SOFTWARE ENGINEERING

SPRINT 1 - KICKOFF PROJECT

Functional & Non-Functional Requirements

In this section we briefly describe our functional and non-functional requirements.

Functional Requirements

As stated in the project proposal, we were committed to deliver **functional requirements** and we were committed in sprint one to deliver the following:

- Application for court owning institutes under the name **COURT OWNER** to manage their courts.
- Ability to view the **owner's profile**.
- Ability to **add a new court** to the owning institute.
- Ability to **view courts** associated with the owning institute.
- Ability to **add a booking** to a court.
- Ability to **view bookings** in a court.

Non-Functional Requirements

Non-functional requirements elicited were:

The Backend Structure

- Creating a database to:
  - hold the information on court owning institutes (COURT OWNERS). The database used is **SQL**.
  - hold information on courts owned by institutes.
  - hold information on reservations and their relations to courts and so their owner.

    - generally hold additional relations to be implemented and featured in the upcoming sprints.
    - Kickoff database is completely handled in the backend.
    - Each relation in the database schema is associated with a **repository**.
- Another database we used was the **Firebase Cloud**
    - Firebase cloud allows us to save the images and attachments issued not only in the court owner profile, but in the attachments handling in the upcoming sprints.
- Creating a model-controller structure
    - **Java Spring Boot Framework** is the spine of this structure.
    - Backend structure consists of
        - Controllers
        - Model
        - Repositories (Database)
        - Services
- Handling the requests issued by application users through controllers to support the functional requirements are done through these controllers
    - **Booking Agent Controller** - handles:
        - Requests of adding an online/offline reservation to a court.
        - ***(Court Owner Feature)*** Request to view reservations and their info.
    - **Court Owner Agent Controller** - handles:
        - Requests of getting courts associated with the owner.
        - Requests of creating/adding a new court to the owner's courts.
        - Requests of adding/editing the profile picture of the owner.
        - Requests of editing profile related info ***(not featured in this sprint)***.
    - **Login Controller** - handles:
        - Requests of logging into an existing account.
    - **Signup Controller** - handles:
        - Requests of signing a new account for court.
- Each controller has a servicer associated by its name and it interfaces an implementation to the services provided by each controller.
    - ***Agents & Services***
        - Booking Agent

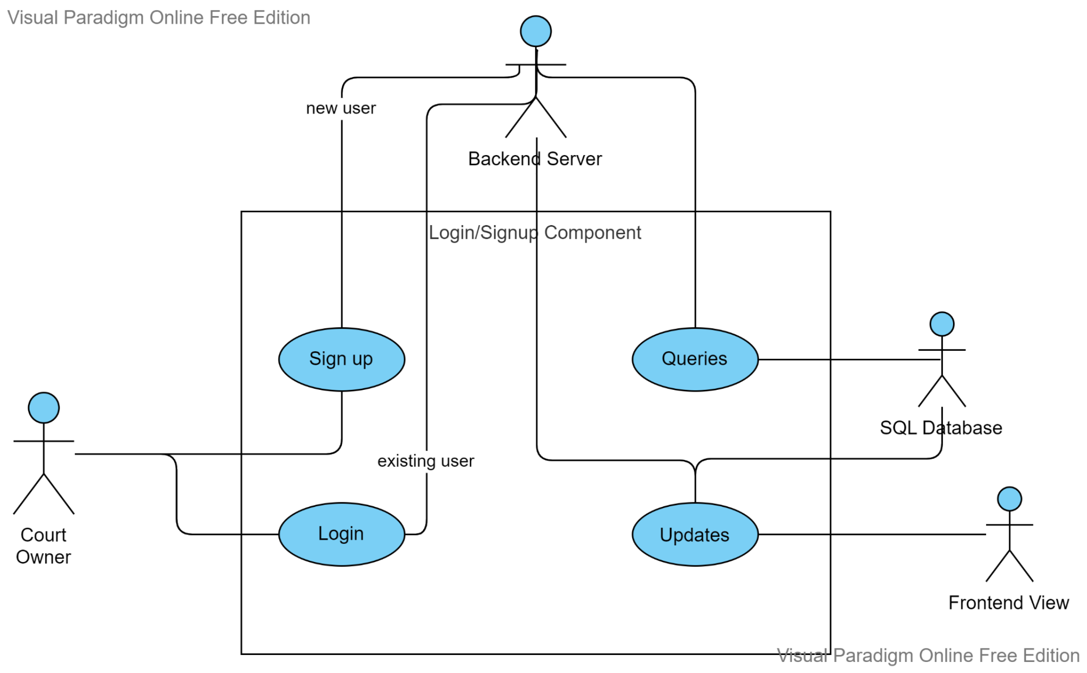
        - Court Owner Agent
        - Login Service
        - Reservations Service
        - Signup Service
        - Empty Json Response
        - Schedule Agent
- Our models are
    - ***Classes***
        - Court
        - CourtOwner
        - CourtSchedule
        - Player
        - Reservation
    - ***Enumerations***
        - CourtState - *Active, Out of order*.
        - ReservationState - *Expired, Pending, Active*.

The Frontend Structure

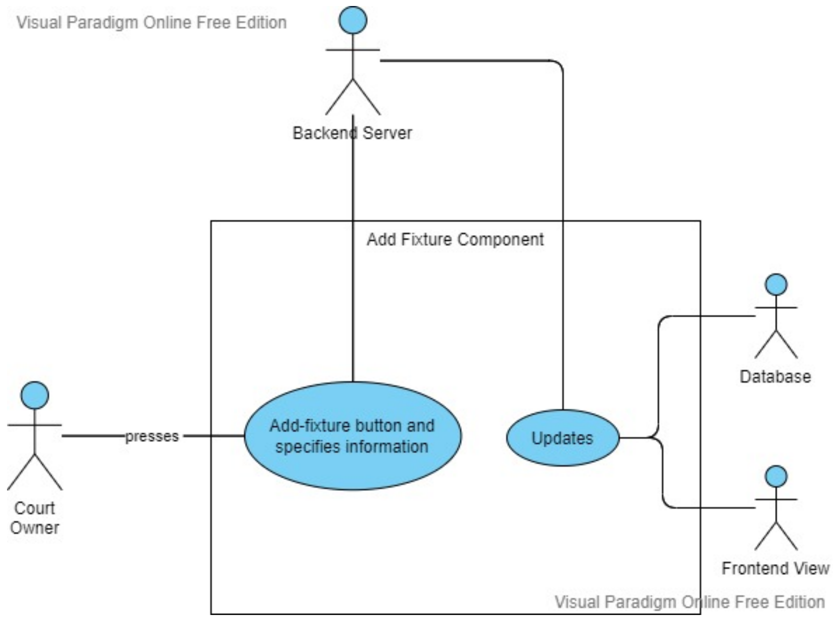
- To model the **view** of our backend **model-controller and repository** structure, we used **Flutter Framework** which uses **Dart Programming Language**.
- Main system components
    - Login Screen - *Contains the logic of logging in or registering a new account.*
    - Profile Screen - *Contains the logic of viewing owner profile info and courts associated with the owner.*
    - Announcements Screen - *Upcoming sprints feature.*
    - Reservations Screen - *Contains the logic of viewing reservations associated with each court on a dedicated date and adding a new offline fixture through the owner.*
- Other non-functional requirements were elicited in handling the validity of the data entry through services and validators in cooperation between both back and front ends.

