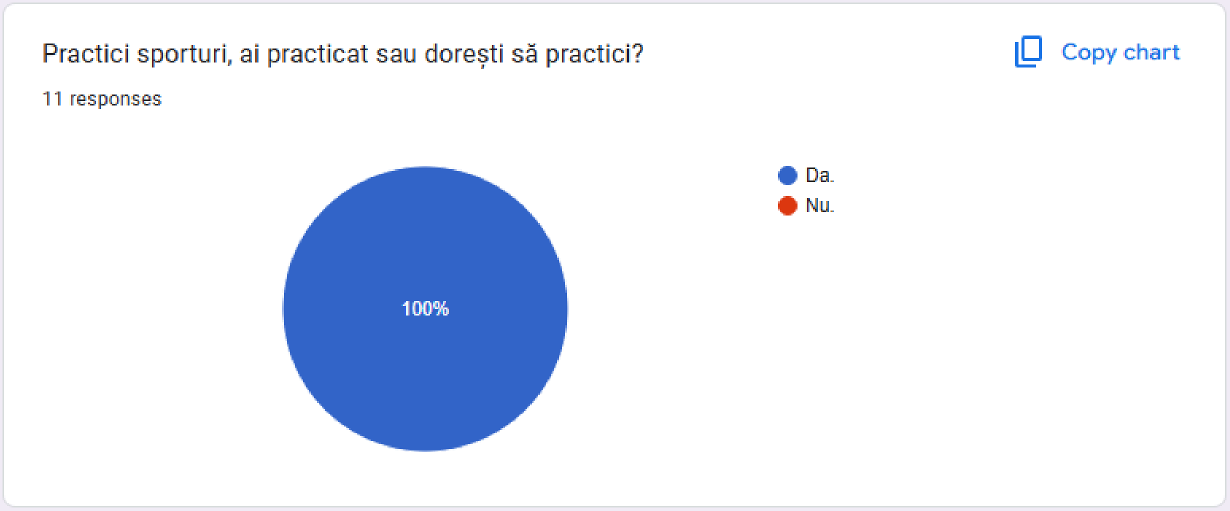
Cloud Computing - Final Project

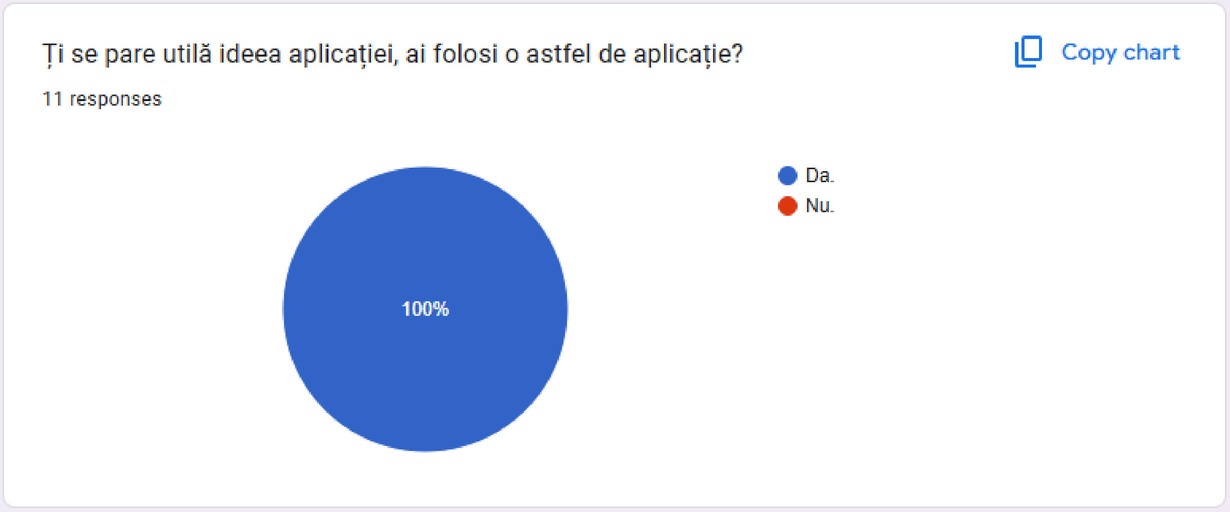
Application for Managing the Availability of Different Sport Facilities

