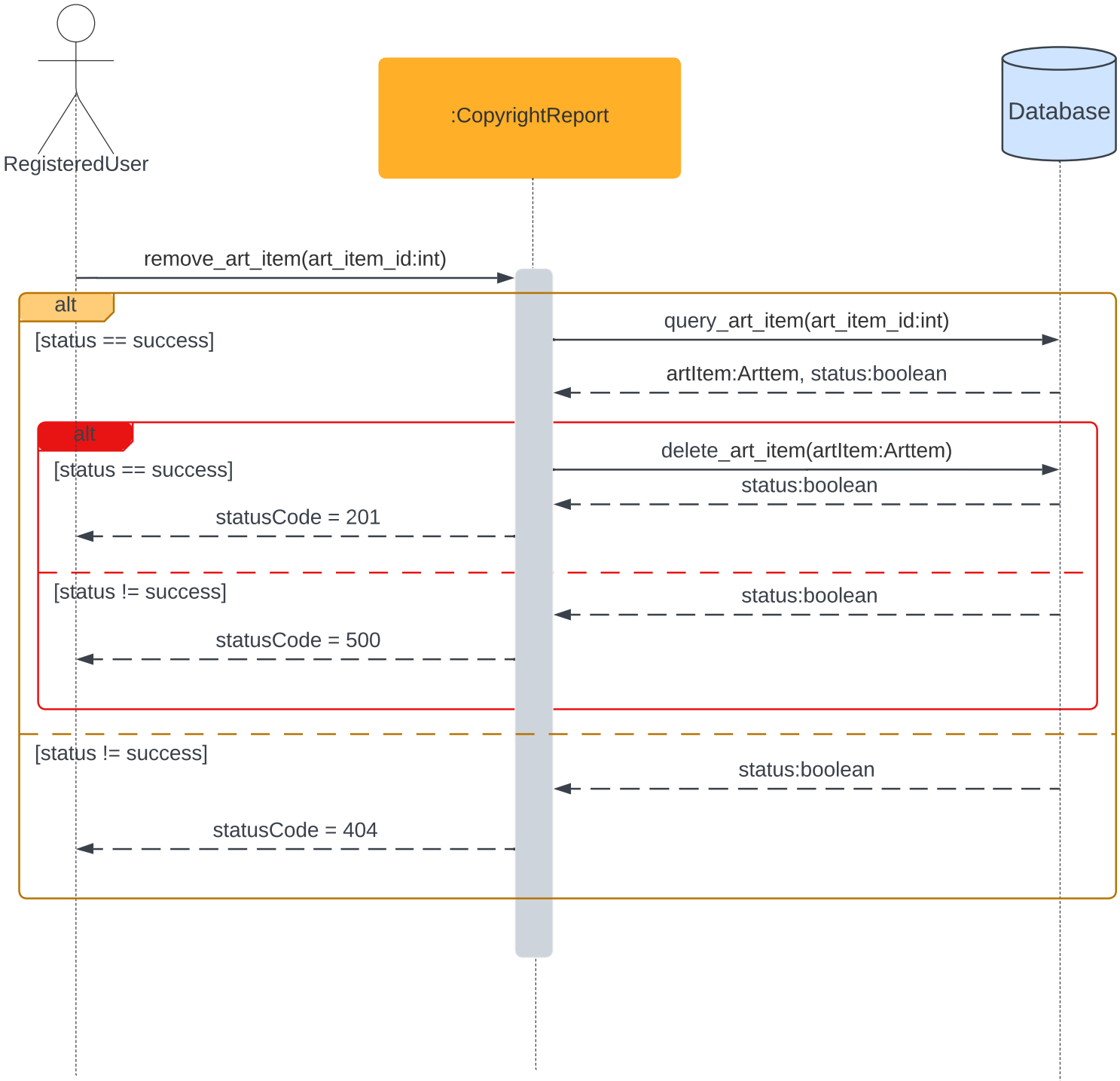
Create Copyright Infringement Reports

Ali Can Milani



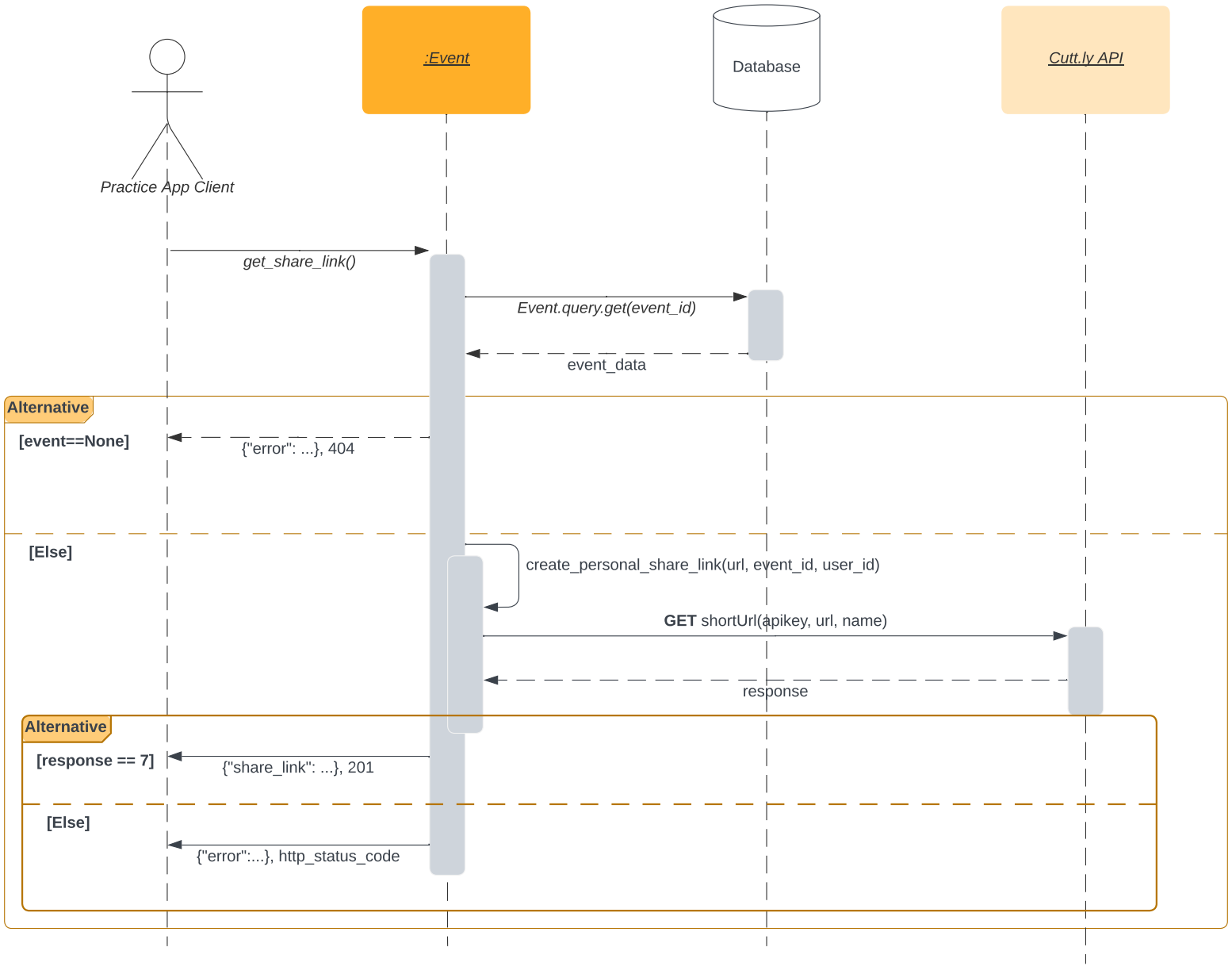
Remove Reported Art Item

Ali Can Milani



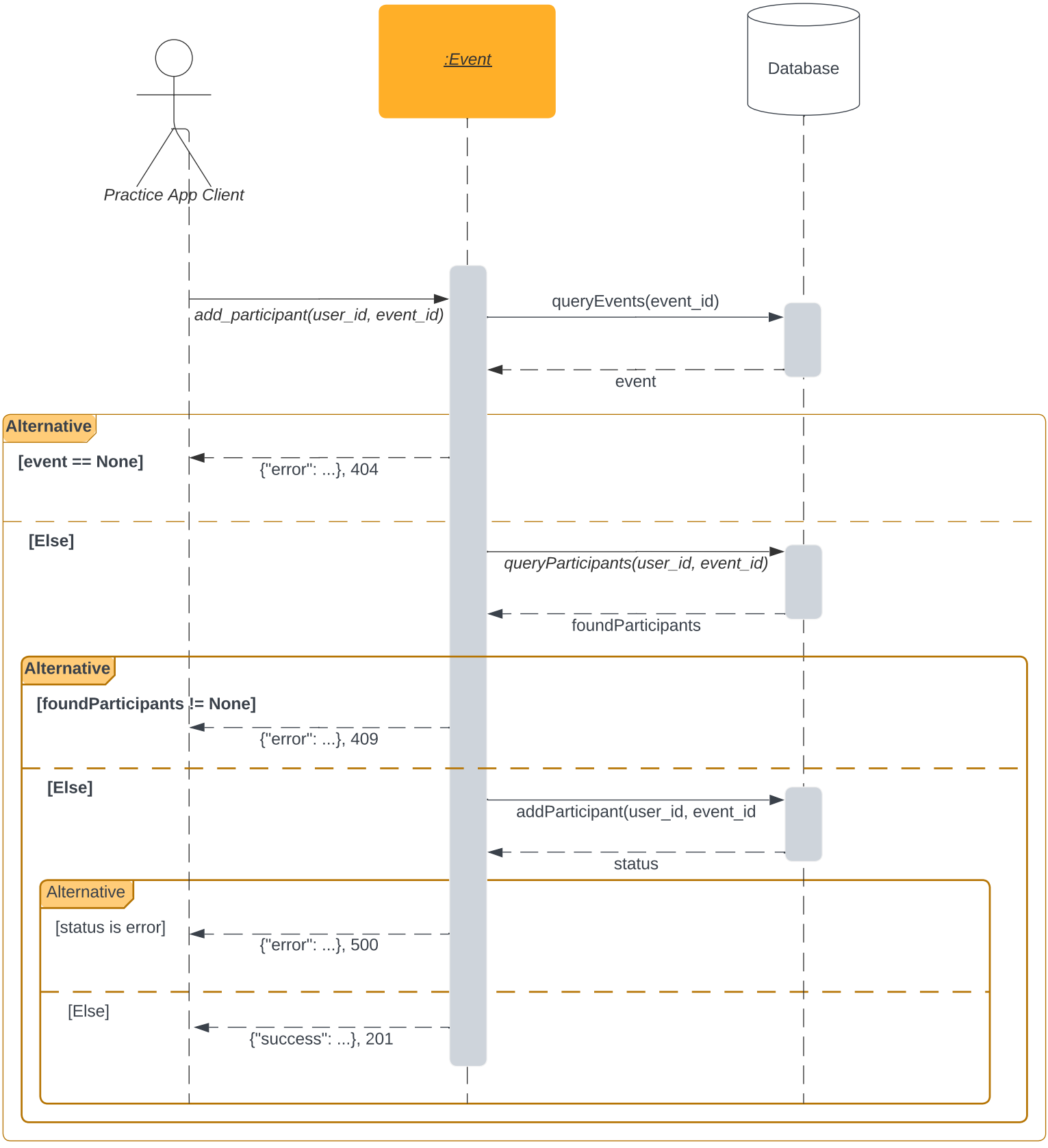
Get Share Link Sequence Diagram

Erim Erkin Doğan



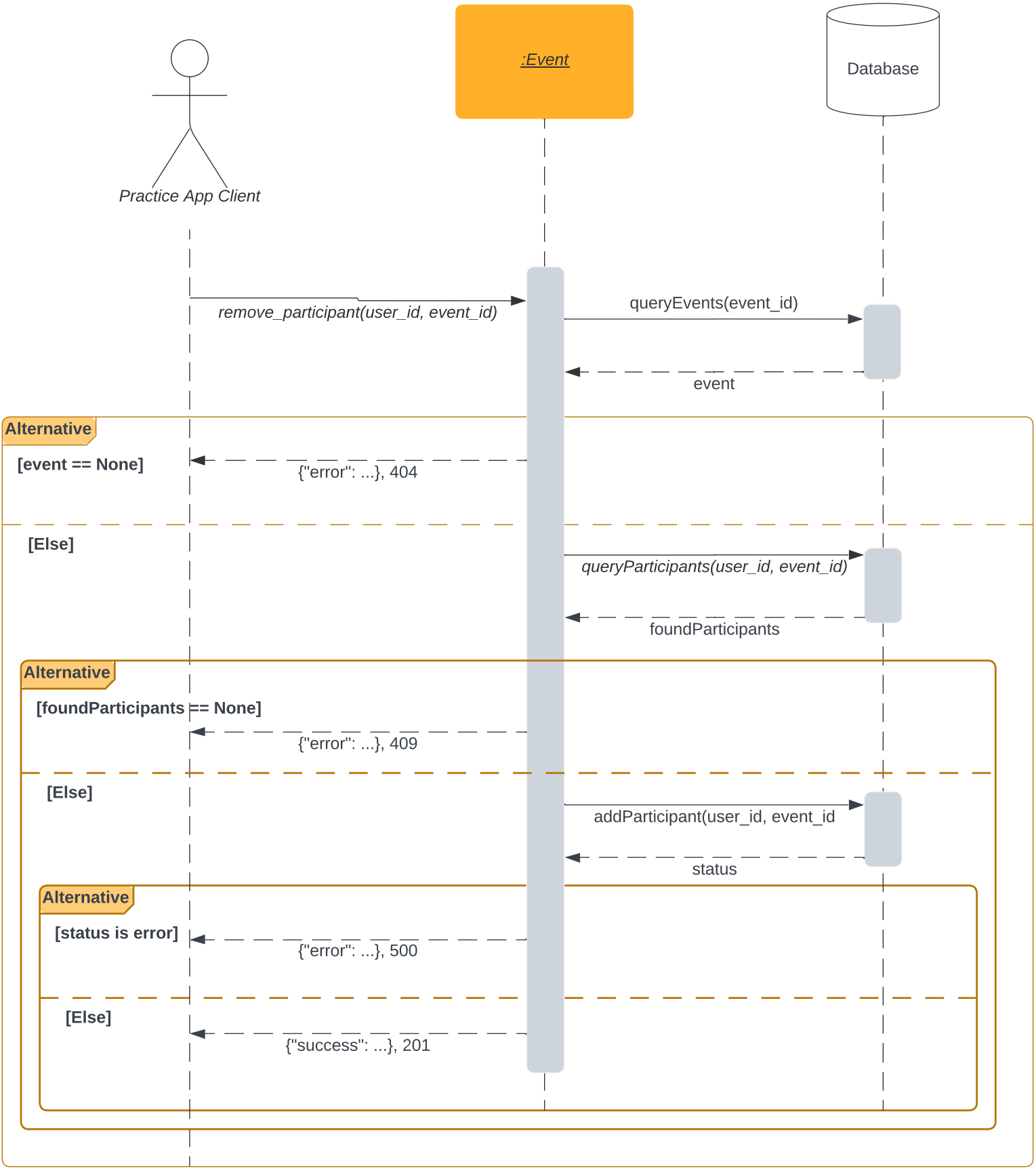
Add Participant Sequence Diagram

Erim Erkin Doğan



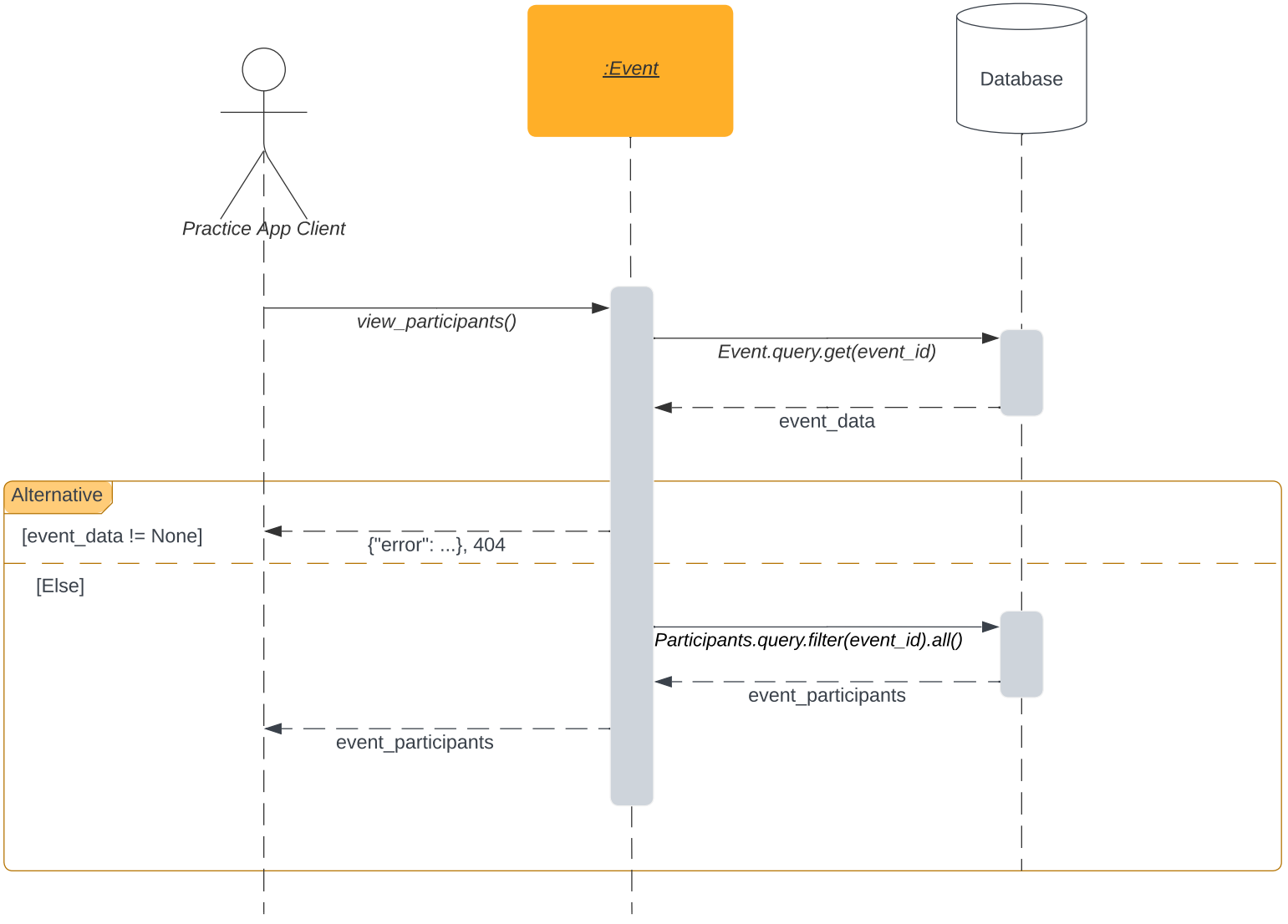
Remove Participant Sequence Diagram

Erim Erkin Doğan



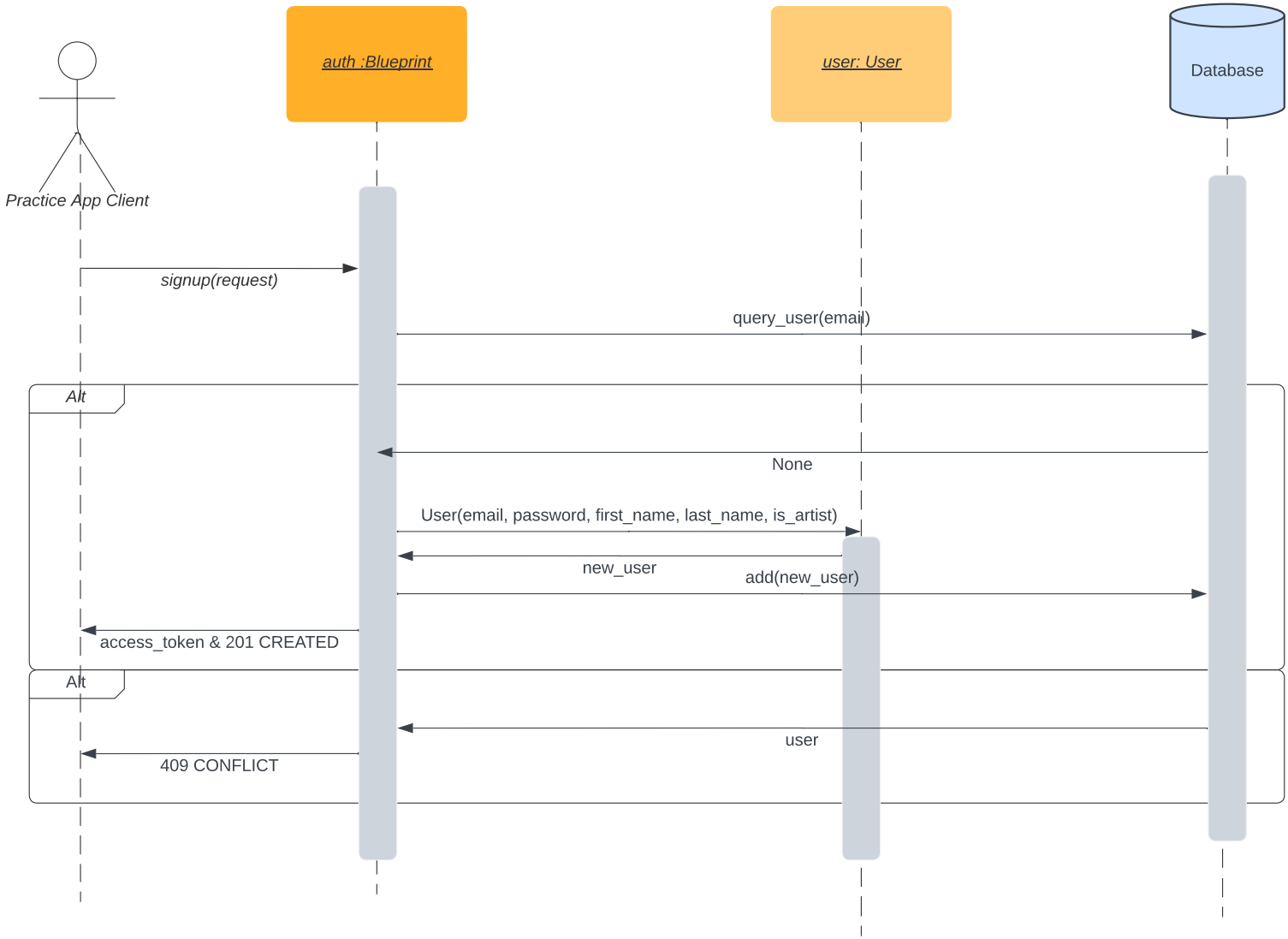
Viev Sequence Diagram

Erim Erkin Doğan | May 20, 2022



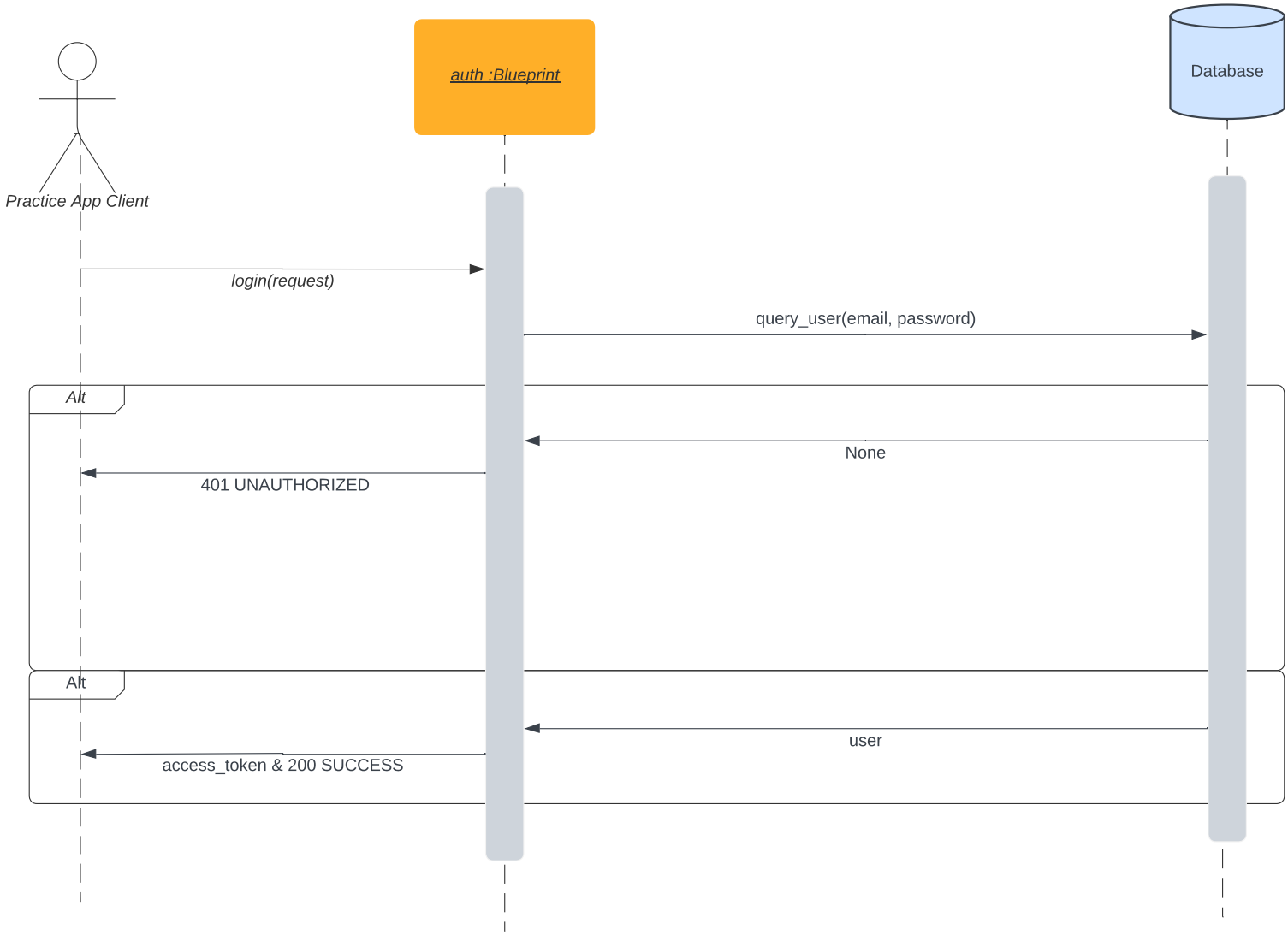
Sign Up

Güney İzol | May 20, 2022



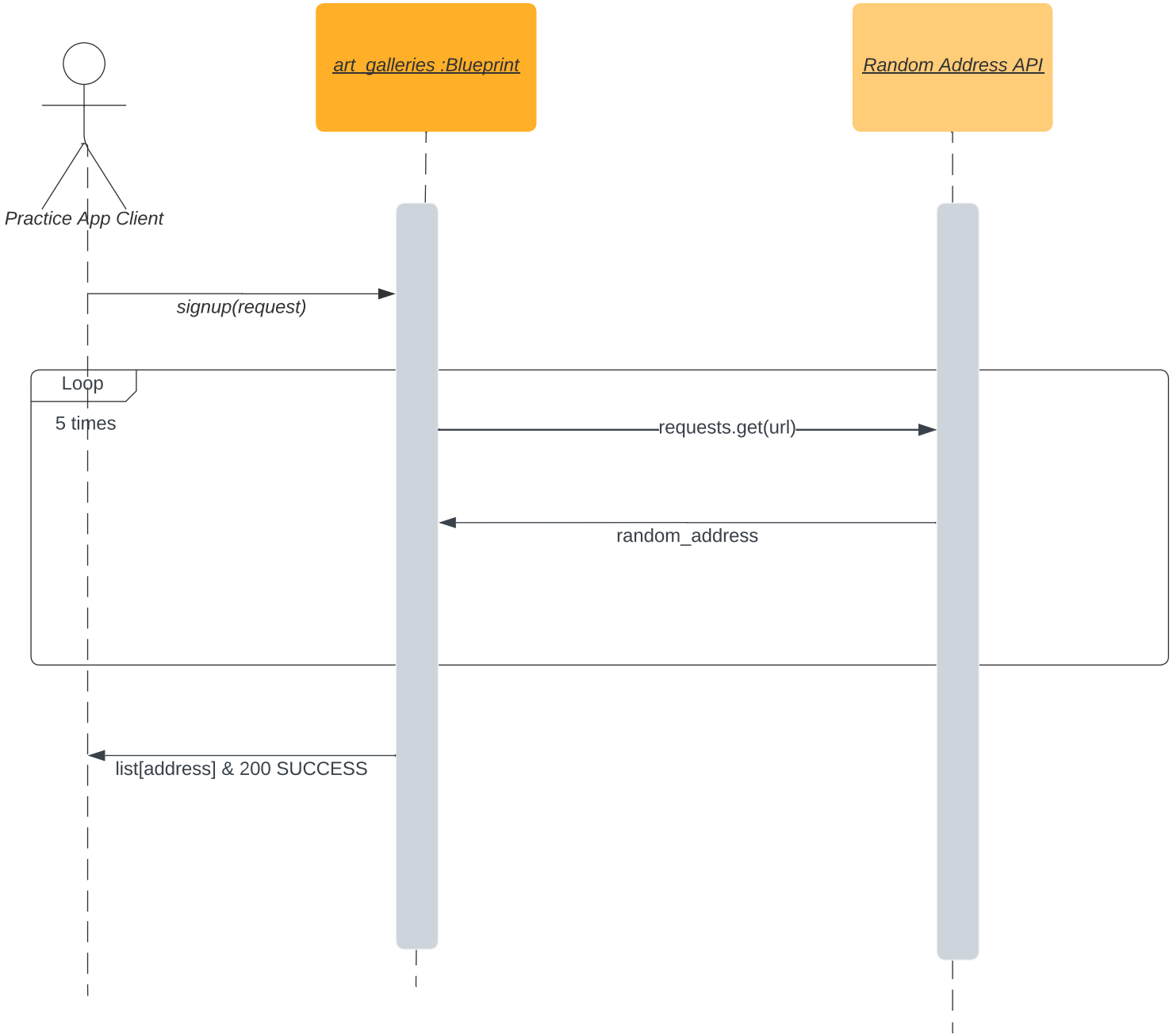
Login

Güney İzol | May 20, 2022



View Art Galleries

Güney İzol | May 20, 2022



Project Plan:
First Semester

		Name	Duration	Start	Finish	Prede...	Resource Names
1		**Infastructure Setup**	14 days?	**3/1/22 8:00 AM**	**3/18/22 5:00 PM**		
2		Personal Wiki Page	5 days?	3/1/22 8:00 AM	3/7/22 5:00 PM		All
3		Git research	5 days?	3/1/22 8:00 AM	3/7/22 5:00 PM		All
4		Communication Plan	5 days	3/1/22 8:00 AM	3/7/22 5:00 PM		Demet Yayla;Atilla Türkmen;Aziza Mankenova;Güney İzol
5		Customizing Issue Tags	5 days	3/1/22 8:00 AM	3/7/22 5:00 PM		Aziza Mankenova;Can Atakan Uğur;Güney İzol
6		Repository Search	5 days?	3/1/22 8:00 AM	3/7/22 5:00 PM		All
7		Similar Project Research	3 days?	3/16/22 8:00 AM	3/18/22 5:00 PM		All
8		**Software Requirements Specification**	9 days?	**3/8/22 8:00 AM**	**3/18/22 5:00 PM**		
9		User requirements	5 days?	3/8/22 8:00 AM	3/14/22 5:00 PM		All
10		System requirements	5 days?	3/8/22 8:00 AM	3/14/22 5:00 PM		All
11		Non-functional requirements	5 days?	3/8/22 8:00 AM	3/14/22 5:00 PM		Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Erim Erkin Doğan;Güney İzol;Mehmet Emin İpekdal;Musa Ş...