Detailed Use Case Diagrams

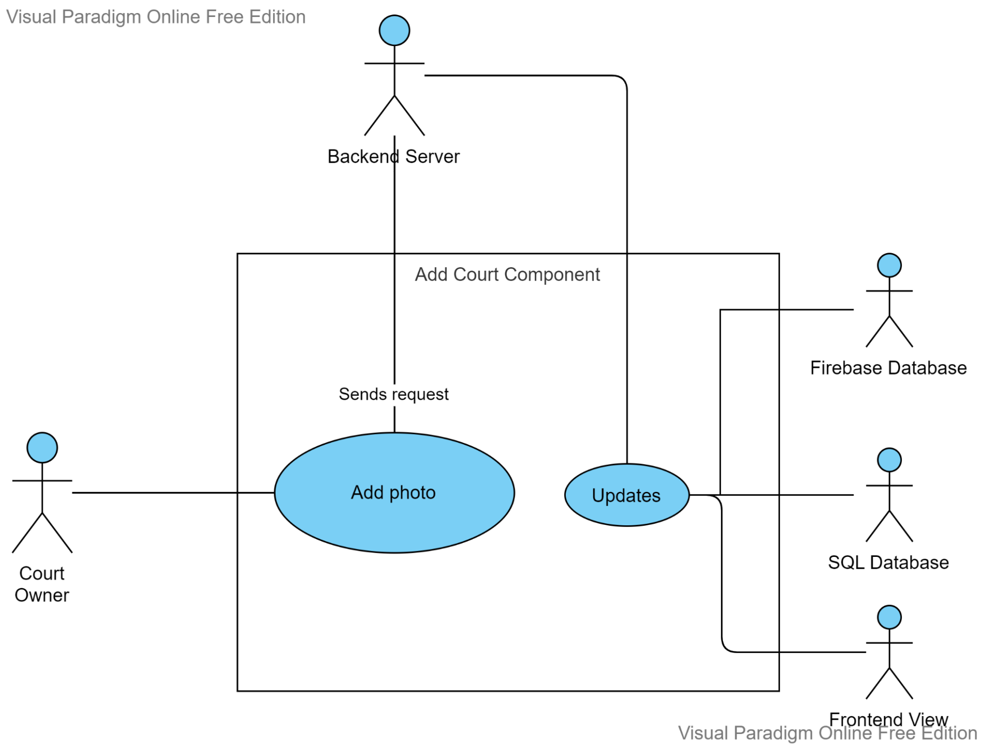
1. Use Case Diagram: Login/Signup Component



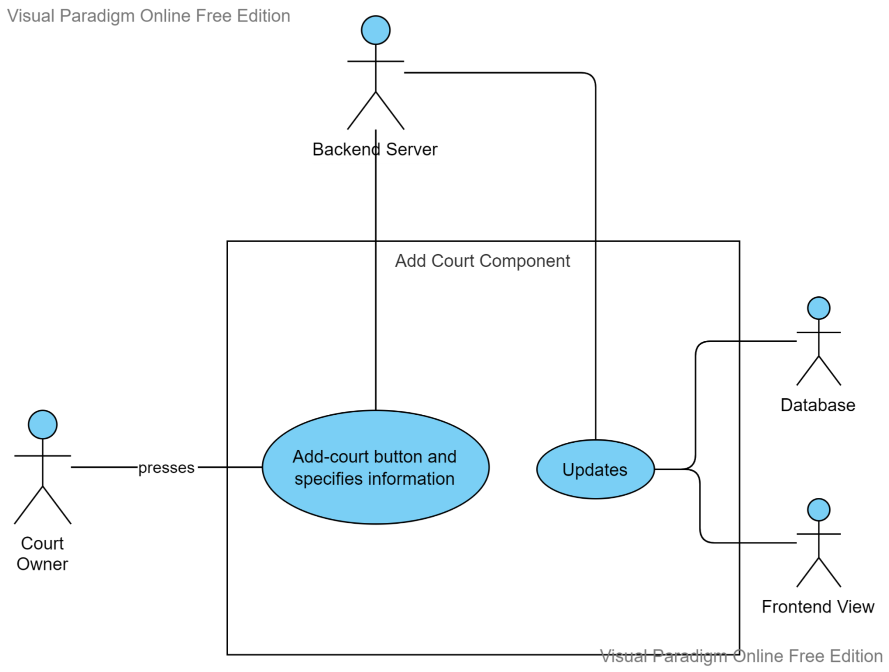
2. Use Case Diagram: Add Offline Reservation


Visual Paradigm Online Free Edition
Backend Server
Add Fixture Component
Database
presses
Court Owner
Add-fixture button and specifies information
Updates
Frontend View
Visual Paradigm Online Free Edition