Case Study:

- **Conclusions:**
  - On a survey on 11 random people from different regions of Romania, 100% of them said they already did practice some sports, they do, or they would like to do and also 100% of them said they like the idea of the application and that they would use it.
- **Proof:**

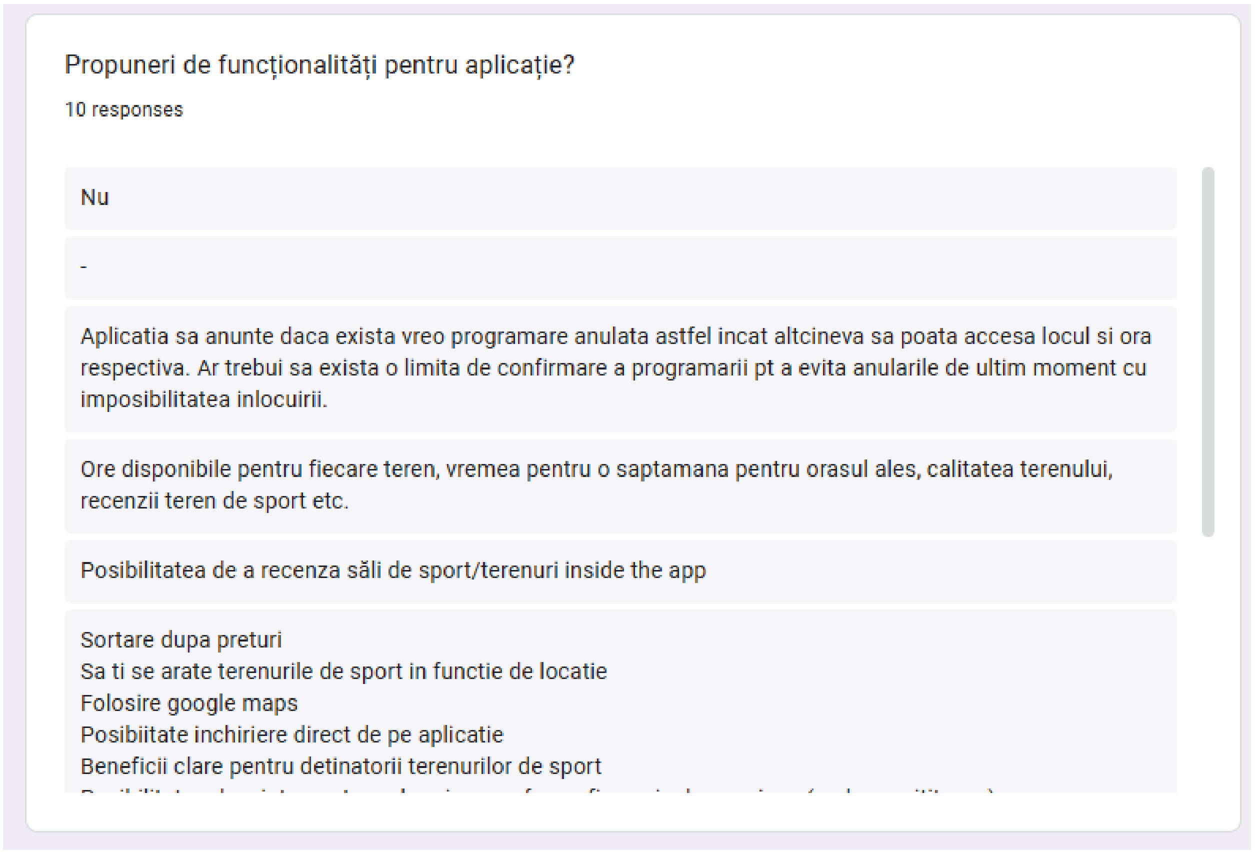

Practici sporturi, ai practicat sau dorești să practici?

