FOOTBALL MATCH APPOINTMENT PROJECT

https://football-appointment.herokuapp.com/

ERDİ BAYIR (Product Owner)

MUHAMMED GEÇGELOĞLU (Product Manager)

ÖMER FARUK AKDUMAN

BERK YİĞİT DURSUN

MUSA KARAŞ

WHAT IS THE FMA?

- Our aim in this project is a web application application for a football match appointment system.

- The main purpose of the web application is for those who want to organize a football match to examine the football match pitches close to them and make an appointment, and to make such organizations much easier by adding their own businesses to this application.

- In the development phase;

    - the players find a team by logging into the application, and the teams find their opponents.

    - Player scoring system

    - Organizing tournaments with application.

USER STORIES

First User Story

Emre lives in Istanbul and owns a football field. Emre loves his job very much, but thinks that he is not able to show himself in this business because he thinks that there are too many football field owners in the city and the number of people he can reach. For this reason, Emre asks her colleagues if there is an online application where he can introduce her field and people can make an appointment from his field so he thinks he can promote his site and find more customers.

One of his colleagues suggested the FMA web application. FMA(Football Match Appointment App) is a web application where people can make an appointment at football fields by signing up, and at the same time, football field owners can promote their fields by uploading their pitches and can very easily fill the appointment hours of their pitches and earn money. By using this web application, He can promote her field and come to much better points in her work.

Second User Story

İsmail lives in Kocaeli and he very likes football. Especially he loves play football match with friends in area.But for the football area match, it was very difficult for him to find out whether the appropriate hours for them were available by contacting the match area field officials one by one by phone or personally.He was thinking and sighing if I could reach the available hours of all the fields at once, all these works online. Thus, the football matches could easily handle the difficult part of organizing a football match.

Then one day he encountered the FMA web application on the internet.Their slogans and service idea caught his attention.After browsing the site for a while, he was delighted to see that it was a site made to solve the ordeal of getting an appointment and that exactly what he wanted was realized.

He immediately had the luxury of setting up a team with his friends and being able to easily handle the hardest part of the job, making appointments.

He talked about going to the match area field owners in and around and talking about that it would be a perfect thing for both parties to join this site.It also looks forward to features such as participating in tournaments with your team coming soon to the site.

Third User Story

Ahmet lives in Ankara and is a football lover. Playing rugs on the field is his greatest pleasure, but at the same time arranging a football field is always a task and a process like torture. Because, in addition to setting the time zone according to the available times of everyone, it was a separate pain for him to search and find the available times on the fields.

Then one day he encountered the FMA web application on the internet, when he saw the service they offered, he realized that it was the application he needed.

He immediately got the luxury of teaming up with his friends and easily tackling the hardest part of the job and making an appointment.

He talked about going to the field owners inside and around the field and it would be a great thing for both parties to join this site.

USER SCENERIOS

USER SCENERIO 1

Title: Book An Appointment

Initial assumption: As a user, I want to book an appointment from the application. When I register in the application and log in, I want to go to the appointment page and make an appointment according to the choice of day, month, hour and city.

Normal: The user must first sign up with the application. After sign up, he / she has to log in. Then, go to the Appointment page to see the information of the fields and select the field where they want to make an appointment. After selecting the site, the hours when the field is full and empty will appear on the Book Appointment page. After completing the payment process, the appointment process is completed.The user must first register with the application. After registering, he / she has to log in. Then, go to the Appointment page to see the information of the fields and select the field where they want to make an appointment. After selecting the site, the hours when the field is full and empty will appear on the Book Appointment page. After completing the payment process, the appointment process is completed.

System state on completion: User logged on. Time selected according to the information from the system database data was filled. Information was forwarded to the owner of the football field.

USER SCENERIOS 2

Title:Add Photo For User Profile

Initial assumption: As a user, I would like to upload a profile photo for my profile page.

Normal: The user must first sign up with the application. After sign up, he / she has to log in. After logging in, the user can go to the profile page and click the change photo button, select the photo they want from their local files and upload them.

System state on completion: After the user uploads the photo, the path of the photo is added to the database.

USER SCENERIO 3

Title:Add Football Areas to Web App

Initial assumption: As a football pitch owner, I want to upload my football pitch so users can view pitch and time information and make an appointment.

Normal: The user must first sign up for owner with the application. After sign up, he / she has to log in. Then he/see has to go to the profile page and click on the Add Area button. After clicking this button, the page to add football field will appear. On this page, there are information that should be entered such as the field address and phone number. After filling these correctly, the field will be added to the application.

System state on completion: The football field added by the user is added to the database and starts to appear in the field list on the Appointment page.

USER SCENERIO 4

Title:Add Comment For Football Areas

Initial assumption: As a user, I would like to comment on the football field I want according to my experience. In this way, those who want to make an appointment on that field can get an idea by looking at the comments.

Normal: The user must first sign up with the application. After sign up, he / she has to log in. After the user logs in, he/she can make an appointment from the field they want and after making an appointment, they go and experience the field. After they experience, they can go to the book appointment page and write their comments in the comment section.

System state on completion:After the user comments, the comment is added to the database and can now be seen by other users on the book appointment page.

USER SCENERIOS 5

Title: Sign Up

Inıtial assumption: As a user and site visitor , I want to signup with my email.I want to go to sign up page for registration , after go to sign in page for login.

Normal:The user must first sign up for sign in the application. To login , he/she must go to signup page for registration.After entiring the sign up page information forms will appear.He/she must fill in information such as name, surname, username, email, password with correct format and accept the terms.The system will not accept it if it is on another username so he/she must enter a unique username. Person to register must choose how to register , user or owner .

System state on completion:The system checks the entered information to see if someone else has the username or if the password is suitable for the format, and if it is suitable, it saves it in the database, if not, it asks us to enter it again.

USER SCENERIOS 6

Title: Search Football Pitches

Initial assumption: As a user , I want to search a specific football pitch. When I login the application , ı have to go to appointment page and he/she has to use select bar to select the city and click the button

Normal: The user first must sign up the application.If he/she already registered the application , he/she can just login.Then must go to appointment page.On the Appointment page, the city selection section (if no city is selected, all fields will have their information, appointment button, dislike button), if the user is in the system, they can select the city they want from there, and only the fields in the city they want appear on the appointment page.

System state on completion: The system looks at the city of that field every time a field is added and adds that city to the filtering section if the city is different from the previous ones, and if the user selects a certain city from the filtering section, it displays the fields matching that city on the page.

USER SCENERIOS 7

Title: Cancel Appointment

Initial assumption: As a user , I want to cancel an appointment ı have already made.First of all the user must sign in the application and go to profil page.Then appointment tab will appear on the screen .He/she should press the tab and the appointment and cancellation button will appear on that tab. Just press the cancel button to cancel the appointment.

Normal:The user first sign in the application.When you log in, the application will automatically redirect you to the profile page.There is an about and appointment tab on his profile page. He/she needs to click on the Appointment tab.The appointment will appear in the appointment tab.The only thing he/she needs to do to cancel the appointment is to tap the cancel button.

System state on completion: When the user wants to cancel, the system finds out which field it is from the field information and goes to the clock data table of that field and changes the value there.

USER SCENERIOS 8

Title: Change Personal Information

Initial assumption: As a user , I want to change update my informations , name , phone number and adress.When he/she sign in the application , he/she has to go to profil page and click edit profil button . And he/she can update any information from this tab.

Normal: The user must log in the application.When the user enters the application, the site automatically starts the site by redirecting to the profile page. The edit profile tab will appear on the top right of the profile page.When this edit profile button is pressed, a page will be opened for the user to update her information, and he/she can update the information she wants by entering the correct formats into the forms here

System state on completion:While updating the user information, the system first determines which user it is and accesses that user in the database and re-enters the information belonging to it as the user entered.

USER SCENERIOS 9

Title: Change Football Pitch Features

Initial assumption: As a user , I want to edit my football area.When I log in the site , The system automatically redirects to the profile page.The user must click Edit Fooball Area button and It should go to the edit the football field page.He/she is free to update the information she wants from this page as she wishes.

Normal:The user must log in to the existing owner account.The system will automatically take the user to the profile page when he or she logs.On the profile page, the section about the profile editing section, the appointment cancellation and field edit section.The user has to press the field edit among them.Has the right to edit the field by typing the data that they want to change to the forms given on the page that opens and clicking the save changes button.

System state on completion: The system completes the process by finding which field will be arranged from the data of the person who enters it and replacing the data belonging to that field with the newly entered data.

USE CASE TABLES

ID:	1.Use Case
Title:	Sign Up Use Case Table
Description:	User can sign up to web application with information.
Primary Actor:	Normal User / Owner
Main Success Scenario:	The user must first sign up for sign in the application. To login , he/she must go to signup page for registration.After entiring the sign up page information forms will appear.He/she must fill in information such as name, surname, username, email, password with correct format and accept the terms.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 10

ID:	2.Use Case
Title:	Sign In Use Case Table
Description:	Users can sign in to web application with username and password if the information is true.
Primary Actor:	Normal User / Owner
Main Success Scenario:	When the user wants to sign in with the application, they can sign in by coming to the sign in page of the application and entering user name and password.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 9

ID:	**3**
Title:	Book An Apointment
Description:	Users can book an appointment from the application.
Primary Actor:	User/Owner
Main Success Scenario:	Appointment page to see the information of the fields and select the field where they want to make an appointment. After selecting the site, the hours when the field is full and empty will appear on the Book Appointment page. After completing the payment process, the appointment process is completed.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority Point = 8

ID:	4.Use Case
Title:	Add Area to Web App Use Case
Description:	Users can to add theirs football area to web application with phone number,address and with photos.
Primary Actor:	Owner
Main Success Scenario:	When the owner wants to add area to application, they can firstly have to sign in to web app than going to add area section of the web app and entering the required information for football areas.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 7

ID:	5.Use Case
Title:	Add Comment For Football Areas
Description:	Users can comment for football areas in web application.
Primary Actor:	User/Owner
Main Success Scenario:	Users has to go to the profile page and click on the Add Area button. After clicking this button, the page to add football field will appear. On this page, there are information that should be entered such as the field address and phone number. After filling these correctly, the field will be added to the application.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 7

ID:	6.Use Case
Title:	Add Photo For User Profile
Description:	Users can upload a profile photo for their profile page.
Primary Actor:	User/Owner
Main Success Scenario:	The user must first sign up with the application. After sign up, he / she has to log in. After logging in, the user can go to the profile page and click the change photo button, select the photo they want from their local files and upload them.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 6

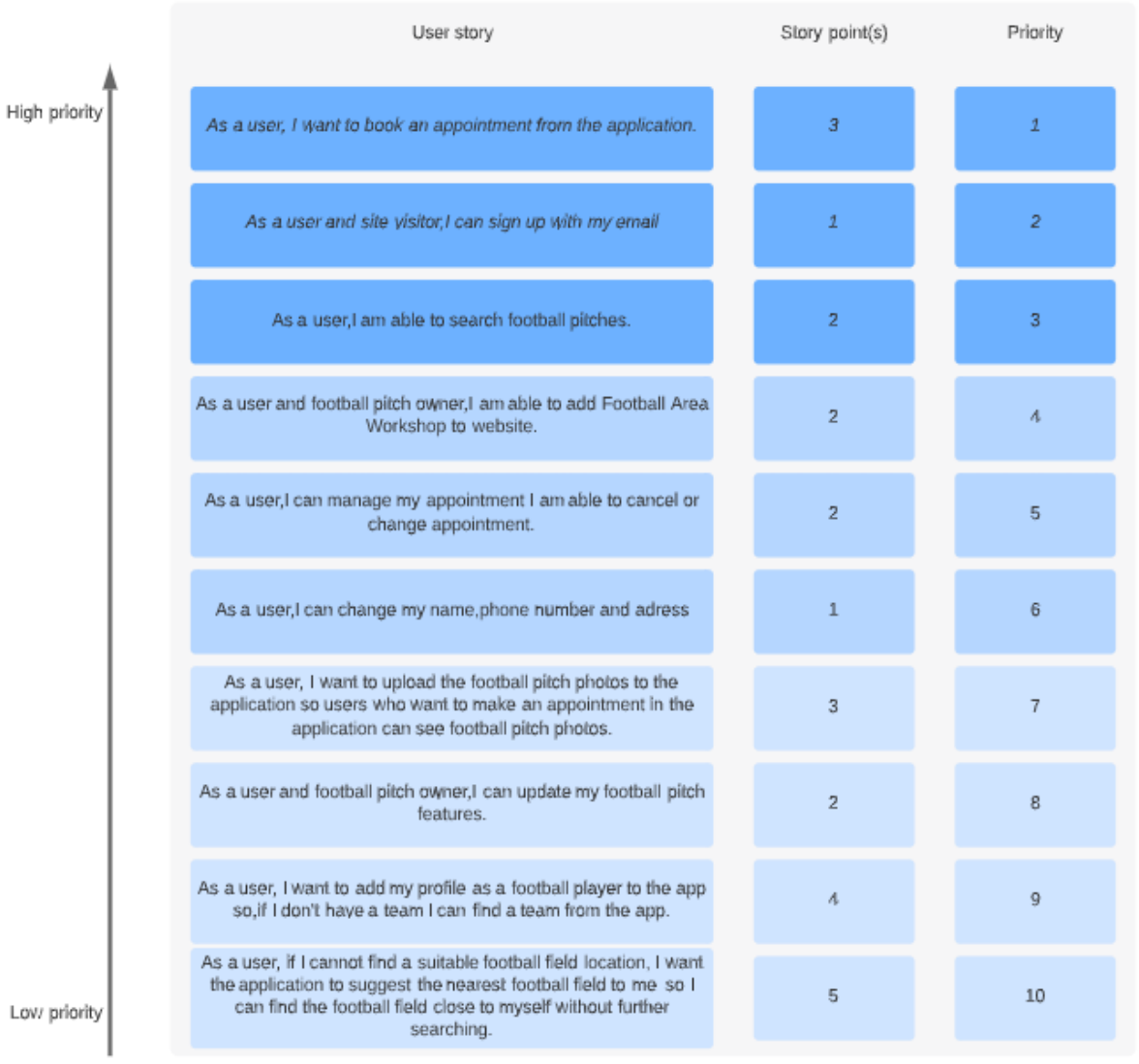
ID:	7.Use Case
Title:	Search Football Pitches
Description:	Users can search a specific football pitch with selection city.
Primary Actor:	User/Owner
Main Success Scenario:	On the Appointment page, the city selection section (if no city is selected, all fields will have their information, appointment button, dislike button), if the user is in the system, they can select the city they want from there, and only the fields in the city they want appear on the appointment page.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 4

