CMPE 352
Milestone Report 2
Group 2

Altay Acar (Communicator)
Bahrican Yeşil
Ecenur Sezer
Egemen Atik
Ezgi Aysel Batı
Hasan Can Erol
Mehmet Batuhan Çelik
Muhammed Enes Sürmeli
Onur Kömürcü

Table of Contents

1. Executive Summary
    - 1.1. Introduction/Project Description
    - 1.2. Project Status
    - 1.3 Basic Functionality of the Project
    - 1.4. Challenges
    - 1.5. List and Status of Deliverables
    - 1.6. API URI
    - 1.7. URL of Deployed Application
2. Table of Work Done by Each Team Member
3. Evaluation of Tools and Processes
    - 3.1. Evaluation of Tools
        - 3.1.1. Discord
        - 3.1.2. GitHub
        - 3.1.3. Google Docs
        - 3.1.4. Node.js & Express.js
        - 3.1.5. Vue.js
        - 3.1.6. Postman
        - 3.1.7. MongoDB
        - 3.1.8. Docker
        - 3.1.9 AWS
        - 3.1.10 Lucidchart
    - 3.2. Evaluation of Processes
        - 3.2.1. Team Meetings
        - 3.2.2. TA Meetings
        - 3.2.3. Issue Creation and Management
        - 3.2.4 Pull Request Creation and Management
4. Deliverables
    - 4.1. Project Repository - Code
    - 4.2. Requirements
        - 4.2.2. Requirements
            - 4.2.2.1. Functional Requirements
                - 4.2.2.1.1. User Requirements
                - 4.2.2.1.2. System Requirements
            - 4.2.2.2. Non-Functional Requirements
    - 4.3. Software Design Documents in UML
        - 4.3.1. Use-Case Diagrams
        - 4.3.2. Class Diagrams
        - 4.3.3. Sequence Diagrams
    - 4.4 URI Tag of the project
5. URI
6. Project Plan

1. Executive Summary

1.1 Introduction/Project Description

As group 2 of CMPE 352 course, our task is to design a learning platform where people can freely give courses and become a learner. For this purpose, we have lecturers and learners as users of our application. Additionally, many other features such as organizing events, filtering, searching courses, and rating lessons exist. We ought to implement a web application for this platform.

Before actual implementation of our project, we have a task to design a practice app to test our frontend and backend skills. We designed this application to have similar features and requirements. Each team member has a task to create at least two, GET and POST endpoints based on our needs, and implement a UI. In our application, we can view lessons, create events where learners of a lesson can get together in real life, search and filter lessons based on categories and ratings, rate a lesson. Our last task is to deploy the application using Docker and AWS systems.

1.2 Project Status

Over the whole term, we have worked on designing our application as stated in Milestone 1. We first decided on requirements, what features are required for our application and we split them into sections. When questions related to deciding requirements arised, we attended customer meetings to clarify. After that, we created mockups and scenarios for basic actions users are capable to do in our app. For detailing the plan of implementation, we created class-user-sequence diagrams. To observe our work done so far, we created project plans and RAM.

One of the most important research topics for our implementation was becoming skilled in Git and Github since we have many pull requests, issues and merges. As a team we developed our skills on using Git. Next step was determining on technologies we will use in our app. We had meetings and researches about finding the most optimal tools for our app and we choose Vue.js and Node.js frameworks. Finally, after researching Docker and AWS to deploy our app, we started implementing practice app.

As stated in 1.1 we made a job share for implementing required features. Every team member created issues for implementing endpoints first since we started from backend. After a endpoint is completed, we created PR’s for corresponding issues. Next step of finalizing endpoints was coding unit tests for each. After tests and manual tests from Postman all passed, each team member implemented a UI for the endpoints they are responsible. During this process, reviewing of each others work was a integral part. We stated the errors and necessities of each other’s work during implementation. Finally, we documented API and Milestone.

1.3 Basic Functionality of the Project

Our project's aim was establishing a web application that has functionalities that eclipses with the work we have done until our previous milestone report 1. Since we have aimed to develop an online learning platform according to the project description provided, our practice application is developed to have some of the basic functionalities of the online learning platform we have planned throughout the semester.

Our practice application as mentioned above is a web application and provides a graphical user interface for users. A signup page welcomes users and they can sign up to our database by filling the required fields. Each user has a unique email and a hashed password stored in our database. A signed up user can log in to our application via the login page.

Since our practice application is some sort of a learning platform, there are three main components included alongside the user: lessons, categories, and events. Each user can create a lesson by providing required fields and then search for lessons available throughout the application. Each lesson belongs to a specific category, which can also be added to our application's database by the users. To add a category, providing a valid category name is enough. Our application connects to Wikipedia's public API to fetch category description and displays each category alongside its name in the categories page. Users can also filter the categories according to their names. When its category is added in our application, a user can create a lesson. Since each lesson has a category and a lecturer, which is the user who created it, a user can filter lessons by category and lecturers. According to their preferences a user can enroll to a lesson and rate it in the rating page. Rating page also provides users' the opportunity to filter lessons according to their ratings.

For each lesson, a user can create an event that will occur in a specific location and on a specific date, by providing those fields alongside the title and the lesson ID of the lesson that the event belongs to. Each user can create an event for an existing lesson and upon creation our system assigns them as the host of the event. During creation, the user needs to provide a valid location and our application converts it to a valid full address via an API call to the Google Maps Geocoding API. Each user can view a lesson's events alongside its specific event details that are specified during the creation and attend to an event. After a user attends an event, they can see the events they are attending in the attended events page.

To sum up, our application is a basic online learning platform that contains lessons under the section of categories and enables users to create and enroll to a lesson alongside creating and attending to an event about the lesson.

1.4 Challenges

First and foremost, all of us faced a big hurdle in learning all of the new ideas required to implement the practice app. Learning and implementing new concepts, libraries, and programming languages in which we had no prior expertise in a relatively short period of

time was really tough, time-consuming, and mentally exhausting. Fortunately, we were able to aid and educate each other at all times by effectively employing communication technologies with some productive meetings.

The decision of which framework to utilize was the second big hurdle we faced throughout this milestone. Surprisingly, there was no structure with which we were all accustomed, so we decided to go with Node.js for the backend part. However, Node.js in general, the REST Framework, Dockerizaton, and AWS deployment all required research and practice. Experienced ones gave other members of our group instruction and a demo so they could feel comfortable working with the framework. Fortunately, we had almost no trouble dealing with the git related problems with the help of the Git PS lecture given to us prior to our starting point. Nonetheless, Problems we had faced using external APIs were another issue.

Finally, yet importantly, we had some issues while implementing the frontend part which we designed to present our work and functions in a visually appealing manner with a lot of time spent fixing bugs and learning new concepts of vue.js, the progressive javascript framework. Aside from the technological problems, another challenge for us was the increase in the workload per group member as the number of group members decreased from 11 to 9 as the term proceeded given that we were initially allocated as a group of eleven.

In general, the issues originated from the fact that the subjects and tools were all new to us, and we had to step outside of our comfort zones to learn new things and struggle. We solved all of these issues quickly by working together and tackling the obstacles collectively. The ultimate outcome was very satisfying for all of us, but we had to work very hard to get there.

1.5 List and Status of Deliverables

Deliverable Name	Delivery Status	Due Date	Delivery Date
Up To Date Project Repository	Delivered	20/05/2022	20/05/2022
Requirements: Software Requirement Specification	Delivered	20/05/2022	20/05/2022
Software Design Documents in UML: Use Case, Class, and Sequence Diagrams.	Delivered	20/05/2022	20/05/2022
URI of the Tag of the project	Delivered	20/05/2022	20/05/2022

Project Plan	Delivered	20/05/2022	20/05/2022
Code	Delivered	20/05/2022	20/05/2022
Milestone Report	Delivered	20/05/2022	20/05/2022

1.6. API URI

API Documentation Link

1.7. URL of Deployed Application

API:: http://3.69.203.16:3000
User Interface:: http://3.69.203.16:8000
DNS: ec2-3-69-203-16.eu-central-1.compute.amazonaws.com

2. Table of Work Done by Each Team Member

Member Name	Work Detail	Issue Link	Wiki/Resource Link
Altay Acar	Attended all team meetings, including regular meetings on Thursdays and additional task-specific meetings	-	All meeting notes
	Examined previous years' group repositories, documents, and wiki pages regularly	-	-
	Wrote weekly efforts to the personal weekly effort table on a weekly basis	-	Personal weekly effort table
	Researched a collective list of free and public APIs.	-	-
	Researched on public APIs for Geocoding and Geolocation	-	-
	Identified the best available public APIs that we can use for geocoding and decided the one we will use	Issue #169	-
	Create the event model and validators in the back-end of the practice app.	Issue #193	Event model
	Implement the geocoding API call for the create event POST endpoint in backend of the practice app.	Issue #195	Related file
	Implement the create event POST endpoint in the back-end of the practice app.	Issue #195	Related file
	Test the create event POST endpoint with Postman and create both automatic and manual documentation.	Issue #195	Link to the API documentation
	Reviewed the pull request about creating the user	Issue #151	Related pull request

	model		
	Reviewed the pull request about get lesson by category endpoint	Issue #159	Related pull request
	Reviewed the pull request about creating the category model	Issue #163	Related pull request
	Write the unit tests of the create event POST endpoint, and run the tests.	Issue #211	Related file
	Implement the get attended events GET endpoint in the back-end of the practice app.	Issue #217	Related file
	Test the get attended events GET endpoint with Postman and create both automatic and manual documentation.	Issue #217	Link to the API documentation
	Write the unit tests of the get attended events GET endpoint, and run the tests.	Issue #218	Related file
	Revised the unit tests of the create event POST endpoint according to the feedback received.	Issue #211	Related file
	Revised the implementation of the create event POST endpoint according to the feedback received.	Issue #195	Related file
	Reviewed the pull request about creating category page in frontend	Issue #204	Related pull request
	Reviewed the pull request about creating signup page in frontend	Issue #232	Related pull request
	Implement the create event page in the front-end of the practice app.	Issue #220	Related file
	Implement the attended events page in the front-end of the practice app.	Issue #221	Related file
	Updated the unit tests for POST create event endpoint in the backend of the practice app.	Issue #274	Related file
	Filled the basic functionality of the project part in the milestone report 2	Issue #257	Milestone Report 2
	Provided a use case diagram for our practice application	Issue #258	Wiki Page
	Provided requirements about event creation and viewing attended events for our practice application	Issue #259	Wiki Page
	Reviewed the deployment process of the practice application	Issue #268	Deployed URL
	Reviewed the requirements about signup and viewing lessons for practice application	Issue #271	Wiki Page

	Filled my part of the individual work done part of our milestone report 2	-	Milestone Report 2
	Prepared individual milestone report	-	Individual Milestone Report
Bahrican Yeşil	Attended all team meetings, including regular meetings on Thursdays and additional task-specific meetings	-	All meeting notes
	Examined previous years' group repositories, documents, and wiki pages regularly	-	-
	Wrote weekly efforts to the personal weekly effort table on a weekly basis	-	Personal weekly effort table
	Researched a collective list of free and public APIs.	Issue #147	-
	Researched on public APIs for Geocoding and Geolocation	Issue #147	-
	Researched on public APIs for Weather Forecast	Issue #147	-
	Create the user model and validators in the back-end of the practice app.	Issue #151	User Model
	Open a PR for creating the user model and validators, get review and at least one approval	Issue #151	PR #156
	Implement the signup POST endpoint in the back-end of the practice app.	Issue #152	Signup Controller
	Create a PR for the implementation of the signup endpoint, get review and at least one approval.	Issue #152	PR #157
	Test the signup POST endpoint with Postman and create both automatic and manual documentation.	Issue #152	Documentation
	Write the unit tests of the signup POST endpoint, and run the tests.	Issue #153	signup.test.js
	Create a PR for the unit tests of the signup endpoint, get review and at least one approval.	Issue #153	PR #158
	Implement the get lessons by category GET endpoint in the back-end of the practice app.	Issue #159	get_lessons_by_category.js
	Create a PR for the get lessons by category endpoint, get review and at least one approval.	Issue #159	PR #161
	Test the get lessons by category GET endpoint with Postman and create both automatic and manual documentation.	Issue #159	Documentation
	Write the unit tests of the get lessons by category GET endpoint, and run the tests.	Issue #160	get_lessons_by_category.test.js