11 responses


Ți se pare utilă ideea aplicației, ai folosi o astfel de aplicație?

11 responses

- **Useful User Proposals:**



Propuneri de funcționalități pentru aplicație?

10 responses

Nu

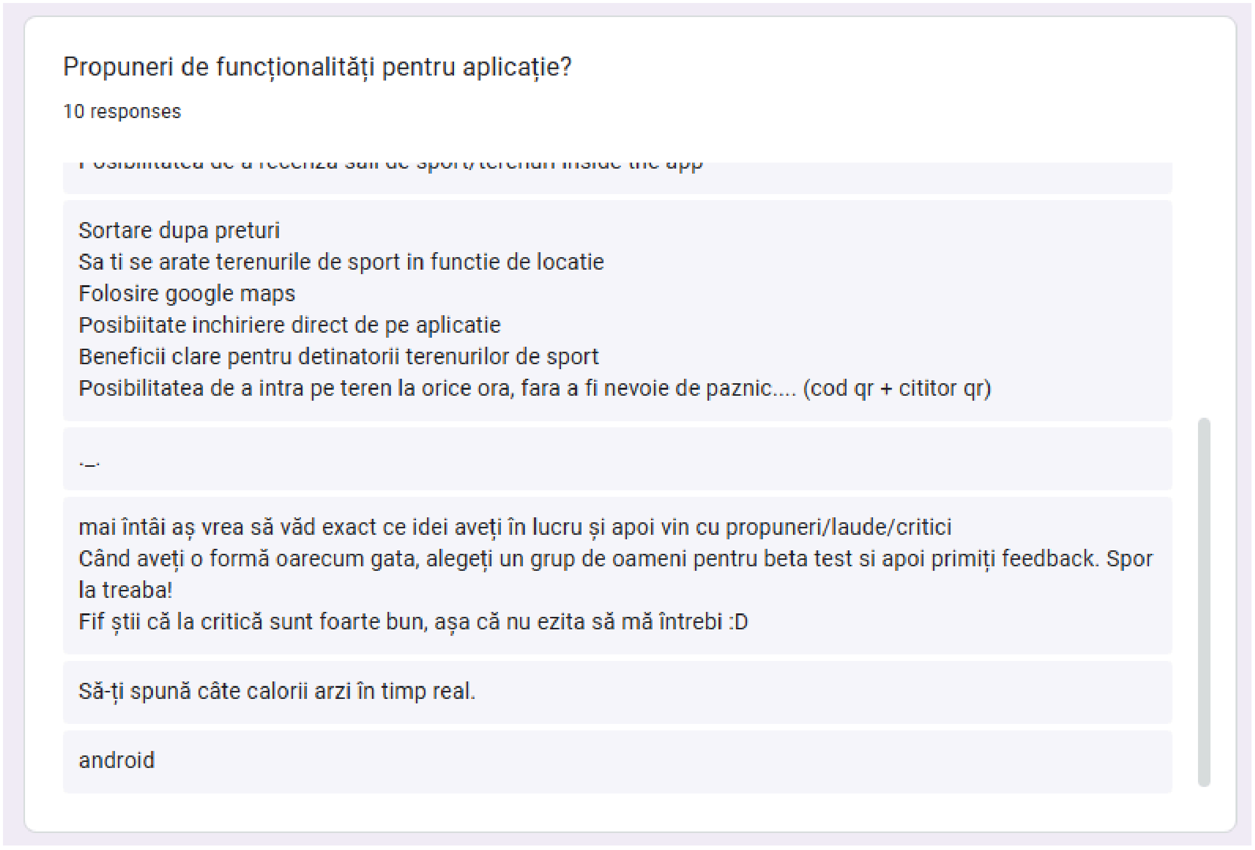
-

Aplicatia sa anunte daca exista vreo programare anulata astfel incat altcineva sa poata accesa locul si ora respectiva. Ar trebui sa existe o limita de confirmare a programarii pt a evita anularile de ultim moment cu imposibilitatea inlocuirii.

Ore disponibile pentru fiecare teren, vremea pentru o saptamana pentru orasul ales, calitatea terenului, recenzii teren de sport etc.