ID:	8.Use Case
Title:	Cancel Appointment
Description:	Users can manage their appointment, Users are able to cancel or change appointment.
Primary Actor:	User/Owner
Main Success Scenario:	He/she needs to click on the Appointment tab.The appointment will appear in the appointment tab.The only thing he/she needs to do to cancel the appointment is to tap the cancel button.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 9

ID:	9.Use Case
Title:	Change Personal Information
Description:	Users want to change update their informations , name , phone number and address.
Primary Actor:	User/Owner
Main Success Scenario:	The edit profile tab will appear on the top right of the profile page.When this edit profile button is pressed, a page will be opened for the user to update her information, and he/she can update the information she wants by entering the correct formats into the forms here.
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 8

ID:	10.Use Case
Title:	Change Football Pitch Features
Description:	Users can edit their football areas.
Primary Actor:	Owner
Main Success Scenario:	On the profile page, the section about the profile editing section, the appointment cancellation and field edit section.The user has to press the field edit among them.Has the right to edit the field by typing the data that they want to change to the forms given on the page that opens and clicking the save changes button..
Status:	Done
Owner:	Erdi Bayır
Priority:	Priority point = 8

BACKLOG



User story	Story point(s)	Priority
As a user, I want to book an appointment from the application.	3	1
As a user and site visitor,I can sign up with my email	1	2
As a user,I am able to search football pitches.	2	3
As a user and football pitch owner,I am able to add Football Area Workshop to website.	2	4
As a user,I can manage my appointment I am able to cancel or change appointment.	2	5
As a user,I can change my name,phone number and adress	1	6
As a user, I want to upload the football pitch photos to the application so users who want to make an appointment in the application can see football pitch photos.	3	7
As a user and football pitch owner,I can update my football pitch features.	2	8
As a user, I want to add my profile as a football player to the app so,if I don't have a team I can find a team from the app.	4	9
As a user, if I cannot find a suitable football field location, I want the application to suggest the nearest football field to me so I can find the football field close to myself without further searching.	5	10

User story	Story point(s)	Priority
As a user,I can manage my appointment I am able to cancel or change appointment.	2	1
As a user, I want to add my profile as a football player to the app so,if I don't have a team I can find a team from the app.	4	2
As a user, when I want to play a match, I want to find the one that suits me from the matches around me.	3	3
As a user, I want to set up my own team and save it to the application.	2	4
As a user, I want to find a reliable referee to direct my own team's match	3	5
As a user, I want to register as a referee and manage the matches	3	6
As a user, I want to be able to invite appropriate people to my team by accessing other players' profile.	3	7
As a user, I want to have a blog section and discuss our matches with other members there.	3	8
As a user, I want to be able to communicate with other players so that I can borrow boots or jerseys	4	9
As a user, if I cannot find a suitable football field location, I want the application to suggest the nearest football field to me so I can find the football field close to myself without further searching.	5	10

TASKS and SPRINTS

WEEK 3

TASK	ASSIGNEE
Choose tool for unit test	TEAM
Design a wireframe for homepage	BERK + MUSA
Design a wireframe for sign in page	BERK + MUSA
Design a wireframe for appointment, contact us and about us pages	ÖMER
Design a wireframe for user signup page	MUHAMMED
Design a wireframe for owner signup page	ERDİ
Write user stories + Create use case diagrams	ERDİ

WEEK 4

TASK	ASSIGNEE
The front end design of the owner signup page	ERDİ
The front end design of the homepage	BBERK + MUSA
The front end design of the sign in page	BERK + MUSA
The front end design of the appointment, contact us and about us pages	ÖMER
The front end design of the user signup page	MUHAMMED
Requirement analysis	ERDİ + MUHAMMED

WEEK 5

TASK	ASSIGNEE
Start searching resources for database operations	TEAM

WEEK 6

TASK	ASSIGNEE
Presentation Report	TEAM
Documantation Test Cases	ÖMER + MUSA
Graphical Interface Documantation + front end design of the profile page	BERK YİĞİT
Use case diagram and user stories documentation	ERDİ
Project backlog document	MUHAMMED

WEEK 8

TASK	ASSIGNEE
Create profile page	Erdi Bayır
Create appointment pages	Erdi Bayır
Database works of sign up page	Muhammed Geçgeloğlu
Database works of sign in page	Muhammed Geçgeloğlu
Write tests for project	Ömer Faruk Akduman Musa Karaş

WEEK 9

TASK	ASSIGNEE
Continue profile page	Erdi Bayır
Continue appointment pages	Erdi Bayır
Create edit profile page	Berk Yiğit Dursun Musa Karaş
Database work of session	Ömer Faruk Akduman
Write tests for project	Ömer Faruk Akduman Musa Karaş

WEEK10

TASKS	ASSIGNEE
Searching for football areas	Erdi Bayır
Like, dislike and score for football areas	Berk Yiğit Dursun
Unique username check	Berk Yiğit Dursun
Payment page	Muhammed Geçgeloğlu
Adding photo for profile	Ömer Faruk Akduman
Write tests for project	Musa Karaş Ömer Faruk Akduman

WEEK11

TASKS	ASSIGNEE
Completing unfinished works	TEAM
Prepare documentation	TEAM
Add comment section to areas	Berk Yiğit Dursun

WEEK 12

TASKS	ASSIGNEE
Completing unfinished works	TEAM
Prepare documentation	TEAM
Add comment section to areas	Berk Yiğit Dursun

WEEK 13

TASKS	ASSIGNEE
Completing unfinished works	TEAM
Prepare documentation	TEAM
Add comment section to areas	Berk Yiğit Dursun

WEEK14

TASKS	ASSIGNEE
Deploy the project	ÖMER
Prepare documentation	TEAM
Moving the project to online database	TEAM
Test for project	ÖMER + MUSA
Adding cancel appointment	ERDİ
Design logo	MUHAMMED

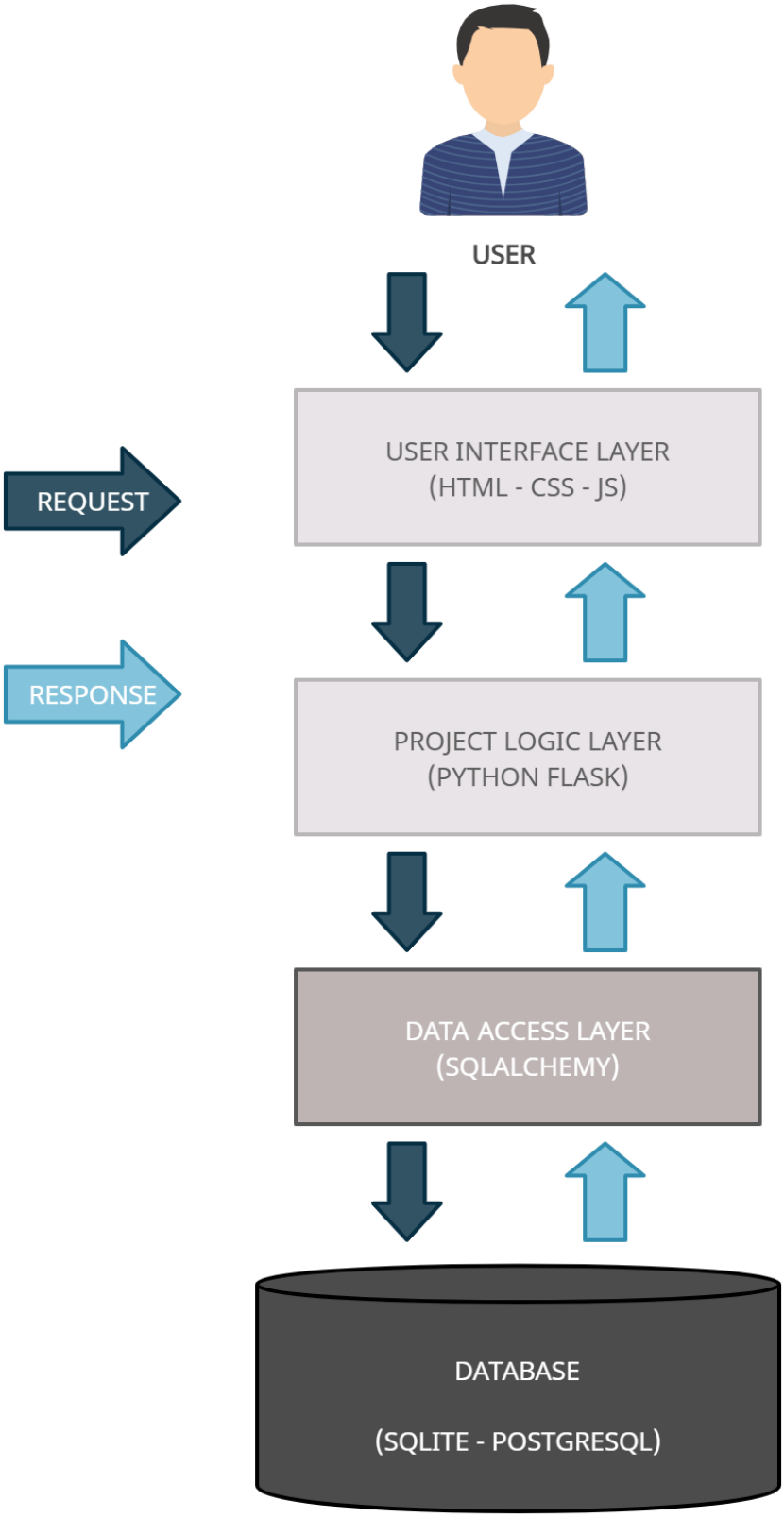
Simple System Architecture Document

Considering our inexperience in management and our lack of knowledge about software, we found it appropriate for our project to separate the systems from each other and develop them separately.

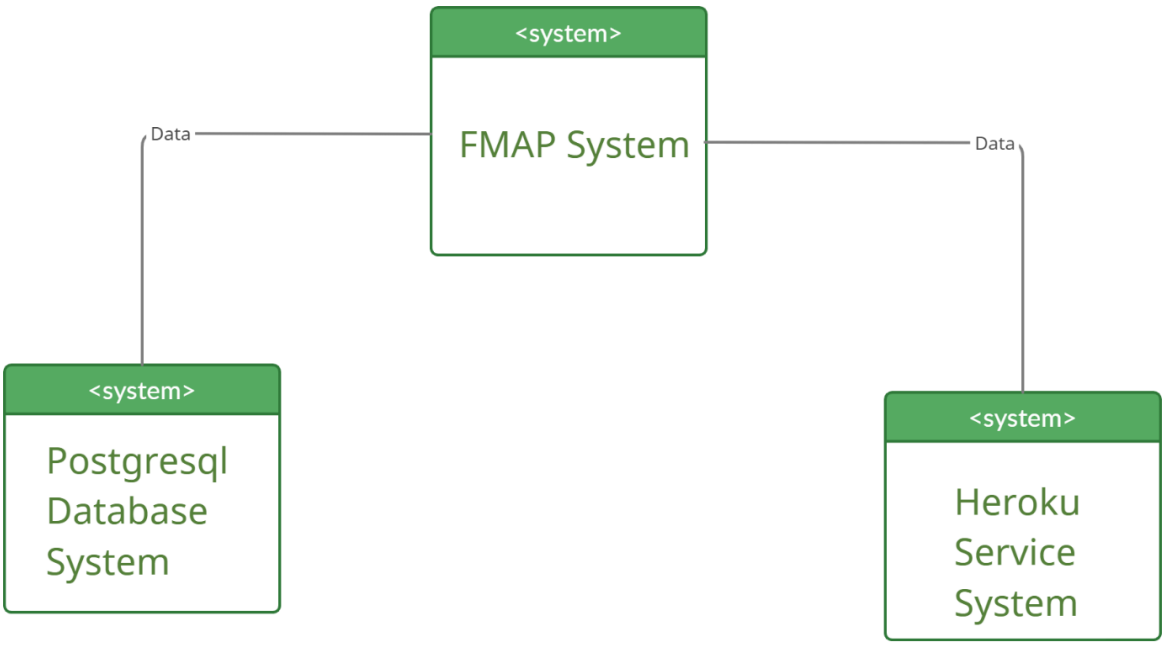
The Layered Architecture pattern is way of achieving separation and independence. Layering the system in this approach supports the incremental development of systems. This architecture allows adding new layers or enhancing existing layers without interfering with other layers.