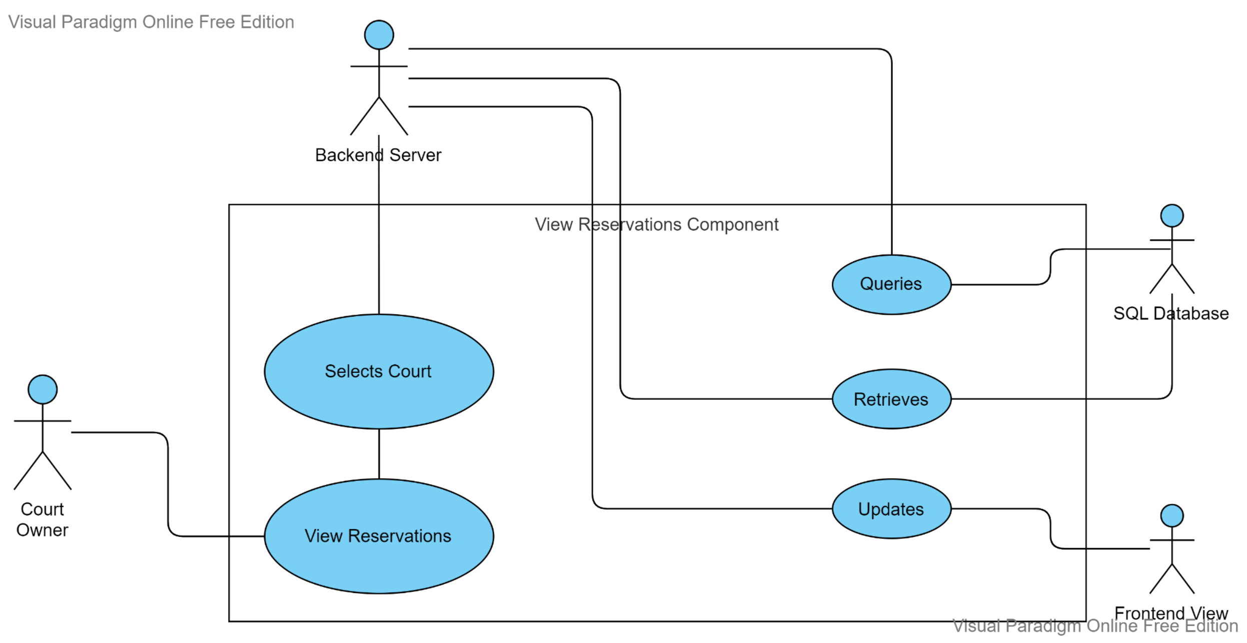
3. Use Case Diagram: Profile Interactions - Upload Profile Picture



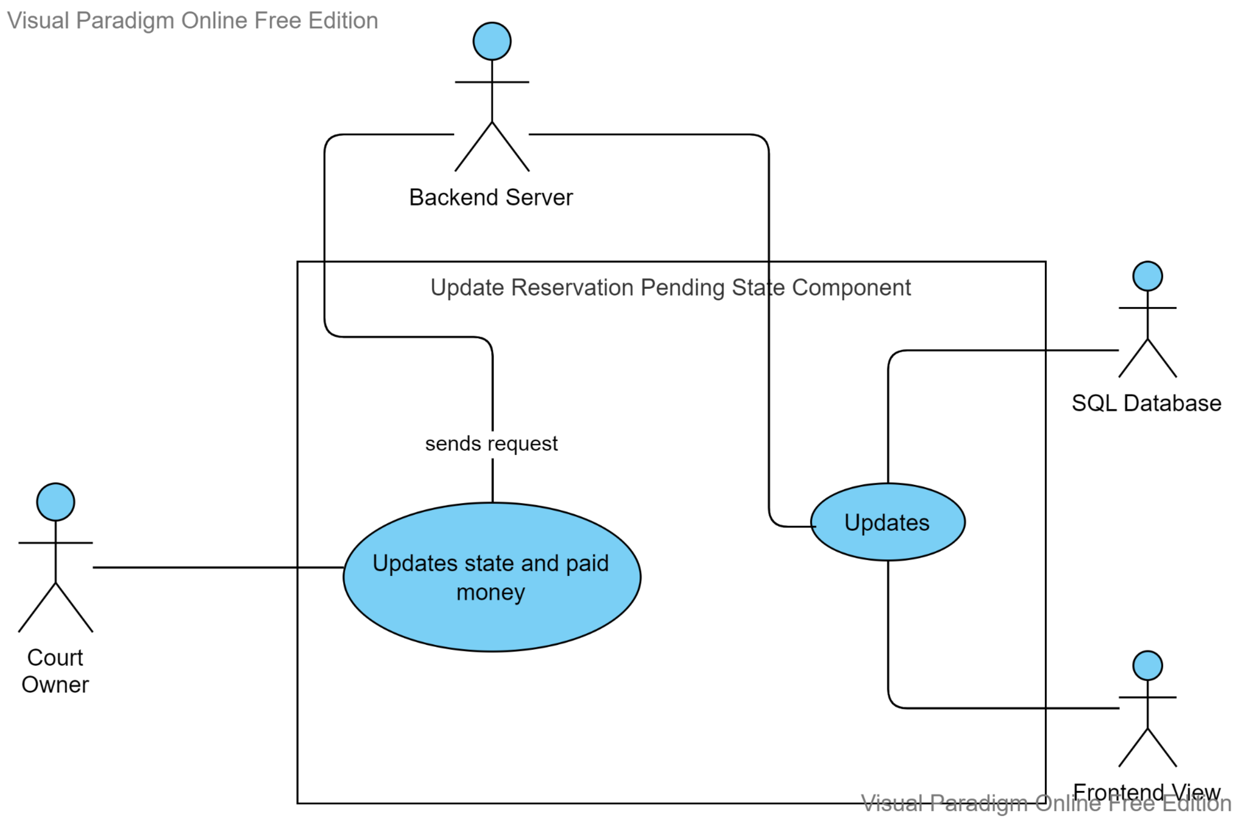
4. Use Case Diagram: Profile Interactions - Add a new court


Visual Paradigm Online Free Edition
Backend Server
Add Court Component
Database
presses
Add-court button and specifies information
Updates
Court Owner
Frontend View
Visual Paradigm Online Free Edition