	Create a PR for the unit tests of the get lessons by category endpoint, get review and at least one approval.	Issue #160	PR #162
	Revised the unit tests of the get lessons by category GET endpoint according to the feedback received.	PR #267	get_lessons_by_category.test.js
	Model update on the unit tests of the get lessons by category GET endpoint.	PR #230	get_lessons_by_category.test.js
	Implement the signup page in the front-end of the practice app.	Issue #232	Signup.vue
	Open a PR for the signup page in the front-end, get review and at least one approval.	Issue #232	PR #235
	Implement the get lessons by category page in the front-end of the practice app.	Issue #233	CategoryLessons.vue
	Open a PR for the get lessons by category page in the front-end, get review and at least one approval.	Issue #233	PR #266
	Bugfix: update the lesson item in the lessons by category screen with the updated model of the lesson.	PR #283	CategoryLessons.vue
	Provided requirements about signup and get lessons by category for our practice application	Issue #271	Wiki Page
	Reviewed and communicated with Batuhan about the deployment process of the practice application front-end	Issue #222	Front-end Deployed URL
	Reviewed and communicated with Batuhan about the deployment process of the practice application back-end	Issue #222	Back-end Deployed URL
	Filled my part of the individual work done part of our milestone report 2	Issue #293	Milestone Report 2
	Prepared the individual milestone report	-	Individual Milestone Report - SENT VIA MOODLE
	Practice App: Initialization Steps of the Project - Create .gitignore and README files	Issue #143	Initialized Project
	Create a pull request for the initialization steps, get review and at least one approval	Issue #143	PR #144
	Practice App: Identify Base API Features and Create a Do-Do List	Issue #146	-
	Documenting Meeting Notes - Meeting 9	Issue #148	Meeting Notes - 9

	Practice App: Set up the base structure of the back-end	Issue #149	Base Setup of the Back-end - express.js
	Create a PR for set up the base structure of the back-end, get review and at least one approval	Issue #149	PR #154
	Practice App: Build the connection with the MongoDB database	Issue #150	mongoose.js
	Create a PR for building the connection with the MongoDB database	Issue #150	PR #155
	Milestone-2: Evaluation of Tools - Discord, GitHub, Google Docs, Node.js & Express.js, Vue.js	Issue #270	Milestone Report 2
	Reviewed Search Lecture by Name Page PR	PR #294	SearchLesson.vue
	Reviewed Enrolled Lessons Page PR	PR #282	EnrolledLessons.vue
	Reviewed Hotfix - Fixing error messages on frontend: Rating PR	PR #280	Rating.vue
	Reviewed Bugfix drop lesson and get lesson events test update PR	PR #272	get_lesson_events.test.js
	Reviewed Login endpoint unit tests PR	PR #260	login.test.js
	Reviewed Create Event Page PR	PR #255	Event.vue
	Reviewed changing static localhost to URI PR	PR #252	Signup.vue
	Reviewed Front-end initizialition PR	PR #185	Initialized Frontend Project
	Reviewed bugfix, inserting beforeach to describe body PR	PR #177	get_lessons_by_category.test.js
	Reviewed unit tests for POST category PR	PR #172	post.category.test.js
	Reviewed HOTFIX: POST /api/category, adding no title is given to bad request cases PR	PR #171	createCategoryEndpoint.js
	Reviewed Generalization of .gitignore and .dockerignore in frontend PR	PR #231	.dockerignore
	Reviewed Writing the Unit Tests for the Attend Event Endpoint PR	PR #228	attend_event.test.js
	Reviewed Implementing unit tests for GET api/user/attendedEvents endpoint PR	PR #225	get.attended_events.test.js
	Reviewed Implementation of GET /api/user/attendedEvents endpoint PR	PR #223	getAttendedEvents.js

	Reviewed Implementation of POST /api/event/createEvent endpoint PR	PR #209	createEvent.js
	Reviewed Practice App: Implementing POST Login Endpoint PR	PR #208	login.js
	Reviewed Test for drop lesson endpoint is implemented PR	PR #207	drop_lesson.test.js
	Reviewed Implementing the Attend Event Endpoint PR	PR #206	attendEvent.js
	Reviewed Practice app/frontend/categories page 204 PR	PR #205	Category.vue
	Reviewed mocking MongoDB connection during tests PR	PR #201	mongoose.js
Ecenur Sezer	Filled the weekly work in my personal wiki page	-	Personal Wiki
	Examined previous years' repositories	-	-
	Made research about API's and endpoints	-	-
	Attended weekly meetings	-	Meeting Notes
	Documented Meeting Notes - Meeting 7	Issue #145	Meeting Notes
	Reviewed POST Category Endpoint	Issue #165	Related repository
	Implemented POST Rating Endpoint	Issue #181	Related repository
	Implemented GET Rating Endpoint	Issue #188	Related repository
	Created validators for rating endpoints	-	Related file
	Filled individual work	Issue #288	Milestone-2
	Implemented unit tests for POST/rating	Issue #196	Related repository
	Implemented unit tests for GET/rating	Issue #203	Related repository
	Created manual tests for rating endpoints and documented using Postman	-	Postman Documentation
	Fixed bugs & changed the implementation of GET/rating and POST/rating endpoints	Issue #246	Related repository
	Fixed bugs & changed the implementation of unit tests of GET/rating and POST/rating endpoints	Issue #247	Related repository
	Implemented the UI for Rating page of Practice App	Issue #248	Related file
	Fixed bugs on error messages of Rating page	Issue #281	Related file

	Created and Documented Class Diagram for Practice-App	Issue #285	Class Diagram for Practice-App
	Filled Introduction/Project Description & Project Status Parts	Issue #284	Milestone-2
	Reviewed bugfix for POST/api/category	PR#165	Related repository
	Reviewed Basic Functionality of the Project Part of Milestone-2	Issue #257	Milestone-2
	Created individual report	-	Individual Report
Egemen Atik	Attended all team meetings, including regular meetings on Thursdays and additional task-specific meetings	-	Meeting Notes Page
	Examined previous years' group repositories, documents, and wiki pages regularly	-	-
	Wrote weekly efforts to the personal weekly effort table on a weekly basis	-	Personal Effort Table
	Attended to every problem session except the last one, along with the customer meetings	-	-
	Made research about APIs, endpoints, Postman, Node.js, Express.js, Vue.js, MongoDB	-	-
	Watch tutorial videos about technologies I will use		-
	Helped creation of Event model	Issue #186	Initial Event Model
	Implemented the POST method for dropping lesson functionality	Issue #174	PR # 178 dropLesson.js
	Prepared a hand-written and Postman documentation for endpoint	Issue #174	Postman Documentation
	Implemented the GET method for getting lesson events functionality	Issue #186	PR #191 get_lesson_events.js
	Prepared a hand-written and Postman documentation for endpoint	Issue #186	Postman Documentation
	Tried to decrease database connections in my code but then discarded	PR #182	PR #182
	Changed error status codes in my code	PR #198	PR #198
	Implemented unit tests for dropping lessons endpoint	Issue #199	PR #207 drop_lesson.test.js
	Implemented unit tests for getting lesson events endpoint	Issue #224	PR #234 get_lesson_events.te

			st.js
	Fixed a bug occured due to a change in Lesson model	PR #272	PR #272
	Implemented frontend for drop lesson endpoint	Issue #251	PR #282 EnrolledLessons.vue
	Implemented GET method for getting enrolled lessons of a user	Issue #275	PR #276 getEnrolledLessons.js
	Prepared a hand-written and Postman documentation for endpoint	Issue #275	Postman Documentation
	Implemented unit tests for getting enrolled lessons of a user endpoint	Issue #277	PR #279 get_enrolled_lessons.test.js
	Implemented frontend for getting enrolled lessons of a user endpoint	PR #282	PR#282 EnrolledLessons.vue
	Implemented frontend for getting lesson events endpoint	Issue #289	PR #290 LessonEvents.vue
	Wrote evaluation of tools for Postman, MongoDB, Lucidchart	Issue #297	This document
	Filled my part of the individual work done part of our milestone report 2	Issue #303	This document
	Reviewed PR for implementing tests for GET /api/categories	PR #179	get_categories.test.js
	Reviewed PR for update event model	PR #194	event.js
	Reviewed PR for implementing the enroll lesson endpoint	PR #190	enrollLesson.js
	Reviewed PR for implementing GET Lecture by Name Endpoint	PR #210	get_lessons_by_name.js
	Reviewed PR for Implementing the Get Specific Event Details Endpoint	PR #216	getEventDetails.js
	Reviewed PR for Implementation of Searching Categories by Name	PR #229	filterByName.js
	Reviewed PR for BUGFIX: Get Lessons By Category Unit Tests - Category Model Update	PR #230	get_lessons_by_category.test.js
	Reviewed PR for Writing the Unit Tests for the Getting Specific Event Details Endpoint	PR #237	get_event_details.test.js
	Reviewed PR for implement GET lesson by lecturer endpoint	PR #240	get_lesson_by_lecturer.js

	Reviewed PR for Create enroll lesson unit test	PR #253	enroll_lesson.test.js
	Reviewed PR for Create get_categories_by_name.test.js	PR #254	get_categories_by_name.test.js
	Reviewed PR for Create Get Event Details Screen, Update getEventDetails and Test Part	PR #301	GetEventDetails.vue
	Reviewed PR for Practice App: Frontend | Implemented Search Lecture by Name Page	PR #294	SearchLesson.vue
	Reviewed PR for Practice App: Create Attend Event Screen	PR #287	Attend.vue
	Reviewed PR for BUGFIX: Update lesson item by adding lecturer id column	PR #283	CategoryLessons.vue
	Reviewed PR for BUGFIX: Create Event Unit Tests - Lesson Model Update	PR #273	post_events.test.js
	Reviewed PR for Practice App: Implement Attended Events Page	PR #256	AttendedEvents.vue
	Reviewed PR for Practice App: Test | Implemented GET Lecture by Name Endpoint Test	PR #261	get_lessons_by_name.test.js
	Created my individual report	-	-
Ezgi Aysel Batı	Attended team meetings regarding practice-app and contributed to the discussions	-	Discord Channel
	Implemented POST endpoint for creating a new lesson	#180 #238	Practice-App
	Implemented GET method for filtering lessons by instructor		Practice-App
	Tested developed endpoints using postman, especially the error parts	-	-
	Learned node.js	-	-
	Learned about API development and http request handling	-	-
	Reviewed work done by other team members	-	Github, practice - app
	Organized milestone report 2, including adding this table		This Document
	Added functional requirements regarding my implemented methods		This Document