We have implemented this system in our project to provide these advantages to our project. In this way, we did not encounter any serious problems in terms of management and operation. When we had problems with advancing some layers, our project continued to evolve and we did not have to wait until the problem was resolved.

System Architecture Diagram

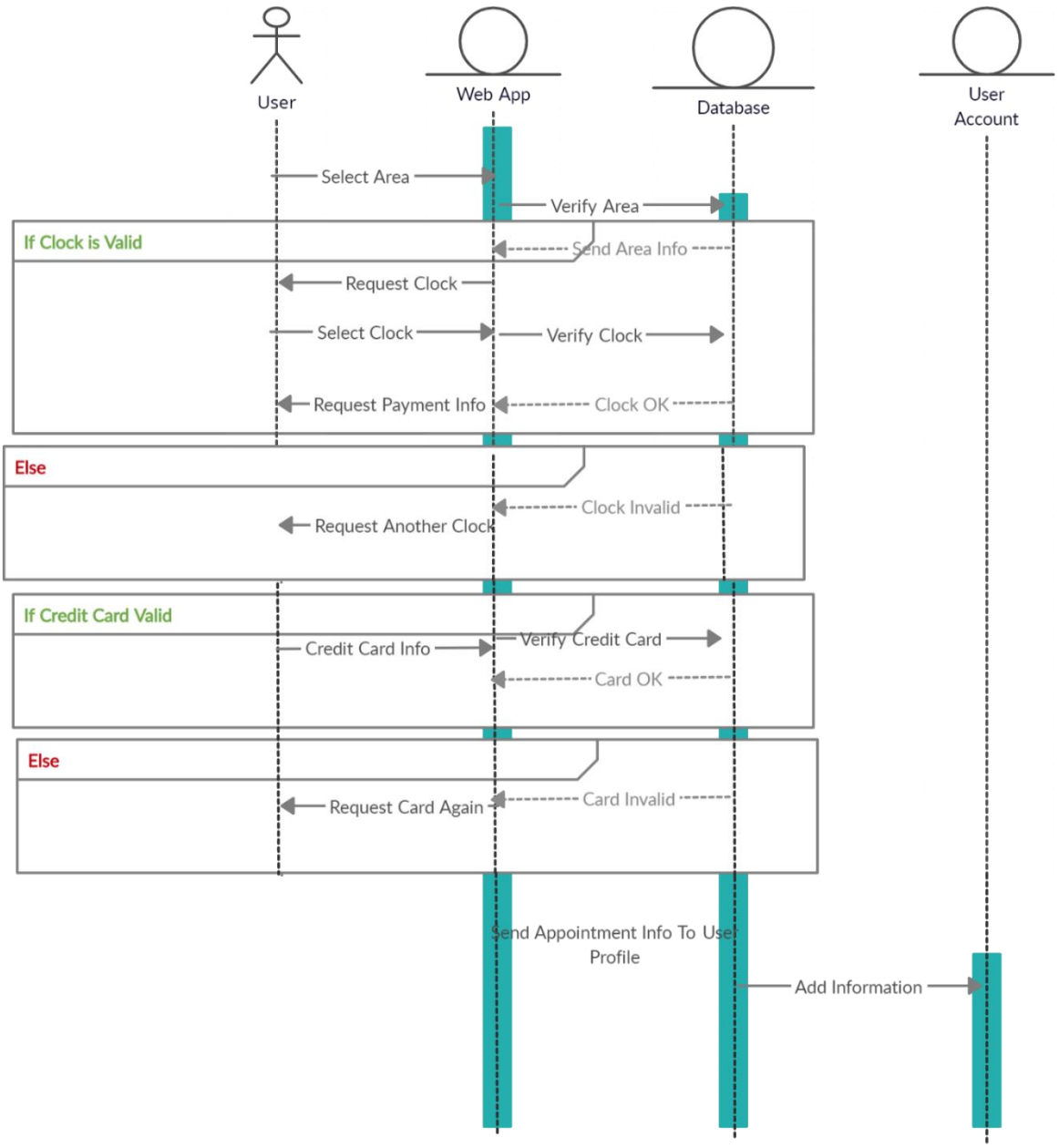