5. Use Case Diagram: View Reservations

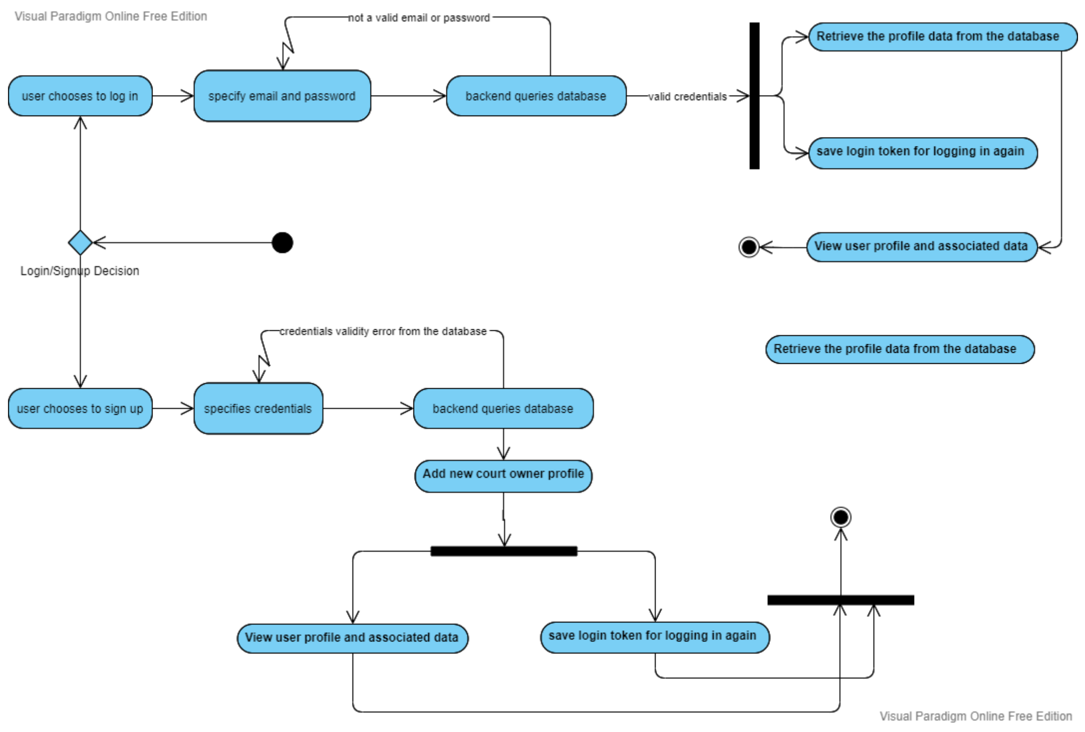


6. Use Case Diagram: Edit Reservation State

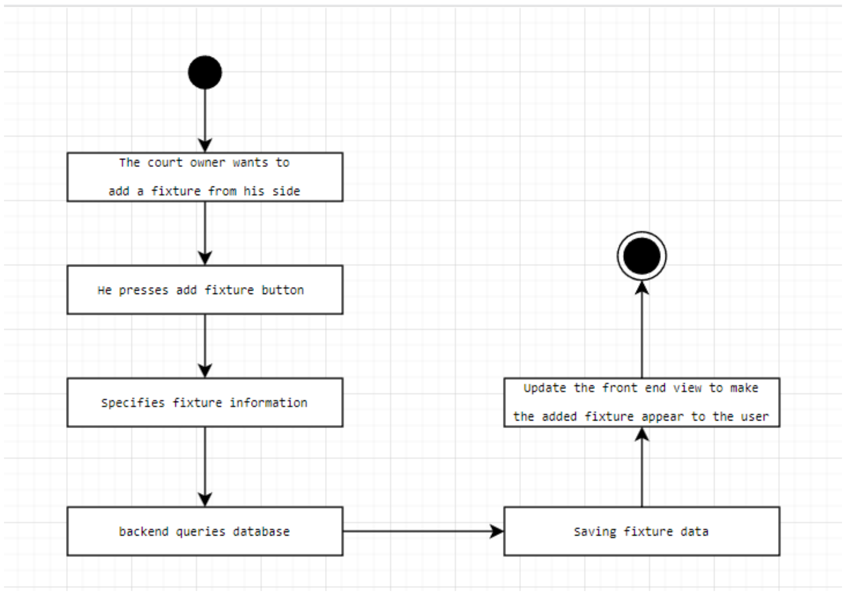

Visual Paradigm Online Free Edition
Backend Server
Update Reservation Pending State Component
SQL Database
sends request
Updates
Court Owner
Updates state and paid money
Frontend View
Visual Paradigm Online Free Edition

Activity Flow Diagrams

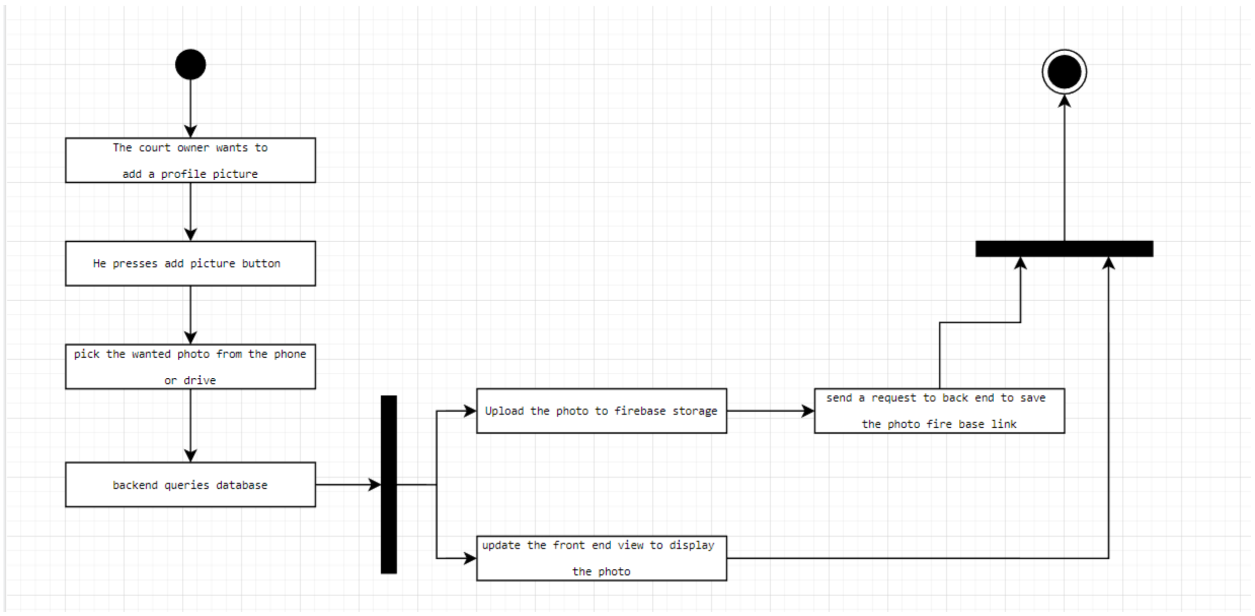
1. Activity Diagram: Login/Signup


Visual Paradigm Online Free Edition
not a valid email or password
user chooses to log in
specify email and password
backend queries database
valid credentials
Retrieve the profile data from the database
save login token for logging in again
View user profile and associated data
Login/Signup Decision
credentials validity error from the database
user chooses to sign up
specifies credentials
backend queries database
Retrieve the profile data from the database
Add new court owner profile
View user profile and associated data
save login token for logging in again
Visual Paradigm Online Free Edition