	Organised & Updated all subsections of 3.2 and section 4.1 in Milestone Report 2		This Document
Hasan Can Erol	Wrote weekly efforts to the weekly effort table	-	Personal Wiki Page
	Attended weekly meetings and review meetings	-	-
	Determined the usability and disaster recovery non-functional requirements	Issue #34	Requirements Wiki Page
	POST method for Lesson Enrollment	Issue#189	Pull190
	Unit Test for Lesson Enrollment	Issue#250	Pull253
	GET method for Search Categories by Name	Issue#227	Pull229
	Unit Test for Search Categories by Name	Issue#249	Pull254
	Documented Personal Effort part of the Milestone-2 report	Issue#291	It can be seen in this document
	Designed Class Diagram of the Practice-app project	Issue#285	Lucid link
	Documented Postman documentations for Milestone-2 report	Issue#292	Link
	Documented requirements of my part of the practice app to our wiki page	Issue#278	Wiki Page
Mehmet Batuhan Çelik	Attended to all the meetings ever happened	-	All meeting notes
	Attended to all the pses	-	-
	Kept weekly efforts table	-	At the bottom of my profile
	Code review for Node project initialization	PR#154	
	Code review for drop lesson endpoint	PR#178	
	Code review for GET/POST /rating endpoint	PR#187	
	Code review for enroll lesson endpoint	PR#190	
	Code review for unit tests of GET/POST /rating endpoint	PR#197	

	Code review for POST /login endpoint	PR#208	
	Review of user interface of ratings section	PR#243	
	Code review for unit tests of get lessons by category endpoint	PR#267	
	Review of user interface of filter lessons by category section	PR#266	
	Review of user interface of login functionality	PR#265	
	Initialization of the category model	PR#164	
	Implemented POST /api/category endpoint	PR#166	
	Implemented GET /api/category endpoint	PR#168	
	Implemented unit tests for the POST /api/category endpoint	PR#172	
	Some testing files were broken and were resulting in an uncatched exception that blocked the testing main thread. Fixed that	PR#177	
	Implemented unit tests for the GET /api/category endpoint	PR#179	
	Initialized a Vue project for the practice-app with its routing, material library, and gitignore, and nav-bar	PR#185	
	Added an in memory mongo mock up to be used during testing. Thus, tests stopped affecting the real mongo database	PR#201	
	Implemented Categories tab from the frontend	PR#205	
	Dockerized the db, frontend, and backend in 3 separate containers and connected them with a docker-compose file and .env.production files	PR#222	
	Environment management for the frontend(on others code): API URI must be stored in a environment variable	PR#252	
	Fixed an “initializing a value twice” error but Bahrican was quicker on his hand and merged a solution before my request was approved.	PR#269	
	Environment management for the frontend(on others code): API URI must be stored in a environment variable (Yes, someone created a static link to the localhost again)	PR#298	

	Fixed the vue router problem of pages not reloading	PR#300	Issue#219
	Deployed the project to the AWS	Issue#268	
Onur Kömürcü	Attended to all the meetings ever happened	-	All meeting notes
	Attend all PS sessions except for one of them		
	Examined previous years' group repositories, documents, and wiki pages regularly in our group meetings	-	-
	Documented Meeting Notes #10	Issue #175	Meeting Notes 10
	Kept weekly efforts table in my profile wiki page.	-	Profile
	Shared screen in every weekly meetings	-	-
	Taked responsibility to distribute weekly works evenly among all team members	-	-
	Made research about APIs, endpoints, Postman, Node.js, Express.js, Vue.js, MongoDB to contribute Practice App development	-	-
	Finished a course about Node.js	-	-
	Research about error codes to implement res.status more accurately	-	-
	Implemented Post Login Endpoint for Practice App. It was tested with Postman and documented with it. Results of the unit tests were provided	Issue #176	PR
	Implemented Get Lecture by Name Endpoint for Practice App. It was tested with Postman and documented with it. Results of the unit tests were provided	Issue #184	PR
	Implemented unit test for Login endpoint .Checked the results are correct	Issue #213	PR
	Implemented unit test for get lesson by name endpoint. Checked the results are correct	Issue #214	PR
	Implemented frontend for the login endpoint. Tested its functions manually on my localhost	Issue #262	PR
	Implemented frontend for the get lesson by name endpoint. Tested its functions manually on my localhost	Issue #263	PR
	Reviewed pull request about practice app folder, gitignore and readme files	Issue #143	PR

	Reviewed pull request about implementation of get/api/category endpoint	Issue #167	PR
	Reviewed pull request about implemention of unit tests for post/api/category endpoint	Issue #170	PR
	Reviewed pull request about implementation of post create lesson endpoint	Issue #180	PR
	Reviewed pull request about implementation of unit tests for post/api/event/createEvent endpoint	Issue #211	PR
	Provided and documented Sequence Diagrams for the Practice app	Issue #296	Lucidchart
	Filled the my section of individual work table in Milestone-2	Issue #295	Here
	Reviewed whole Practice App to check if there is any inconvenience occurred. This minor bug fix PR was great example	-	PR
	Prepared individual milestone report	-	-
	Documented lesson search and login related requirements for the Practice App	-	Wiki Page
Muhammed Enes Sürmeli	Attended all team meetings, including weekly meetings on Thursday as well as additional task-specific meetings	-	All Meeting Notes
	On a weekly basis, I added weekly efforts to my personal weekly effort tables	-	Personal Weekly Effort Tables
	Created my personal wiki page	Issue #5	Personal Wiki Page
	Regularly reviewed previous year's group repositories, documents, and wiki pages	-	-
	Attended to every problem session, along with the customer meetings	-	-
	Made research about API's and endpoints	-	-
	Researched on public APIs for Geocoding and Geolocation	-	-
	Visually edited Milestone Report 2 for a more modular and clear look before finalization	-	This Document
	Reviewed the pull request about Fixing of the Bug for enroll_lesson.test.js	-	Related pull request
	Reviewed the pull request about Practice	Issue #289	Related pull request

	app/feature/frontend get lesson events		
	Reviewed the pull request about Added get enrolled lessons tests	Issue #277	Related pull request
	Reviewed the pull request about Practice app/feature/get enrolled lessons endpoint	Issue #275	Related pull request
	Reviewed the pull request about Practice App: FIX | Deleted additional import and use of event	-	Related pull request
	Reviewed the pull request about Practice app/feature/get lesson events tests	Issue #224	Related pull request
	Reviewed the pull request about Practice app/feature/get lesson events endpoint	Issue #186	Related pull request
	Reviewed the pull request about Create a local db and establish the connection with mongoose	Issue #150	Related pull request
	Implemented the attend event POST endpoint in the back-end of the practice app.	Issue #192	Related file
	Tested the attend event POST endpoint with Postman and created both automatic and manual documentation.	Issue #192	Link to the API documentation
	Wrote the unit tests of the attend event POST endpoint, and ran the tests.	Issue #226	Related file
	Implemented the get event details GET endpoint in the back-end of the practice app.	Issue #217	Related file
	Tested the get event details GET endpoint with Postman and created both automatic and manual documentation.	Issue #215	Link to the API documentation
	Wrote the unit tests of the get event details GET endpoint, and ran the tests.	Issue #236	Related file
	Implemented the attend event page in the front-end of the practice app.	Issue #286	Related file
	Implemented the get event details page in the front-end of the practice app.	Issue #299	Related file
	Filled the List and Status of Deliverables part in the milestone report 2	-	Milestone Report 2
	Filled the challenges of the project part in the milestone report 2	-	Milestone Report 2
	Reviewed the issue for Practice App API Documentation	Issue #292	Wiki Page

	Reviewed the issue for Milestone-2: Filling the individual work table of Hasan Can	Issue #291	Milestone Report 2
	Reviewed the issue for Milestone-2: Filling the individual work table of Ecenur	Issue #288	Milestone Report 2
	Reviewed the requirements about Event Creation and Viewing Attended Events for practice application	Issue #259	Wiki Page
	Documented requirements of my part of the practice app to our wiki page	Issue#305	Wiki Page
	Filled my part of the individual work done part of our milestone report 2	Issue#306	Milestone Report 2
	Wrote challenges our team has faced during the implementation of the practice app	Issue #297	Milestone Report 2
	Filled the List and Status of Deliverables Table	Issue#307	Milestone Report 2
	Prepared individual milestone report	-	Individual Milestone Report - SENT VIA MOODLE

3. Evaluation of Tools and Processes

3.1. Evaluation of Tools

Throughout developing our project we had to use many different tools in order to achieve different objectives. When choosing which tool to use for a task, there were several points to consider. One of the most important points was that the tool should enable us to work collaboratively and observe each other's work easily, since we always had to divide tasks among ourselves so that every member could contribute. Another criteria was ease of use since we didn't have any experience and had to learn as we progressed. Following subsections will include a short summary on the tools we used, their purpose and our evaluation.

3.1.1. Discord

We used Discord as our main communication tool. All team meetings were held in voice channel and we created different text channels for different needs such as meeting notes, research material, general and random chat. We found Discord to be useful as our main communication tool since it is easy to use and many of us were already familiar with it. Features such as having multiple voice channels in the same group and separate text channels were particularly useful. Besides these, using threads to separate messages within the same text channel (such as having a new thread for each meeting or research topic) made our channel easy to navigate and organized. Overall, Discord provided all the basic features we required from a team communication environment and we plan on keeping the same channel and structure for the development phase of this project as well.

3.1.2. GitHub

The github repository is where all progress made as well as all the deliverables can be viewed. Our repository provides a portfolio of all the work we have done. It includes information about who we are and all the design and research we have done in the wiki section. It includes information about what

should be expected from the project we are developing and also who has done what via the regularly created and maintained issues. Github has been possibly the most important tool during this project for us because it collects everything we have done in one place. Almost all types of work we have done include Github in one way, every task of design, documentation, research, assignment tracking and even reviews include and can be reached from our Github repository. As the project progresses further we will also start using version control aspects of Github such as committing changes and merging branches. Even during the design phase Github was critical, and as we start working on implementation it will be crucial.
We found Github to be useful for this purpose as it contains every feature we needed for such a tracking and management system. Also it has good user interaction for both the contributors and outsiders observing. Alternatives such as BitBucket provide similar features for version management but we find Github is more useful in terms of merging version control and information wiki together in one space.

3.1.3. Google Docs

Generally, for the documentation related tasks of our project we used the Wiki section of our Github Repository. This was useful enough as we could individually work on our parts and easily check change history if needed. Since most of our documentation deliverables were meant to be published on Wiki anyway this made our task easier and we didn't have to include an additional step of migrating. However, when it came to a more detailed work of reporting, such as the Milestone report, Github wiki was not a good enough option. We needed a space where everyone could work simultaneously for longer times without conflicting versions. We found Google Docs to be the most useful tool for such a task. It is easily accessible, we are all familiar with using it, and it provides good collaboration with synced work and commenting features. One disadvantage of Google docs was that with tables and images we had the issue of position stability. As more content was added above the design changed. However, considering 9 people worked on the same document at once it is an understandable issue and did not pose a great problem to us.

3.1.4. Node.js & Express.js

Node.js is an open-source back-end JavaScript runtime environment and Express.js is a free and open source back-end web application framework for Node.js. We chose to use Node.js and Express.js while developing the back-end of the application. The reasons behind this decision are as follows: Node.js and its most popular and easy to use web application framework, Express.js, are so common in the industry. This leads to lots of resources about tutorials, questions, answers, code sections, example implementations and so on. Other than this reason, since we are new to the real-world server implementation area, we had to choose comparably easier technologies while developing the back-end. All team members firstly learned the Javascript syntax to be familiar with, then gained knowledge about Node.js runtime environment and finally had practice on Express.js framework, its routings, controllers and so on. With the help of the modularity of the project setup, we also separated models, controllers, validators, routers and so on. We also benefit from some external packages in required places such as Joi to validate models. All of these many different helpful packages and features are the result of the fact that Node.js & Express.js (so that Javascript) is widespread.

3.1.5. Vue.js

Vue.js is an open-source model–view–viewmodel front-end JavaScript framework for developing user interfaces for web applications. Initially, we decreased our options for the technology we will use while developing the front-end into 2: React.js and Vue.js. After the discussion among the team members, we eliminated React since it is a bit more complex than Vue.js for the ones who are not familiar with it before (even though there are some team members who used React

previously among the team). Even though there is no-one in the team who has experience with Vue.js before, we chose to start a challenge for ourselves and wanted to learn it. Since it is a popular and getting more popular day by day, we thought that learning it will also be beneficial for our future career. We created a main app and routing file in the core of the project. Other than that, we basically created separate screens for each unique feature and created these screens under the '/components' directory. Each component file includes mainly 3 separate parts inside: html, javascript, css (view, script, style). We mostly used html to draw the corresponding components and screens and javascript to communicate with our back-end, store network responses.