CONTEXT MODEL

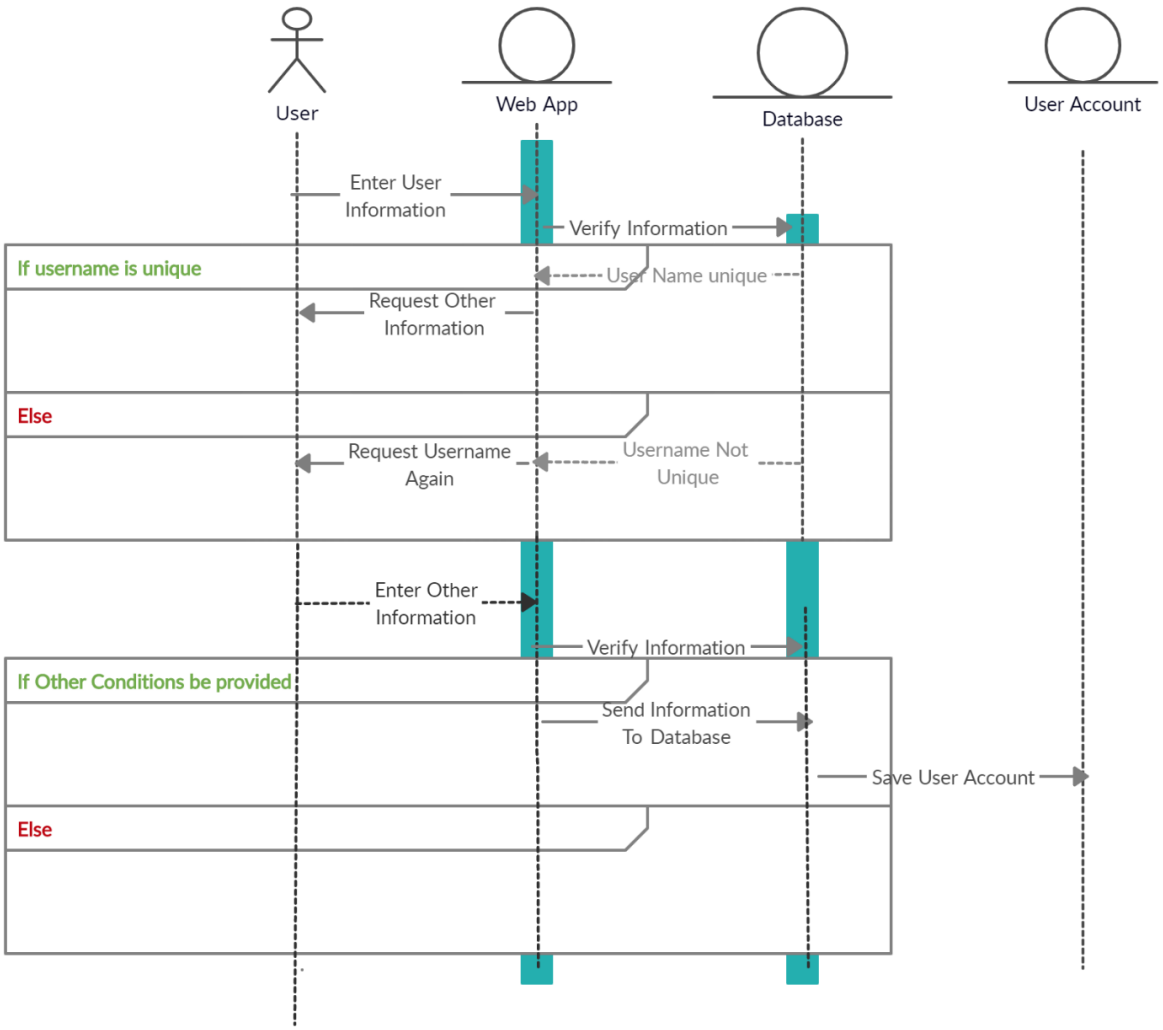


SEQUENCE DIAGRAM

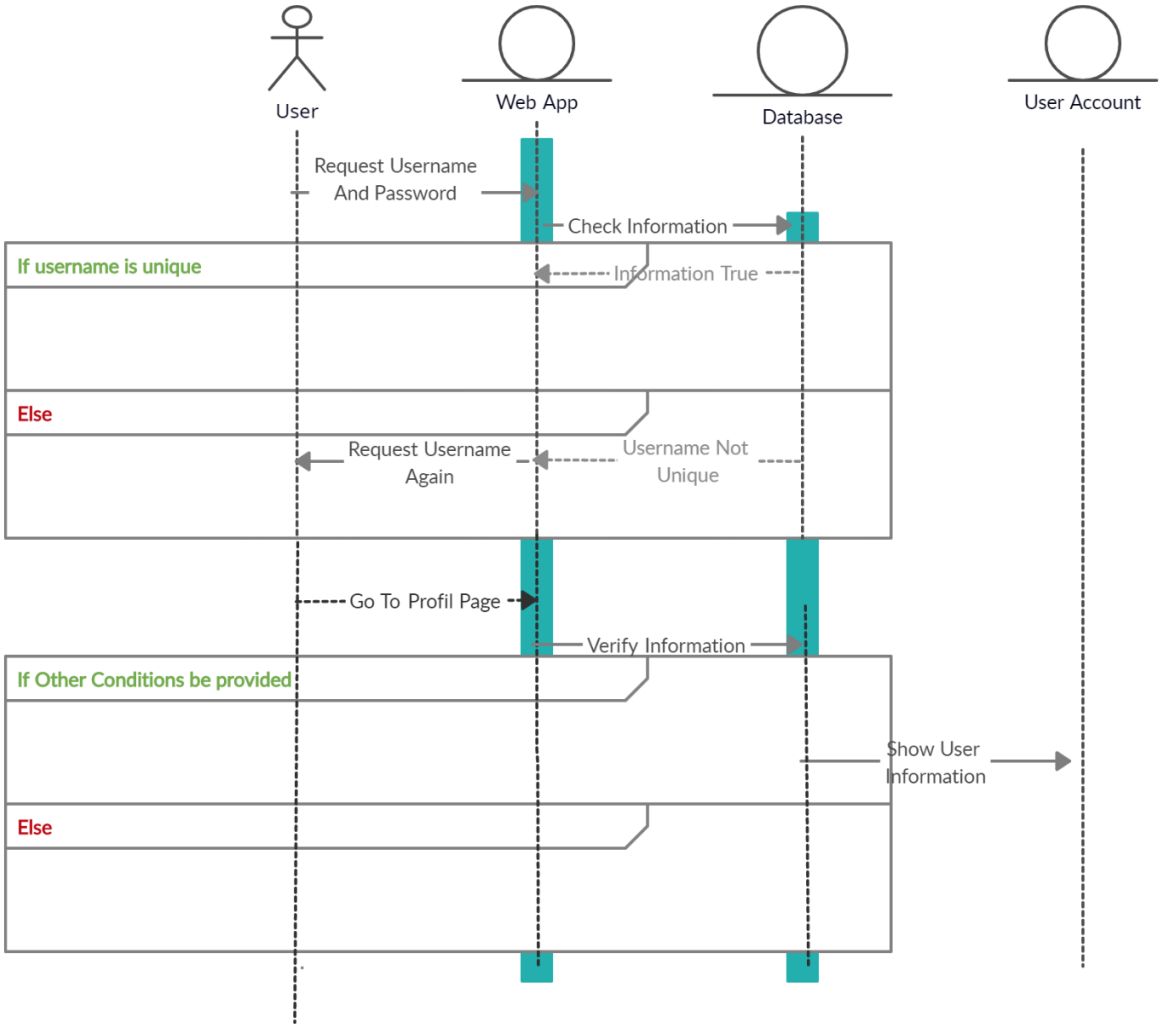
BOOK AN APPOİNTMENT SEQUENCE DIAGRAM



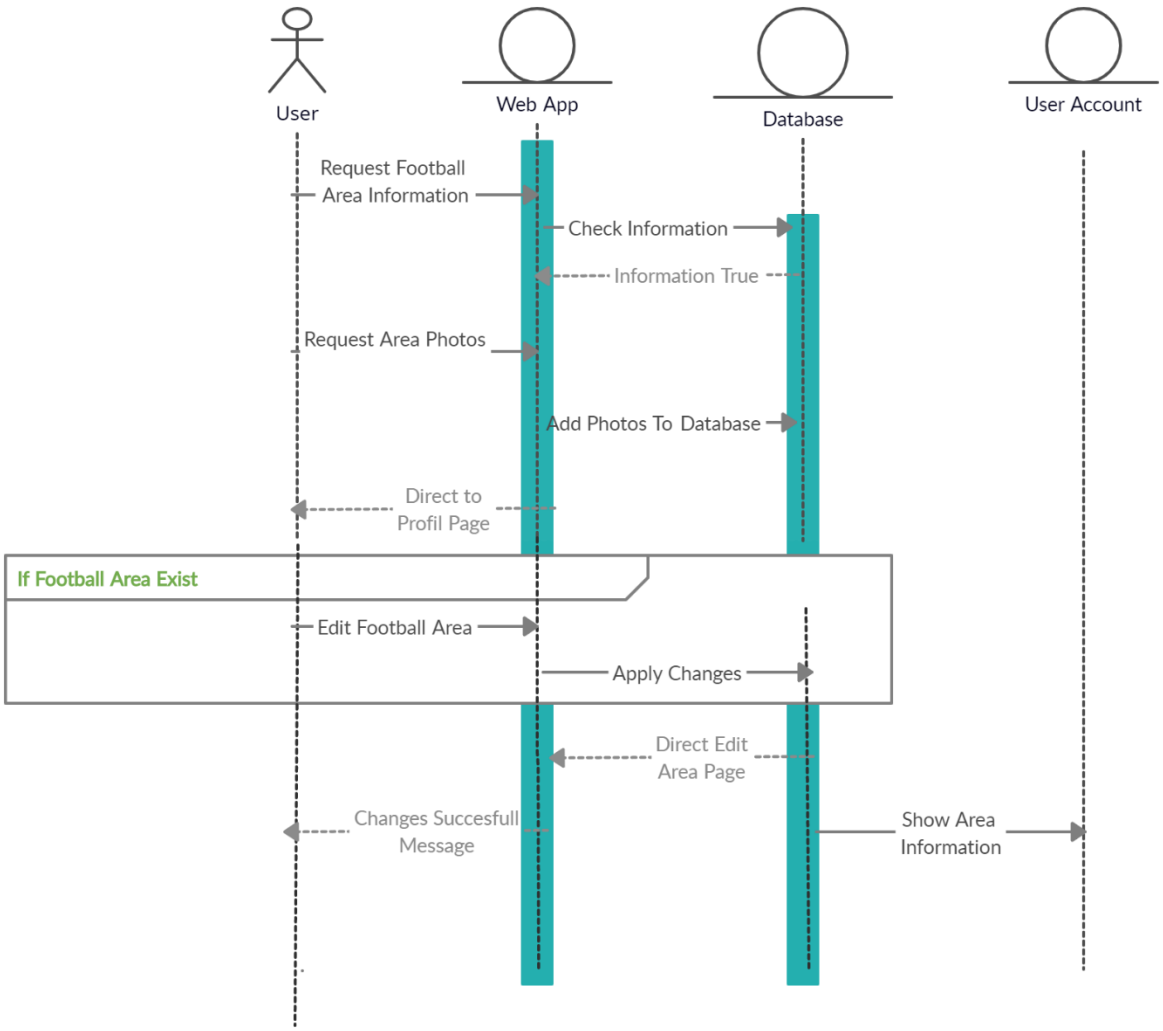
SIGN UP SEQUENCE DIAGRAM



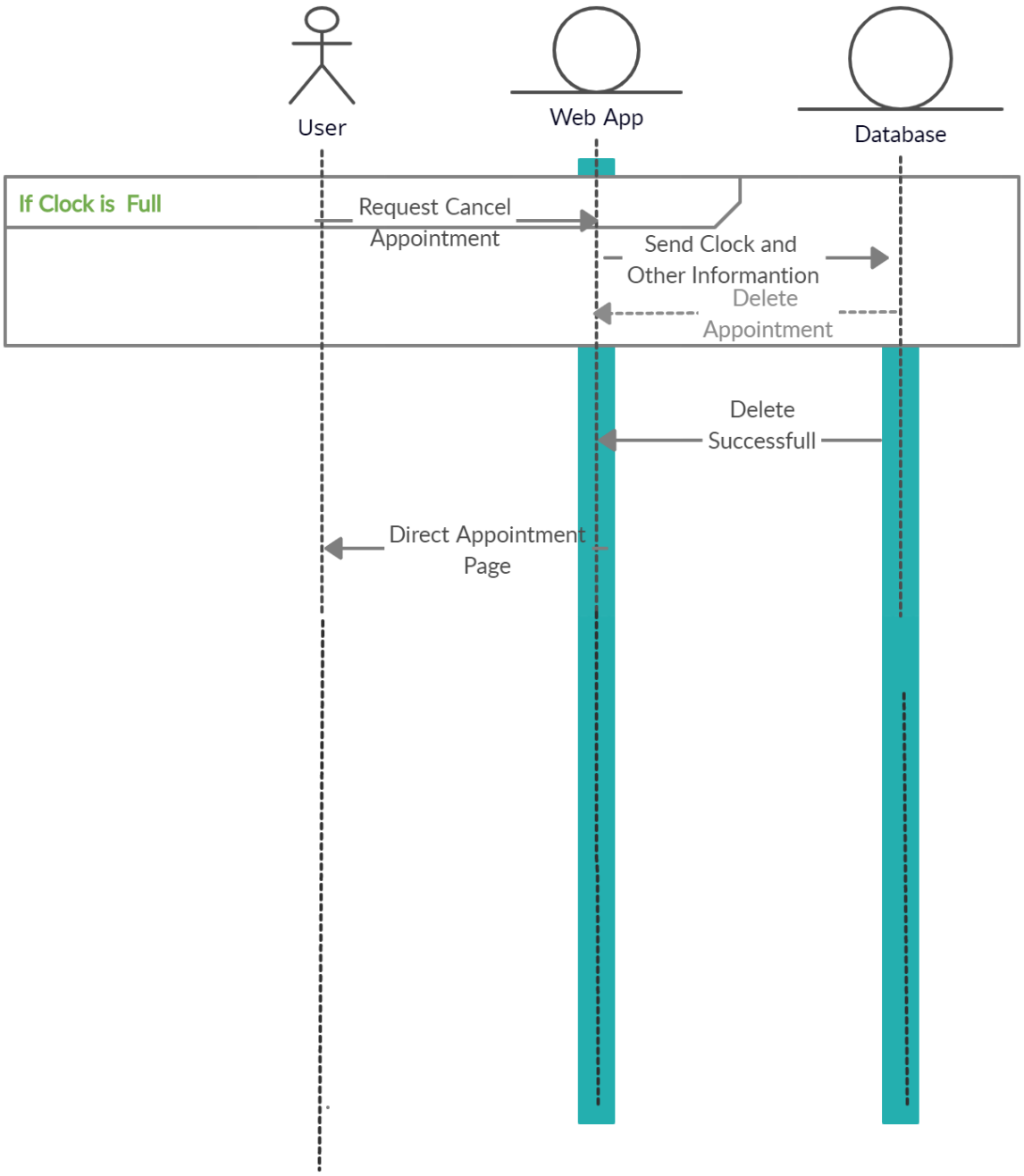
SIGN UP SEQUENCE DIAGRAM



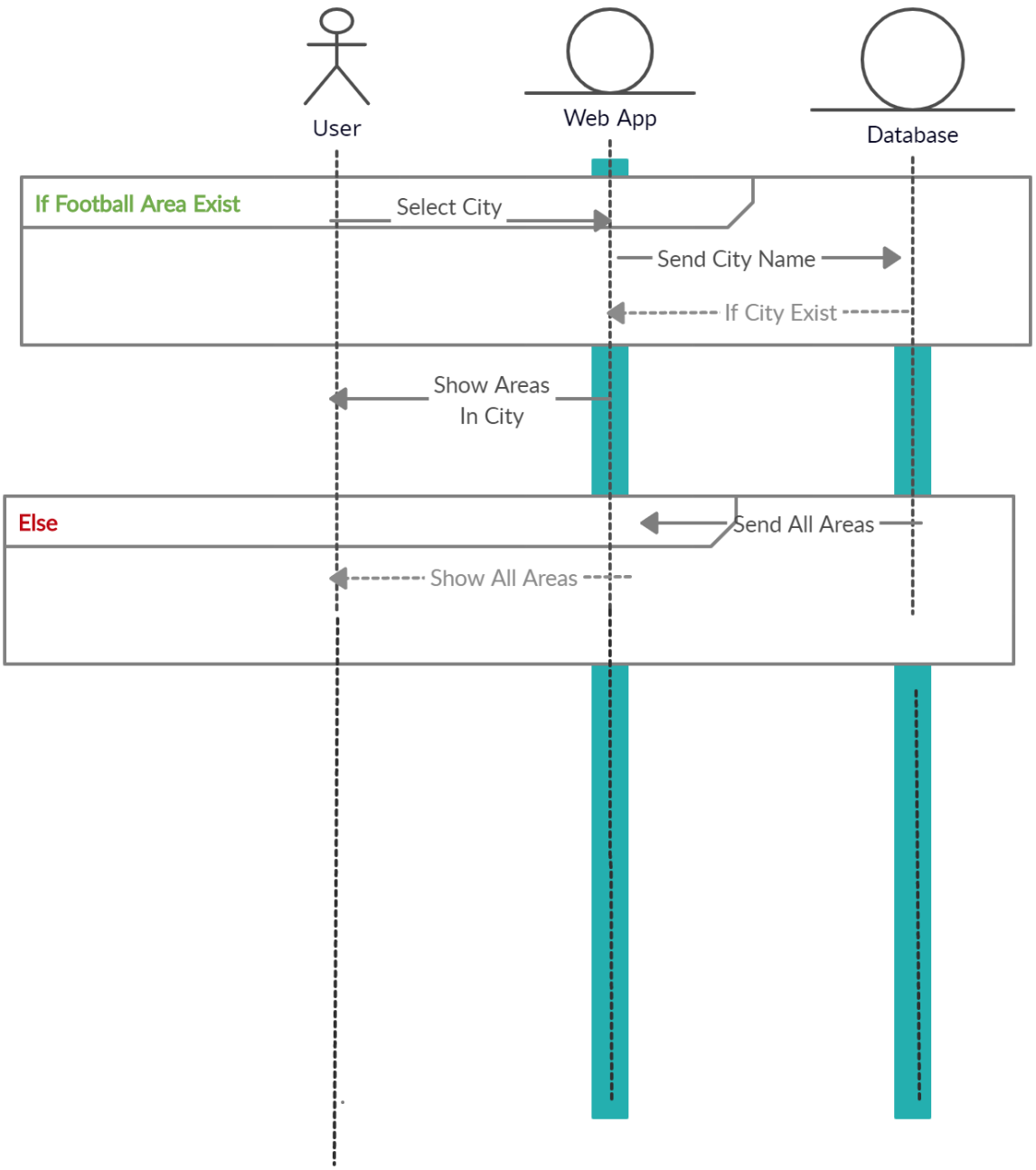
ADD/EDİT FOOTBALL AREA SEQUENCE DIAGRAM