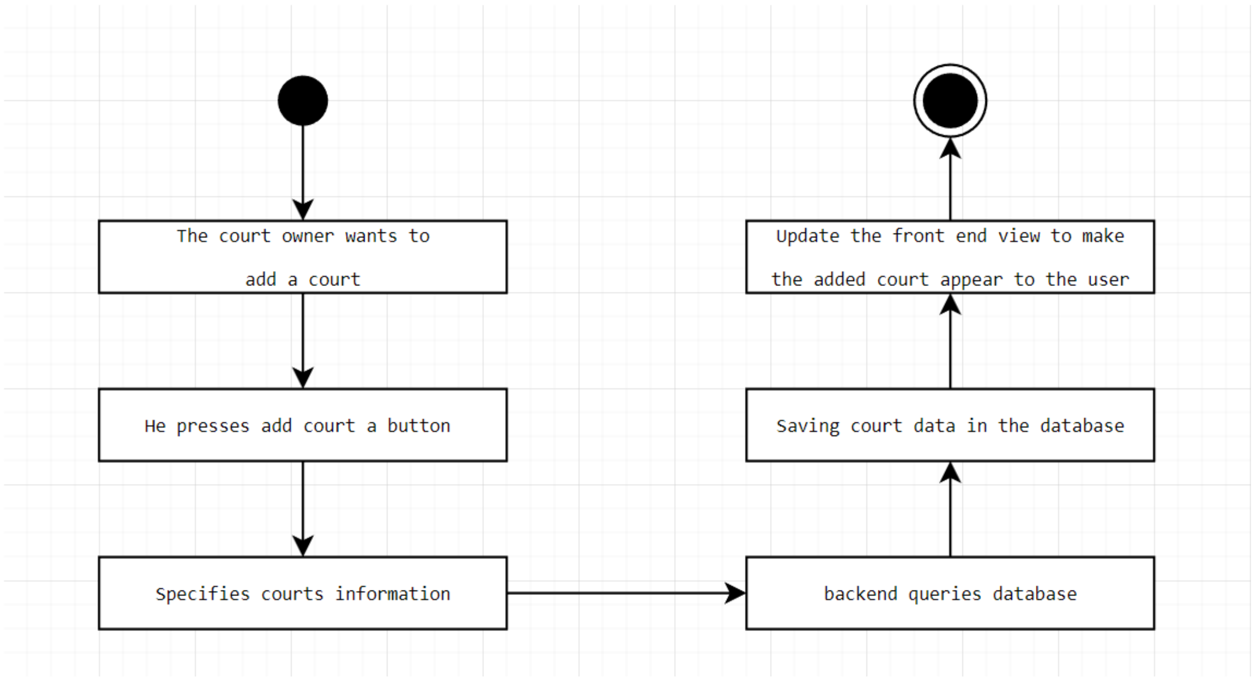
2. Activity Diagram: Add Offline Reservation


The court owner wants to add a fixture from his side
He presses add fixture button
Specifies fixture information
backend queries database
Saving fixture data
Update the front end view to make the added fixture appear to the user

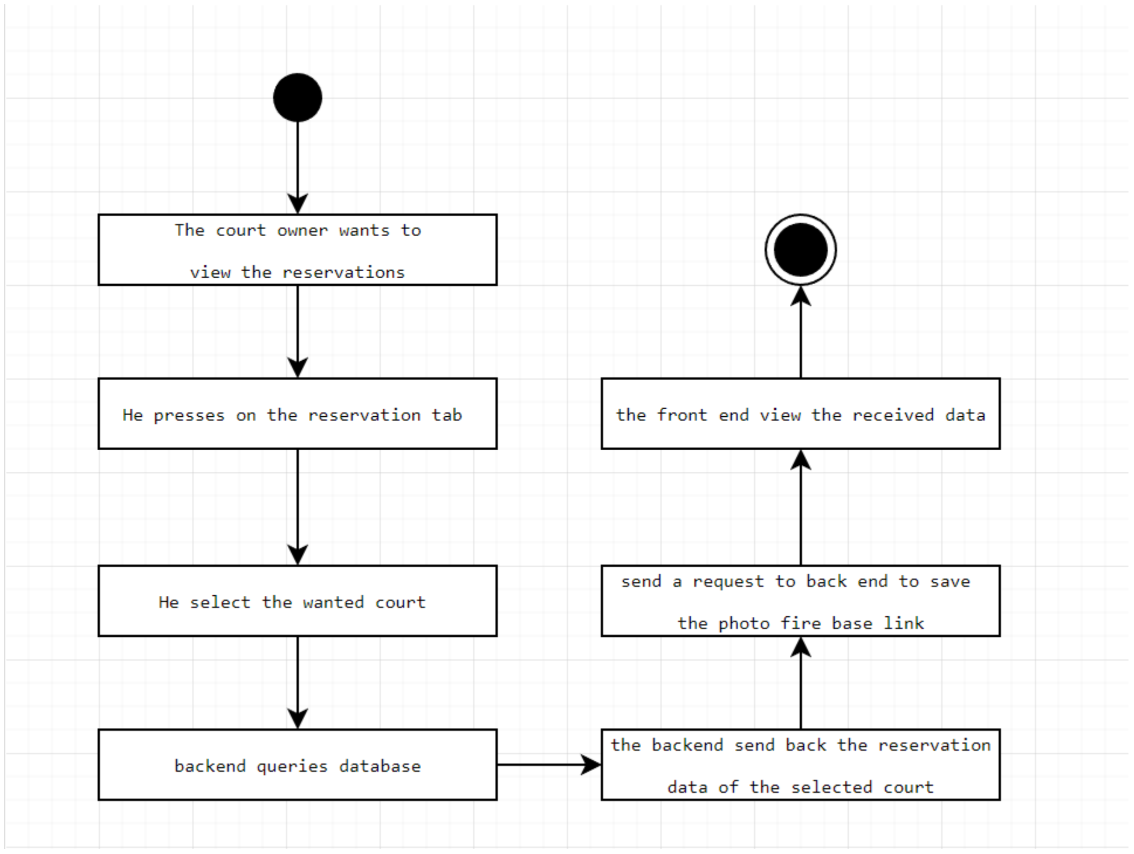
3. Activity Diagram: Profile Interactions - Upload Profile Picture