12		Description Analysis	3 days	3/16/22 8:00 AM	3/18/22 5:00 PM		Aziza Mankenova;Can Atakan Uğur;Cahid Arda Öz;Demet Yayla
13		**Scenarios and Mockups**	4 days?	**3/16/22 8:00 AM**	**3/21/22 5:00 PM**		
14		Scenario 1	4 days?	3/16/22 8:00 AM	3/21/22 5:00 PM		Demet Yayla;Erim Erkin Doğan;Güney İzol;Rafet Oğuz Pançuk
15		Scenario 2	4 days?	3/16/22 8:00 AM	3/21/22 5:00 PM		Ali Can Milani;Güney İzol;Musa Şimşek
16		Scenario 3	4 days?	3/16/22 8:00 AM	3/21/22 5:00 PM		Atilla Türkmen;Can Atakan Uğur
17		Mock-up for Scenario 1	2 days?	3/18/22 8:00 AM	3/21/22 5:00 PM		Aziza Mankenova;Erim Erkin Doğan;Mehmet Emin İpekdal;Musa Şimşek
18		Mock-up for Scenario 2	2 days?	3/18/22 8:00 AM	3/21/22 5:00 PM		Cahid Arda Öz;Erim Erkin Doğan;Mehmet Emin İpekdal;Musa Şimşek
19		Mock-up for Scenario 3	2 days?	3/18/22 8:00 AM	3/21/22 5:00 PM		Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Mehmet Emin İpekdal;Musa Şimşek
20		Post-customer Meeting Review	3 days	3/30/22 8:00 AM	4/1/22 5:00 PM		Cahid Arda Öz;Can Atakan Uğur
21		**Software Design**	11 days?	**4/1/22 8:00 AM**	**4/15/22 5:00 PM**		
22		Requirements Review	8 days?	4/1/22 8:00 AM	4/12/22 5:00 PM		All
23		Use Case Diagram	2 days?	4/1/22 8:00 AM	4/4/22 5:00 PM		All
24		Class Diagram	6 days?	4/5/22 8:00 AM	4/12/22 5:00 PM		All
25		Sequence Diagrams	5 days?	4/11/22 8:00 AM	4/15/22 5:00 PM		All
26		**Planning**	5 days?	**4/11/22 8:00 AM**	**4/15/22 5:00 PM**		
27		RAM	5 days?	4/11/22 8:00 AM	4/15/22 5:00 PM		All
28		Project Planning	2 days	4/14/22 8:00 AM	4/15/22 5:00 PM		Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Hasan Bingölbali
29		Milestone 1 Report	3 days?	4/13/22 8:00 AM	4/15/22 5:00 PM		All