Posibilitatea de a recenza săli de sport/terenuri inside the app

Sortare dupa preturi
Sa ti se arate terenurile de sport in functie de locatie
Folosire google maps
Posibiitate inchiriere direct de pe aplicatie
Beneficii clare pentru detinatorii terenurilor de sport



Propuneri de funcționalități pentru aplicație?

10 responses

Sortare dupa preturi
Sa ti se arate terenurile de sport in functie de locatie
Folosire google maps
Posibiitate inchiriere direct de pe aplicatie
Beneficii clare pentru detinatorii terenurilor de sport
Posibilitatea de a intra pe teren la orice ora, fara a fi nevoie de paznic.... (cod qr + cititor qr)

-_-

mai întâi aș vrea să văd exact ce idei aveți în lucru și apoi vin cu propuneri/laude/critici
Când aveți o formă oarecum gata, alegeți un grup de oameni pentru beta test si apoi primiți feedback. Spor la treaba!
Fif știi că la critică sunt foarte bun, așa că nu ezita să mă întrebi :D

Să-ți spună câte calorii arzi în timp real.

android

Technical Details of Competitor Products:

1. ClassPass

- **Architecture**:
    - Multi-tenant, microservices-based backend to isolate functionalities (user accounts, bookings, payments).
    - Responsive web client plus native iOS/Android apps.
    - Real-time availability updates via message queues (e.g. Kafka) and push notifications.
- **Technologies**:
    - **Front-end**: React (Web), Swift (iOS), Kotlin (Android)
    - **Back-end**: Java/Spring Boot microservices, Node.js for edge/API gateway
    - **Data**: PostgreSQL for relational data, Redis for caching, Elasticsearch for search
    - **Infra/DevOps**: AWS (EKS, Lambda, RDS), Docker/Kubernetes, GitHub Actions CI/CD
- **Marketing**:
    - B2B partnerships with studios & gyms (revenue-share model)
    - Heavy app-store optimization (targeting “fitness classes near me”)
    - Social-media influencer campaigns and referral credits
    - Seasonal promotions (“first month free,” “holiday pass”)

2. Skedda

- **Architecture**:
    - Monolithic web-app core (can be broken into services as scale demands).
    - Purely browser-based SaaS with real-time calendar updates via WebSockets.
    - Embedded booking widgets for partner sites.

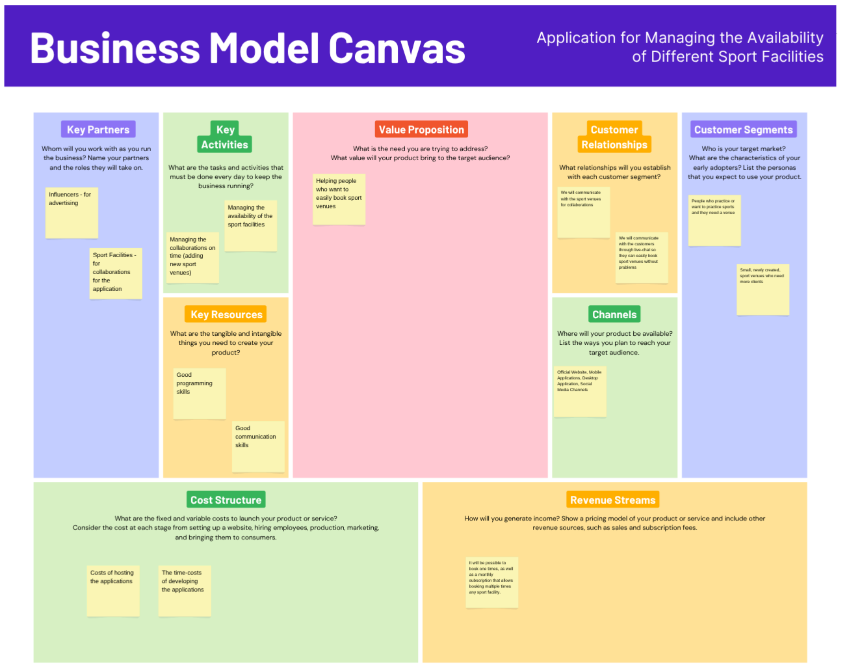
- **Technologies**:
    - **Front-end**: Vue.js, Vuetify components
    - **Back-end**: Node.js/Express
    - **Data**: PostgreSQL, Redis for session management
    - **Infra/DevOps**: Heroku (initially), now AWS EC2/RDS; Docker for local dev
- **Marketing**:
    - SEO-driven content targeting facility managers ("how to schedule community sports halls")
    - Free-tier for small venues, upsell to premium plans (advanced rules & integrations)
      Partnerships with municipal councils & university sports departments
    - Webinars and case-studies showcasing large installations