The court owner wants to add a profile picture
He presses add picture button
pick the wanted photo from the phone or drive
backend queries database
Upload the photo to firebase storage
send a request to back end to save the photo fire base link
update the front end view to display the photo

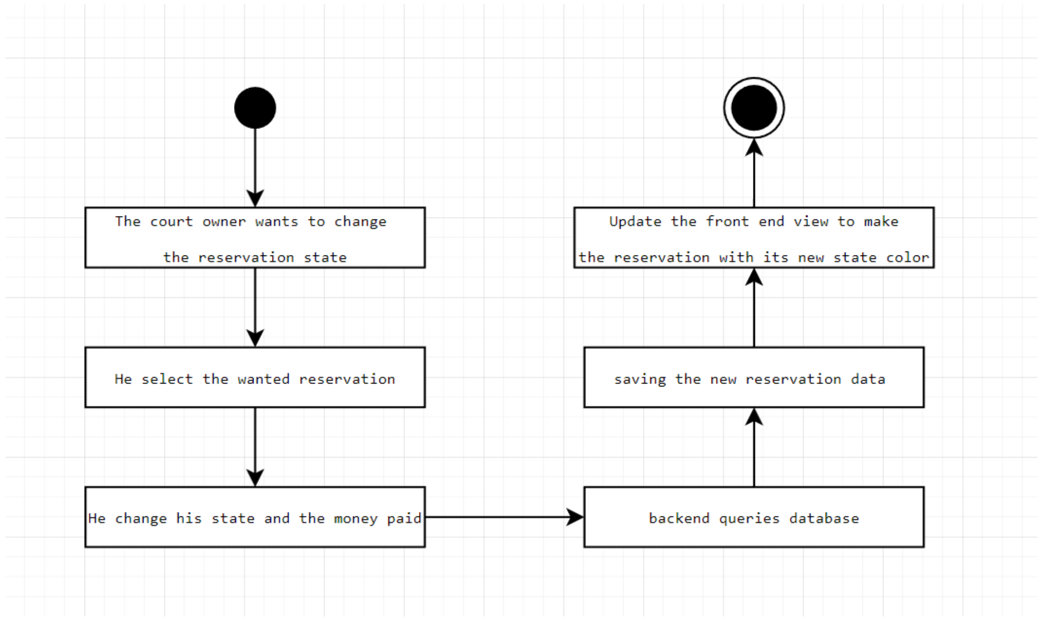
4. Activity Diagram: Profile Interactions - Add a new court


The court owner wants to add a court
He presses add court a button
Specifies courts information
backend queries database
Saving court data in the database
Update the front end view to make the added court appear to the user

5. Activity Diagram: View Reservations


The court owner wants to view the reservations
He presses on the reservation tab
He select the wanted court
backend queries database
the backend send back the reservation data of the selected court
send a request to back end to save the photo fire base link
the front end view the received data

6. Activity Diagram: Edit Reservation State


The court owner wants to change the reservation state
He select the wanted reservation
He change his state and the money paid
backend queries database
saving the new reservation data
Update the front end view to make the reservation with its new state color

Changes/Refactoring

No significant changes were issued after the initial project proposal.

Scrum Master Report

Sprint Planning Report

The sprint plan was represented in the following steps:

1. Requirement specification and creating the stories.
2. Creating tasks and assigning each task. Making sure that there are 2 or 3 people working in each story.
3. Developing and implementing each task will merging the related task iteratively
4. Testing and refactoring each task and component then the whole system
5. Fixing any bugs and reviewing the codes.

Please refer to *GIT/GITHUB REPORT* as it reports the pull requests, code reviews, etc.,.

Jira Report

Current Jira Roadmap

Milestone sprint listing the included requirements Jira Stories

Stories, tasks, and subtasks estimates

Stories

- As a court owner, I need to have the ability to log into the system.
- As a court owner, I need to have the ability to upload timetables of courts.
- As a court owner, I need to have the ability to add courts.
- As a court owner , I need to have ability to modify and cancel reservation.

Tasks & Subtasks Estimates

- Front End
- Back End

Task	Assigned-to	Estimated Time	Description
Backend package	Omar Khairat	1 day	Creating the packages on the backend. Controller, Repositories, Model and Services
Booking agent	Omar Khairat	2 days	
Design Courts	Zeyad Zidan	1 day	Design the court model and view in the frontend. Design court services and http requests
Design Reservations	Zeyad Zidan	2 days	Design the reservation model and view in the frontend. Design the add offline reservation feature.
Signup Page for Court Owner	Abdelaziz Mohamed	2 days	Design Signup page for Court owner
Create Court Request	Abdelrhman Gad	1day	Handling the requests of creating the court
Schedule Model	Abdelrhman Gad	1 day	The design of the class of the court schedule.

Merge Schedule with Reservations	Abdelrhman Gad	2 days	Manage the mapping between court schedule and reservation.
Profile Design	AbdEl-Rahman El-Sayed	3 days	Represent the profile data of the user after logging in
get all reservations on time interval at Schedule agent	Abdelrhman Gad	1 day	Filtering the reservations of a court to take the reservation on certain date
Add image to login. (subtask)	Abdelrhman Gad	0 day (1 hour)	Add the image link to the response of the login
Create signup response like login. (subtask)	Abdelrhman Gad	0 day (2 hours)	Editing the response of signup request to make it return the profile attributes.
getAttachments Controller (subtask)	Abdelrhman Gad	0 day (2 hours)	Making the get request to get and add the image to the CourtOwner
Reservation Model and Builder	Abdelrhman Gad	2 day	Build the model of the reservation.
Login Page for Court Owner	Abdelaziz Mohamed	2 days	Design Login page for CourtOwner
Court Model	Abdelaziz Mohamed	1 day	Build Model of Court
Navigation Bar	AbdEl-Rahman El-Sayed	1 day	Add a bar to navigate between tabs
Court Owner Model	Omar Khairat	1 day	
Backend Signup Services	Omar Khairat	1 day	
Backend Signup Controller	Omar Khairat	1 day	

Implement stay logged in	AbdEl-Rahman El-Sayed	2 days	Saved the login data on the user phone in order not to make the user login every time he open the app
implement log out	AbdEl-Rahman El-Sayed	0 day	Log out from the app
Showing Attachments	AbdEl-Rahman El-Sayed	2 days	Make the user able to select image from the phone and make it appear every time he login
Attachments	Abdelaziz Mohamed	2 days	Store image link in back end and connect flutter with firebase
CourtOwner Agent	Omar Khairat	2 days	
Backend Login Controller	Abdelaziz Mohamed	1 day	Making the post request to get information of login of the court
Backend Login Services	Abdelaziz Mohamed	1 day	Check the information of the CourtOwner if it is found.