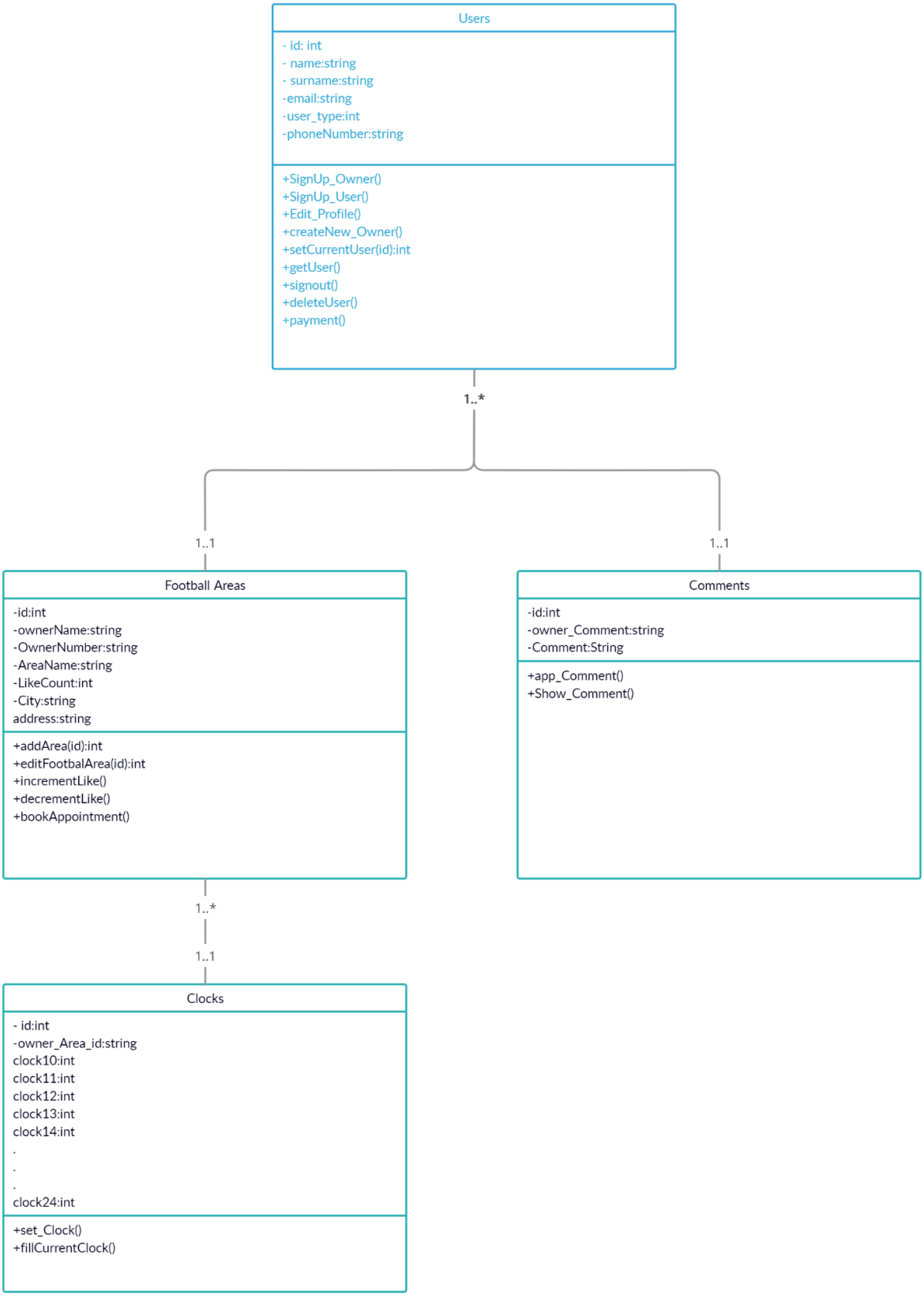
CANCEL APPOINTMENT SEQUENCE DIAGRAM



SELECT CITY SEQUENCE DIAGRAM

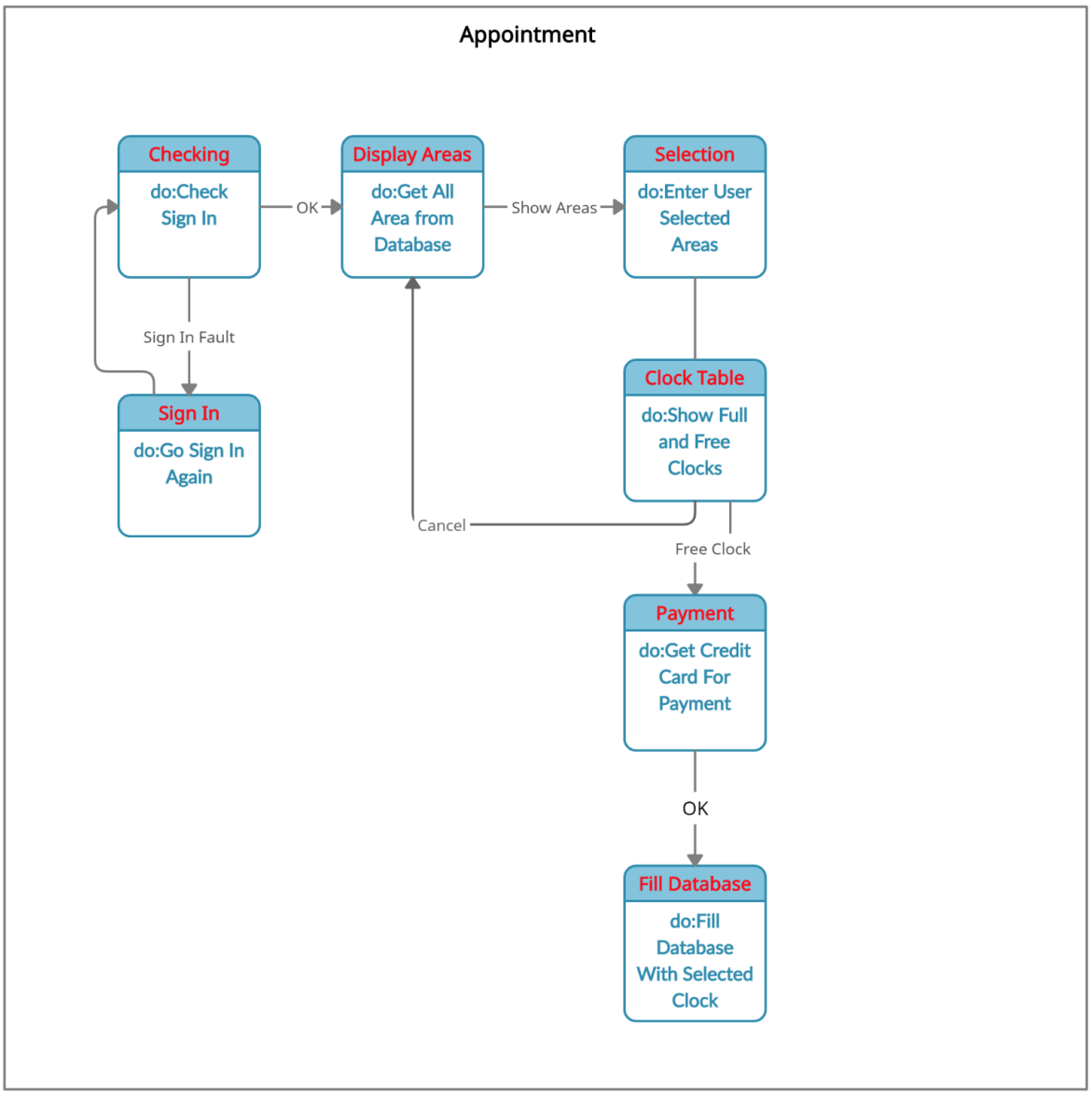


CLASS DIAGRAMS

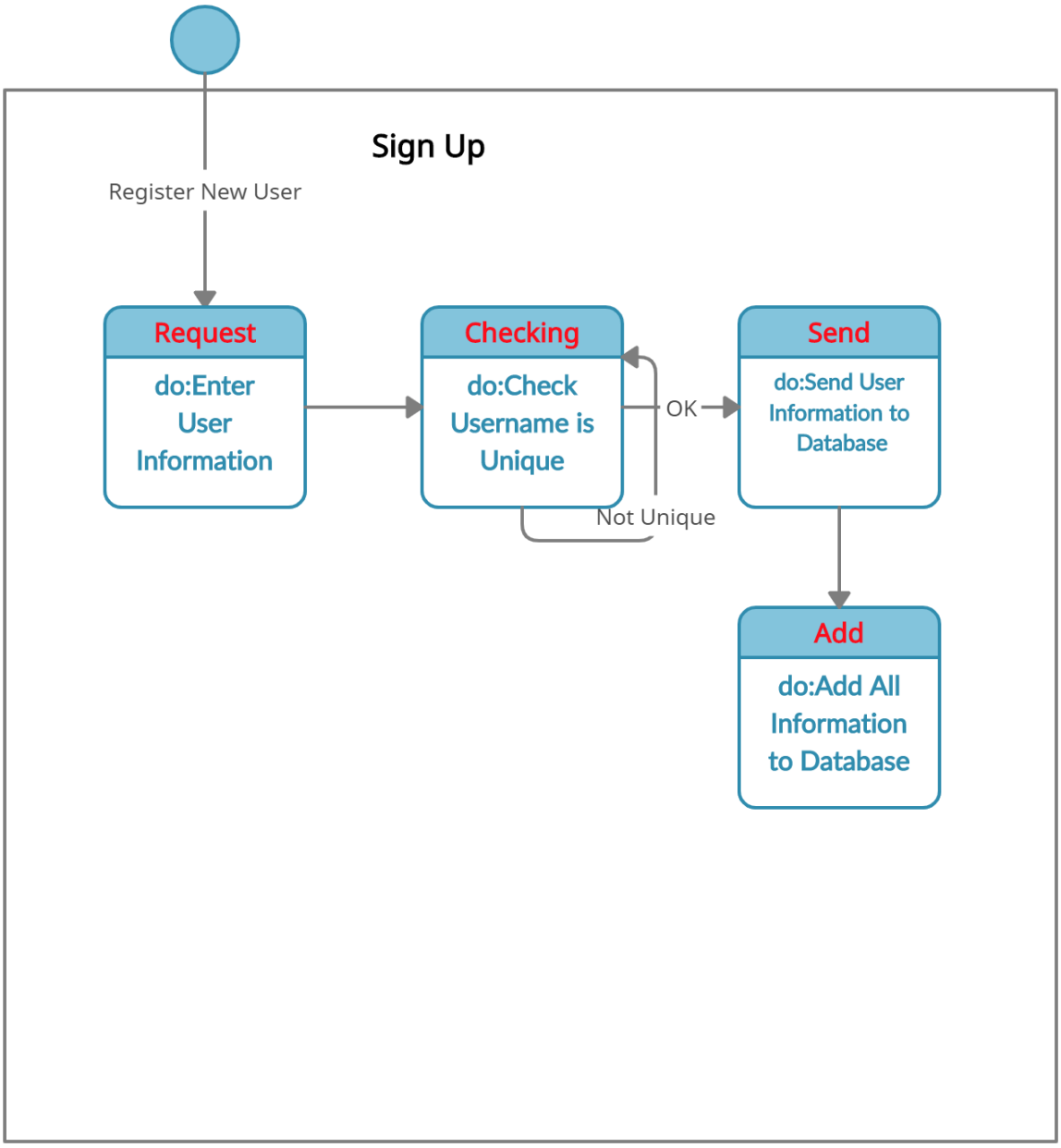


STATE DIAGRAMS

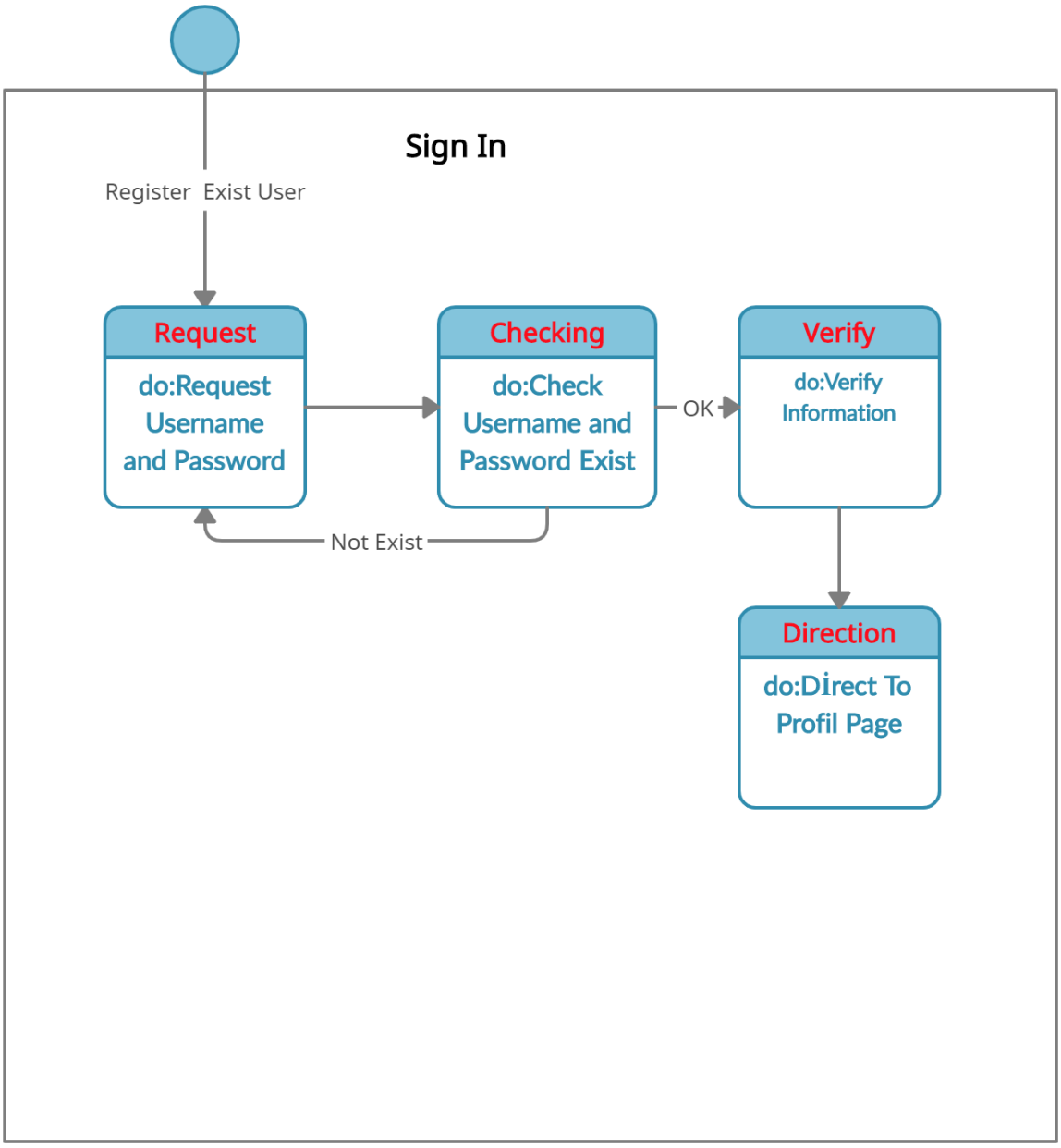
BOOK APPOİNTMENT STATE DIAGRAM



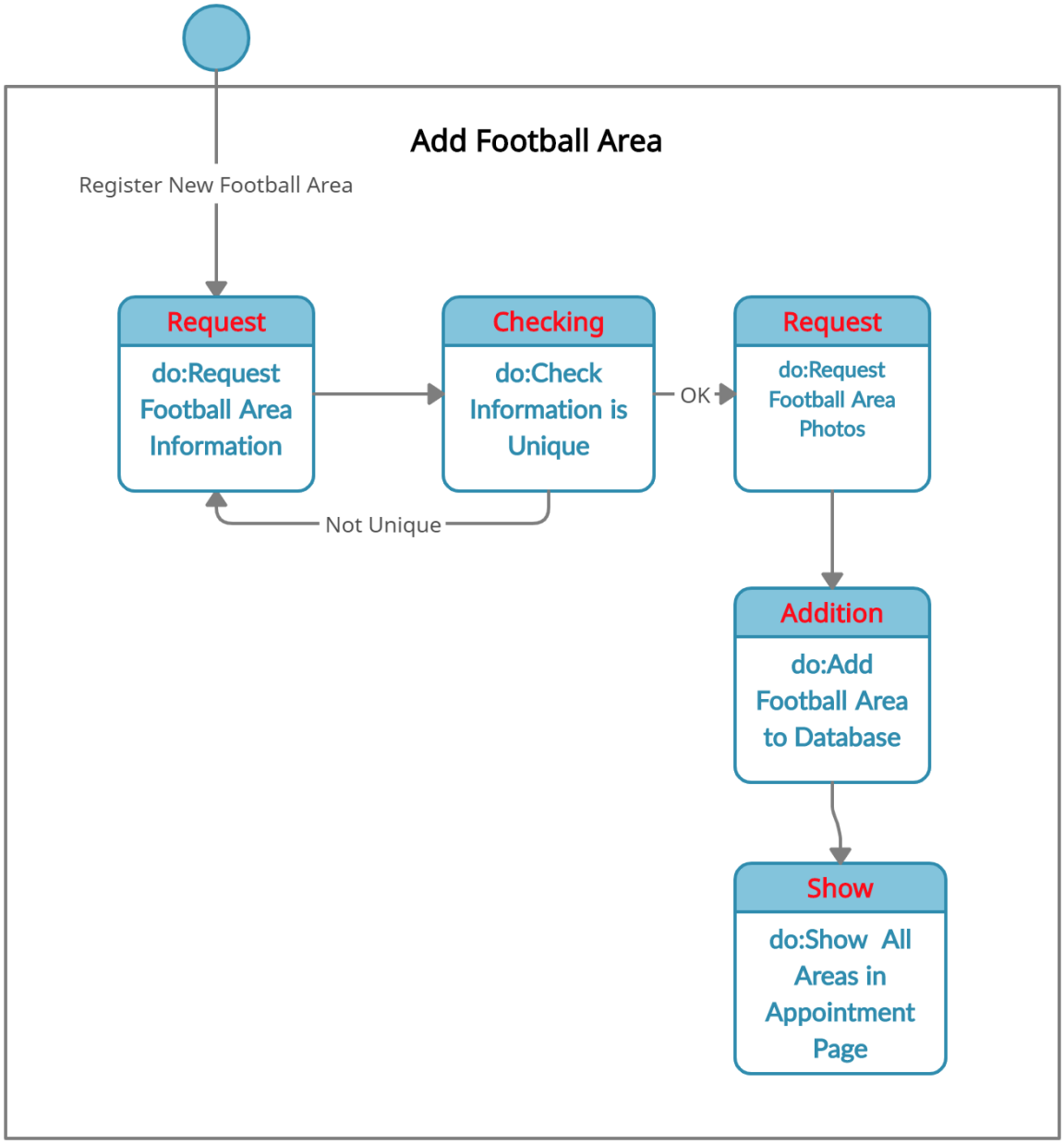
SIGN UP STATE DIAGRAM



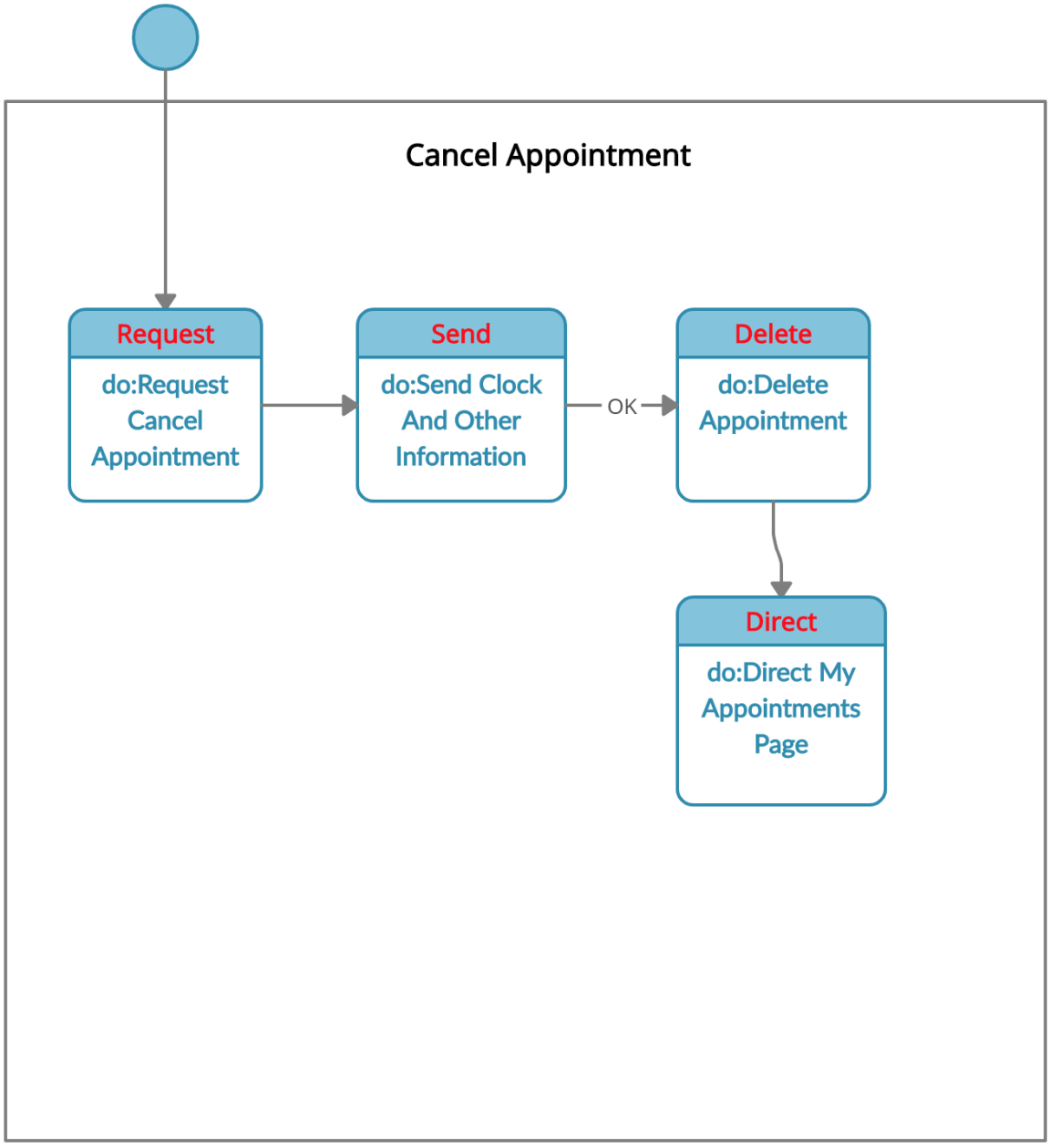