3. CourtReserve

- **Architecture**:
    - Modular SaaS platform with separate modules for reservations, memberships, billing, and events.
    - Offers both hosted web portal and on-premise API integration for larger clubs.
    - Synchronous booking engine to prevent double-booking, with audit logging.
- **Technologies**:
    - **Front-end**: Angular
    - **Back-end**: .NET Core services, RESTful APIs
    - **Data**: Microsoft SQL Server, Azure Redis Cache
    - **Infra/DevOps**: Microsoft Azure (App Service, SQL Database), Azure DevOps pipelines
- **Marketing**:
    - Direct sales outreach to tennis and pickleball clubs

    ○ Trade-show presence at industry expos (USTA, Pickleball conventions)
    ○ Affiliate program with club-management associations
    ○ Targeted email campaigns highlighting ROI and member-engagement features

Business Canvas:



Business Model Canvas

Application for Managing the Availability of Different Sport Facilities

Key Partners

Whom will you work with as you run the business? Name your partners and the roles they will take on.

Influencers - for advertising

Sport Facilities - for collaborations for the application

Key Activities

What are the tasks and activities that must be done every day to keep the business running?

Managing the availability of the sport facilities

Managing the collaborations on time (adding new sport venues)

Key Resources

What are the tangible and intangible things you need to create your product?

Good programming skills

Good communication skills

Value Proposition

What is the need you are trying to address?
What value will your product bring to the target audience?

Helping people who want to easily book sport venues

Customer Relationships

What relationships will you establish with each customer segment?

We will communicate with the sport venues for collaborations

We will communicate with the customers through live-chat so they can easily book sport venues without problems

Channels

Where will your product be available? List the ways you plan to reach your target audience.

Official Website, Mobile Applications, Desktop Application, Social Media Channels

Customer Segments

Who is your target market?
What are the characteristics of your early adopters? List the personas that you expect to use your product.

People who practice or want to practice sports and they need a venue

Small, newly created, sport venues who need more clients

Cost Structure

What are the fixed and variable costs to launch your product or service?
Consider the cost at each stage from setting up a website, hiring employees, production, marketing, and bringing them to consumers.

Costs of hosting the applications

The time-costs of developing the applications

Revenue Streams

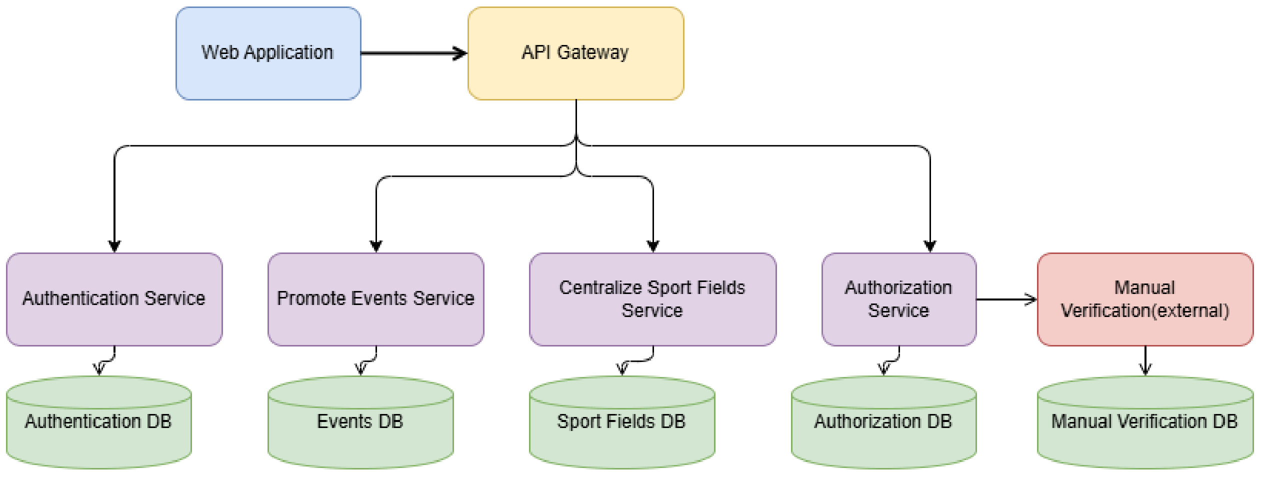
How will you generate income? Show a pricing model of your product or service and include other revenue sources, such as sales and subscription fees.