Git/Github Report

Branches

Main branch: main

Milestone: Sprint1

Features branches:

- Sprint1_Backend_andCourtModel

- ReservationModelBuilder
- Frontend_Profile_initiation
- CourtOwner-Signup
- Sprint1_frontend_login
- Frontend-profile
- Fixtures
- BookingAgent
- Sprint1_edit_login

Test and reviewing branches

- Test-Branch
- RequestHandling

Pull Requests

- **Pull Request #1 - Package Creation**.
  - Creating the architecture of the system projects (Flutter Frontend project & Spring Boot Backend project) including their packages.
- **Pull Request #5 -** Reservation and Court Models creation in the backend server.
- **Pull Request #8 -** Schedule model and services in the backend server and merge with the reservation.
- **Pull Request #9 -** Court Owner, Signup, and Login models and services in the backend server.
- **Pull Request #10 -** Frontend Signup and Login component creations.
- **Pull Request #11 -** Frontend Profile component creation.
- **Pull Request #12 -** Frontend Fixtures, which are the frontend model of the reservations backend model, representing the reservations associated with each court.
- **Pull Request #13 -** The Backend's booking agent, which handles reservations pending status, booking a reservation, deletion of a reservation and viewing the reservations for specified court).
- **Pull Request #15 -** Login frontend component editing after reviewing and testing.

- **Pull Request #16 -** Login and Signup frontend and backend components and models requests handling.
- **Pull Request #19 -** Fixtures phase 2, which was the refactoring of fixtures after code reviewing and fixing errors and bugs.
- **Pull Request #20 -** Handling requests for attachments and major UI enhancements.
- **Pull Request #24 -** Handling the rest of the components requests.
- **Pull Request #25 -** Test branches merge after testing, reviewing, and resolving conflicts, errors, and bugs.
- **Pull Request #27 -** Requests handling and merging all branches into one branch under the name Sprint1.

Code Review

- **Creating Signup backend functionalities**

  Author: Omar Khairat

  Review: Abdelaziz Mohamed

  Created: 28/11/2022

  Last Update:6/12/2022

  Status: Done

- **Creating login backend functionalities**

  Author: Abdelaziz Mohamed

  Review: Omar Khairat

  Created: 28/11/2022

  Last Update:6/12/2022

  Status: Done

- **Creating CourtOwnerAgent functionalities**

  Author: Abdelrahman Gad - Omar Khairat

Review: Abdelaziz Mohamed

Created: 28/11/2022

Last Update:6/12/2022

Status: Done

- **Creating BookingAgent functionalities**

  Author: Abdelrahman Gad - Omar Khairat

  Review: Abdelaziz Mohamed

  Created: 28/11/2022

  Last Update:6/12/2022

  Status: Done

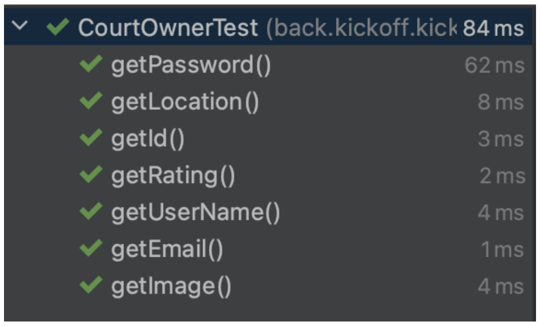
Unit Testing Report

Unit tests are created to test all our components in the following backend packages:

- **Model package**

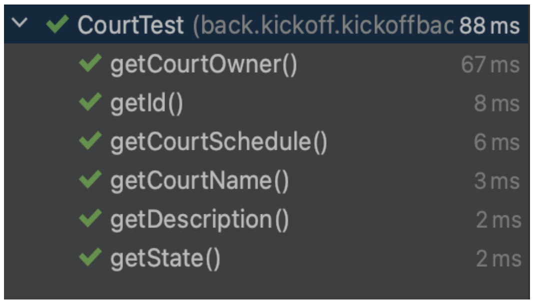
  Unit tests are created in that package to ensure the attributes of each model class are persisted and stored correctly as they are essential in our business logic. The classes that are attributes are tested are the following:

  - *CourtOwner*



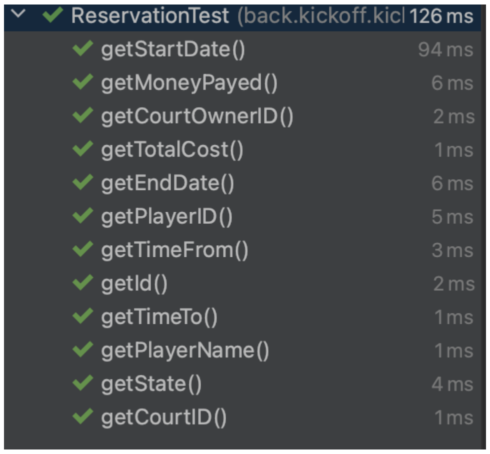
```
✓ CourtOwnerTest (back.kickoff.kick 84 ms
    ✓ getPassword()               62 ms
    ✓ getLocation()                8 ms
    ✓ getId()                      3 ms
    ✓ getRating()                  2 ms
    ✓ getUserName()                4 ms
    ✓ getEmail()                   1 ms
    ✓ getImage()                   4 ms
```