SIGN IN STATE DIAGRAM



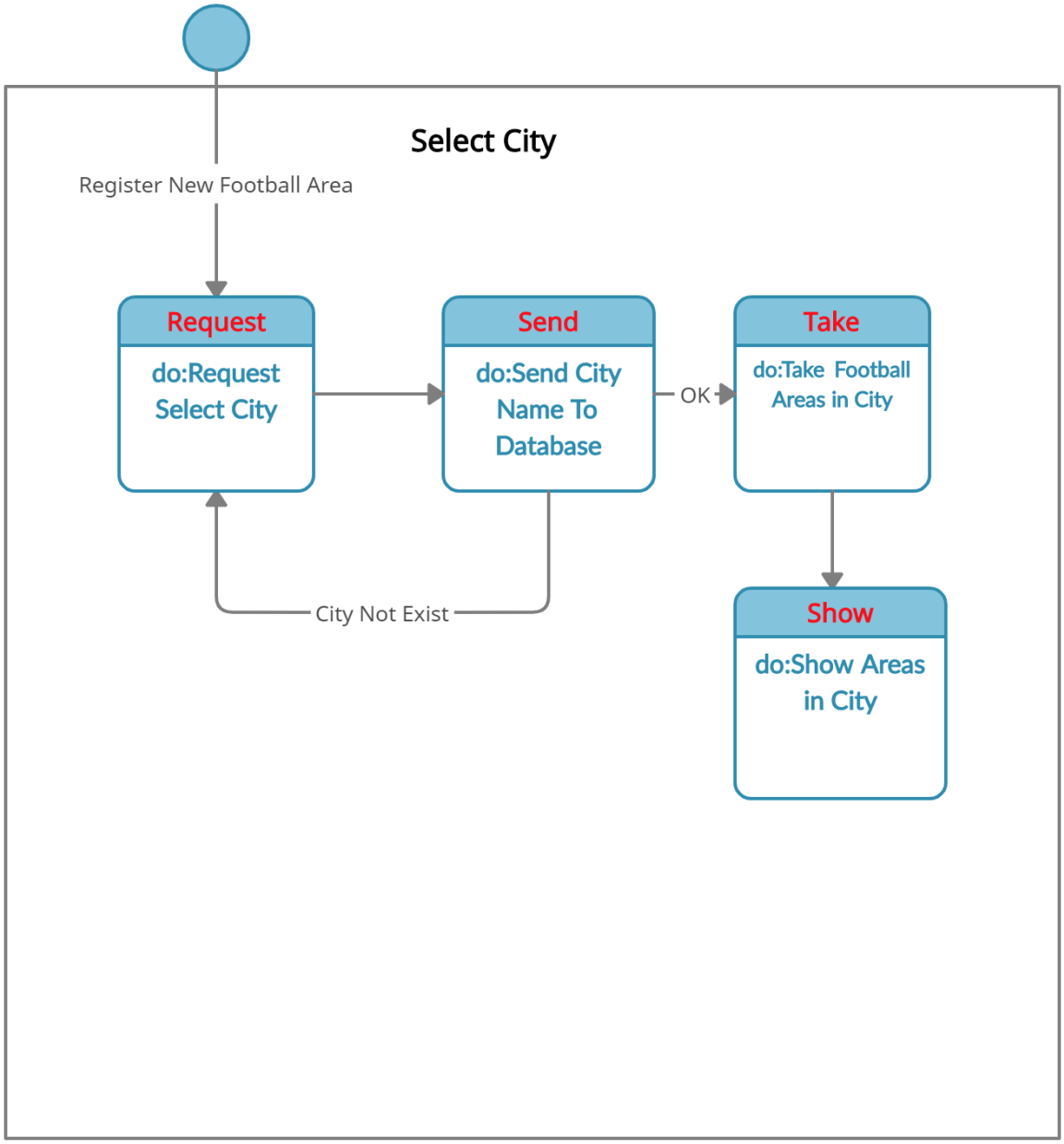
ADD FOOTBALL AREA STATE DIAGRAM



CANCEL APPOINTMENT STATE DIAGRAM



SELECT CITY STATE DIAGRAM



CONTINUOUS INTEGRATION – CI

The continuous integration architecture ensures that the new version, which is formed as a result of changes / improvements, is automatically merged with the current version.