30		**Practice Application**	24 days?	**4/18/22 8:00 AM**	**5/19/22 5:00 PM**	**29**	
31		**Initial Planning**	5 days	**4/18/22 8:00 AM**	**4/22/22 5:00 PM**		
32		Research on different frameworks	5 days	4/18/22 8:00 AM	4/22/22 5:00 PM		All
33		Deciding which API functions to implement	5 days	4/18/22 8:00 AM	4/22/22 5:00 PM		All
34		Studying APIs	5 days	4/18/22 8:00 AM	4/22/22 5:00 PM		All
35		**Research**	5 days?	**4/25/22 8:00 AM**	**4/29/22 5:00 PM**	**31**	
36		Pull Request and Review Tools Research	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Ali Can Milani
37		Research on Flask Framework	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Erim Erkin Doğan;Atilla Türkmen
38		Docker Research	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Can Atakan Uğur;Musa Şimşek
39		Research on testing API using Postman	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Aziza Mankenova
40		Unittest Research	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Demet Yayla
41		AWS Research	5 days?	4/25/22 8:00 AM	4/29/22 5:00 PM		Cahid Arda Öz
42		**Software Design for Practice App**	13 days?	**5/2/22 8:00 AM**	**5/18/22 5:00 PM**	**35**	
43		Requirements	13 days?	5/2/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa...
44		Use Case Diagram	13 days?	5/2/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa...
45		Class Diagram	13 days?	5/2/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa...
46		Sequence Diagrams	13 days?	5/2/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa...
47		**Forming Initial Code Structure**	5 days	**5/2/22 8:00 AM**	**5/6/22 5:00 PM**	**35**	
48		Creating the Fundamental Back-end Structure	5 days	5/2/22 8:00 AM	5/6/22 5:00 PM		Güney İzol;Erim Erkin Doğan;Musa Şimşek;Hasan Bingölbali
49		Creating the Fundamental Database Structure	5 days	5/2/22 8:00 AM	5/6/22 5:00 PM		Can Atakan Uğur;Aziza Mankenova;Mehmet Emin İpekdal
50		Creating the Fundamental Front-end Structure	5 days	5/2/22 8:00 AM	5/6/22 5:00 PM		Cahid Arda Öz;Atilla Türkmen;Demet Yayla
51		**Preparing Additional Project Configurations**	8 days?	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**	**47**	
52		Creating API documentation template on Wiki	5 days?	5/9/22 8:00 AM	5/13/22 5:00 PM		Erim Erkin Doğan
53		Docker Configuration	5 days	5/9/22 8:00 AM	5/13/22 5:00 PM		Can Atakan Uğur;Erim Erkin Doğan
54		AWS Deployment Configuration	5 days	5/9/22 8:00 AM	5/13/22 5:00 PM		Cahid Arda Öz;Aziza Mankenova
55		Integrating Authentication	4 days?	5/9/22 8:00 AM	5/12/22 5:00 PM		Cahid Arda Öz;Güney İzol
56		Integrating Swagger for API Documentation	8 days?	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
57		Creating CI/CD Workflows	8 days?	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
58		**Implementation with Testing & Documentation**	8 days	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**	**47**	

59		**Art Item Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Can Atakan Uğur**
60		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Can Atakan Uğur**
61		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
62		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
63		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
64		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
65		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Can Atakan Uğur**
66		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
67		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
68		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Can Atakan Uğur**
69		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
70		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Can Atakan Uğur
71		**Authentication Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Güney İzol**
72		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Güney İzol**
73		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
74		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
75		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
76		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
77		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Güney İzol**
78		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
79		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
80		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Güney İzol**
81		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
82		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Güney İzol
83		**Comment Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Aziza Mankenova**
84		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Aziza Mankenova**
85		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
86		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
87		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova

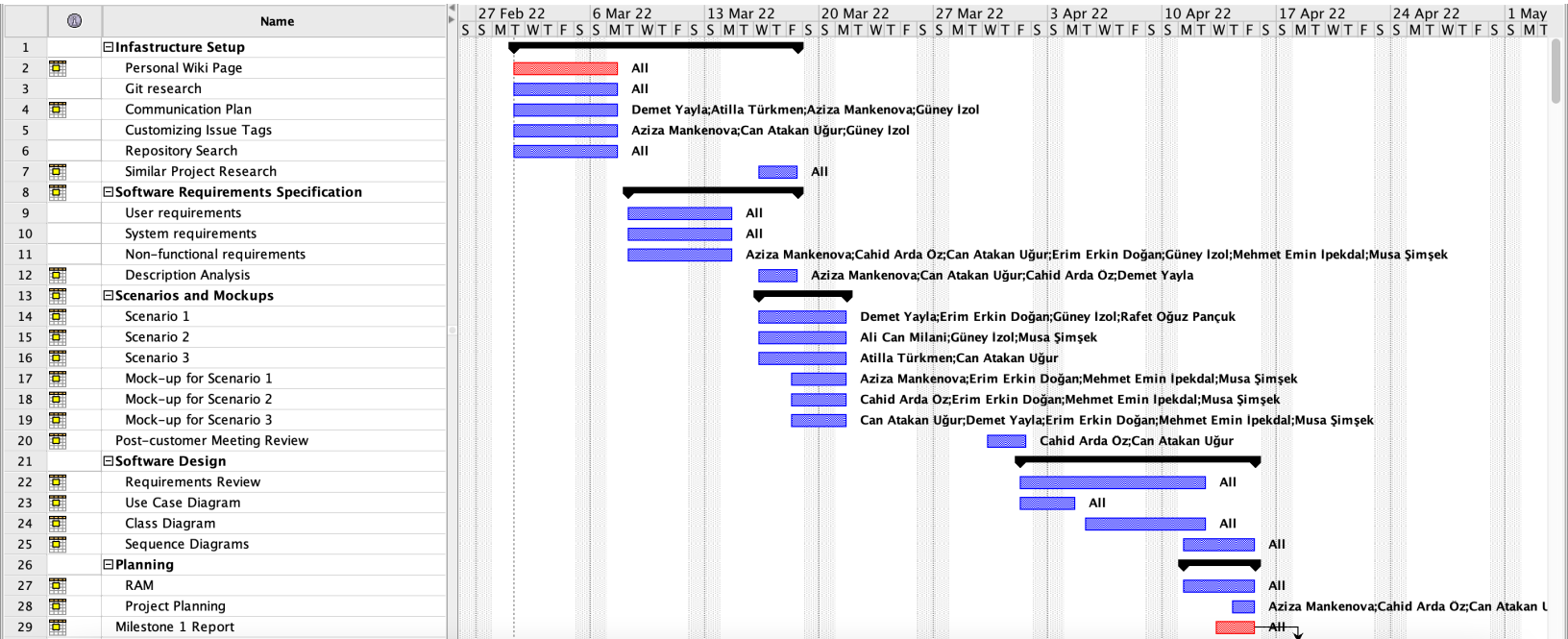
88		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
89		**⊟Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Aziza Mankenova**
90		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
91		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
92		**⊟Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Aziza Mankenova**
93		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
94		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Aziza Mankenova
95		**⊟Copyright Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Ali Can Milani**
96		**⊟Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Ali Can Milani**
97		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
98		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
99		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
100		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
101		**⊟Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Ali Can Milani**
102		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
103		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
104		**⊟Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Ali Can Milani**
105		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
106		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Ali Can Milani
107		**⊟Discussion Forum Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Demet Yayla**
108		**⊟Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Demet Yayla**
109		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
110		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
111		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
112		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
113		**⊟Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Demet Yayla**
114		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
115		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
116		**⊟Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Demet Yayla**

117		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
118		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Demet Yayla
119		**Event Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
120		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
121		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
122		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
123		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
124		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
125		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
126		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
127		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
128		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
129		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
130		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
131		**Event Participation Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Erim Erkin Doğan**
132		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Erim Erkin Doğan**
133		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
134		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
135		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
136		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
137		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Erim Erkin Doğan**
138		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
139		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
140		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Erim Erkin Doğan**
141		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
142		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Erim Erkin Doğan
143		**Home Page & Search Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
144		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
145		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz

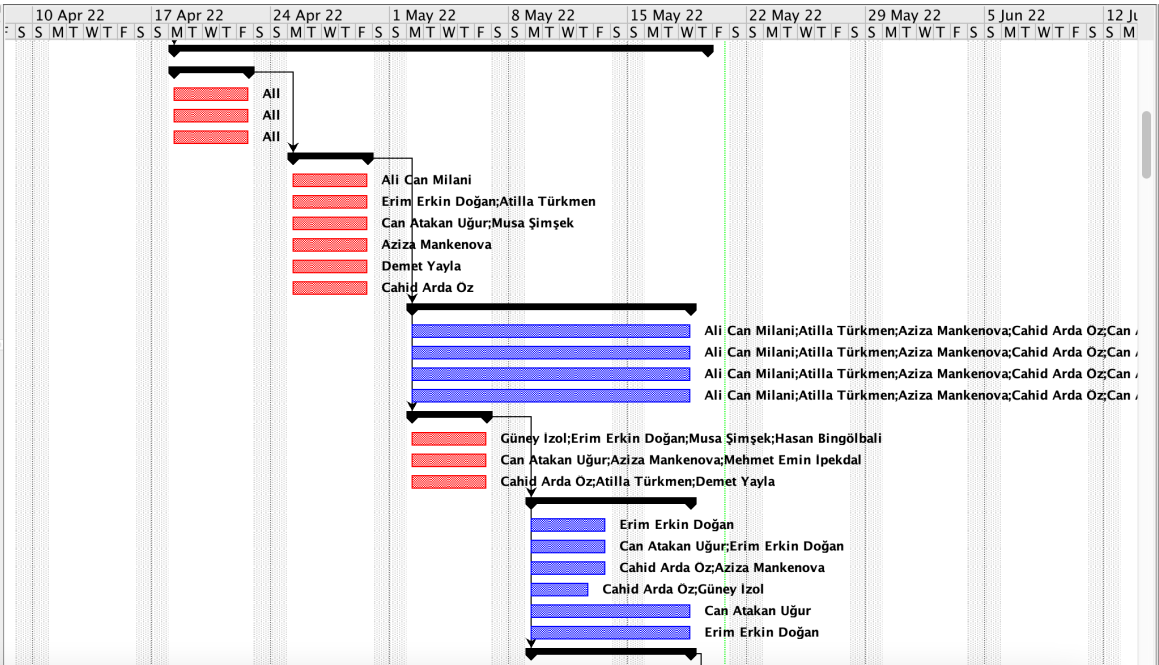
146		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
147		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
148		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
149		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
150		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
151		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
152		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Cahid Arda Öz**
153		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
154		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Cahid Arda Öz
155		**Verification Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Musa Şimşek**
156		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Musa Şimşek**
157		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
158		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
159		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
160		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
161		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Musa Şimşek**
162		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
163		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
164		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Musa Şimşek**
165		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
166		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Musa Şimşek
167		**User Profile Page Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
168		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
169		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
170		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
171		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
172		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
173		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
174		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen

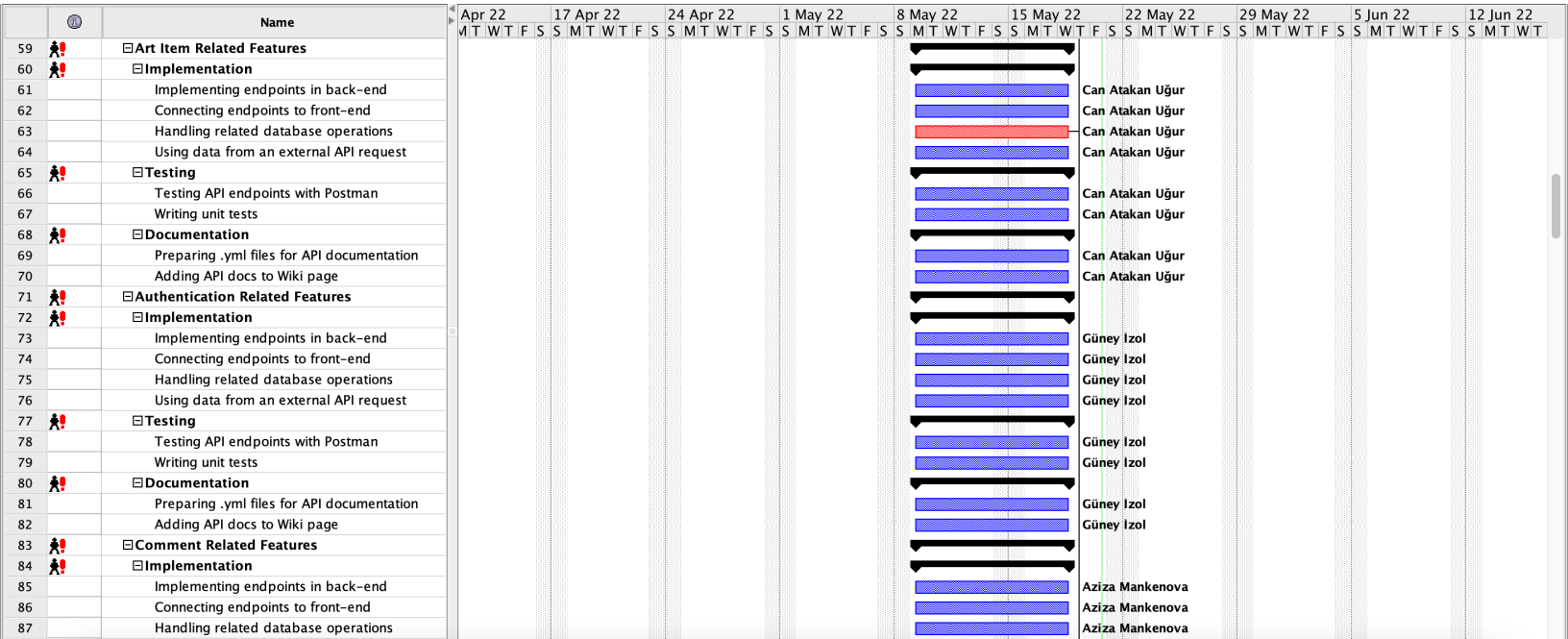
175		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
176		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
177		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
178		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
179		**Wikipedia Page Related Features**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
180		**Implementation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
181		Implementing endpoints in back-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
182		Connecting endpoints to front-end	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
183		Handling related database operations	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
184		Using data from an external API request	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
185		**Testing**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
186		Testing API endpoints with Postman	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
187		Writing unit tests	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
188		**Documentation**	**8 days**	**5/9/22 8:00 AM**	**5/18/22 5:00 PM**		**Atilla Türkmen**
189		Preparing .yml files for API documentation	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
190		Adding API docs to Wiki page	8 days	5/9/22 8:00 AM	5/18/22 5:00 PM		Atilla Türkmen
191		Dockerizing the App in Own Environment	1 day?	5/19/22 8:00 AM	5/19/22 5:00 PM	58	All
192		Deploying the App in Own Environment	1 day?	5/19/22 8:00 AM	5/19/22 5:00 PM	58	All
193		Meeting #10	1 day	4/24/22 8:00 AM	4/25/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa…
194		Meeting #11	1 day	4/30/22 8:00 AM	5/2/22 5:00 PM		Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Mehmet Em…
195		Meeting #12	1 day	5/6/22 8:00 AM	5/6/22 5:00 PM		Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Güney İzol;…
196		Meeting #13	1 day	5/11/22 8:00 AM	5/11/22 5:00 PM		Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Musa Şimşek
197		Meeting #14	1 day	5/15/22 8:00 AM	5/16/22 5:00 PM		Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Öz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğa…
198		Meeting #15	1 day?	5/18/22 8:00 AM	5/18/22 5:00 PM		
199		**Milestone 2 Report & Deliverables**	**2 days?**	**5/19/22 8:00 AM**	**5/20/22 5:00 PM**	**63**	
200		**Milestone 2 Report**	**2 days?**	**5/19/22 8:00 AM**	**5/20/22 5:00 PM**		
201		Executive Summary	2 days?	5/19/22 8:00 AM	5/20/22 5:00 PM		Demet Yayla
202		Project Plan	2 days?	5/19/22 8:00 AM	5/20/22 5:00 PM		Can Atakan Uğur
203		Individual Work Table	2 days?	5/19/22 8:00 AM	5/20/22 5:00 PM		All