  - *Court*



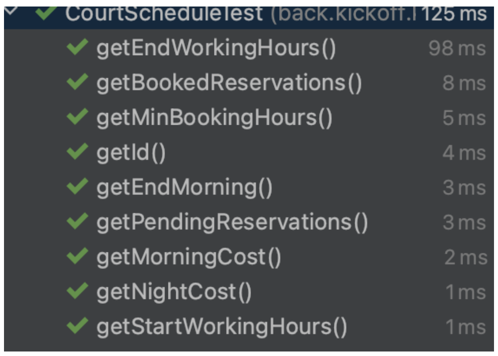
```
✓ CourtTest (back.kickoff.kickoffbac 88 ms
    ✓ getCourtOwner()            67 ms
    ✓ getId()                     8 ms
    ✓ getCourtSchedule()          6 ms
    ✓ getCourtName()              3 ms
    ✓ getDescription()            2 ms
    ✓ getState()                  2 ms
```

- *Reservation*



```
✓ ReservationTest (back.kickoff.kicl 126 ms
    ✓ getStartDate()        94 ms
    ✓ getMoneyPayed()        6 ms
    ✓ getCourtOwnerID()      2 ms
    ✓ getTotalCost()         1 ms
    ✓ getEndDate()           6 ms
    ✓ getPlayerID()          5 ms
    ✓ getTimeFrom()          3 ms
    ✓ getId()                2 ms
    ✓ getTimeTo()            1 ms
    ✓ getPlayerName()        1 ms
    ✓ getState()             4 ms
    ✓ getCourtID()           1 ms
```

- *CourtSchedule*

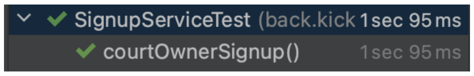


```
✓ CourtScheduleTest (back.kickoff.l 125 ms
    ✓ getEndWorkingHours()          98 ms
    ✓ getBookedReservations()        8 ms
    ✓ getMinBookingHours()           5 ms
    ✓ getId()                        4 ms
    ✓ getEndMorning()                3 ms
    ✓ getPendingReservations()       3 ms
    ✓ getMorningCost()               2 ms
    ✓ getNightCost()                 1 ms
    ✓ getStartWorkingHours()         1 ms
```

- **Service package**

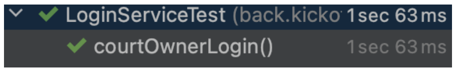
Unit tests are created in that package to ensure that each component in the service package is doing its required functionality efficiently and provide its necessary services and interacting with other packages correctly in the following test cases . The components that are attributes are tested are the following:

- *SignupService*



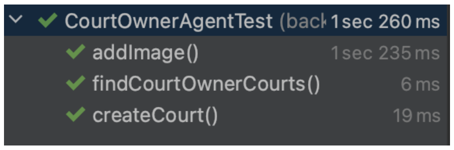
```
✓ SignupServiceTest (back.kick 1 sec 95 ms
    ✓ courtOwnerSignup()          1 sec 95 ms
```

- *LoginService*



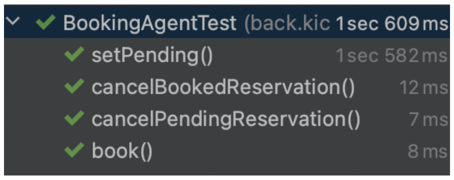
```
✓ LoginServiceTest (back.kicko 1 sec 63 ms
    ✓ courtOwnerLogin()         1 sec 63 ms
```

- *CourtOwnerAgent*



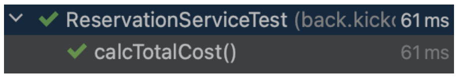
```
✓ CourtOwnerAgentTest (back 1 sec 260 ms
    ✓ addImage()               1 sec 235 ms
    ✓ findCourtOwnerCourts()           6 ms
    ✓ createCourt()                   19 ms
```

- *BookingAgent*



```
✓ BookingAgentTest (back.kic 1 sec 609 ms
    ✓ setPending()              1 sec 582 ms
    ✓ cancelBookedReservation()        12 ms
    ✓ cancelPendingReservation()        7 ms
    ✓ book()                            8 ms
```

- *ReservationService*

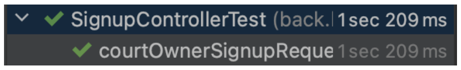


```
✓ ReservationServiceTest (back.kick 61 ms
    ✓ calcTotalCost()                 61 ms
```

- **Controller package**

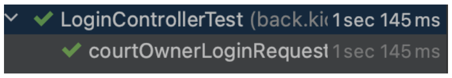
  Unit tests are created in that package to ensure that each controller interacts with desired services to provide correct responses towards the user requests coming from the view layer with data provided. The components that are attributes are tested are the following:

- *SignupController*



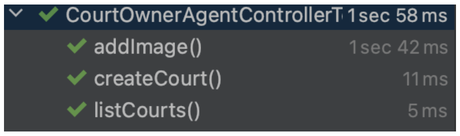
```
✓ SignupControllerTest (back.l 1 sec 209 ms
    ✓ courtOwnerSignupReque 1 sec 209 ms
```

- *LoginController*



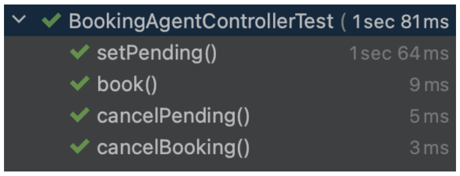
```
✓ LoginControllerTest (back.ki 1 sec 145 ms
  ✓ courtOwnerLoginRequest 1 sec 145 ms
```

- *CourtOwnerAgentController*



```
✓ CourtOwnerAgentControllerT 1 sec 58 ms
  ✓ addImage()               1 sec 42 ms
  ✓ createCourt()                  11 ms
  ✓ listCourts()                    5 ms
```

- *BookingAgentController*



```
✓ BookingAgentControllerTest ( 1 sec 81 ms
  ✓ setPending()                1 sec 64 ms
  ✓ book()                             9 ms
  ✓ cancelPending()                    5 ms
  ✓ cancelBooking()                    3 ms
```

Collaborators & Links

Scrum master for sprint one was ***Abdel-Rahman El-Sayed Gad El-Sayed***.

Collaborators

Zeyad Zidan	**19015709**	**Github**
Abdelrahman Aly	**19015893**	**Github**
Abdelrahman Gad	**19015894**	**Github**
Abdel-Aziz Mohammed	**19015941**	**Github**
Omar Khairat	**19016063**	**Github**

Project

- **Github Repository**
- **JIRA Repository**
- **Firebase Repository**