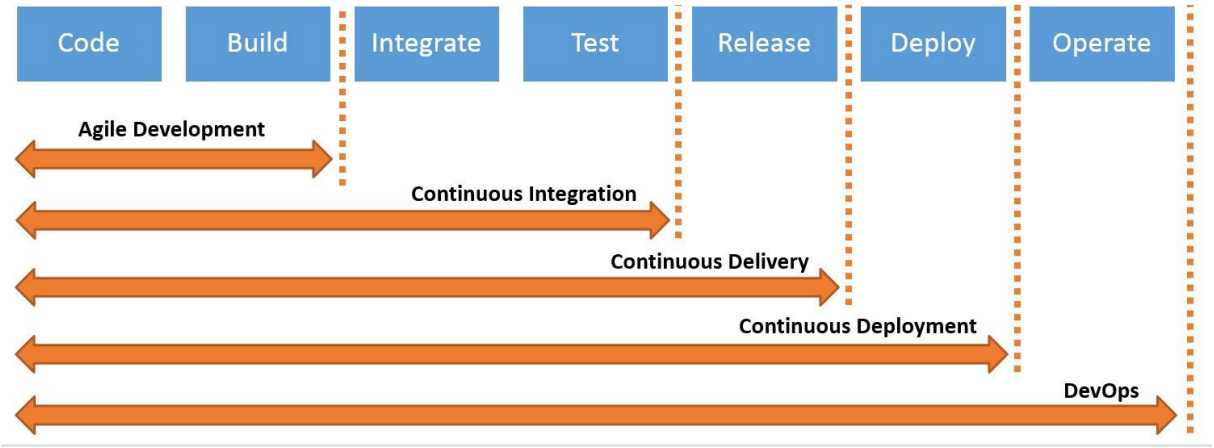
CONTINUOUS DELIVERY – CD

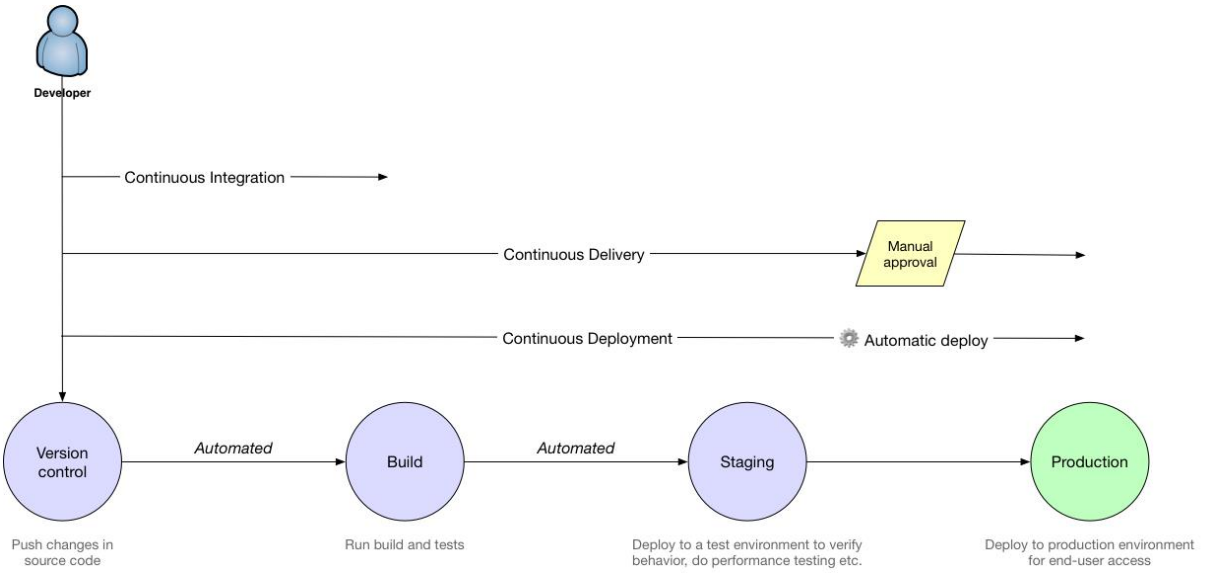
Continuous delivery is that a desired version can be deployed to a desired environment by manually triggering it. In other words, deployment is not provided unless we want it, we start the process manually when we want a deployment.

Continuous integration, continuous deployment, and continuous delivery are like vectors that have the same direction, but different magnitude. Their goal is the same: make our software development and release process faster and more robust.

Benefits

- Smaller code changes are simpler (more atomic) and have fewer unintended consequences.
- Fault isolation is simpler and quicker.
- Mean time to resolution (MTTR) is shorter because of the smaller code changes and quicker fault isolation.
- Testability improves due to smaller, specific changes. These smaller changes allow more accurate positive and negative tests.
- Elapsed time to detect and correct production escapes is shorter with a faster rate of release.
- The backlog of non-critical defects is lower because defects are often fixed before other feature pressures arise.
- The product improves rapidly through fast feature introduction and fast turn-around on feature changes.
- Upgrades introduce smaller units of change and are less disruptive.
- CI-CD product feature velocity is high. The high velocity improves the time spent investigating and patching defects.
- Feature toggles and blue-green deploys enable seamless, targeted introduction of new production features.
- You can introduce critical changes during non-critical (regional) hours. This non-critical hour change introduction limits the potential impact of a deployment problem.
- Release cycles are shorter with targeted releases and this blocks fewer features that aren’t ready for release.
- End-user involvement and feedback during continuous development leads to usability improvements. You can add new requirements based on customer’s needs on a daily basis.





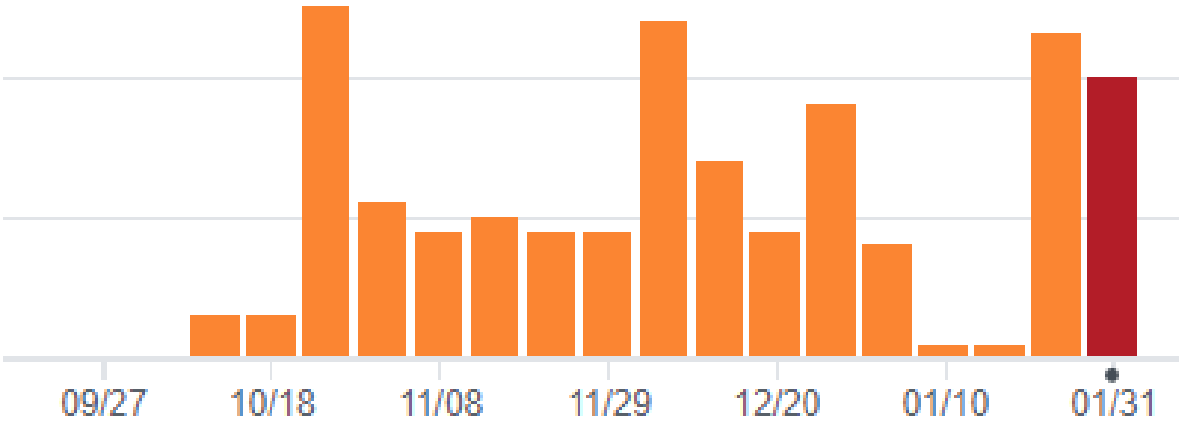
https://semaphoreci.com/blog/2017/07/27/what-is-the-difference-between-continuous-integration-continuous-deployment-and-continuous-delivery.html

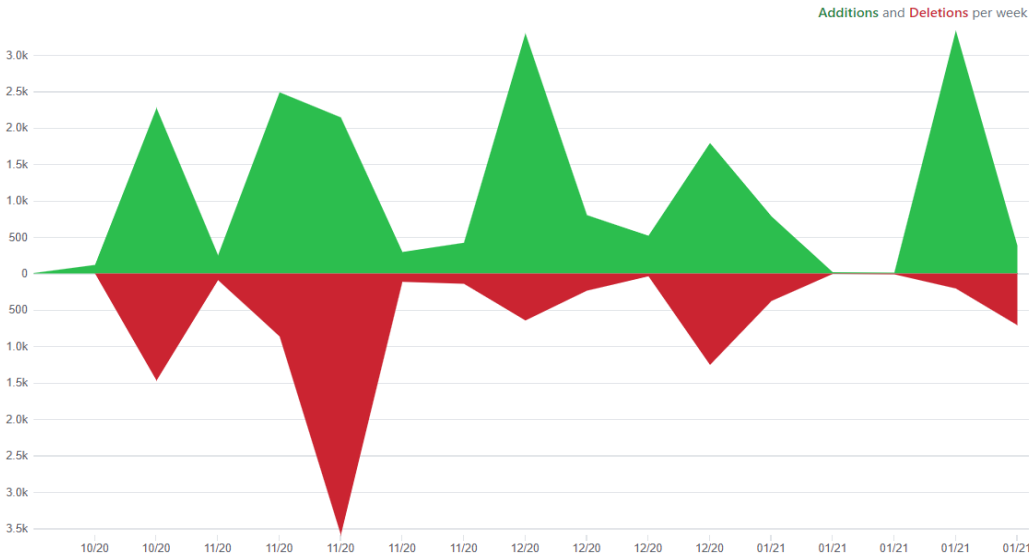
https://medium.com/kodcular/ci-continuous-integration-cd-continuous-delivery-deployment-nedir-cdfbfdc40e4b

https://help.mypurecloud.com/articles/benefits-continuous-integration-continuous-deployment-ci-cd/

Actually , we have been implementing the continuous integration architecture without extra tools since the beginning of our project. We were uploading every change in our code to our github repository every day. By conducting our tests, we always kept our project ready to deploy.

After the research we have done, we have applied the continuous integration and continuous delivery features more systematically in our project using github actions.






```yaml
# This workflow will install Python dependencies, run tests and lint with a single version of Python
# For more information see: https://help.github.com/actions/language-and-framework-guides/using-python-with-github-actions

name: Python application

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.9
      uses: actions/setup-python@v2
      with:
        python-version: 3.9
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install flake8 pytest
        if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Lint with flake8
      run: |
        # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
        # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
        flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics
    - name: Test with pytest
      run: |
        pytest
```

Accessibility and Feedbacks

https://football-appointment.herokuapp.com/

A good project removes the problems we experience. We tried to eliminate a big problem while planning our project. Especially in our country, although many people want to do sports, they cannot meet the appropriate conditions. This causes people to continue their lives more boring and unhealthy. Football appointment project was designed to eliminate the problem of people who want to do sports, not finding places and partners.

We have completed our project designed for this purpose. We presented it in a way that people can access. And some of the feedback we got from those people:

Metehan Kundak – 23 years old – İSTANBUL – Student

Futbol oynamak benim için bir tutku ama çevremde uygun halı saha bulmak beni zorlamaktaydı. Bu uygulama sayesinde çevremdeki en uygun sahayı bulabileceğim. FMAP ekibine teşekkür ediyorum.

Sarpkan Özcan – 24 years old – İSTANBUL – Student

Yıllardır en büyük dertlerimden biri halısaha oynayabilecek takımı kuramamaktı. Bu uygulamayı ilk duyduğumda tam da aradığım uygulama dedim. Pandeminin bitip kullanacağım ilk günü sabırsızlıkla bekliyorum.

Ertuğrul Semiz – 23 years old – İSTANBUL – Engineer

Bir hayal gerçek oldu. Ben bir futbol aşığıyım ama uzun zamandır maç yapmak için doğru sahayı bulamıyordum. Bu uygulama sayesinde bu dertten kurtulacağımı düşünüyorum. Umarım uygulamada en güvenilir sahalara ulaşabilirim.

Alper Uslu – 22 years old – İSTANBUL – Student

Uygulamayı çok beğendim ama başka isteklerim olacak. Çocukluğumdan beri hakem olmak istiyorum. Umarım FMAP ekibi uygulamaya istediğimiz maçlarda hakem olabileceğimiz bir özellik ekler. Böylece hem hayallerimiz gerçek olur hem de para kazanabiliriz.

Unittest of Flask Project Application

In this project's unittest section We divided it into 4 Main Parts

1. Database Testing: We try to test Database using helper functions which are created by ourself. Our main goal is to detect database errors that the user may encounter while using this application, so we tried to detect certain errors that may occur by simulating the database part of the application.

2. Flask-Web Page Testing: In this part we tested pages to ensure load successfully.

3. User Login-Out Testing: In this part We checked if any user can login and logout successfully.

4. Helper Functions: We needed some codes for simulate some cases in that reason we added helper functions.

0-Total Testing (*32/32 Successful test*)

No.	Func. Name	Purpose	Result
1-	test_notToSignIn(self):	To Ensure that start application with no user -not yet login	OK
2-	test_signupLogin(self):	To Ensure that signup page requires user login	OK
3-	test_main_route_requires_addArea(self):	To Ensure that addArea page requires user login	OK

4-	test_main_route_requires_myprofil(self):	To Ensure that myprofil page requires user login	OK
5-	test_main_route_requires_addFootballArea(self):	To Ensure that addFootballArea page requires user login	OK
6-	test_ToSignIn(self):	To test user login succesfully	OK
7-	test_signout(self):	To test user signout succesfully	OK
8-	test_homepage(self):	For ensure that homepage was load correctlly when user signin	OK
9-	test_signup(self):	For ensure that signup page was load correctlly	OK
10-	test_appointment(self):	For ensure that appoinment page was load correctlly	OK
11-	test_myprofil(self):	For ensure that myprofile page was load correctlly	OK
12-	test_contactus(self):	For ensure that contactus page was load correctlly	OK
13-	test_aboutus(self):	For ensure that aboutus page was load correctlly	OK
14-	test_signin(self):	For ensure that signin page was load correctlly	OK
15-	test_editMyProfil(self):	For ensure that edit my profile page was load correctlly	OK
16-	test_addFootballArea(self):	For ensure that addFootballArea page was load correctlly	OK
17-	test_payment(self):	For ensure that payment page was load correctlly	OK
18-	test_userAddedSuccesfully(self):	To test user added succesfully to Db	OK
19-	test_commentAddedSuccesfully(self):	To test user could comment succesfully	OK
20-	test_commentDeleteSuccesfully(self):	To test comment delete succesfully from Db	OK
21-	test_commentUserRelation(self):	To test comment user relation (user -> comment)	OK
22-	test_commentAreaRelation(self):	To test comment area relation (area -> comment)	OK
23-	test_createAClock(self):	To test area's clock created succesfully	OK
24-	test_clockAreaRelation(self):	To test clock area relation (area -> clock)	OK
25-	test_clockDelete(self):	To test area's clock deleted succesfully from Db	OK