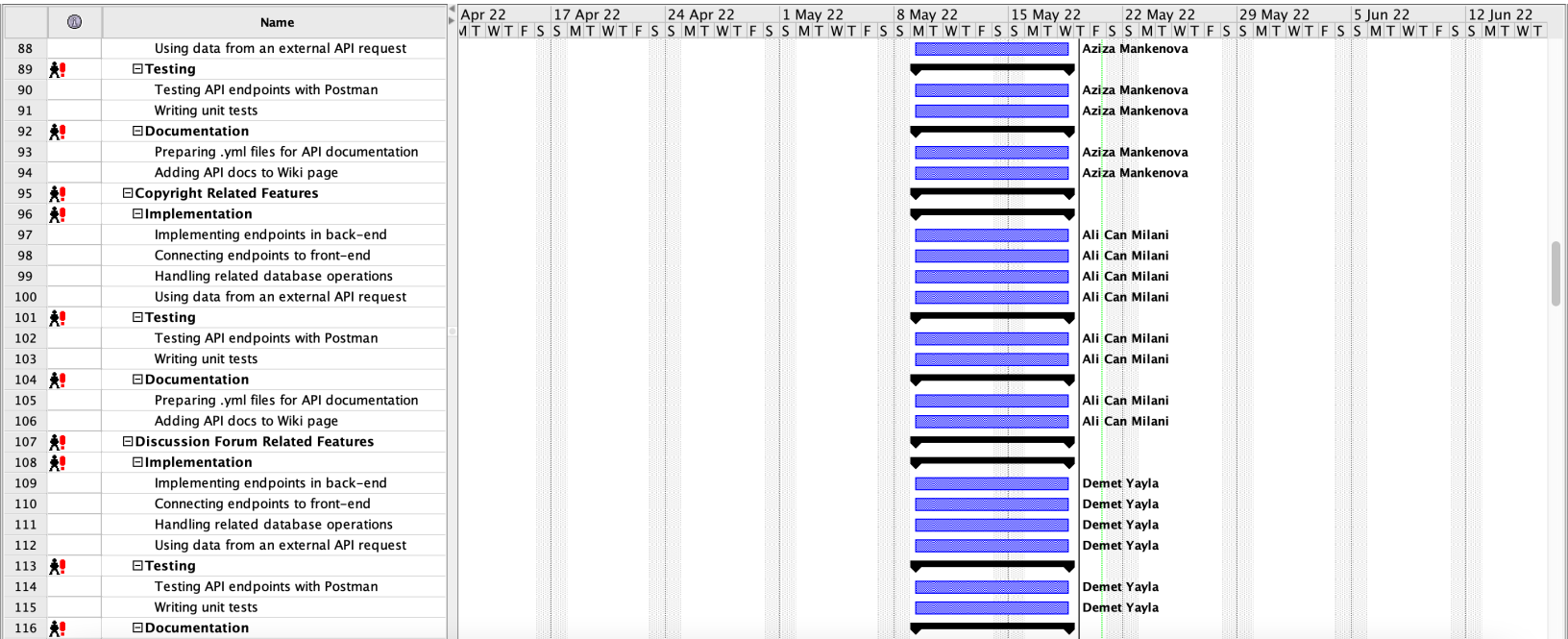
204		Evaluation of Tools and Processes	0.4 days?	5/19/22 8:00 AM	5/19/22 11:12 AM	Cahid Arda Öz;Musa Şimşek;Can Atakan Uğur;Erim Erkin Doğan;Güney İzol
205		Gathering Final Version of Requirements	2 days	5/19/22 8:00 AM	5/20/22 5:00 PM	Cahid Arda Öz;Aziza Mankenova
206		Gathering Final Versions of Software Diagrams	2 days	5/19/22 8:00 AM	5/20/22 5:00 PM	Demet Yayla;Can Atakan Uğur
207		**Preparing the Practice App Deliverable**	**2 days?**	**5/19/22 8:00 AM**	**5/20/22 5:00 PM**	**Cahid Arda Öz**
208		Dockerizing & deploying the final application	2 days?	5/19/22 8:00 AM	5/20/22 5:00 PM	Cahid Arda Öz
209		Adding a brief guide on running the application	2 days?	5/19/22 8:00 AM	5/20/22 5:00 PM	Cahid Arda Öz
210		Tagging the release in the repository	1 day?	5/19/22 8:00 AM	5/19/22 5:00 PM	Cahid Arda Öz;Erim Erkin Doğan

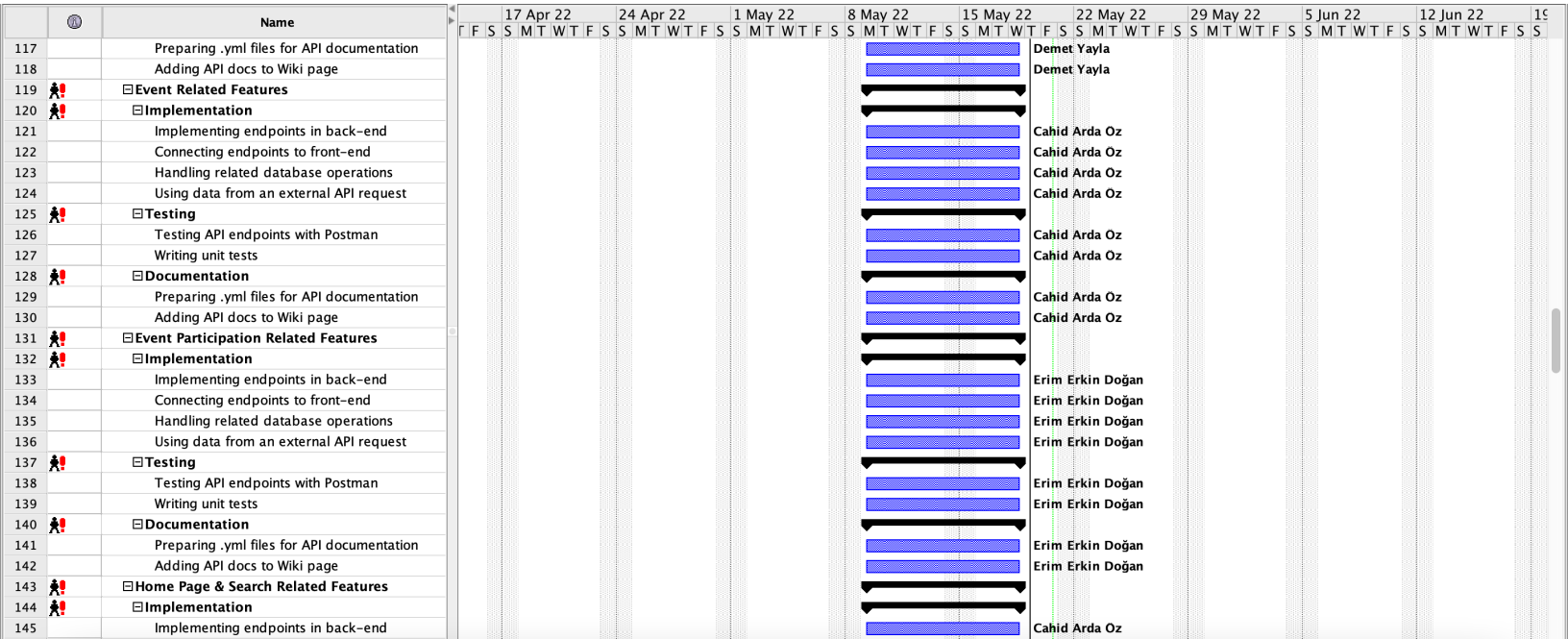

Name
27 Feb 22
6 Mar 22
13 Mar 22
20 Mar 22
27 Mar 22
3 Apr 22
10 Apr 22
17 Apr 22
24 Apr 22
1 May
1 Infrastructure Setup
2 Personal Wiki Page
3 Git research
4 Communication Plan
5 Customizing Issue Tags
6 Repository Search
7 Similar Project Research
8 Software Requirements Specification
9 User requirements
10 System requirements
11 Non-functional requirements
12 Description Analysis
13 Scenarios and Mockups
14 Scenario 1
15 Scenario 2
16 Scenario 3
17 Mock-up for Scenario 1
18 Mock-up for Scenario 2
19 Mock-up for Scenario 3
20 Post-customer Meeting Review
21 Software Design
22 Requirements Review
23 Use Case Diagram
24 Class Diagram
25 Sequence Diagrams
26 Planning
27 RAM
28 Project Planning
29 Milestone 1 Report
All
All
Demet Yayla;Atilla Türkmen;Aziza Mankenova;Güney Izol
Aziza Mankenova;Can Atakan Uğur;Güney Izol
All
All
All
All
Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Erim Erkin Doğan;Güney Izol;Mehmet Emin Ipekdal;Musa Şimşek
Aziza Mankenova;Can Atakan Uğur;Cahid Arda Oz;Demet Yayla
Demet Yayla;Erim Erkin Doğan;Güney Izol;Rafet Oğuz Pançuk
Ali Can Milani;Güney Izol;Musa Şimşek
Atilla Türkmen;Can Atakan Uğur
Aziza Mankenova;Erim Erkin Doğan;Mehmet Emin Ipekdal;Musa Şimşek
Cahid Arda Oz;Erim Erkin Doğan;Mehmet Emin Ipekdal;Musa Şimşek
Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Mehmet Emin Ipekdal;Musa Şimşek
Cahid Arda Oz;Can Atakan Uğur
All
All
All
All
All
All
Aziza Mankenova;Cahid Arda Oz;Can Atakan U
All

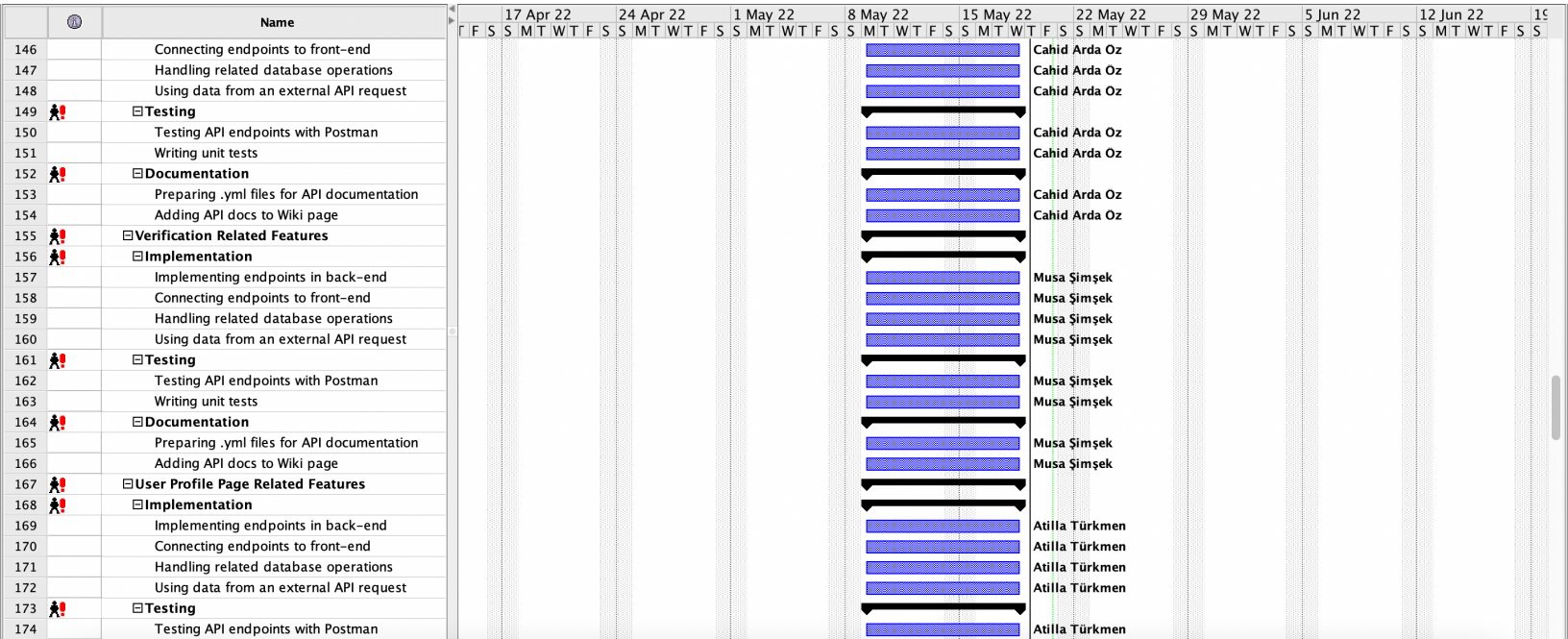
		Name
30		**Practice Application**
31		**Initial Planning**
32		Research on different frameworks
33		Deciding which API functions to implement
34		Studying APIs
35		**Research**
36		Pull Request and Review Tools Research
37		Research on Flask Framework
38		Docker Research
39		Research on testing API using Postman
40		Unittest Research
41		AWS Research
42		**Software Design for Practice App**
43		Requirements
44		Use Case Diagram
45		Class Diagram
46		Sequence Diagrams
47		**Forming Initial Code Structure**
48		Creating the Fundamental Back-end Structure
49		Creating the Fundamental Database Structure
50		Creating the Fundamental Front-end Structure
51		**Preparing Additional Project Configurations**
52		Creating API documentation template on Wiki
53		Docker Configuration
54		AWS Deployment Configuration
55		Integrating Authentication
56		Integrating Swagger for API Documentation
57		Creating CI/CD Workflows
58		**Implementation with Testing & Documentation**

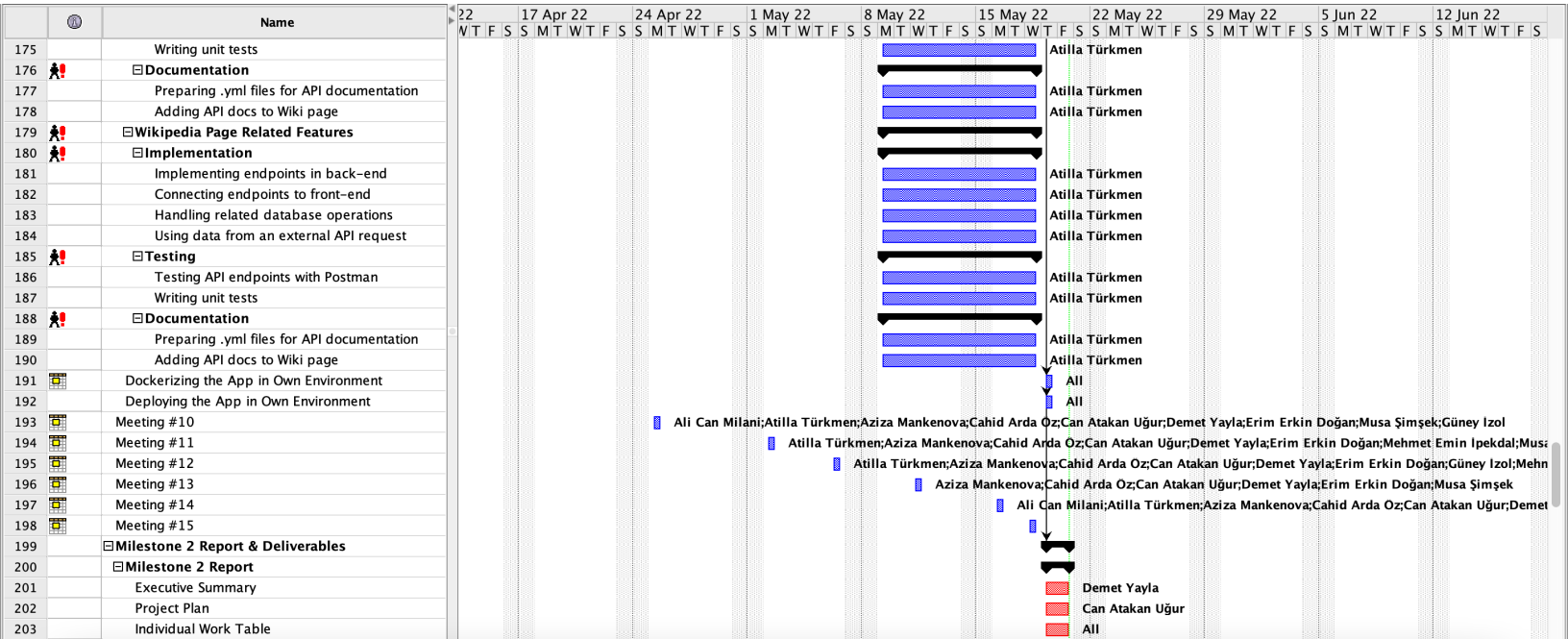



Name
Art Item Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Authentication Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Comment Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Apr 22
17 Apr 22
24 Apr 22
1 May 22
8 May 22
15 May 22
22 May 22
29 May 22
5 Jun 22
12 Jun 22
Can Atakan Uğur
Can Atakan Uğur
Can Atakan Uğur
Can Atakan Uğur
Can Atakan Uğur
Can Atakan Uğur
Can Atakan Uğur
Güney Izol
Güney Izol
Güney Izol
Güney Izol
Güney Izol
Güney Izol
Güney Izol
Güney Izol
Aziza Mankenova
Aziza Mankenova
Aziza Mankenova