3.1.6. Postman

Postman is an API platform for developers to design, build, test and iterate their APIs. Today, the application has more than 20 million registered users and 75000 open APIs. It has many products like API repository and API builder. By unanimous vote, we chose to use Postman to test our APIs, because firstly it is free to use it individually. Another nice feature of it is that it provides users easy documentation. We all prepared our documentations by using that feature. Most obvious reason we chose Postman is that it is the most popular API testing application in today's world. Some of our members already had experiences with Postman, some of us were not very familiar. At the end of our practice-app, we all improved ourselves in using Postman. As I mentioned, we tested our APIs using Postman and shared Postman documentation links in our issues. We also plan to create a wiki page and include those Postman documentation links there. They include examples of failing requests, successful requests and corresponding bodies for those requests. Every request includes an example response and they are very easy to read and understand.

3.1.7. MongoDB

For our practice app to be maintainable, usable we needed a database management system to keep or data in, get or change data when necessary. For that purpose, we had a few options like MySQL, MongoDB, PostgreSQL. Most of our members take CMPE321 course this semester, so we were familiar with MySQL and SQL and some of us were also experienced in MongoDB. We discussed about which system we should use, made a list of pros and cons and decided to go with MongoDB. We chose to go with it because it is one of the most popular, maybe the most popular, database management systems in the world right now, so we thought it could be nice to learn it as soon as possible. Other than that, it has more functionalities than other database management systems, for example we can keep lists as fields in MongoDB. It took a while for many of us to adapt ourselves to using MongoDB but in the end we managed to complete our practice app by using MongoDB as our DBMS. MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of tables and rows, we use documents and collections in MongoDB. Documents consists of key-value pairs and those pairs are basic unit of data in MongoDB. Database contains collections and collections contain documents inside of them. Also, each document can differ in terms of number of fields, even if they are in the same collection. Size and content of documents in same collection can differ. Documents do not have to have a schema beforehand. It also has more complex structures. As summary, it provides more freedom to users

.

3.1.8. Docker

As you can see, we have 3 different services keeping our application running: a Mongo client, an Express server and an HTTP server. For the sake of making our app easy to set-up in any environment without nasty installations, we used docker. By conternirezing these three services, we were able to easily deploy our app to the AWS and different Linux, MacOS, Windows environments(our locals). To conternerize these apps: For mongo we used the official custom

container, for Express and Vue, we followed the tutorials from their documentation. Finally, by creating a docker-compose file, we were able to manage ports and volumes of the containers easily and run our app with a single command!

3.1.9. AWS

Finally, by using the EC2 service of the AWS, we created an instance in Frankfurt. Firstly we installed docker to it using the tutorial from AWS documentation and got our app running. Then we got an elastic IP to reach our deployed application and allocated ports for the API and the HTTP server. One thing I like about port allocation is that mongo client is not open to the world. This provided us with a nice thing: we did not need to set security rules for Mongo. For containers to talk to each other, we have to allow a port to the mongodb service and if we had to open the port to the world, we had to set lots of security rules for accessibility.

3.1.10 Lucidchart

We used Lucidchart to create our system design diagrams: use-case diagram, sequence diagrams and class diagram. From our previous research we found Lucidchart to be the most useful tool for this task, since it was developed specifically for the goal of making the system design process easier. It included all the specific notation elements. Useful features of Lucidchart include many notation elements that are easy to add to your design, simple usage when it comes to making edits and a good collaborative page to work on together. In contrast to Figma, Lucidchart elements were much easier to learn and use and their tutorials were quite helpful so every member of the team was able to create in Lucidchart together. We were happy with the collaborative working system and the outcome of our efforts. Overall, Lucidchart provided all the features we required and we were satisfied.

3.2. Evaluation of Processes

3.2.1. Team Meetings

As we have done through the semester, we followed the same communication plan for the development and planning of the practice project. We as the whole team gather on our Discord channel on a weekly basis to discuss the status of our project and the works that are due that week. Before each meeting we outline the topics we will discuss and form a meeting plan. We usually start with reviewing the previous work done by the meeting time and continue with evaluation of the current status of our project. Generally, a discussion on the weekly tasks follows and we distribute the tasks, which will be done until the next week's meeting. We use Discord's screen sharing feature for a better visualization of our meeting and when it is necessary we collaboratively finish tasks. In each meeting one of our team members takes the meeting notes and uploads them on our GitHub repository later on for everyone to track what has been discussed during the meeting. We decided the path of our project in these team meetings, including which tehcnologies should be used. We also used these team meetings as a peer learning environment and helped each other with the parts we struggle on.

3.2.2. TA Meetings

Once a week, we hold a customer meeting with our teaching assistant. In those meetings, our teaching assistant gives us feedback upon our project's current status. If we have questions about our tasks or things we have done, they answer those questions and make everything as clear as possible for us to move forward. After each customer meeting session, we make necessary changes and update our project according to the feedback given by our teaching assistant.

In the case of practice app, TA meetings were mostly about learning important concepts we will need such as docker and git usage. We didn't get as detailed feedback and instead recieved general tips for moving forward with the project.

3.2.3. Issue Creation and Management

We have prepared three templates for issue creation: generic issue template, group issue template, and research issue template. We mainly used the generic issue template within the context of the practice app.

In each team meeting, we distribute the tasks we will do among the team members and then manage these tasks by creating issues in our project repository in GitHub. According to our templates we provide a title, a detailed and clear description of the issue, step details about the task and final actions to be taken by resolving the issue. There is a deadline for each issue. There is also a reviewer assigned to each issue alongside an additional deadline for the review.

We manage the issues mainly with the labels attached to them. There are priority levels of the issues and also we specify the type of the issue with the labels. We also provide a communication baseline with the labels: When an issue is created we assign a "status-new" tag, then when the assignee starts to work on the issue they change the label to "status-inprogress." After the tasks are finished, the assignee labels the issue as "status-needreview." When the assigned reviewer sees this label, they review the issue and provide feedback if necessary. In order to alert the assignee, the reviewer changes the label into "status-waitingresponse" and assignee make the necessary adjustments. After all is done, we close the issue and label it as "status-completed."

With this structure we are able to ensure each team member is aware of their tasks and the deadlines, alongside a review process for each issue for double check before completion.

3.2.4 Pull Request Creation and Management

As our work got more code heavy pull requests became a part of our project management. In order to keep a systematic structure, with the help of our TA we introduced a branch protection rule for the master branch. With this rule we blocked merging to master without recieving approval from at least one reviewer, which made our code more protected. In our meetings we decided on a naming convention for our branches as follows:
practice-app/[feature/bug/fix]/[details-of-work]/[issue-number]
Creating a new branch for every feature and enhancement helped organise work and avoid conflicts as much as possible. We also connected issues with pull requests to further enhance organisation, but since instructions for the code in pull requests were clear for the instance of this work management was established mostly with code reviews under the pull request, and issues stood secondary in relation. We found the pull request system of Github to be quite helpful for version management and especially the organisation of a project that has 9 contributors working on the same parts.

4. Deliverables

4.1. Project Repository

Our project repository is named "bounswe2022group2" and the content of the code section is as follows:

- a README.md file
- Github Issue Template directory
- deliverables directory
- practice app
- package-lock.json (for practice app)
- .gitignore (for practice app)

Our project repository is up-to-date and by the submission of this report document, there are [8] open issues, [218] closed issues and 77 merged pull requests.

Below a link to our project repository is provided.

https://github.com/bounswe/bounswe2022group2

4.2. Requirements

- **4.2.2.1. Functional Requirements**
  - **4.2.2.1.1. User Requirements**
    - **4.2.2.1.1.1. Authentication**
      - **4.2.2.1.1.1.1. Signup**
        - 4.2.2.1.1.1.1.1. Guests shall enter their name, email address and passwords to be able to sign up. These are the required information that users have to provide to enter the app.
      - **4.2.2.1.1.1.1. Login**
        - 4.2.2.1.1.1.1.1. Users shall provide their username and passwords to be able to login if they logged out or they are kicked out due to long inactivity.
    - **4.2.2.1.1.2. User Interaction**
      - **4.2.2.1.1.2.1. User - Lesson Interaction**
        - **4.2.2.1.1.2.1.1 Creating Lessons**
          - 4.2.2.1.1.2.1.1.1 Users shall be able to create lessons on the create lesson page by entering a lesson name and an existing category name.
        - **4.2.2.1.1.2.1.2. Rating Lessons**
          - 4.2.2.1.1.2.1.2.1 Users are able to rate a lesson from a rating page using the lessons name and the rating value.
      - **4.2.2.1.1.2.2. User - Event Interaction**
        - **4.2.2.1.1.2.1.1. Creating Events**
          - Users shall be able to create events by providing a title, description, a valid

date, a valid location, and a valid lesson ID.

- **4.2.2.1.1.2.1.2. Viewing Events**
  - Users shall be able to view events they have attended.
- **4.2.2.1.1.2.1.3. Attending Events**
  - Users shall be able to attend events by providing the event id.
- **4.2.2.1.1.2.1.4. Getting Specific Event Details**
  - Users shall be able to see the specific event details, such as “title”, “location”, “host id”, “lesson id”, and “date”, by providing the event id.

- **4.2.2.1.2. System Requirements**
  - **4.2.2.1.2.1. Lessons**
    - **4.2.2.1.2.1.1. Searching and Browsing**
      - 4.2.2.1.2.1.1.1. The system shall allow to filter the lessons according to their categories.
      - 4.2.2.1.2.1.1.2. The system shall allow to filter the lessons according to their ratings. Names of the filtered lessons and the ratings will be shown.
      - 4.2.2.1.2.1.1.3 The system shall allow to filter the lessons according to their lecturer.