26-	test_areaAddedSuccesfully(self):	To test Owner can add area succesfully	OK
27-	test_areaDeletedSuccesfully(self):	To test Owmer can delete area succesfully	OK
28-	test_userAreaRelation(self):	To test user area relation (user -> area)	OK
29-	test_imageAddedSuccesfully(self):	To test user can add img	OK
30-	test_userImageRelation(self):	To test user image relation (user -> image)	OK
31-	test_imageDeletedSuccesfully(self):	To test user can delete img from Db	OK
32-	test_userDeleteSuccesfully(self):	To test user delete her/his account from Db	OK

1-Database Testing *(15/15 Successful test)*

```
5
6   class userCreateDatabase(unittest.TestCase):
7       #To test user added succesfully to Db
8       def test_userAddedSuccesfully(self):
9           self.assertEqual(createNewUser(), 'User added succesfully!')
10
11      #To test user could comment succesfully
12      def test_commentAddedSuccesfully(self):
13          self.assertEqual(createNewComment(), 'Comment added succesfully!')
14
15      #To test comment delete succesfully from Db
16      def test_commentDeleteSuccesfully(self):
17          self.assertEqual(deleteAComment(), 'Comment deleted succesfully!')
18
19      #To test comment user relation (user -> comment)
20      def test_commentUserRelation(self):
21          self.assertEqual(getCheckCommentUserRelation(), True)
22
23      #To test comment area relation (area -> comment)
24      def test_commentAreaRelation(self):
25          self.assertEqual(getCheckCommentAreaRelation(), True)
26
27      #To test area's clock created succesfully
28      def test_createAClock(self):
29          self.assertEqual(createAClock(), 200)
30
31      #To test clock area relation (area -> clock)
32      def test_clockAreaRelation(self):
33          self.assertEqual(getCheckClockAreaRelation(), True)
34
35      #To test area's clock deleted succesfully from Db
36      def test_clockDelete(self):
37          self.assertEqual(deleteClock(), 200 )
38
39
```

```python

class Database(unittest.TestCase):
    #To test Owner can add area succesfully
    def test_areaAddedSuccesfully(self):
        self.assertEqual(createArea(), 'Area added succesfully!')

    #To test Owmer can delete area succesfully
    def test_areaDeletedSuccesfully(self):
        self.assertEqual(deleteArea(), 'Area deleted succesfully!')

    #To test user area relation (user -> area)
    def test_userAreaRelation(self):
        self.assertEqual(getCheckAreaUserRelation(),True)

    #To test user can add img
    def test_imageAddedSuccesfully(self):
        self.assertEqual(createImage(), 200)

    #To test user can delete img from Db
    def test_imageDeletedSuccesfully(self):
        self.assertEqual(deleteImage(), 200)

    #To test user image relation (user -> image)
    def test_userImageRelation(self):
        self.assertEqual(getCheckImageUserRelation(),True)

    #To test user delete her/his account from DbF
    def test_userDeleteSuccesfully(self):
        self.assertEqual(deleteAUser(), 'User deleted succesfully!')


if __name__ == '__main__':
    unittest.main()
```

Output of the Database Unit Tests

```
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\test\Project-main\FMAP> python databaseTest.py
D:\Program Files\Python\lib\site-packages\flask_sqlalchemy\__init__.py:833: FSADeprecationWarning
: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the
 future.  Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
...............
----------------------------------------------------------------------
Ran 15 tests in 0.159s

OK
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\test\Project-main\FMAP>
```

2-Flask-Web Page Testing *(12/12 Successful test)*

```python
    #For ensure that myprofile page was load correctlly
    def test_myprofil(self):
        tester = app.test_client(self)
        response = tester.get('/myprofil', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)

    #For ensure that contactus page was load correctlly
    def test_contactus(self):
        tester = app.test_client(self)
        response = tester.get('/contactus', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)

    #For ensure that aboutus page was load correctlly
    def test_aboutus(self):
        tester = app.test_client(self)
        response = tester.get('/aboutus', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)

    #For ensure that signin page was load correctlly
    def test_signin(self):
        tester = app.test_client(self)
        response = tester.get('/signin', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)

    #For ensure that signin page was load correctlly
    def test_editMyProfil(self):
        tester = app.test_client(self)
        response = tester.get('/editMyProfil', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)

    #For ensure that addFootballArea page was load correctlly
    def test_addFootballArea(self):
        tester = app.test_client(self)
        response = tester.get('/addFootballArea', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)
```

```python
    #For ensure that payment page was load correctlly
    def test_payment(self):
        tester = app.test_client(self)
        response = tester.get('/payment', content_type = 'html/text')
        self.assertEqual(response.status_code, 200)
```

Output of the FlaskTestCase Unit Tests

```
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\Project\FMAP> python flaskTestCase.py
D:\Program Files\Python\lib\site-packages\flask_sqlalchemy\__init__.py:833: FSADeprecati
onWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled
 by default in the future.  Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
............
----------------------------------------------------------------------
Ran 12 tests in 0.087s

OK
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\Project\FMAP>
```

3-User Login-Out Testing *(5/5 Successful test)*

```python
class requireUserLoginTest(unittest.TestCase):

    # Ensure that start application with no user -not yet login-
    def test_notToSignIn(self):
        self.assertEqual(getUser(), 0)

    # Ensure that signup page requires user login
    def test_signupLogin(self):
        tester = app.test_client(self)
        response = tester.get('/signup', content_type = 'html/text')
        self.assertEqual(getUser(), 0)

    # Ensure that addArea page requires user login
    def test_main_route_requires_addArea(self):
        tester = app.test_client(self)
        response = tester.get('/addArea', content_type = 'html/text')
        self.assertEqual(getUser(), 0)

    # Ensure that myprofil page requires user login
    def test_main_route_requires_myprofil(self):
        tester = app.test_client(self)
        response = tester.get('/myprofil', content_type = 'html/text')
        self.assertEqual(getUser(), 0)

    # Ensure that addFootballArea page requires user login
    def test_main_route_requires_addFootballArea(self):
        tester = app.test_client(self)
        response = tester.get('/addFootballArea', content_type = 'html/text')
        self.assertEqual(getUser(), 0)
```

Output of the RequireUserLoginTest Unit Tests

```
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\Project\FMAP> python requireUserLoginTest.py
                                                        D:\Program Files\Py
thon\lib\site-packages\flask_sqlalchemy\__init__.py:833: FSADeprecationWarning: SQLALCHE
MY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the
future.  Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
.....
----------------------------------------------------------------------
Ran 5 tests in 0.019s

OK
PS C:\Users\OmerF\OneDrive\Masaüstü\Proje\Project\FMAP>
```

4-Helper Functions

HELPERS FUCTIONS FOR UNIT TESTS

CreateArea Function

```python
def createArea():
    global check_area_user
    user = Users.query.filter_by(user_type = -1).first()
    if not user:
        return 'Error, there is not any user whose created for test! Look back createNewUser function.'

    OwnerName = 'TemelReis'
    AreaName = 'test_case'
    City = 'test_case'
    adress = 'test_case'
    OwnerNumber = 'test_case'
    owner_name = 'test_case'
    newArea = FootballArea(OwnerName = OwnerName,AreaName = AreaName,
        OwnerNumber=OwnerNumber,City = City,adress=adress,LikeCoun=0, users = user)

    db.session.add(newArea)
    db.session.commit()
    area = FootballArea.query.filter_by(adress = 'test_case').first()

    if area.users_id == user.id:
        check_area_user = True  #area and user connection is OK if check_area_user is True

    if area:                #If added there must be a user who has user_type -1
        return 'Area added succesfully!'

    return 'Area didnt add, Error!'
```

CreateNewUser Function

```python
def createNewUser():
    newOwner = Users(name = 'TemelReis',surname = 'owner1' ,username='owner1',
        email = 'owner1',password = 'owner1',user_type = -1)
    db.session.add(newOwner)
    db.session.commit()
    user = Users.query.filter_by(user_type = -1).first()

    if user:                #If added there must be a user who has user_type -1
        return 'User added succesfully!'

    return 'User didnt add, Error!'
```

DeleteArea Function

```python
def deleteArea():
    area = FootballArea.query.filter_by(adress = 'test_case').first()

    if not area:                #If added there must be a user who has user_type -1
        return 'Error, There is no area which is created by createArea function! Look for createArea function.'

    db.session.delete(area)
    db.session.commit()
    area = FootballArea.query.filter_by(adress = 'test_case').first()

    if not area:
        return 'Area deleted succesfully!'

    return 'Error, Area didnt deleted!'
```

DeleteAUser Function

```
def deleteAUser():
    user = Users.query.filter_by(user_type = -1).first()
    if not user:                #If didnt user add so didnt delete
        return 'Error, there is no user for delete! Look back to createNewUser function'
    db.session.delete(user)
    db.session.commit()

    user = Users.query.filter_by(user_type = -1).first()

    if not user:
        return 'User deleted succesfully!'
    return 'Error, User didnt deleted!'
```

DeleteImage Function

```
def deleteImage():

    image = Img.query.filter_by(mimetype = "this is an example!" ).first()
    db.session.delete(image)
    db.session.commit()
    image = Img.query.filter_by(mimetype = "this is an example!" ).first()


    if not image:
        return 200 # "Blob deleted succesfully!" ,

    if  image:
        return "Blob didnt delete!" , 400
```

ImageCreated Function

```
def createImage():
    global check_image_user

    user = Users.query.filter_by(user_type = -1).first()
    if not user:
        return 'no user' , 400

    filename = "this is an example!"
    mimetype = "this is an example!"
    img = "this is an example!"


    image = Img(img = img, mimetype = mimetype, name = filename, users = user)

    db.session.add(image)
    db.session.commit()

    image = Img.query.filter_by(mimetype = "this is an example!" ).first()

    if image.users_id == user.id:
        check_image_user = True

    if image:
        return 200 # "Blob created succesfully!" ,

    if not image:
        return "Blob didnt create!" , 400
```

Globals variable to use Database Testing and getters to Check one-to-many relation in this project

```
#global variables for check one to many relation with user
check_area_user = False
check_image_user = False
check_comment_area = False
check_clock_area = False
check_comment_user = False

#getters
def getCheckAreaUserRelation():
    global check_area_user
    return check_area_user

def getCheckImageUserRelation():
    global check_image_user
    return check_image_user

def getCheckCommentUserRelation():
    global check_comment_user
    return check_comment_user

def getCheckCommentAreaRelation():
    global check_comment_area
    return check_comment_area

def getCheckClockAreaRelation():
    global check_clock_area
    return check_clock_area
```

createAClock Function

```
def createAClock():
    global check_clock_area


    newArea = FootballArea(OwnerName = "OwnerName",AreaName = "AreaName",OwnerNumber="OwnerNumbe
        adress="adress",LikeCoun=0)
    newClock = Clocks(c10 = 0,owner_area = newArea, c11 = -99,c12 = 0,c13 = 0,c14 = 0,c15 = 0,c1
    db.session.add(newArea)
    db.session.add(newClock)
    db.session.commit()



    if newClock.owner_area == newArea:
        check_clock_area = True

    db.session.delete(newArea)
    db.session.commit()

    if newClock:
        return 200

    if not newClock:
        return 400
```

deleteAClock Function

```
56
57  def deleteClock():
58      clock = Clocks.query.filter_by(c11 = -99).first()
59      db.session.delete(clock)
60      db.session.commit()
61
62      clock = Clocks.query.filter_by(c11 = -99).first()
63
64      if clock:
65          return 400
66
67      if not clock:
68          return 200
69
70
```

createNewComment Function

```
84
85  def createNewComment():
86
87
88      newOwner = Users(name = 'TemelReis',surname = 'owner1' ,username='owner1',
89          email = 'owner1',password = 'owner1',user_type = -1)
90      db.session.add(newOwner)
91      db.session.commit()
92
93
94      newComment = Comment(owner_Com = -1,owner_User = newOwner.name,Com = 'newCommentCom')
95      db.session.add(newComment)
96      db.session.commit()
97
98      global check_comment_user
99      global check_comment_area
100
101     if newComment.owner_User == newOwner.name:
102         check_comment_user = True
103
104     if newComment.owner_Com == -1:
105         check_comment_area = True
106
107     db.session.delete(newOwner)
108     db.session.commit()
109
110     comment = Comment.query.filter_by(owner_Com = -1).first()
111
112     if comment:                #If added there must be a user who has user_type -1
113         return 'Comment added succesfully!'
114
115     return 'Comment didnt add, Error!'
116
117
```

deleteComment Function

```python
def deleteAComment():
    comment = Comment.query.filter_by(owner_Com = -1).first()
    if not comment:                #If didnt user add so didnt delete
        return 'Error, there is no comment for delete! Look back to createNewComment function'

    db.session.delete(comment)
    db.session.commit()

    comment = Comment.query.filter_by(owner_Com = -1).first()

    if not comment:
        return 'Comment deleted succesfully!'

    return 'Error, Comment didnt deleted!'
```