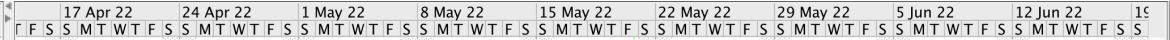

Name
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Copyright Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Discussion Forum Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Apr 22
17 Apr 22
24 Apr 22
1 May 22
8 May 22
15 May 22
22 May 22
29 May 22
5 Jun 22
12 Jun 22
Aziza Mankenova
Aziza Mankenova
Aziza Mankenova
Aziza Mankenova
Aziza Mankenova
Ali Can Milani
Ali Can Milani
Ali Can Milani
Ali Can Milani
Ali Can Milani
Ali Can Milani
Ali Can Milani
Ali Can Milani
Demet Yayla
Demet Yayla
Demet Yayla
Demet Yayla
Demet Yayla
Demet Yayla


Name
17 Apr 22
24 Apr 22
1 May 22
8 May 22
15 May 22
22 May 22
29 May 22
5 Jun 22
12 Jun 22
117 Preparing .yml files for API documentation
118 Adding API docs to Wiki page
119 Event Related Features
120 Implementation
121 Implementing endpoints in back-end
122 Connecting endpoints to front-end
123 Handling related database operations
124 Using data from an external API request
125 Testing
126 Testing API endpoints with Postman
127 Writing unit tests
128 Documentation
129 Preparing .yml files for API documentation
130 Adding API docs to Wiki page
131 Event Participation Related Features
132 Implementation
133 Implementing endpoints in back-end
134 Connecting endpoints to front-end
135 Handling related database operations
136 Using data from an external API request
137 Testing
138 Testing API endpoints with Postman
139 Writing unit tests
140 Documentation
141 Preparing .yml files for API documentation
142 Adding API docs to Wiki page
143 Home Page & Search Related Features
144 Implementation
145 Implementing endpoints in back-end
Demet Yayla
Demet Yayla
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Erim Erkin Doğan
Cahid Arda Oz


Name
17 Apr 22
24 Apr 22
1 May 22
8 May 22
15 May 22
22 May 22
29 May 22
5 Jun 22
12 Jun 22
146 Connecting endpoints to front-end
147 Handling related database operations
148 Using data from an external API request
149 Testing
150 Testing API endpoints with Postman
151 Writing unit tests
152 Documentation
153 Preparing .yml files for API documentation
154 Adding API docs to Wiki page
155 Verification Related Features
156 Implementation
157 Implementing endpoints in back-end
158 Connecting endpoints to front-end
159 Handling related database operations
160 Using data from an external API request
161 Testing
162 Testing API endpoints with Postman
163 Writing unit tests
164 Documentation
165 Preparing .yml files for API documentation
166 Adding API docs to Wiki page
167 User Profile Page Related Features
168 Implementation
169 Implementing endpoints in back-end
170 Connecting endpoints to front-end
171 Handling related database operations
172 Using data from an external API request
173 Testing
174 Testing API endpoints with Postman
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Cahid Arda Oz
Musa Şimşek
Musa Şimşek
Musa Şimşek
Musa Şimşek
Musa Şimşek
Musa Şimşek
Musa Şimşek
Musa Şimşek
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen


Name
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Wikipedia Page Related Features
Implementation
Implementing endpoints in back-end
Connecting endpoints to front-end
Handling related database operations
Using data from an external API request
Testing
Testing API endpoints with Postman
Writing unit tests
Documentation
Preparing .yml files for API documentation
Adding API docs to Wiki page
Dockerizing the App in Own Environment
Deploying the App in Own Environment
Meeting #10
Meeting #11
Meeting #12
Meeting #13
Meeting #14
Meeting #15
Milestone 2 Report & Deliverables
Milestone 2 Report
Executive Summary
Project Plan
Individual Work Table
22
17 Apr 22
24 Apr 22
1 May 22
8 May 22
15 May 22
22 May 22
29 May 22
5 Jun 22
12 Jun 22
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
Atilla Türkmen
All
All
Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Musa Şimşek;Güney Izol
Atilla Türkmen;Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Mehmet Emin Ipekdal;Musa
Atilla Türkmen;Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Güney Izol;Mehn
Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Demet Yayla;Erim Erkin Doğan;Musa Şimşek
Ali Can Milani;Atilla Türkmen;Aziza Mankenova;Cahid Arda Oz;Can Atakan Uğur;Demet
Demet Yayla
Can Atakan Uğur
All

		Name
204		Evaluation of Tools and Processes
205		Gathering Final Version of Requirements
206		Gathering Final Versions of Software Diagrams
207		**Preparing the Practice App Deliverable**
208		Dockerizing & deploying the final application
209		Adding a brief guide on running the application
210		Tagging the release in the repository



Project Plan:
Second Semester

		Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Project Initialization**	5 days?	**10/17/22 8:00 AM**	**10/21/22 5:00 PM**		
2		**Frontend**	5 days?	**10/17/22 8:00 AM**	**10/21/22 5:00 PM**		
3		Frontend Meeting #1	0 days	10/17/22 8:00 AM	10/17/22 8:00 AM		Frontend Team
4		Project Initialization	5 days?	10/17/22 8:00 AM	10/21/22 5:00 PM		Frontend Team
5		**Mobile**	5 days?	**10/17/22 8:00 AM**	**10/21/22 5:00 PM**		
6		Mobile Meeting #1	0 days	10/17/22 8:00 AM	10/17/22 8:00 AM		Mobile Team
7		Project Initialization	5 days?	10/17/22 8:00 AM	10/21/22 5:00 PM		Mobile Team
8		**Backend**	5 days?	**10/17/22 8:00 AM**	**10/21/22 5:00 PM**		
9		Backend Meeting #1	0 days	10/17/22 8:00 AM	10/17/22 8:00 AM		Backend Team
10		Project Initialization	5 days?	10/17/22 8:00 AM	10/21/22 5:00 PM		Backend Team
11		**Implement: Authorization**	5 days?	**10/24/22 8:00 AM**	**10/28/22 5:00 PM**	1	
12		Endpoint Decision: Authorization	1 day	10/24/22 8:00 AM	10/24/22 5:00 PM		
13		**Frontend**	4 days?	**10/24/22 5:00 PM**	**10/28/22 5:00 PM**	12	
14		Frontend Meeting #2	0 days	10/24/22 5:00 PM	10/24/22 5:00 PM		Frontend Team
15		Implement Register	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Frontend Team
16		Implement Sign-in	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Frontend Team
17		Implement Password Reset	4 days	10/25/22 8:00 AM	10/28/22 5:00 PM		Frontend Team
18		**Mobile**	4 days?	**10/24/22 5:00 PM**	**10/28/22 5:00 PM**	12	
19		Mobile Meeting #2	0 days	10/24/22 5:00 PM	10/24/22 5:00 PM		Mobile Team
20		Implement Register	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Mobile Team
21		Implement Sign-in	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Mobile Team
22		Implement Password Reset	4 days	10/25/22 8:00 AM	10/28/22 5:00 PM		Mobile Team
23		Implement Sign-in/Register with Google Account	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Mobile Team
24		**Backend**	4 days?	**10/24/22 5:00 PM**	**10/28/22 5:00 PM**	12	
25		Backend Meeting #2	0 days	10/24/22 5:00 PM	10/24/22 5:00 PM		Backend Team
26		Implement Register	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Backend Team
27		Implement Sign-in	4 days?	10/25/22 8:00 AM	10/28/22 5:00 PM		Backend Team
28		Implement Password Reset	4 days	10/25/22 8:00 AM	10/28/22 5:00 PM		Backend Team
29		**Implement: Profile Page**	5 days?	**10/31/22 8:00 AM**	**11/4/22 5:00 PM**	11	

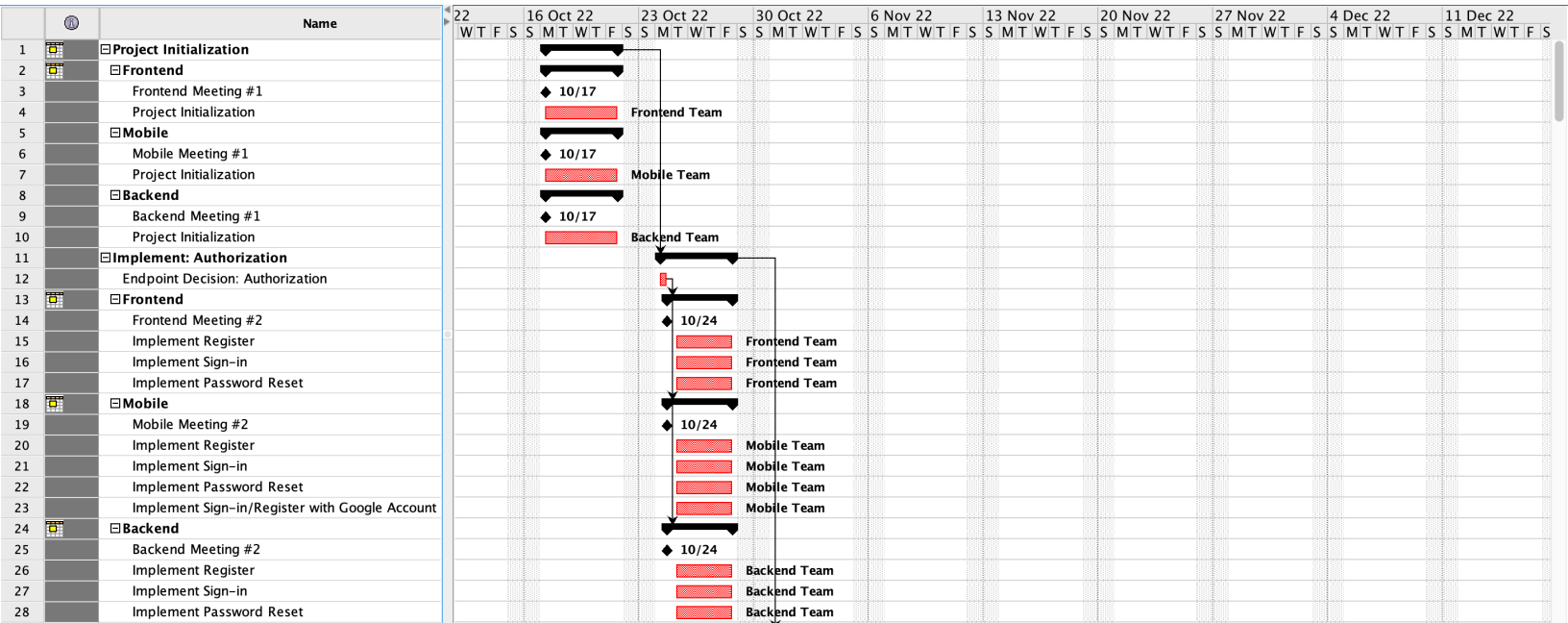
30		Endpoint Decision: Profile Page	1 day	10/31/22 8:00 AM	10/31/22 5:00 PM		
31		**Frontend**	4 days?	**10/31/22 5:00 PM**	**11/4/22 5:00 PM**	**30**	
32		Frontend Meeting #3	0 days	10/31/22 5:00 PM	10/31/22 5:00 PM		Frontend Team
33		Implement Profile Page	4 days?	11/1/22 8:00 AM	11/4/22 5:00 PM		Frontend Team
34		**Mobile**	4 days?	**10/31/22 5:00 PM**	**11/4/22 5:00 PM**	**30**	
35		Mobile Meeting #3	0 days	10/31/22 5:00 PM	10/31/22 5:00 PM		Mobile Team
36		Implement Profile Page	4 days?	11/1/22 8:00 AM	11/4/22 5:00 PM		Mobile Team
37		**Backend**	4 days?	**10/31/22 5:00 PM**	**11/4/22 5:00 PM**	**30**	
38		Backend Meeting #3	0 days	10/31/22 5:00 PM	10/31/22 5:00 PM		Backend Team
39		Implement Profile Page	4 days?	11/1/22 8:00 AM	11/4/22 5:00 PM		Backend Team
40		**Implement: Settings and Notifications**	5 days?	**11/7/22 8:00 AM**	**11/11/22 5:00 PM**	**29**	
41		Endpoint Decision: Settings, GDPR requirements and notifications	1 day	11/7/22 8:00 AM	11/7/22 5:00 PM		
42		**Frontend**	4 days?	**11/7/22 5:00 PM**	**11/11/22 5:00 PM**	**41**	
43		Frontend Meeting #4	0 days	11/7/22 5:00 PM	11/7/22 5:00 PM		Frontend Team
44		Implement Profile Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Frontend Team
45		Implement Account Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Frontend Team
46		Implement Notifications	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Frontend Team
47		**Mobile**	4 days?	**11/7/22 5:00 PM**	**11/11/22 5:00 PM**	**41**	
48		Mobile Meeting #4	0 days	11/7/22 5:00 PM	11/7/22 5:00 PM		Mobile Team
49		Implement Profile Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Mobile Team
50		Implement Account Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Mobile Team
51		Implement Notifications	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Mobile Team
52		**Backend**	4 days?	**11/7/22 5:00 PM**	**11/11/22 5:00 PM**	**41**	
53		Backend Meeting #4	0 days	11/7/22 5:00 PM	11/7/22 5:00 PM		Backend Team
54		Implement Profile Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Backend Team
55		Implement Account Settings	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Backend Team
56		Implement Notifications	4 days?	11/8/22 8:00 AM	11/11/22 5:00 PM		Backend Team
57		**Implement: Home Page**	5 days?	**11/14/22 8:00 AM**	**11/18/22 5:00 PM**	**40**	

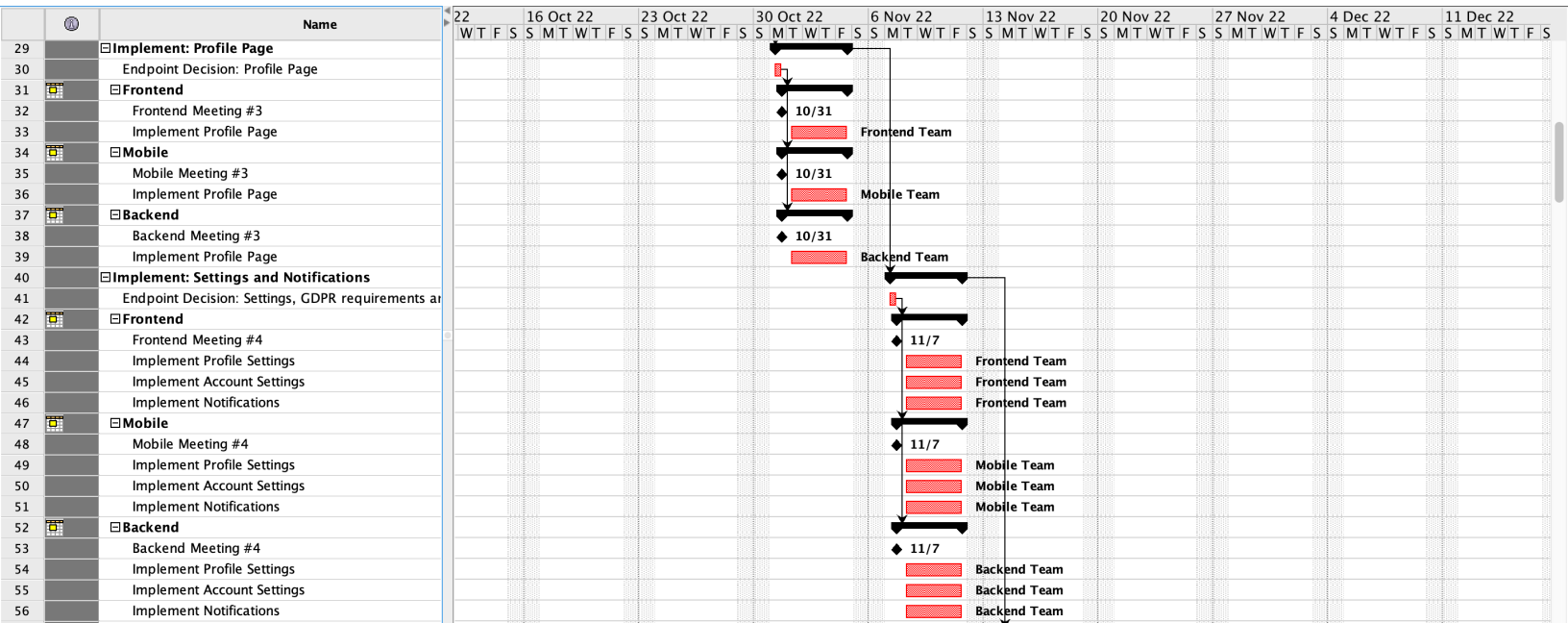
58		Endpoint Decision: Home page and recommendations	1 day	11/14/22 8:00 AM	11/14/22 5:00 PM		
59		**Frontend**	4 days?	**11/14/22 5:00 PM**	**11/18/22 5:00 PM**	**58**	
60		Frontend Meeting #5	0 days	11/14/22 5:00 PM	11/14/22 5:00 PM		Frontend Team
61		Implement Home Page	4 days?	11/15/22 8:00 AM	11/18/22 5:00 PM		Frontend Team
62		**Mobile**	4 days?	**11/14/22 5:00 PM**	**11/18/22 5:00 PM**	**58**	
63		Mobile Meeting #4	0 days	11/14/22 5:00 PM	11/14/22 5:00 PM		Mobile Team
64		Implement Home Page	4 days?	11/15/22 8:00 AM	11/18/22 5:00 PM		Mobile Team
65		**Backend**	4 days?	**11/14/22 5:00 PM**	**11/18/22 5:00 PM**	**58**	
66		Backend Meeting #4	0 days	11/14/22 5:00 PM	11/14/22 5:00 PM		Backend Team
67		Implement Recommendations	4 days?	11/15/22 8:00 AM	11/18/22 5:00 PM		Backend Team
68		Implement Home Page	4 days?	11/15/22 8:00 AM	11/18/22 5:00 PM		Backend Team
69		**Implement: Discussion Forum**	5 days?	**11/21/22 8:00 AM**	**11/25/22 5:00 PM**	**57**	
70		Endpoint Decision: Discussion Forum	1 day	11/21/22 8:00 AM	11/21/22 5:00 PM		
71		**Frontend**	4 days?	**11/21/22 5:00 PM**	**11/25/22 5:00 PM**	**70**	
72		Frontend Meeting #6	0 days	11/21/22 5:00 PM	11/21/22 5:00 PM		Frontend Team
73		Implement Discussion Forum	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Frontend Team
74		Implement Discussion Annotations	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Frontend Team
75		**Mobile**	4 days?	**11/21/22 5:00 PM**	**11/25/22 5:00 PM**	**70**	
76		Mobile Meeting #6	0 days	11/21/22 5:00 PM	11/21/22 5:00 PM		Mobile Team
77		Implement Discussion Forum	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Mobile Team
78		Implement Discussion Annotations	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Mobile Team
79		**Backend**	4 days?	**11/21/22 5:00 PM**	**11/25/22 5:00 PM**	**70**	
80		Backend Meeting #6	0 days	11/21/22 5:00 PM	11/21/22 5:00 PM		Backend Team
81		Implement Discussion Forum	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Backend Team
82		Implement Discussion Annotations	4 days?	11/22/22 8:00 AM	11/25/22 5:00 PM		Backend Team
83		**Implement: Art Item Posts and Replies**	5 days?	**11/28/22 8:00 AM**	**12/2/22 5:00 PM**	**69**	
84		Endpoint Decision: Art Item Posts and Replies	1 day	11/28/22 8:00 AM	11/28/22 5:00 PM		
85		**Frontend**	4 days?	**11/28/22 5:00 PM**	**12/2/22 5:00 PM**	**84**	