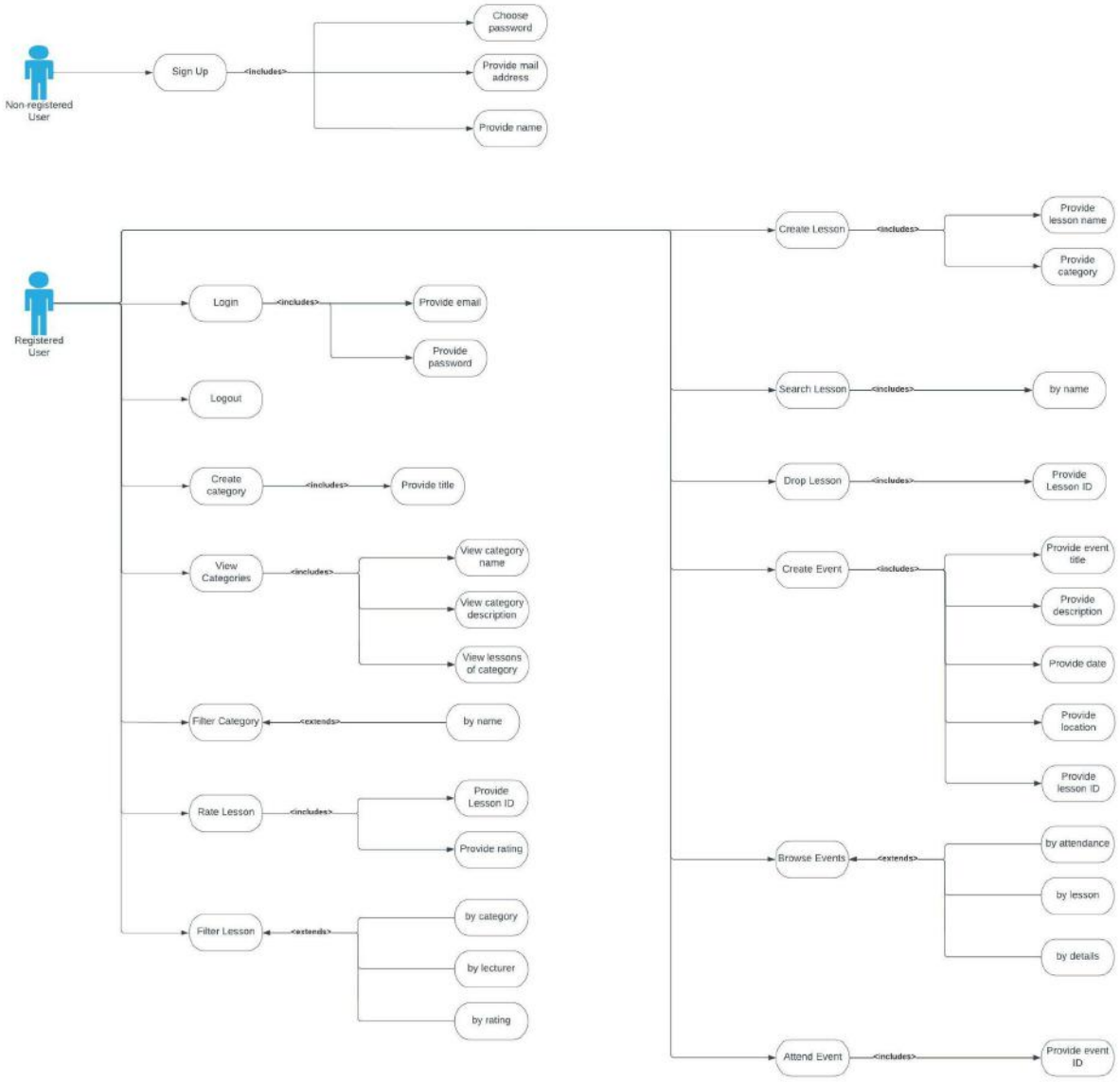
- **4.2.2.2. Non-Functional Requirements**
  - **4.2.2.2.1 Usability**
    - 4.2.2.2.1.1 The system should provide a user-friendly interface. It should contain a basic site map and should not contain a confusing linking system.
    - 4.2.2.2.1.2 The system should provide a direct feedback mechanism. The moment people interact with the system, we should offer an indication of the success or failure of their actions.
    - 4.2.2.2.1.3 The system should have a well-chosen typeface that should be readable and clean without diverting too much attention from the rest of the design.

System shall provide a description for the categories fetched from the Wiki API

4.3. Software Design Documents in UML

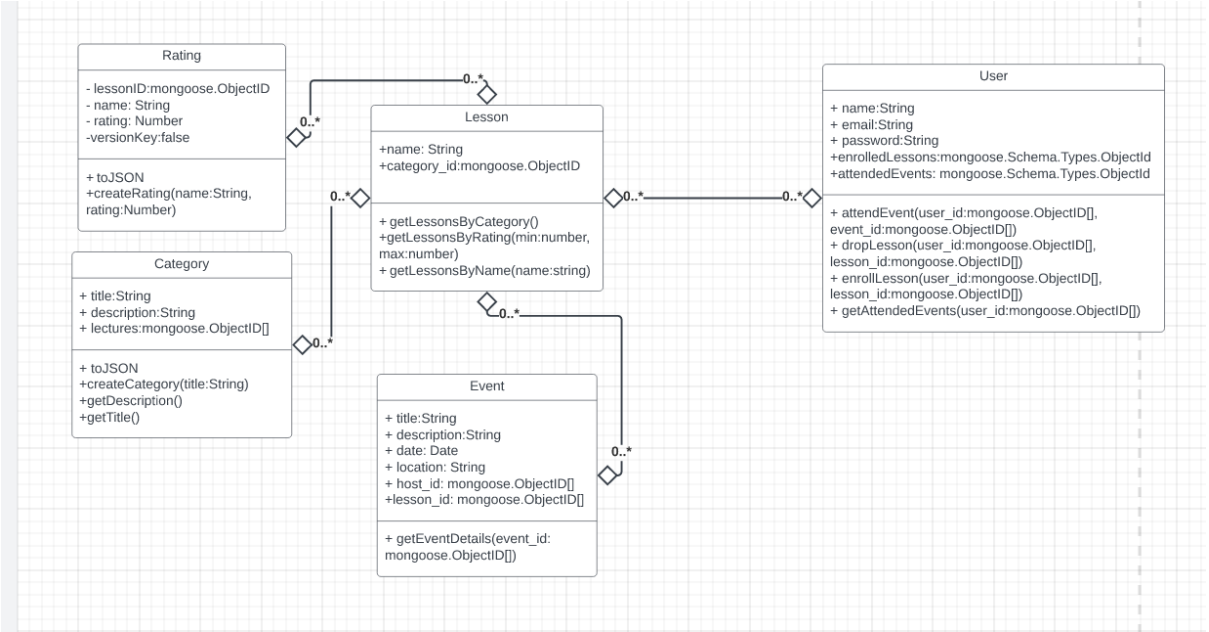
4.3.1. Use-Case Diagrams

Our use case diagram can be viewed from this link.



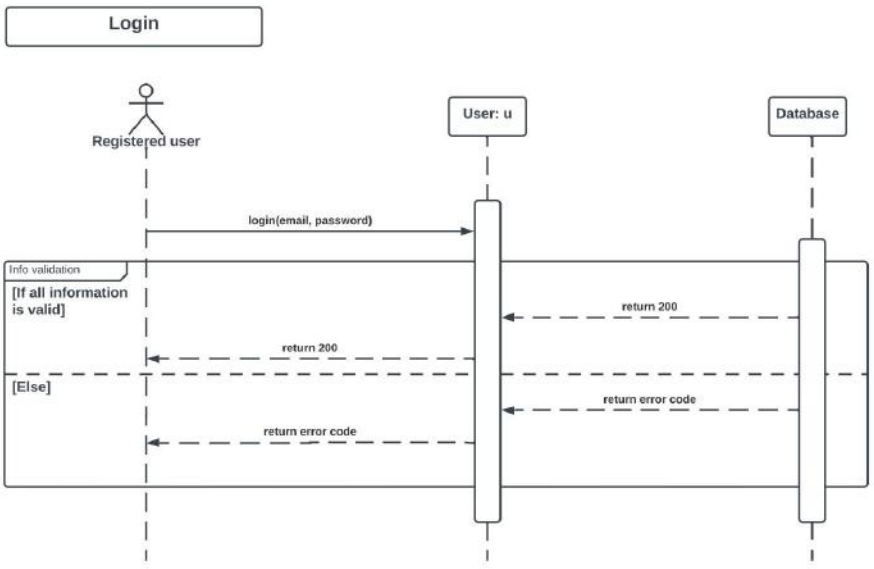
4.3.2. Class Diagrams

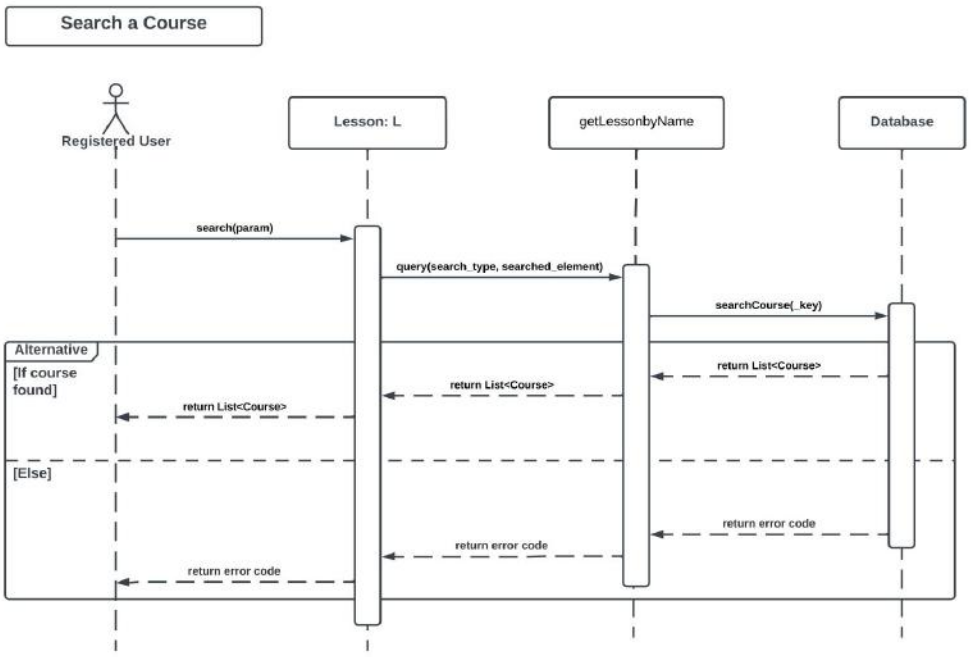
Our class diagram for practice app can be reached from this link.

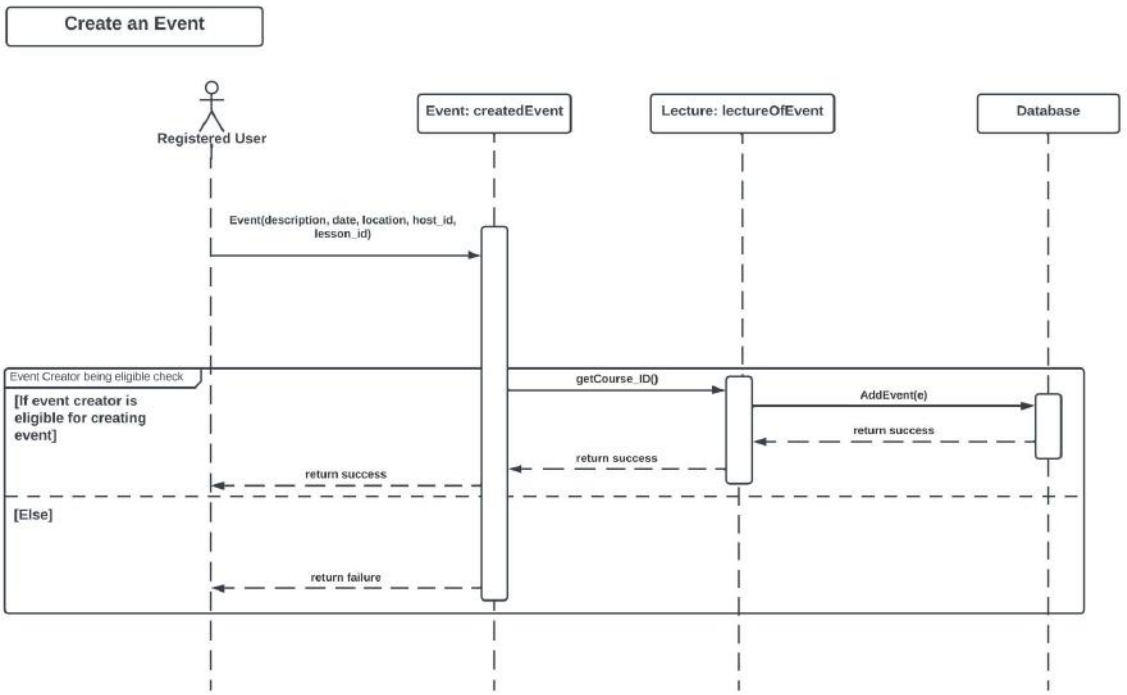


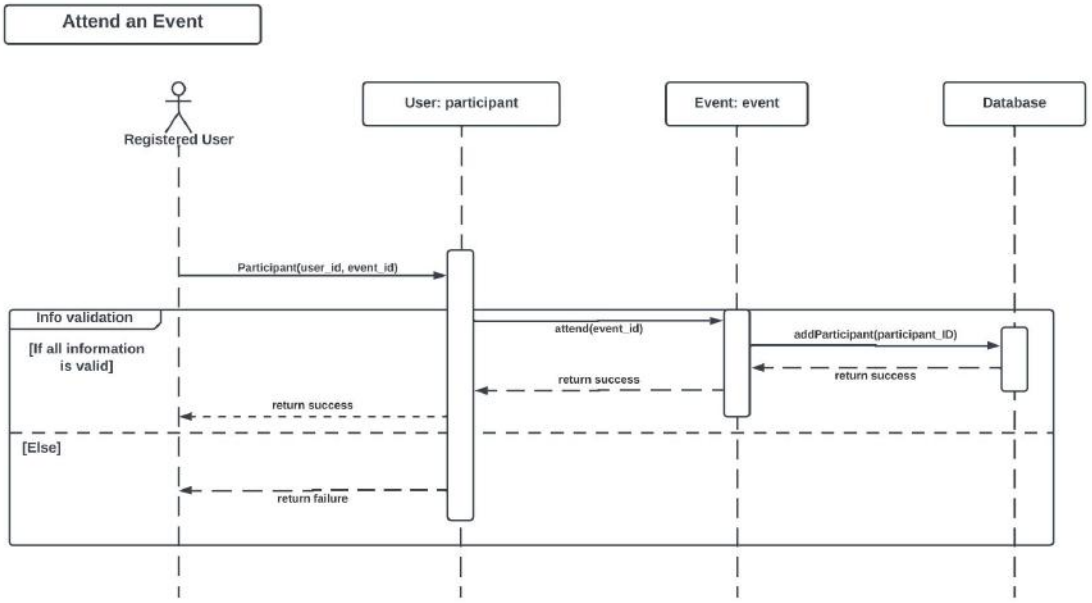
4.3.3. Sequence Diagrams

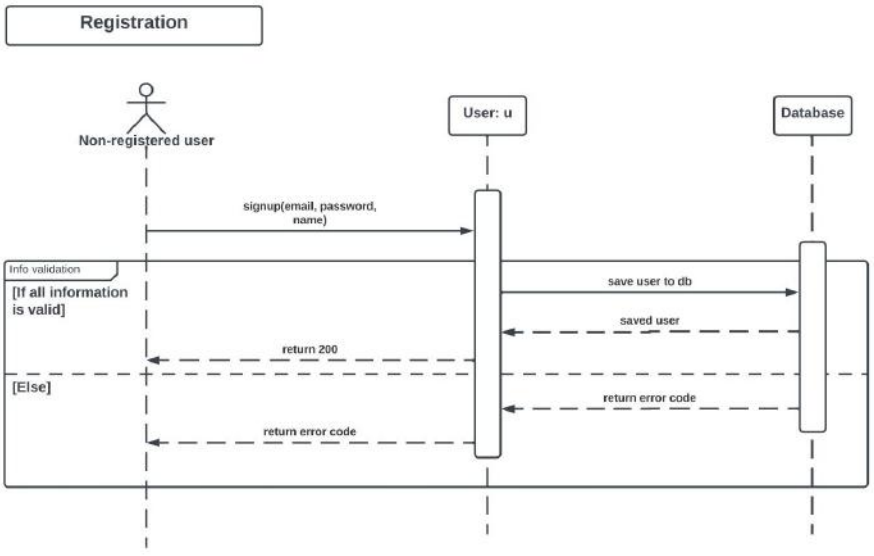
Our sequence diagrams for practice app can be reached from this link.




Search a Course
Registered User
Lesson: L
getLessonbyName
Database
search(param)
query(search_type, searched_element)
searchCourse(_key)
Alternative
[If course found]
return List<Course>
return List<Course>
return List<Course>
[Else]
return error code
return error code
return error code


Create an Event
Registered User
Event: createdEvent
Lecture: lectureOfEvent
Database
Event(description, date, location, host_id, lesson_id)
Event Creator being eligible check
[If event creator is eligible for creating event]
getCourse_ID()
AddEvent(e)
return success
return success
return success
[Else]
return failure


Attend an Event
Registered User
User: participant
Event: event
Database
Participant(user_id, event_id)
Info validation
[If all information is valid]
attend(event_id)
addParticipant(participant_ID)
return success
return success
return success
[Else]
return failure


Registration
Non-registered user
User: u
Database
signup(email, password, name)
Info validation
[If all information is valid]
save user to db
saved user
return 200
[Else]
return error code
return error code

4.4. URI Tag of the project

Tag of the project

5. URI

Tag URI:
https://github.com/bounswe/bounswe2022group2/releases/tag/Group-2-Practice-App-Deliverable

API URI - API Documentation:
https://github.com/bounswe/bounswe2022group2/wiki/Practice-App-API-Documentation

6. How to run the project?

You can initialize the project by running **docker-compose up** in the practice app folder, but firstly you need to add an API key for Create Event to the .env.production file under the server folder.

6 Project Plan

		Ad	Süre	Balat	Bitirme
1		**Week 1**	**7,203 günler**	**01.03.2022 12:00**	**08.03.2022 17:00**
2		Infrastructure Setup	7,203 günler	01.03.2022 12:00	08.03.2022 17:00
3		Create Discord channel	0,375 günler	04.03.2022 08:00	04.03.2022 17:00
4		Add new labels	0,208 günler	04.03.2022 20:00	05.03.2022 01:00
5		Create a template for personal wiki pages	0,874 günler	04.03.2022 20:00	05.03.2022 17:00
6		Create personal wiki pages	2,998 günler	05.03.2022 17:00	08.03.2022 17:00
7		Create a template for meeting notes	1,874 günler	04.03.2022 20:00	06.03.2022 17:00
8		Modify Readme File	2,29 günler	05.03.2022 10:00	07.03.2022 17:00
9		Create Home Wiki Page and Linking	0,292 günler	05.03.2022 10:00	05.03.2022 17:00
10		Document Communication Plan	0,292 günler	07.03.2022 10:00	07.03.2022 17:00
11		Study Git&Github	1,874 günler	04.03.2022 20:00	06.03.2022 17:00
12		Document Git History	0,292 günler	07.03.2022 10:00	07.03.2022 17:00
13		Document Git Summary	0,292 günler	07.03.2022 10:00	07.03.2022 17:00
14		Document Useful Git Commands	0,292 günler	07.03.2022 10:00	07.03.2022 17:00
15		Research about repositories	1,874 günler	04.03.2022 20:00	06.03.2022 17:00
16		Document favorite repositories	0,292 günler	07.03.2022 10:00	07.03.2022 17:00
17		Document Meeting#1 Notes	2,873 günler	04.03.2022 20:00	07.03.2022 17:00
18		**Week 2**	**8,453 günler**	**08.03.2022 10:00**	**16.03.2022 21:00**
19		Project Assignment	1,291 günler	08.03.2022 10:00	09.03.2022 17:00
20		Requirements	4,872 günler	10.03.2022 20:00	15.03.2022 17:00
21		Research on W3 Web Annotation Model	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
22		Document W3 Web Annotation Model Summary	0,874 günler	13.03.2022 20:00	14.03.2022 17:00
23		Research on Semantic Search	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
24		Document Semantic Search	0,874 günler	13.03.2022 20:00	14.03.2022 17:00
25		Research on Rival Companies	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
26		Document Rival Company Analysis	0,874 günler	13.03.2022 20:00	14.03.2022 17:00
27		User requirements of Authentatication	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
28		User requirements of Profile Page	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
29		User requirements of User Interaction	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
30		System requirements of Searching and Browsing	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
31		System requirements of Recommendation	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
32		System requirements of Notification	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
33		Non-functional requirements of Security, Performance and Reliability	1,79 günler	11.03.2022 22:00	13.03.2022 17:00

CMPE352_Group2_ProjectPlan- Sayfa1

Önceki	Kaynak Adlar
	Onur Kömürcü
	Altay Acar
	Bahrican Yeil;Mehmet Batu...
5	Everyone
	Ezgi Aysel Bat
	Egemen Atik
	Bahrican Yeil
	Onur Kömürcü
	Everyone
11	Onur Kömürcü
11	Ecenur Sezer;Egemen Atik
11	Muhammed Enes Sürmeli
	Everyone
15	Hasan Can Erol
	Ezgi Aysel Bat
19	
	Everyone
21	Altay Acar
	Everyone
23	Ezgi Aysel Bat
	Everyone
25	Muhammed Enes Sürmeli
	Bahrican Yeil
	Altay Acar
	Mehmet Batuhan Çelik
	Ezgi Aysel Bat
	Ecenur Sezer
	Muhammed Enes Sürmeli
	Onur Kömürcü