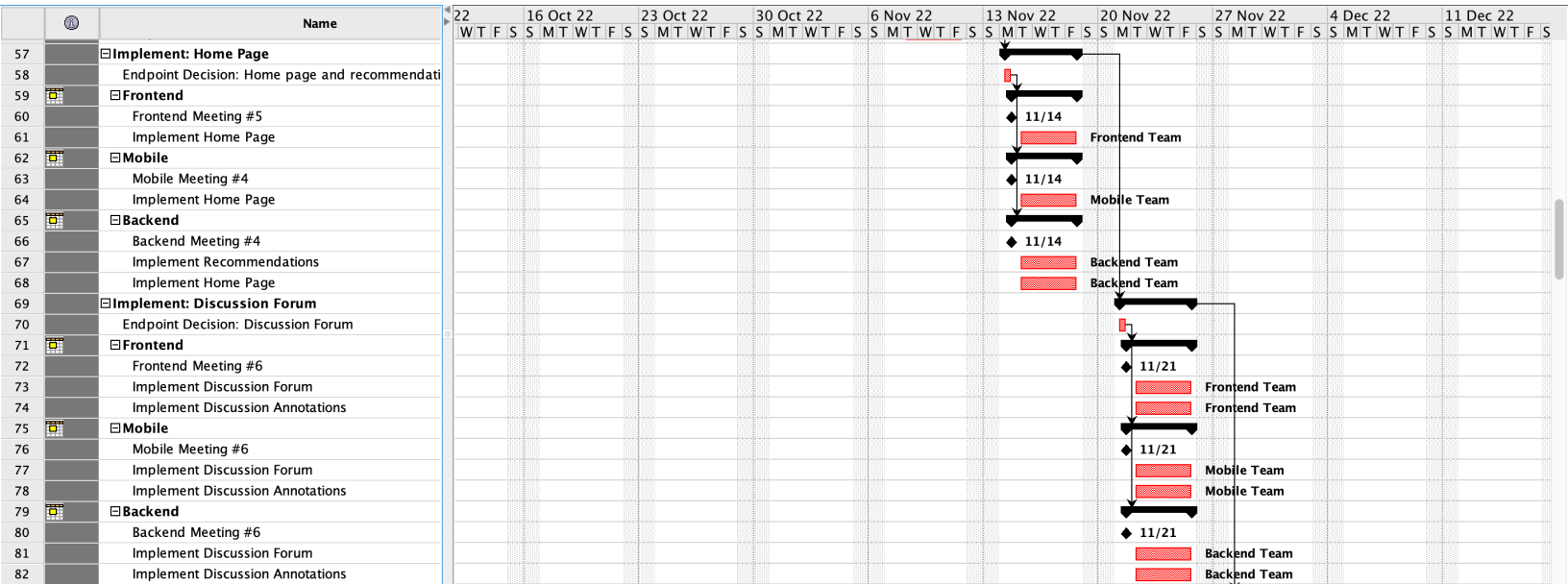
86		Frontend Meeting #7	0 days	11/28/22 5:00 PM	11/28/22 5:00 PM		Frontend Team
87		Implement Art Item Posts	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Frontend Team
88		Implement Replies	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Frontend Team
89		Implement Art Item Annotations	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Frontend Team
90		**Mobile**	4 days?	**11/28/22 5:00 PM**	**12/2/22 5:00 PM**	**84**	
91		Mobile Meeting #7	0 days	11/28/22 5:00 PM	11/28/22 5:00 PM		Mobile Team
92		Implement Art Item Posts	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Mobile Team
93		Implement Replies	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Mobile Team
94		Implement Art Item Annotations	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Mobile Team
95		**Backend**	4 days?	**11/28/22 5:00 PM**	**12/2/22 5:00 PM**	**84**	
96		Backend Meeting #7	0 days	11/28/22 5:00 PM	11/28/22 5:00 PM		Backend Team
97		Implement Art Item Posts	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Backend Team
98		Implement Replies	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Backend Team
99		Implement Art Item Annotations	4 days?	11/29/22 8:00 AM	12/2/22 5:00 PM		Backend Team
100		**Implement: Search & Filtering**	5 days?	**12/5/22 8:00 AM**	**12/9/22 5:00 PM**	**83**	
101		Endpoint Decision: Search & Filtering	1 day	12/5/22 8:00 AM	12/5/22 5:00 PM		
102		**Frontend**	4 days?	**12/5/22 5:00 PM**	**12/9/22 5:00 PM**	**101**	
103		Frontend Meeting #8	0 days	12/5/22 5:00 PM	12/5/22 5:00 PM		Frontend Team
104		Implement Search	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Frontend Team
105		Implement Filtering	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Frontend Team
106		**Mobile**	4 days?	**12/5/22 5:00 PM**	**12/9/22 5:00 PM**	**101**	
107		Mobile Meeting #8	0 days	12/5/22 5:00 PM	12/5/22 5:00 PM		Mobile Team
108		Implement Search	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Mobile Team
109		Implement Filtering	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Mobile Team
110		**Backend**	4 days?	**12/5/22 5:00 PM**	**12/9/22 5:00 PM**	**101**	
111		Backend Meeting #8	0 days	12/5/22 5:00 PM	12/5/22 5:00 PM		Backend Team
112		Implement Search	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Backend Team
113		Implement Filtering	4 days?	12/6/22 8:00 AM	12/9/22 5:00 PM		Backend Team

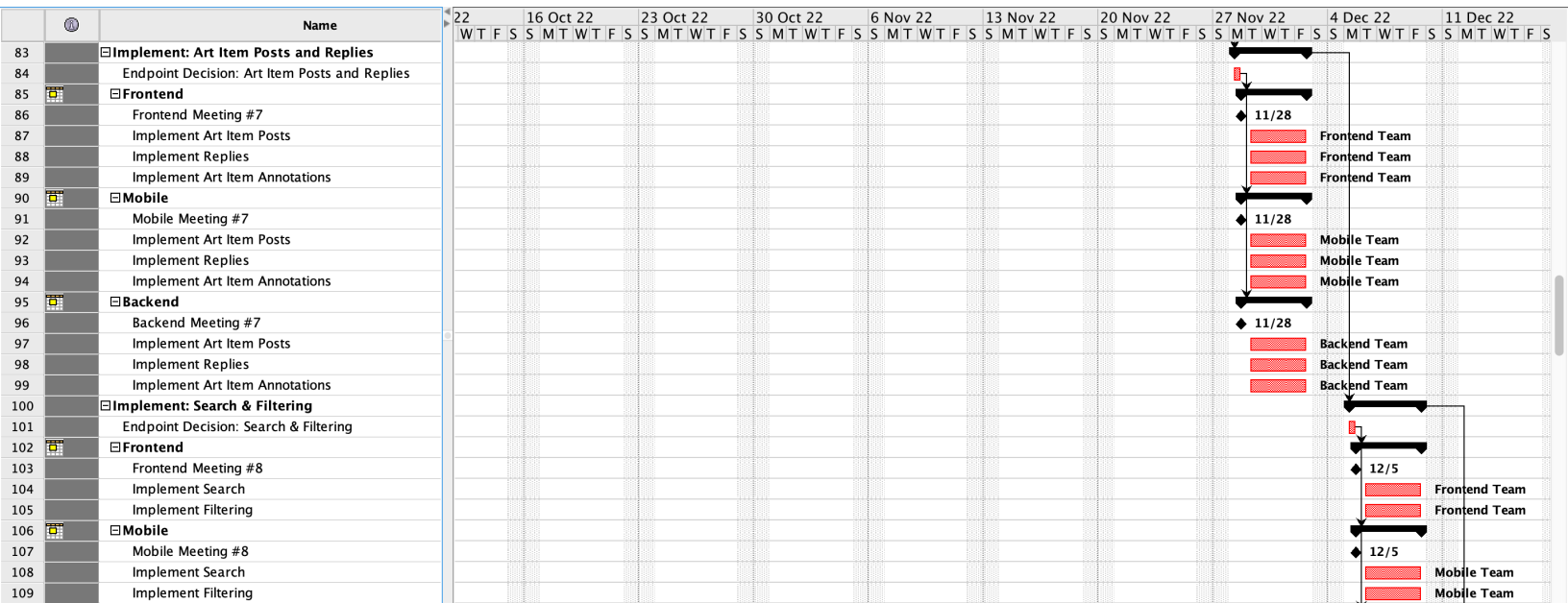
114		**Implement: Events**	5 days?	**12/12/22 8:00 AM**	**12/16/22 5:00 PM**	**100**	
115		Endpoint Decision: Exhibitions & Online Galleries	1 day	12/12/22 8:00 AM	12/12/22 5:00 PM		
116		**Frontend**	4 days?	**12/12/22 5:00 PM**	**12/16/22 5:00 PM**	**115**	
117		Frontend Meeting #9	0 days	12/12/22 5:00 PM	12/12/22 5:00 PM		Frontend Team
118		Implement Exhibitions	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Frontend Team
119		Implement Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Frontend Team
120		Implement Annotations for Exhibitions & Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Frontend Team
121		**Mobile**	4 days?	**12/12/22 5:00 PM**	**12/16/22 5:00 PM**	**115**	
122		Mobile Meeting #9	0 days	12/12/22 5:00 PM	12/12/22 5:00 PM		Mobile Team
123		Implement Exhibitions	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Mobile Team
124		Implement Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Mobile Team
125		Implement Annotations for Exhibitions & Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Mobile Team
126		**Backend**	4 days?	**12/12/22 5:00 PM**	**12/16/22 5:00 PM**	**115**	
127		Backend Meeting #9	0 days	12/12/22 5:00 PM	12/12/22 5:00 PM		Backend Team
128		Implement Exhibitions	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Backend Team
129		Implement Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Backend Team
130		Implement Annotations for Exhibitions & Online Galleries	4 days?	12/13/22 8:00 AM	12/16/22 5:00 PM		Backend Team
131		**Implement: Auction & Bidding**	5 days?	**12/19/22 8:00 AM**	**12/23/22 5:00 PM**	**114**	
132		Endpoint Decision: Auction & Bidding	1 day	12/19/22 8:00 AM	12/19/22 5:00 PM		
133		**Frontend**	4 days?	**12/19/22 5:00 PM**	**12/23/22 5:00 PM**	**132**	
134		Frontend Meeting #10	0 days	12/19/22 5:00 PM	12/19/22 5:00 PM		Frontend Team
135		Implement Auctions	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Frontend Team
136		Implement Bidding	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Frontend Team
137		**Mobile**	4 days?	**12/19/22 5:00 PM**	**12/23/22 5:00 PM**	**132**	
138		Mobile Meeting #10	0 days	12/19/22 5:00 PM	12/19/22 5:00 PM		Mobile Team
139		Implement Auctions	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Mobile Team
140		Implement Bidding	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Mobile Team
141		**Backend**	4 days?	**12/19/22 5:00 PM**	**12/23/22 5:00 PM**	**132**	
142		Backend Meeting #10	0 days	12/19/22 5:00 PM	12/19/22 5:00 PM		Backend Team

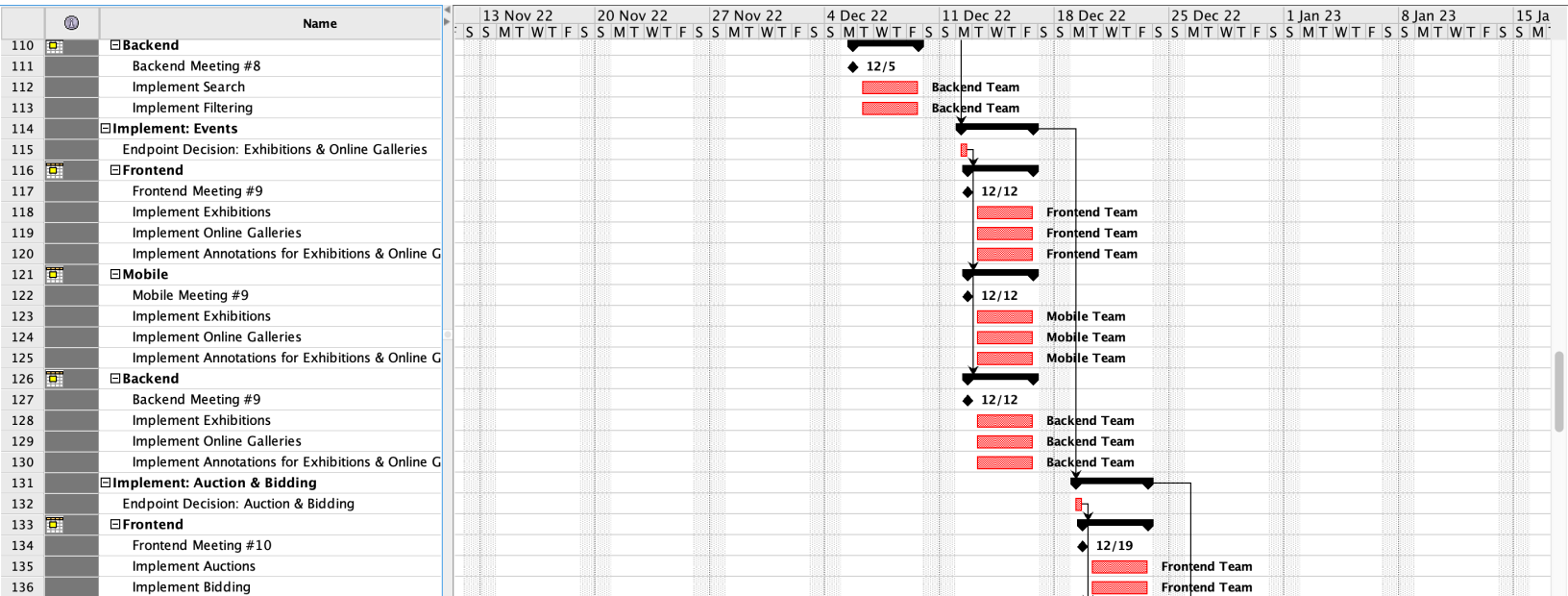
143		Implement Auctions	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Backend Team
144		Implement Bidding	4 days?	12/20/22 8:00 AM	12/23/22 5:00 PM		Backend Team
145		**Implement: Platform Management**	5 days?	**12/26/22 8:00 AM**	**12/30/22 5:00 PM**	**131**	
146		Endpoint Decision: Admin functions, reporting and verification	1 day	12/26/22 8:00 AM	12/26/22 5:00 PM		
147		**Frontend**	4 days?	**12/26/22 5:00 PM**	**12/30/22 5:00 PM**	**146**	
148		Frontend Meeting #11	0 days	12/26/22 5:00 PM	12/26/22 5:00 PM		Frontend Team
149		Implement Admin User Functions	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Frontend Team
150		Implement Copyright Reporting System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Frontend Team
151		Implement Verification System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Frontend Team
152		**Mobile**	4 days?	**12/26/22 5:00 PM**	**12/30/22 5:00 PM**	**146**	
153		Mobile Meeting #11	0 days	12/26/22 5:00 PM	12/26/22 5:00 PM		Mobile Team
154		Implement Admin User Functions	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Mobile Team
155		Implement Copyright Reporting System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Mobile Team
156		Implement Verification System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Mobile Team
157		**Backend**	4 days?	**12/26/22 5:00 PM**	**12/30/22 5:00 PM**	**146**	
158		Backend Meeting #11	0 days	12/26/22 5:00 PM	12/26/22 5:00 PM		Backend Team
159		Implement Admin User Functions	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Backend Team
160		Implement Copyright Reporting System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Backend Team
161		Implement Verification System	4 days?	12/27/22 8:00 AM	12/30/22 5:00 PM		Backend Team
162		Unit Testing	2 days	1/2/23 8:00 AM	1/3/23 5:00 PM	145	All
163		Dockerization & Deployment	1 day	1/4/23 8:00 AM	1/4/23 5:00 PM	162	All