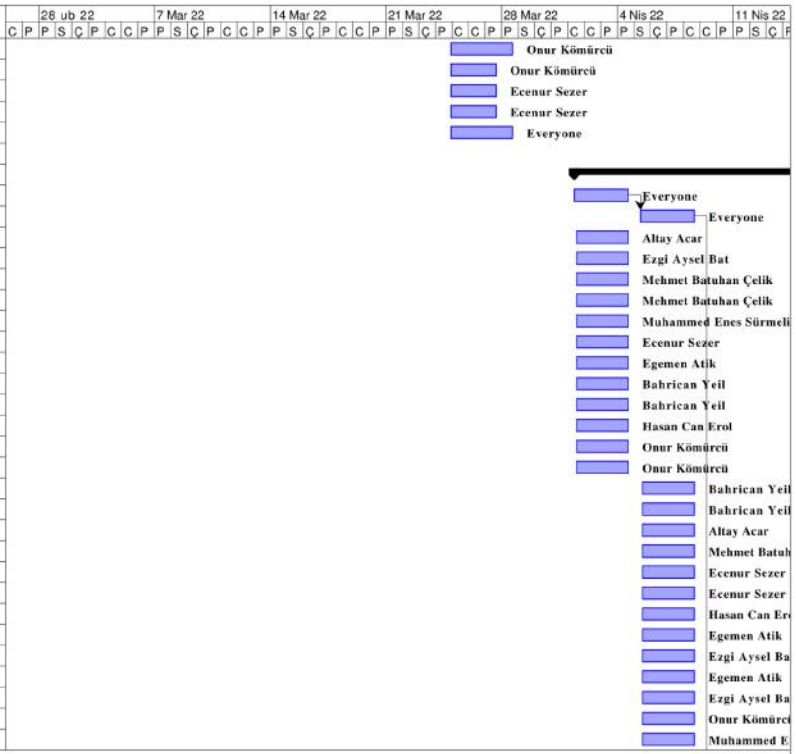
CMPE352_Group2_ProjectPlan- Sayfa2

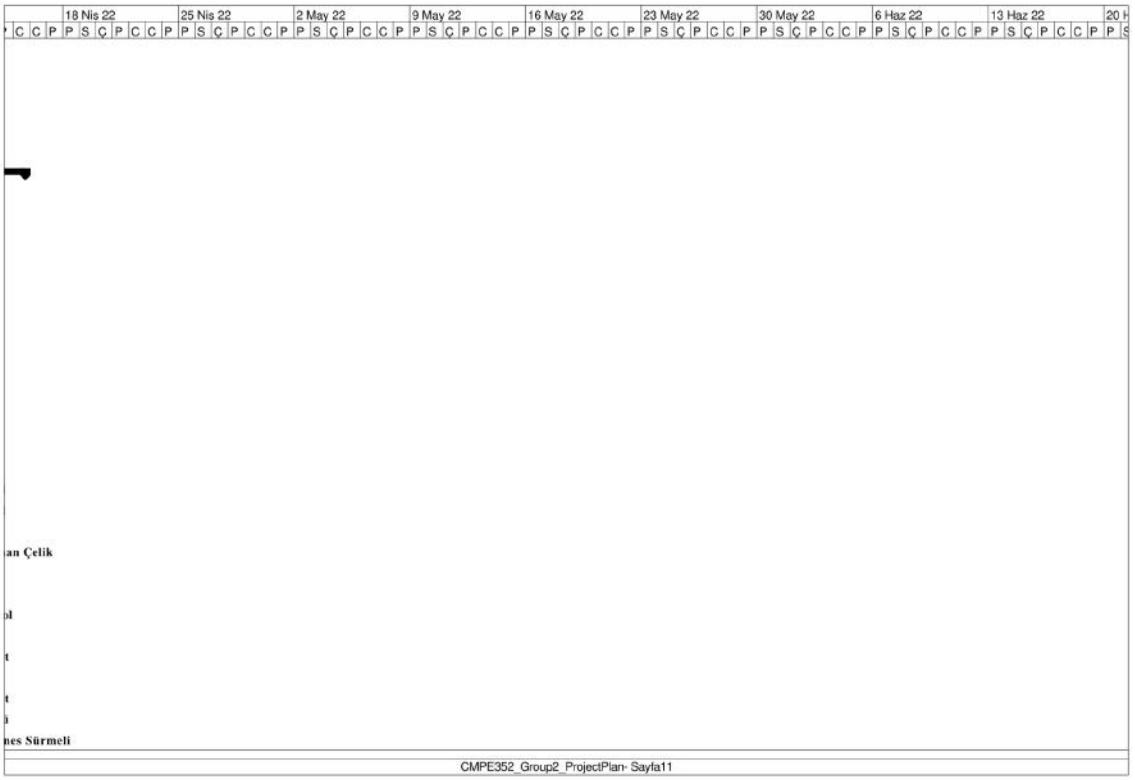
		Ad	Süre	Balat	Bitirme
34		Non-functional requirements of Usability and Disaster Recovery	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
35		Non-functional requirements of Availability and Accessibility, Privacy	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
36		Document Meeting#2 Notes	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
37		Create Glossary	1,79 günler	11.03.2022 22:00	13.03.2022 17:00
38		Reviewing Requirements	0,874 günler	13.03.2022 20:00	14.03.2022 17:00
39		Formulate Questions to Client	2,873 günler	11.03.2022 20:00	14.03.2022 17:00
40		Customer Meeting	0,167 günler	16.03.2022 17:00	16.03.2022 21:00
41		**Week #3**	**8,203 günler**	**15.03.2022 12:00**	**23.03.2022 17:00**
42		Scenarios and Mockups	8,203 günler	15.03.2022 12:00	23.03.2022 17:00
43		Create Issue Templates	0,791 günler	17.03.2022 22:00	18.03.2022 17:00
44		Determine Lecture Requirements	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
45		Document Meeting #3 Notes	2,79 günler	17.03.2022 22:00	20.03.2022 17:00
46		Create Community Event Requirements	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
47		Revise Glossary	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
48		Determine the Persona, User Story, Goals and Pre-conditions for Registered L...	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
49		Create a template for mockups	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
50		Decide Scenario and Acceptance Criteria for Registered Lecturer Scenario	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
51		Determine Persona and User Story for the Non-registered Learner Scenario	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
52		Determine Goals and Pre-conditions for the Non-registered Learner Scenario	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
53		Determine Scenario and Acceptance Criteria for Non-registered Learner Scenari	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
54		Create Creating Community Event Scenario	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
55		Providing mockups for non-registered user scenario, homepage, profile page	4,788 günler	17.03.2022 22:00	22.03.2022 17:00
56		Determine Chat Requirements	3,789 günler	17.03.2022 22:00	21.03.2022 17:00
57		Provide Mockups for Creating Community Event Scenario	4,788 günler	17.03.2022 22:00	22.03.2022 17:00
58		Review of weekly work	5,787 günler	17.03.2022 22:00	23.03.2022 17:00
59		**Week #4**	**3,789 günler**	**24.03.2022 22:00**	**28.03.2022 17:00**
60		Revise Privacy Requirements and check correct usage of words	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
61		Revise Authentication Requirements	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
62		Create a Meeting Schedule	1,79 günler	24.03.2022 22:00	26.03.2022 17:00
63		Create the initial version of User and Token classes	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
64		Revise Lecture Requirements	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
65		Revise User Interaction Requirements	2,79 günler	24.03.2022 22:00	27.03.2022 17:00

Önceki	Kaynak Adlar
	Hasan Can Erol
	Egemen Atik
	Bahrican Yeil
	Egemen Atik
	Everyone
	Everyone
	Everyone
	Bahrican Yeil
	Mehmet Batuhan Çelik
	Egemen Atik
	Ecenur Sezer
	Ezgi Aysel Bat;Onur Kömürcü
	Bahrican Yeil
	Altay Acar;Hasan Can Erol
	Muhammed Enes Sürmeli
	Mehmet Batuhan Çelik
	Egemen Atik
	Onur Kömürcü
	Ecenur Sezer;Ezgi Aysel Bat
	Altay Acar
	Hasan Can Erol
	Hasan Can Erol
	Everyone
	Egemen Atik
	Bahrican Yeil
	Bahrican Yeil
	Bahrican Yeil
	Mehmet Batuhan Çelik
	Mehmet Batuhan Çelik

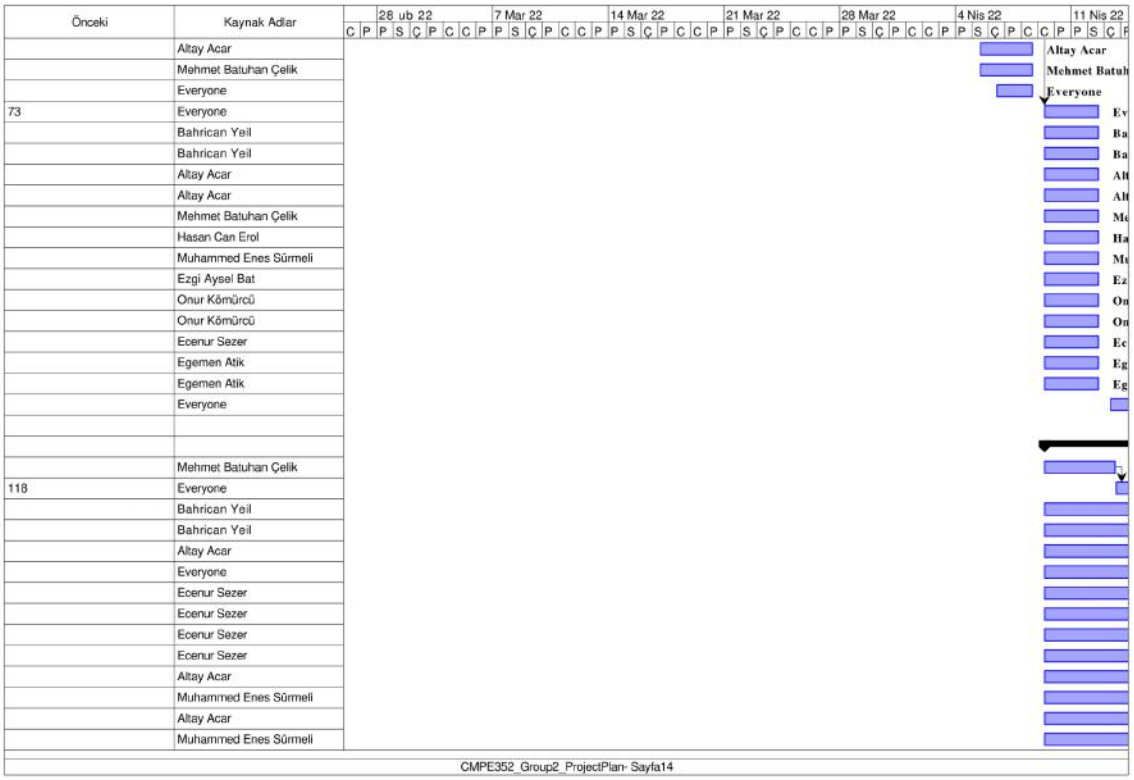
		Ad	Süre	Balat	Bitirme
66		Document Meeting #4 Notes	3,789 günler	24.03.2022 22:00	28.03.2022 17:00
67		Determine Admin Requirements	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
68		Revise Community Event Requirements	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
69		Add a new requirement to User Profile Page	2,79 günler	24.03.2022 22:00	27.03.2022 17:00
70		Review of weekly work	3,789 günler	24.03.2022 22:00	28.03.2022 17:00
71		**Week #5 & #6**	**14,282 gün...**	**01.04.2022 10:00**	**15.04.2022 17:00**
72		Create Use Case Diagrams	3,29 günler	01.04.2022 10:00	04.04.2022 17:00
73		Create Class Diagrams	3,29 günler	05.04.2022 10:00	08.04.2022 17:00
74		Use Case Diagram: Course Creation	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
75		Use Case Diagram: Admin Operations	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
76		Use Case Diagram: Signing and logging in	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
77		Use Case Diagram: Authentication	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
78		Use Case Diagram: View profile page	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
79		Use Case Diagram: Searching-Browsing	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
80		Use Case Diagram: Chat	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
81		Use Case Diagram: Note actions	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
82		Use Case Diagram: Review a course	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
83		Use Case Diagram: Annotation	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
84		Use Case Diagram: Attending and viewing events	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
85		Use Case Diagram: View Content	3,206 günler	01.04.2022 12:00	04.04.2022 17:00
86		Class Diagram: User Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
87		Class Diagram: Token Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
88		Class Diagram: Annotation Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
89		Class Diagram: Course Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
90		Class Diagram: Message Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
91		Class Diagram: Profile Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
92		Class Diagram: App Info Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
93		Class Diagram: Event Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
94		Class Diagram: Note Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
95		Class Diagram: Participant Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
96		Class Diagram: Tag Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
97		Class Diagram: Lecture Admin Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
98		Class Diagram: Enrollment Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00

Önceki	Kaynak Adlar
	Onur Kömürcü
	Onur Kömürcü
	Ecenur Sezer
	Ecenur Sezer
	Everyone
	Everyone
72	Everyone
	Altay Acar
	Ezgi Aysel Bat
	Mehmet Batuhan Çelik
	Mehmet Batuhan Çelik
	Muhammed Enes Sürmeli
	Ecenur Sezer
	Egemen Atik
	Bahrican Yeil
	Bahrican Yeil
	Hasan Can Erol
	Onur Kömürcü
	Onur Kömürcü
	Bahrican Yeil
	Bahrican Yeil
	Altay Acar
	Mehmet Batuhan Çelik
	Ecenur Sezer
	Ecenur Sezer
	Hasan Can Erol
	Egemen Atik
	Ezgi Aysel Bat
	Egemen Atik
	Ezgi Aysel Bat
	Onur Kömürcü
	Muhammed Enes Sürmeli