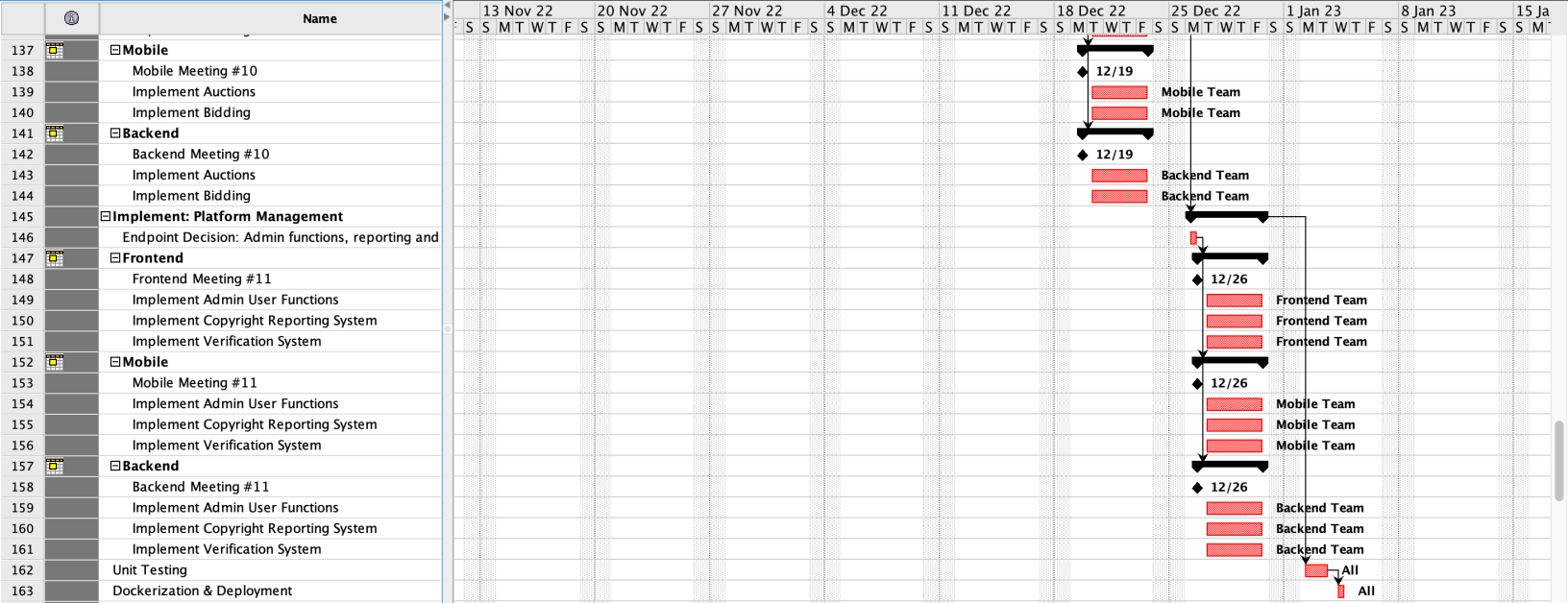

Name
1 Project Initialization
2 Frontend
3 Frontend Meeting #1
4 Project Initialization
5 Mobile
6 Mobile Meeting #1
7 Project Initialization
8 Backend
9 Backend Meeting #1
10 Project Initialization
11 Implement: Authorization
12 Endpoint Decision: Authorization
13 Frontend
14 Frontend Meeting #2
15 Implement Register
16 Implement Sign-in
17 Implement Password Reset
18 Mobile
19 Mobile Meeting #2
20 Implement Register
21 Implement Sign-in
22 Implement Password Reset
23 Implement Sign-in/Register with Google Account
24 Backend
25 Backend Meeting #2
26 Implement Register
27 Implement Sign-in
28 Implement Password Reset
22
16 Oct 22
23 Oct 22
30 Oct 22
6 Nov 22
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
10/17
Frontend Team
10/17
Mobile Team
10/17
Backend Team
10/24
Frontend Team
Frontend Team
Frontend Team
10/24
Mobile Team
Mobile Team
Mobile Team
Mobile Team
10/24
Backend Team
Backend Team
Backend Team


Name
Implement: Profile Page
Endpoint Decision: Profile Page
Frontend
Frontend Meeting #3
Implement Profile Page
Mobile
Mobile Meeting #3
Implement Profile Page
Backend
Backend Meeting #3
Implement Profile Page
Implement: Settings and Notifications
Endpoint Decision: Settings, GDPR requirements ar
Frontend
Frontend Meeting #4
Implement Profile Settings
Implement Account Settings
Implement Notifications
Mobile
Mobile Meeting #4
Implement Profile Settings
Implement Account Settings
Implement Notifications
Backend
Backend Meeting #4
Implement Profile Settings
Implement Account Settings
Implement Notifications
22
16 Oct 22
23 Oct 22
30 Oct 22
6 Nov 22
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
10/31
Frontend Team
10/31
Mobile Team
10/31
Backend Team
11/7
Frontend Team
Frontend Team
Frontend Team
11/7
Mobile Team
Mobile Team
Mobile Team
11/7
Backend Team
Backend Team
Backend Team


Name
22
16 Oct 22
23 Oct 22
30 Oct 22
6 Nov 22
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
57 Implement: Home Page
58 Endpoint Decision: Home page and recommendati
59 Frontend
60 Frontend Meeting #5
61 Implement Home Page
62 Mobile
63 Mobile Meeting #4
64 Implement Home Page
65 Backend
66 Backend Meeting #4
67 Implement Recommendations
68 Implement Home Page
69 Implement: Discussion Forum
70 Endpoint Decision: Discussion Forum
71 Frontend
72 Frontend Meeting #6
73 Implement Discussion Forum
74 Implement Discussion Annotations
75 Mobile
76 Mobile Meeting #6
77 Implement Discussion Forum
78 Implement Discussion Annotations
79 Backend
80 Backend Meeting #6
81 Implement Discussion Forum
82 Implement Discussion Annotations
11/14
Frontend Team
11/14
Mobile Team
11/14
Backend Team
Backend Team
11/21
Frontend Team
Frontend Team
11/21
Mobile Team
Mobile Team
11/21
Backend Team
Backend Team


Name
22
16 Oct 22
23 Oct 22
30 Oct 22
6 Nov 22
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
83 Implement: Art Item Posts and Replies
84 Endpoint Decision: Art Item Posts and Replies
85 Frontend
86 Frontend Meeting #7
87 Implement Art Item Posts
88 Implement Replies
89 Implement Art Item Annotations
90 Mobile
91 Mobile Meeting #7
92 Implement Art Item Posts
93 Implement Replies
94 Implement Art Item Annotations
95 Backend
96 Backend Meeting #7
97 Implement Art Item Posts
98 Implement Replies
99 Implement Art Item Annotations
100 Implement: Search & Filtering
101 Endpoint Decision: Search & Filtering
102 Frontend
103 Frontend Meeting #8
104 Implement Search
105 Implement Filtering
106 Mobile
107 Mobile Meeting #8
108 Implement Search
109 Implement Filtering
11/28
Frontend Team
Frontend Team
Frontend Team
11/28
Mobile Team
Mobile Team
Mobile Team
11/28
Backend Team
Backend Team
Backend Team
12/5
Frontend Team
Frontend Team
12/5
Mobile Team
Mobile Team


Name
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
18 Dec 22
25 Dec 22
1 Jan 23
8 Jan 23
15 Ja
S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
110 Backend
111 Backend Meeting #8
112 Implement Search
113 Implement Filtering
114 Implement: Events
115 Endpoint Decision: Exhibitions & Online Galleries
116 Frontend
117 Frontend Meeting #9
118 Implement Exhibitions
119 Implement Online Galleries
120 Implement Annotations for Exhibitions & Online G
121 Mobile
122 Mobile Meeting #9
123 Implement Exhibitions
124 Implement Online Galleries
125 Implement Annotations for Exhibitions & Online G
126 Backend
127 Backend Meeting #9
128 Implement Exhibitions
129 Implement Online Galleries
130 Implement Annotations for Exhibitions & Online G
131 Implement: Auction & Bidding
132 Endpoint Decision: Auction & Bidding
133 Frontend
134 Frontend Meeting #10
135 Implement Auctions
136 Implement Bidding
12/5
Backend Team
Backend Team
12/12
Frontend Team
Frontend Team
Frontend Team
12/12
Mobile Team
Mobile Team
Mobile Team
12/12
Backend Team
Backend Team
Backend Team
12/19
Frontend Team
Frontend Team


Name
13 Nov 22
20 Nov 22
27 Nov 22
4 Dec 22
11 Dec 22
18 Dec 22
25 Dec 22
1 Jan 23
8 Jan 23
15 Ja
137 Mobile
138 Mobile Meeting #10
139 Implement Auctions
140 Implement Bidding
141 Backend
142 Backend Meeting #10
143 Implement Auctions
144 Implement Bidding
145 Implement: Platform Management
146 Endpoint Decision: Admin functions, reporting and
147 Frontend
148 Frontend Meeting #11
149 Implement Admin User Functions
150 Implement Copyright Reporting System
151 Implement Verification System
152 Mobile
153 Mobile Meeting #11
154 Implement Admin User Functions
155 Implement Copyright Reporting System
156 Implement Verification System
157 Backend
158 Backend Meeting #11
159 Implement Admin User Functions
160 Implement Copyright Reporting System
161 Implement Verification System
162 Unit Testing
163 Dockerization & Deployment
12/19
Mobile Team
Mobile Team
12/19
Backend Team
Backend Team
12/26
Frontend Team
Frontend Team
Frontend Team
12/26
Mobile Team
Mobile Team
Mobile Team
12/26
Backend Team
Backend Team
Backend Team
All
All

API Documentation

Practice App API

Overview

API for Practice Application of Group 7. A collaborative art platform, ArtShare.

Version information

Version : 1.0

License information

Terms of service : /tos

Security

BearerAuth

Type : apiKey
Name : Authorization
In : HEADER

Paths

Endpoint to view 5 available art galleries

```
GET /api/art_galleries
```

Responses

HTTP Code	Description	Schema
200	The sign up was successful.	< Response 200 > array

Response 200

Name	Description	Schema
address *optional*	Address of the art gallery **Example** : `"Suite 867 8602 Schroeder Point, East Guillermo, NE 00680-8089"`	string
name *optional*	Name of the art gallery **Example** : `"West Loriaview Art Gallery"`	string

Endpoint to create an art item object

```
POST /api/art_item
```

Parameters

Type	Name	Description	Schema
Body	**content_uri** *required*	The link to the resource file of the art item (image, gif, etc.)	string
Body	**description** *required*	The informative text about the art item	string
Body	**name** *required*	Name of the art item	string

Responses

HTTP Code	Description	Schema
201	Art item with the given data is successfully created.	Response 201
400	Missing required parameters.	No Content
401	Access token is missing or invalid.	No Content
409	Duplicate art item tried to be created.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created art item **Example** : 37	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch an art item with a given ID

```
GET /api/art_item/{art_item_id}
```

Parameters

Type	Name	Description	Schema
Path	**art_item_id** *required*	ID of the art item	integer

Responses

HTTP Code	Description	Schema
200	Art item with the given ID is successfully fetched.	Response 200
404	There are no art items with the given ID.	No Content

Response 200

Name	Description	Schema
content_uri *required*	The link to the resource file of the art item (image, gif, etc.) **Example** : "https://en.wikipedia.org/wiki/The_Starry_Night#/media/File:Van_Gogh_-Starry_Night-_Google_Art_Project.jpg"	string
creation_date *required*	The date when the art item is created **Example** : "2022-05-17"	string
creator_artist *required*	User ID of the creator of the art item **Example** : 29	integer

Name	Description	Schema
description *required*	The informative text about the art item **Example** : `"The Starry Night is an oil-on-canvas painting by the Dutch Post-Impressionist painter Vincent van Gogh."`	string
dominant_colors *required*	List of the dominant colors in the art item's content. **Example** : `[ "orange crush", "graphite" ]`	< string > array
id *required*	ID of the art item **Example** : `37`	integer
name *required*	Name of the art item **Example** : `"The Starry Night"`	string

Endpoint to create an copyright report object

```
POST /api/copyright
```

Parameters

Type	Name	Description	Schema
Body	**description** *required*	The informative text about the report	string
Body	**infringement_art_item_id** *required*	ID of the infringement art item	string
Body	**original_art_item_id** *required*	ID of the original art item	string

Responses

HTTP Code	Description	Schema
201	Report with the given data is successfully created.	Response 201
400	Original art item and infringement art item must be different.	No Content

HTTP Code	Description	Schema
404	Art Item with id : <art_item_id> does not exists.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created report **Example** : 5	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to remove an reported art item object

```
DELETE /api/copyright
```

Parameters

Type	Name	Description	Schema
Body	**art_item_id** *required*	ID of the reported art item	string

Responses

HTTP Code	Description	Schema
200	Reported art item is successfully removed.	Response 200
400	You have not provided some of the required fields.	No Content
404	There are no Art Item with the id <art_item_id>.	No Content

Response 200

Name	Description	Schema
id *required*	ID of the removed art item **Example** : `46`	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch an copyright report with a given ID

```
GET /api/copyright/{report_id}
```

Parameters

Type	Name	Description	Schema
Path	**report_id** *required*	ID of the Copyright Infringement Report	integer

Responses

HTTP Code	Description	Schema
200	Copyright Infringement Report with the given ID is successfully fetched.	Response 200
404	There are no reports with the id <report_id>.	No Content

Response 200

Name	Description	Schema
creation_date *required*	The date when the report is created **Example** : `"2022-05-17"`	string
creator *required*	User ID of the report creator **Example** : `2`	integer
creator_name *required*	Name of the report creator **Example** : `"Brian Tennison"`	string

Name	Description	Schema
description *required*	The informative text about the report **Example** : `"These two art items are too similar, there might be a copyright infringement."`	string
id *required*	ID of the report **Example** : `5`	integer
infringement_art_creation_date *required*	The date when the infringement art item is created **Example** : `"2022-05-19"`	string
infringement_art_item_id *required*	ID of the infringement art item **Example** : `46`	integer
infringement_art_name *required*	Name of the infringement art item **Example** : `"The Starry Night Recreation"`	string
infringement_art_uri *required*	The link to the resource file of the infringement art item (image, gif, etc.) **Example** : `"https://cdn.shopify.com/s/files/1/0839/9135/products/Starry_Night_final_large.jpg?v=1438585349"`	string
original_art_creation_date *required*	The date when the original art item is created **Example** : `"2022-05-17"`	string
original_art_item_id *required*	ID of the original art item **Example** : `37`	integer
original_art_name *required*	Name of the original art item **Example** : `"The Starry Night"`	string
original_art_uri *required*	The link to the resource file of the original art item (image, gif, etc.) **Example** : `"https://en.wikipedia.org/wiki/The_Starry_Night#/media/File:Van_Gogh_-Starry_Night-_Google_Art_Project.jpg"`	string

Name	Description	Schema
similarity_score *required*	Number that shows the similarity between the reported art items, Smaller the value, similar the images **Example** : 5	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to create an event object

```
POST /api/event
```

Parameters

Type	Name	Description	Schema
Body	**city** *required*	City where the event takes place	string
Body	**date** *required*	Date of the event	string
Body	**description** *required*	The informative text about the art item	string
Body	**poster_link** *required*	Link to the event poster	string
Body	**title** *required*	Title of the event	string