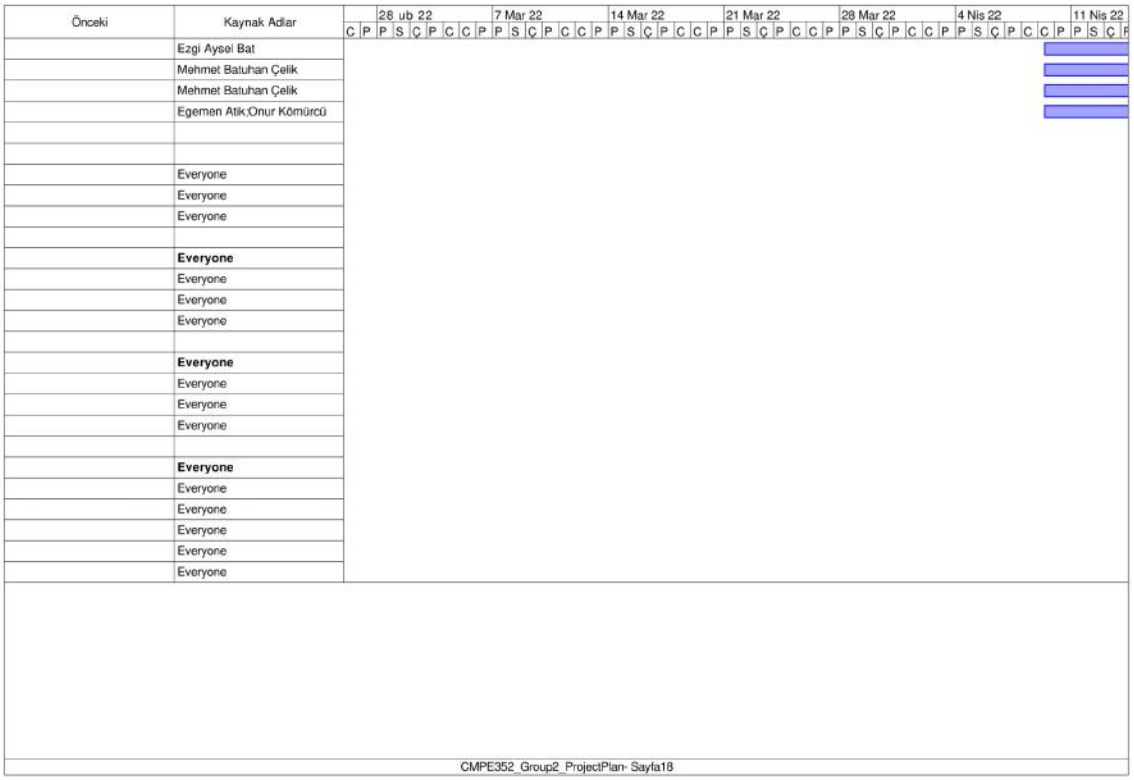

18 Nis 22
25 Nis 22
2 May 22
9 May 22
16 May 22
23 May 22
30 May 22
6 Haz 22
13 Haz 22
an Çelik
nes Sürmeli
CMPE352_Group2_ProjectPlan- Sayfa11

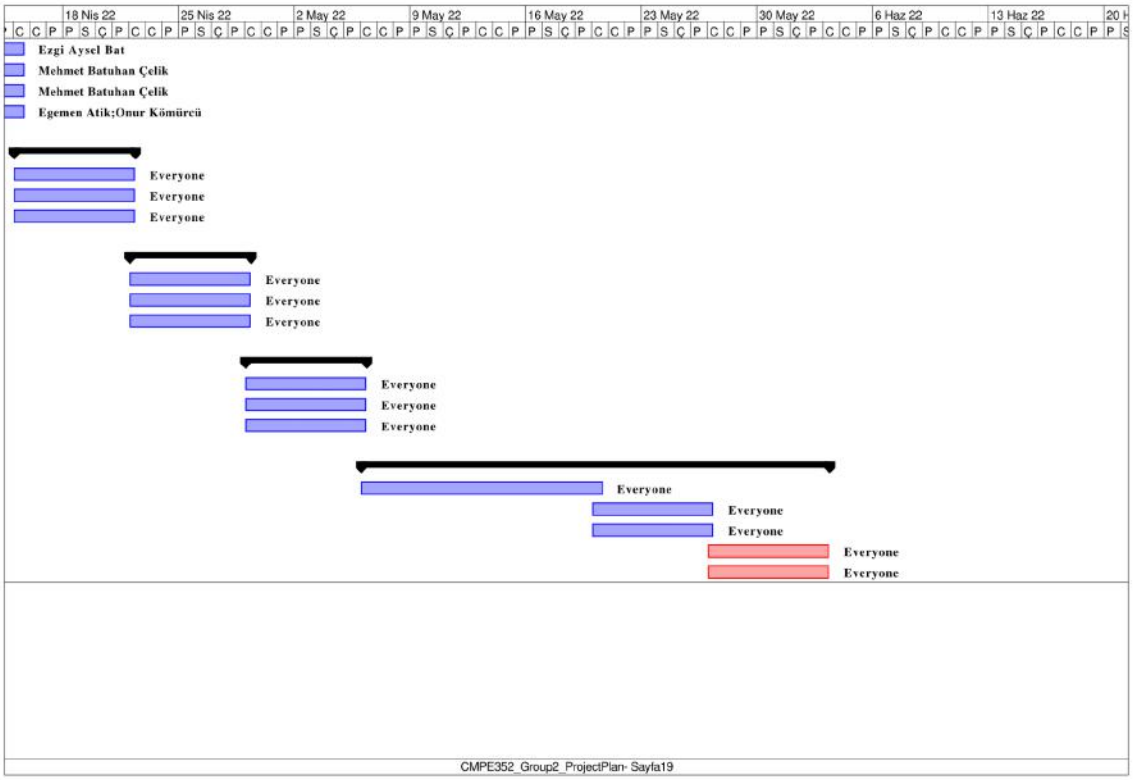
		Ad	Süre	Balat	Bitirme
99		Class Diagram: Search Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
100		Class Diagram: Chat Class	3,206 günler	05.04.2022 12:00	08.04.2022 17:00
101		Revise Use Case and Class Diagrams according to feedback	2,207 günler	06.04.2022 12:00	08.04.2022 17:00
102		Create Sequence Diagrams	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
103		Sequence Diagram: Registration	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
104		Sequence Diagram: Login	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
105		Sequence Diagram: Search User	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
106		Sequence Diagram: Search Course	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
107		Sequence Diagram: Send Message	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
108		Sequence Diagram: Create Course	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
109		Sequence Diagram:Take Course	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
110		Sequence Diagram Take Note	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
111		Sequence Diagram: Edit Course	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
112		Sequence Diagram: Annotate	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
113		Sequence Diagram: Edit Profile	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
114		Sequence Diagram: Create Event	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
115		Sequence Diagram: Join Event	3,29 günler	09.04.2022 10:00	12.04.2022 17:00
116		Revise Sequence Diagrams according to feedback	2,29 günler	13.04.2022 10:00	15.04.2022 17:00
117		**Week #7**	**6,287 günler**	**09.04.2022 10:00**	**15.04.2022 17:00**
118		Milestone-1: Creating Responsibility Assignment Matrix	4,289 günler	09.04.2022 10:00	13.04.2022 17:00
119		Milestone-1: Filling RAM	1,999 günler	13.04.2022 17:00	15.04.2022 17:00
120		Milestone-1: Creating the Introduction & Project Description Part	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
121		Milestone-1: Writing the Forward Plan Part	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
122		Milestone-1: Creating the Online Document with Title page and Table of Conte..	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
123		Milestone-1: Filling the Table of Work	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
124		Milestone-1: Adding Communication Plan to Deliverables	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
125		Milestone-1: Writing Evaluation of Communication Plan	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
126		Milestone-1: Adding Scenarios and Mockups to Deliverables	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
127		Milestone-1: Writing Evaluation of Scenarios and Mockups	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
128		Milestone-1: Writing Evaluation of Processes	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
129		Milestone-1: Writing Evaluation of Software Design Documents in UML	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
130		Milestone-1: Adding Project Repository to Deliverables	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
131		Milestone-1: Filling the Table of List and Status of Deliverables	6,287 günler	09.04.2022 10:00	15.04.2022 17:00


Önceki
Kaynak Adlar
28 ub 22
7 Mar 22
14 Mar 22
21 Mar 22
28 Mar 22
4 Nis 22
11 Nis 22
Altay Acar
Mehmet Batuhan Çelik
Everyone
73
Everyone
Bahrican Yeil
Bahrican Yeil
Altay Acar
Altay Acar
Mehmet Batuhan Çelik
Hasan Can Erol
Muhammed Enes Sürmeli
Ezgi Aysel Bat
Onur Kömürcü
Onur Kömürcü
Ecenur Sezer
Egemen Atik
Egemen Atik
Everyone
Mehmet Batuhan Çelik
118
Everyone
Bahrican Yeil
Bahrican Yeil
Altay Acar
Everyone
Ecenur Sezer
Ecenur Sezer
Ecenur Sezer
Ecenur Sezer
Altay Acar
Muhammed Enes Sürmeli
Altay Acar
Muhammed Enes Sürmeli
CMPE352_Group2_ProjectPlan- Sayfa14


18 Nis 22
25 Nis 22
2 May 22
9 May 22
16 May 22
23 May 22
30 May 22
6 Haz 22
13 Haz 22
Everyone
Mehmet Batuhan Çelik
Everyone
Bahrican Yeil
Bahrican Yeil
Altay Acar
Everyone
Ecenur Sezer
Ecenur Sezer
Ecenur Sezer
Ecenur Sezer
Altay Acar
Muhammed Enes Sürmeli
Altay Acar
Muhammed Enes Sürmeli
CMPE352_Group2_ProjectPlan- Sayfa15

		Ad	Süre	Balat	Bitirme
132		Milestone-1: Writing Evaluation of Tools	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
133		Milestone-1: Adding Requirements to Deliverables	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
134		Milestone-1: Writing Evaluation of Requirements	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
135		Milestone-1: Creating Project Plan	6,287 günler	09.04.2022 10:00	15.04.2022 17:00
136		**Using API' s**	**7,37 günler**	**15.04.2022 01:00**	**22.04.2022 10:00**
137		API for Backend	7,37 günler	15.04.2022 01:00	22.04.2022 10:00
138		API for Frontend	7,37 günler	15.04.2022 01:00	22.04.2022 10:00
139		API for Mobile	7,37 günler	15.04.2022 01:00	22.04.2022 10:00
140		**Determining Teams**	**7,37 günler**	**22.04.2022 01:00**	**29.04.2022 10:00**
141		Backend Team	7,37 günler	22.04.2022 01:00	29.04.2022 10:00
142		Frontend Team	7,37 günler	22.04.2022 01:00	29.04.2022 10:00
143		Mobile Team	7,37 günler	22.04.2022 01:00	29.04.2022 10:00
144		**Reviewing Project**	**7,37 günler**	**29.04.2022 01:00**	**06.05.2022 10:00**
145		Reviewing Requirements	7,37 günler	29.04.2022 01:00	06.05.2022 10:00
146		Reviewing Scenarios & Mockups	7,37 günler	29.04.2022 01:00	06.05.2022 10:00
147		Reviewing Diagrams	7,37 günler	29.04.2022 01:00	06.05.2022 10:00
148		**Implementation & Testing**	**28,356 gün...**	**06.05.2022 01:00**	**03.06.2022 10:00**
149		Implementation	14,657 günler	06.05.2022 01:00	20.05.2022 17:00
150		Testing	7,37 günler	20.05.2022 01:00	27.05.2022 10:00
151		Debugging	7,37 günler	20.05.2022 01:00	27.05.2022 10:00
152		Deployment	7,37 günler	27.05.2022 01:00	03.06.2022 10:00
153		Final Report	7,37 günler	27.05.2022 01:00	03.06.2022 10:00


Önceki
Kaynak Adlar
28 ub 22
7 Mar 22
14 Mar 22
21 Mar 22
28 Mar 22
4 Nis 22
11 Nis 22
Ezgi Aysel Bat
Mehmet Batuhan Çelik
Mehmet Batuhan Çelik
Egemen Atik;Onur Kömürcü
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
CMPE352_Group2_ProjectPlan- Sayfa18


18 Nis 22
25 Nis 22
2 May 22
9 May 22
16 May 22
23 May 22
30 May 22
6 Haz 22
13 Haz 22
Ezgi Aysel Bat
Mehmet Batuhan Çelik
Mehmet Batuhan Çelik
Egemen Atik;Onur Kömürcü
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
Everyone
CMPE352_Group2_ProjectPlan- Sayfa19