Responses

HTTP Code	Description	Schema
201	Event with the given data is successfully created.	Response 201
400	Date you have entered is not valid. Format is "%Y-%m-%d". You entered "<DATE>".	No Content

HTTP Code	Description	Schema
401	Missing Authorization Header	No Content
409	Please enter a title that doesn't exist in the platform.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created event **Example** : 2	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch an event with a given ID

```
GET /api/event/{event_id}
```

Parameters

Type	Name	Description	Schema
Path	**event_id** *required*	ID of the event	integer

Responses

HTTP Code	Description	Schema
200	Art item with the given ID is successfully fetched.	Response 200
404	There are no events with the id <event_id>.	No Content

Response 200

Name	Description	Schema
artist_id *required*	ID of the creator artist **Example** : `2`	integer
artist_name *required*	Name of the artist **Example** : `"Ivan Aivazovsky"`	string
city *required*	City where the event takes place **Example** : `"Istanbul"`	string
date *required*	Date of the event **Example** : `"Fri, 01 Apr 2022 00:00:00 GMT"`	string
description *required*	The informative text about the event **Example** : `"An event to feel as if you are cruising in the ocean without setting foot on a boat."`	string
id *required*	Id of the event **Example** : `3`	integer
poster_link *required*	Link to the event poster **Example** : `"https://en.wikipedia.org/wiki/Ivan_Aivazovsky#/media/File:Aivazovsky_-_Self-portrait_1874.jpg"`	string
title *required*	Title of the event. **Example** : `"The Aivazovsky Day"`	string
weather *required*	Information about the current weather at the event location	weather

weather

Name	Description	Schema
temp *optional*	Temparature at the event location **Example** : `"13.17 C"`	string
weather *optional*	Description of the weather condition **Example** : `"Rain"`	string

Endpoint to create a forum post comment object

```
POST /api/forum_comment_post
```

Parameters

Type	Name	Description	Schema
Body	**content_uri** *required*	The link to the image to be attached to the comment (image, gif, etc.)	string
Body	**parent_post** *required*	ID of the parent forum post	string
Body	**text** *required*	The body of the comment to be posted under the discussion post	string

Responses

HTTP Code	Description	Schema
201	A comment with the given data is successfully created.	Response 201
400	Missing required parameters.	No Content
401	Access token is missing or invalid.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created comment **Example** : 15	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch all forum posts in database.

```
GET /api/forum_get/
```

Responses

HTTP Code	Description	Schema
200	Forum posts in database are successfully fetched.	Response 200
500	Error while extracting posts from database.	No Content

Response 200

Name	Description	Schema
results *required*	Array of all forum posts in database. **Example** : ""	< object > array

Endpoint to create a new forum post

```
POST /api/forum_post/
```

Parameters

Type	Name	Description	Schema
Body	**content_uri** *optional*	The link to the resource file of the visual content being shared (image, gif, etc.)	string
Body	**description** *required*	The main textual body of the forum post.	string
Body	**title** *optional*	Title for the forum post	string

Responses

HTTP Code	Description	Schema
201	Forum post is successfully added to the cluster.	Response 201
500	Error recording post to database.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created forum post. **Example** : 47	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch a forum post with a given ID

```
GET /api/get_discussion_post/{post_id}
```

Parameters

Type	Name	Description	Schema
Path	**post_id** *required*	ID of the forum post	integer

Responses

HTTP Code	Description	Schema
200	Forum post with the given ID is successfully fetched.	Response 200
404	There is no forum post with the given ID.	No Content

Response 200

Name	Description	Schema
comments *required*	a list of comments written under the forum post **Example** : `[ { "content_uri" : "https://random.imagecdn.app/500/150", "creation_date" : "Wed, 18 May 2022 00:00:00 GMT", "creator" : 5, "id" : 35, "name" : "Johnny Depp", "parent_post" : 7, "text" : "Great forum post!", "translation" : "Harika bir forum yazisi!" } ]`	< comments > array

Name	Description	Schema
post *required*	Post information **Example** : `{ "content_uri" : "https://random.imagecdn.app/500/150", "creation_date" : "Wed, 9 May 2022 00:00:00 GMT", "creator" : 9, "description" : "This is the description of the post", "id" : 23, "title" : "Starry Night", "user_name" : "Michael Jackson" }`	post

comments

Name	Description	Schema
content_uri *required*	The link to the image attached to the comment (image, gif, etc.) **Example** : `"https://random.imagecdn.app/500/150"`	string
creation_date *required*	Date of creation of the comment **Example** : `"Wed, 15 May 2022 00:00:00 GMT"`	string
creator *required*	ID of the creator of the comment **Example** : `5`	integer
id *required*	ID of the comment **Example** : `35`	integer
name *required*	name of the creator of the comment **Example** : `"Johnny Depp"`	string
parent_post *required*	ID of the parent post **Example** : `7`	integer
text *required*	The body of the comment to be posted under the discussion post **Example** : `"Great forum post!"`	string
translation *required*	The turkish translation of the body of the comment posted under the discussion post **Example** : `"Harika bir forum yazisi!"`	string

post

Name	Description	Schema
content_uri *required*	The link to the image attached to the forum post (image, gif, etc.) **Example** : `"https://random.imagecdn.app/500/150"`	string
creation_date *required*	Date of creation of the forum post **Example** : `"Wed, 9 May 2022 00:00:00 GMT"`	string
creator *required*	ID of the creator of the post **Example** : `9`	integer
description *required*	Description of the forum post **Example** : `"This is the description of the post"`	string
id *required*	ID of the forum post **Example** : `23`	integer
title *required*	The title of the forum post **Example** : `"Starry Night"`	string
user_name *required*	The name of the user who created the forum post **Example** : `"Michael Jackson"`	string

Endpoint to login

```
POST /api/login
```

Parameters

Type	Name	Description	Schema
Body	**email** *required*	Email of the new user	string
Body	**password** *required*	Password of the new user	string

Responses

HTTP Code	Description	Schema
200	The sign up was successful.	Response 200

HTTP Code	Description	Schema
401	Incorrect email or password.	No Content

Response 200

Name	Description	Schema
access_token *required*	An access token for the logged in user to access restricted content **Example** : `"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2V4YW1wbGUuYXV0aDAuY29tLyIsImF1ZCI6Imh0dHBzOi8vYXBpLmV4YW1wbGUuY29tL2NhbGFuZGFyL3YxLyIsInN1YiI6InVzcl8xMjMiLCJpYXQiOjE0NTg3ODU3OTYsImV4cCI6MTQ1ODg3MjE5Nn0.CA7eaHjIHz5NxeIJoFK9krqaeZrPLwmMmgI_XiQiIkQ"`	string

Endpoint to add a participant to given event

```
POST /api/participants/add/{event_id}
```

Parameters

Type	Name	Description	Schema
Path	**event_id** *required*	ID of the event	integer

Responses

HTTP Code	Description	Schema
201	User is added succesfully as participant to event.	Response 201
404	Event with given parameter ID cannot be found.	Response 404
409	Happens when the user already participates in the event.	Response 409
500	Server cannot process the call, there was an error probably regarding the database.	Response 500

Response 201

Name	Description	Schema
success *required*	Success Message **Example** : `"Successfully added the participant."`	string

Response 404

Name	Description	Schema
error *required*	Error Message **Example** : `"There are no events with the given ID."`	string

Response 409

Name	Description	Schema
error *required*	Error Message **Example** : `"User already participating"`	string

Response 500

Name	Description	Schema
error *required*	Error Message **Example** : `"There was an error adding the user as a participant to event."`	string

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to remove participant(s) from given event

```
POST /api/participants/remove/{event_id}
```

Parameters

Type	Name	Description	Schema
Path	**event_id** *required*	ID of the event	integer

Type	Name	Description	Schema
Body	**participants** *optional*	An array to hold participant's ID	string

Responses

HTTP Code	Description	Schema
200	Participants for event removed successfully.	Response 200
400	Given request body was erroneous key/values were not matching with the schema.	Response 400
403	Happens when the user tries to remove other participants and is not the creator.	Response 403
404	Event with given parameter ID cannot be found.	Response 404
409	Happens when the target user doesn;t participate in the event.	Response 409
500	Server cannot process the call, there was an error probably regarding the database.	Response 500

Response 200

Name	Description	Schema
success *required*	Success Message **Example** : `"Successfully added the participant."`	string

Response 400

Name	Description	Schema
error *required*	Error Message **Example** : `"There was an error on key / value pairs on request body."`	string

Response 403

Name	Description	Schema
error *required*	Error Message **Example** : `"User is not the creator of the event"`	string

Response 404

Name	Description	Schema
error *required*	Error Message **Example** : `"There are no events with the given ID."`	string

Response 409

Name	Description	Schema
error *required*	Error Message **Example** : `"User does not participate in the event"`	string

Response 500

Name	Description	Schema
error *required*	Error Message **Example** : `"There was an error adding the user as a participant to event."`	string

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to get a share link for event with given ID

```
POST /api/participants/share/{event_id}
```

Parameters

Type	Name	Description	Schema
Path	**event_id** *required*	ID of the event	integer
Body	**page_url** *required*	URL of the target	string

Responses

HTTP Code	Description	Schema
200	Participants for event removed successfully.	Response 200
400	Given request body was erroneous key/values were not matching with the schema, or can be result of invalid link in the request.	Response 400
404	Event with given parameter ID cannot be found.	Response 404
409	A conflict happens on the API call	Response 409
500	API call from backend fails, link couldn't be created.	Response 500
502	An unexpected error occured on the server.	Response 502

Response 200

Name	Description	Schema
share_link *required*	Shortened URL **Example** : `"cutt.ly/ornek_link_addres"`	string

Response 400

Name	Description	Schema
error *required*	Error Message **Example** : `"There was an error on key / value pairs on request body."`	string

Response 404

Name	Description	Schema
error *required*	Error Message **Example** : `"There are no events with the given ID."`	string

Response 409

Name	Description	Schema
error *required*	Error Message **Example** : `"The link is already shortened"`	string

Response 500

Name	Description	Schema
error *required*	Error Message **Example** : `"Error when creating the share link. Try again later."`	string

Response 502

Name	Description	Schema
error *required*	Error Message **Example** : `"Unexpected error"`	string

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to list all participants of an event with given ID.

```
GET /api/participants/{event_id}
```

Parameters

Type	Name	Description	Schema
Path	**event_id** *required*	ID of the event	integer

Responses

HTTP Code	Description	Schema
200	Participants for event found successfully.	Response 200
404	There are no events with the given ID."	No Content

Response 200

Name	Description	Schema
event_title *required*	Title of the event **Example** : `"Test Event"`	string
is_creator *required*	Parameter identifying if the user is the creator of the event. **Example** : `true`	boolean
participants *required*	An array containing id of all participants **Example** : `[ 1, 2, 29 ]`	< object > array
user_participating *required*	Identifies user's participation in the event **Example** : `false`	boolean

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to view profile of current user

```
GET /api/profile/
```

Responses

HTTP Code	Description	Schema
200	User is successfully fetched.	Response 200
404	There is no user with the same id as token.	No Content

Response 200

Name	Description	Schema
email *required*	Email of user **Example** : `"user@email.com"`	string
first_name *required*	First name of user **Example** : `"Ali"`	string

Name	Description	Schema
id *required*	Id of user **Example** : `3`	integer
is_verified *required*	True if user is verified **Example** : `true`	boolean
last_name *required*	Last name of user **Example** : `"Yılmaz"`	string

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to fetch all events and art items on the platform

```
GET /api/search
```

Parameters

Type	Name	Description	Schema
Query	**query** *optional*	A keyword with which the content on the platform is filtered.	string

Responses

HTTP Code	Description	Schema
200	Event and Art Item information succesfuly retrieved.	Response 200

Response 200

Name	Description	Schema
art_items *required*	List of art item data	< art_items > array

Name	Description	Schema
events *required*	List of event data	< events > array

art_items

Name	Description	Schema
content_uri *optional*	The link to the resource file of the art item (image, gif, etc.) **Example** : `"https://en.wikipedia.org/wiki/The_Starry_Night#/media/File:Van_Gogh_-Starry_Night-_Google_Art_Project.jpg"`	string
creation_date *optional*	The date when the art item is created **Example** : `"2022-05-17"`	string
creator_artist *optional*	User ID of the creator of the art item **Example** : `29`	integer
description *optional*	The informative text about the art item **Example** : `"The Starry Night is an oil-on-canvas painting by the Dutch Post-Impressionist painter Vincent van Gogh."`	string
id *optional*	ID of the art item **Example** : `37`	integer
name *optional*	Name of the art item **Example** : `"The Starry Night"`	string

events

Name	Description	Schema
artist_id *optional*	ID of the creator artist **Example** : `2`	integer
city *optional*	City where the event takes place **Example** : `"Istanbul"`	string
date *optional*	Date of the event **Example** : `"Fri, 01 Apr 2022 00:00:00 GMT"`	string
description *optional*	The informative text about the event **Example** : `"An event to feel as if you are cruising in the ocean without setting foot on a boat."`	string

Name	Description	Schema
id *optional*	Id of the event **Example** : `3`	integer
poster_link *optional*	Link to the event poster **Example** : `"https://en.wikipedia.org/wiki/Ivan_Aivazovsky#/media/File:Aivazovsky_-_Self-portrait_1874.jpg"`	string
title *optional*	Title of the event. **Example** : `"The Aivazovsky Day"`	string

Endpoint to sign up

```
POST /api/signup
```

Parameters

Type	Name	Description	Schema
Body	**email** *required*	Email of the new user	string
Body	**first_name** *required*	First name of the new user	string
Body	**is_artist** *required*	States whether the new user is an artist	string
Body	**last_name** *required*	Last name of the new user	string
Body	**password** *required*	Password of the new user	string

Responses

HTTP Code	Description	Schema
201	The sign up was successful, a new user has been created.	Response 201
409	A user with the specified email already exists.	No Content

Response 201

Name	Description	Schema
access_token *required*	An access token for the newly created user to access restricted content **Example** : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2V4YW1wbGUuYXV0aDAuY29tLyIsImF1ZCI6Imh0dHBzOi8vYXBpLmV4YW1wbGUuY29tL2NhbGFuZGFyL3YxLyIsInN1YiI6InVzcl8xMjMiLCJpYXQiOjE0NTg3ODU3OTYsImV4cCI6MTQ1ODg3MjE5Nn0.CA7eaHjIHz5NxeIJoFK9krqaeZrPLwmMmgI_XiQiIkQ"	string

Endpoint to create a verification request

```
POST /api/verification/request
```

Responses

HTTP Code	Description	Schema
201	A verification request data is successfully created.	Response 201
409	There is a request, whose status is 'Pending', for the user requesting verification. (OR) The user requesting verification is already verified.	No Content
500	Server Error.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the newly created request **Example** : 14	integer

Security

Type	Name
apiKey	**BearerAuth**

Endpoint to review a verification request

```
POST /api/verification/review/{request_id}
```

Parameters

Type	Name	Description	Schema
Body	**result** *required*	The decision for the verification request	string

Responses

HTTP Code	Description	Schema
201	The decision for the request is processed succesfully.	Response 201
404	There is no available request with the given id	No Content
500	Server error.	No Content

Response 201

Name	Description	Schema
id *required*	ID of the processed request **Example** : 37	integer
status *required*	Status of the request **Example** : 1	integer

Endpoint to fetch a verification request with a given ID

```
GET /api/verification/{request_id}
```

Parameters

Type	Name	Description	Schema
Path	**request_id** *required*	ID of the verification request	integer

Responses

HTTP Code	Description	Schema
201	Verification request with the given ID is successfully fetched.	Response 201
404	There is no verification request with the given ID.	No Content

Response 201

Name	Description	Schema
historical_event *required*	A historical event that happened on the day the user requested verification **Example** : `"Ashina Jiesheshuai of the Eastern Turkic Khaganate failed in an attempt to assassinate Emperor Taizong of the Chinese Tang dynasty and was killed by pursuers."`	string
id *required*	ID of the verification request **Example** : `9`	integer
request_date *required*	The date the user has requested verification, in Epoch format **Example** : `"2022-05-19 19:50:10.936008"`	string
status *required*	Status of the request. Status is 0 when a request is created, 1 if the request is accepted and -1 if the request is rejected **Example** : `"Rejected"`	string
user_id *required*	ID of the user requested verification **Example** : `4`	integer
year *required*	In what year the historical event happened? **Example** : `630`	integer

Endpoint to get a definition of a word from wikipedia

```
POST /api/wikipedia_definition
```

Parameters

Type	Name	Description	Schema
Body	**word** *required*	Word to be searched on wikipedia	string

Responses

HTTP Code	Description	Schema
200	Got a definition from wikipedia successfully	Response 200
400	Did not send any words.	No Content

Response 200

Name	Description	Schema
definition *required*	Definition of the word in query. **Example** : `"Pizza (Italian: ['pittsa], Neapolitan: ['pittsə]) is a dish of Italian origin consisting of a usually round, flat base of leavened wheat-based dough topped with ⋯"`	string