It will be possible to book one times, as well as a monthly subscription that allows booking multiple times any sport facility.

Technologies:

- VUE.js : frontend
- Spring Boot (java) : backend (OOP dev)
- REST/JSON over HTTPS : communication
- Azure SQL Database (relational) : storage
- Stripe/PayPal (SDKs) : payment
- OAuth2 JWT : identity
- Azure Cloud Services : deployment, gateway (Azure API Management)

Architectural diagram:



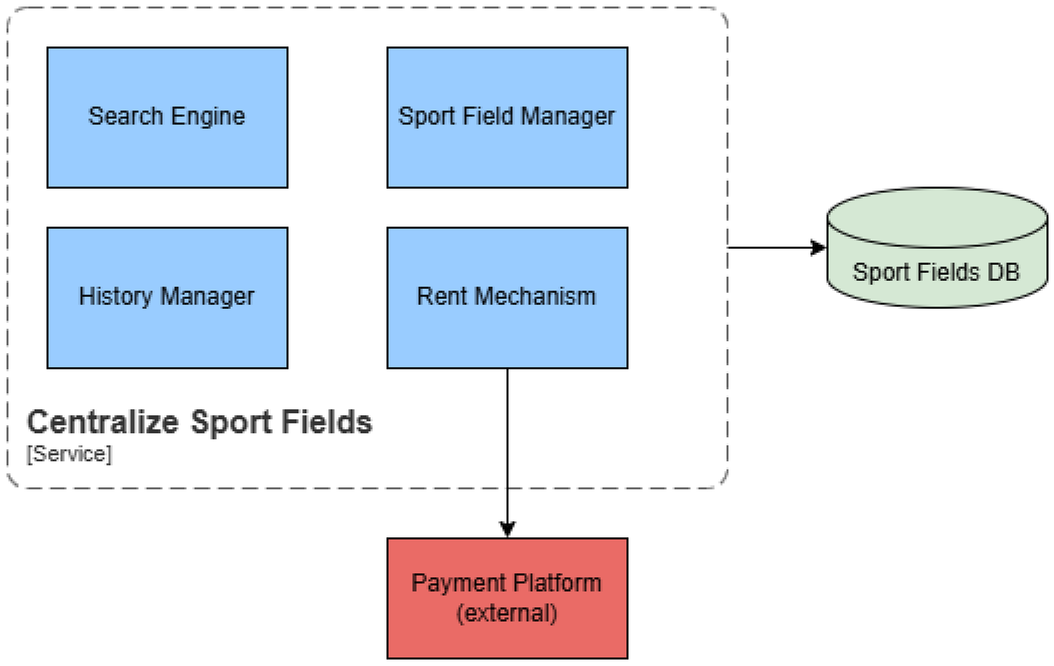
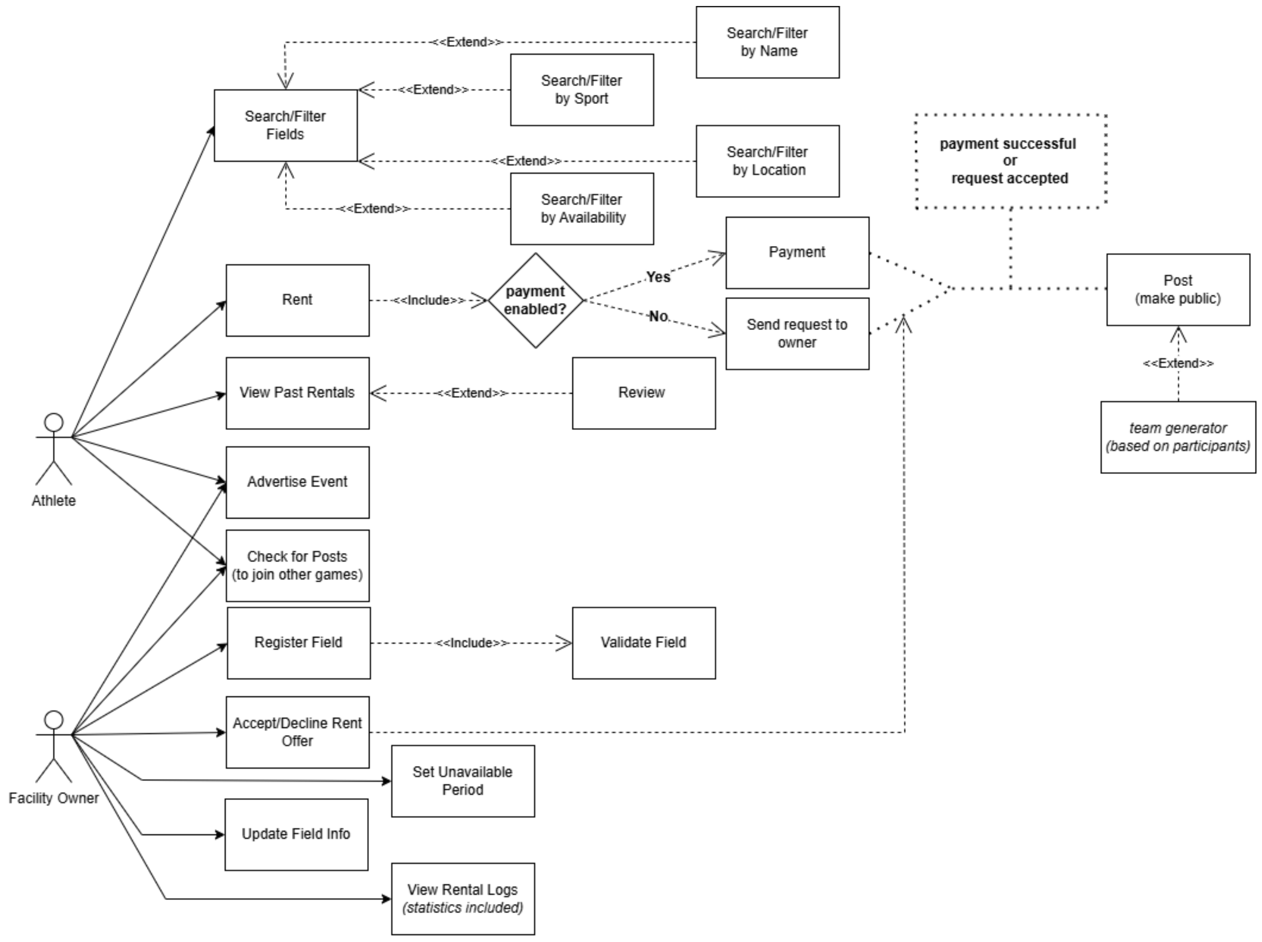


Diagram for service

Use-case diagram:



APIs:

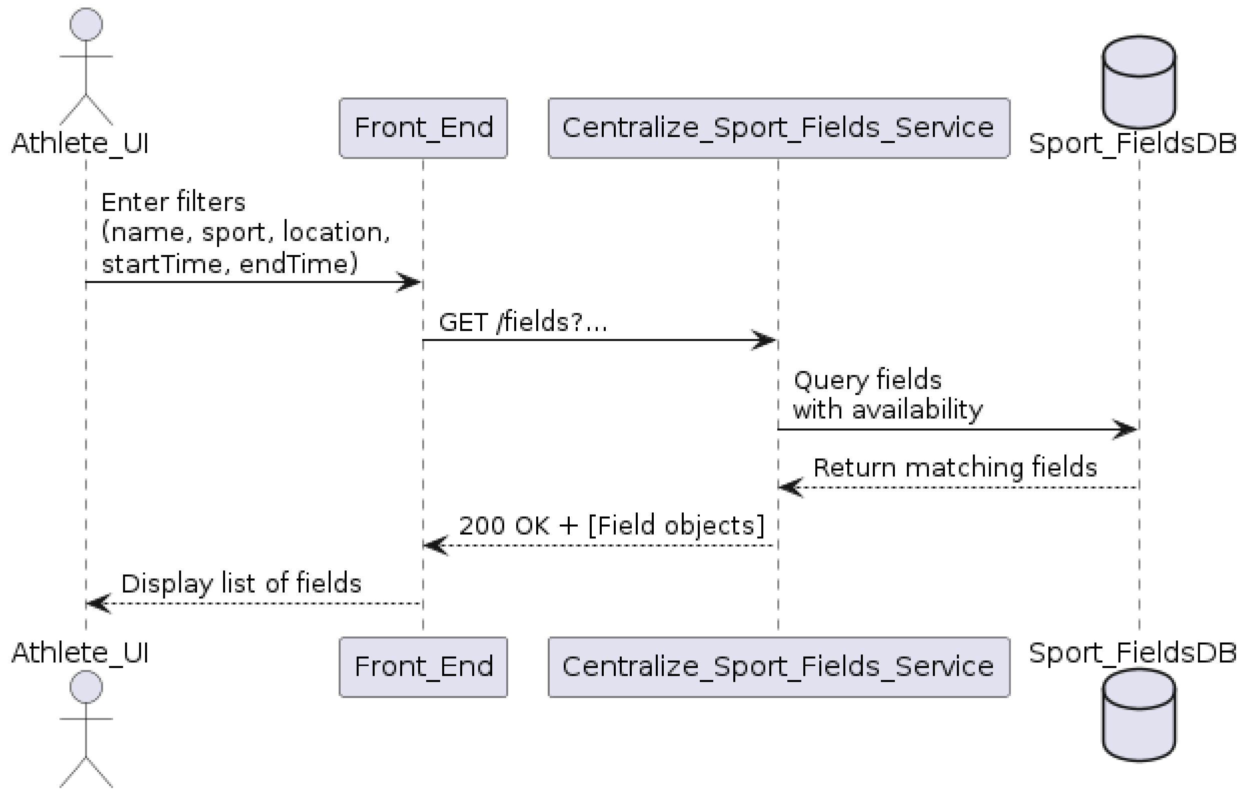
- https://app.swaggerhub.com/apis/ncnf/Homework5CC/1.0.0#/

Functionality flows documentation:

Search and Filter Fields

1. Athlete accesses the search page.
2. Athlete enters search criteria: name, sport, location, desired time window (`startTime`, `endTime`).

3. Front-end calls:
   GET
   /api/v1/fields?name={name}&sport={sport}&location={location}&startTime={ISO}&endTime={ISO}
4. Centralize Sport Fields Service:
   - Parses filters and queries database for matching fields with availability in the time window.
   - Returns list of Field objects.
5. Athlete views available fields and selects one.


Athlete_UI
Front_End
Centralize_Sport_Fields_Service
Sport_FieldsDB
Enter filters (name, sport, location, startTime, endTime)
GET /fields?...
Query fields with availability
Return matching fields
200 OK + [Field objects]
Display list of fields
Athlete_UI
Front_End
Centralize_Sport_Fields_Service
Sport_FieldsDB

Make a Reservation with Online Payment

1. Athlete selects a Field and time slot.
2. Athlete chooses reservation options (public or private) and payment method (`onsite` or `online`).
3. Front-end calls:
   POST /api/v1/fields/{fieldId}/reservations

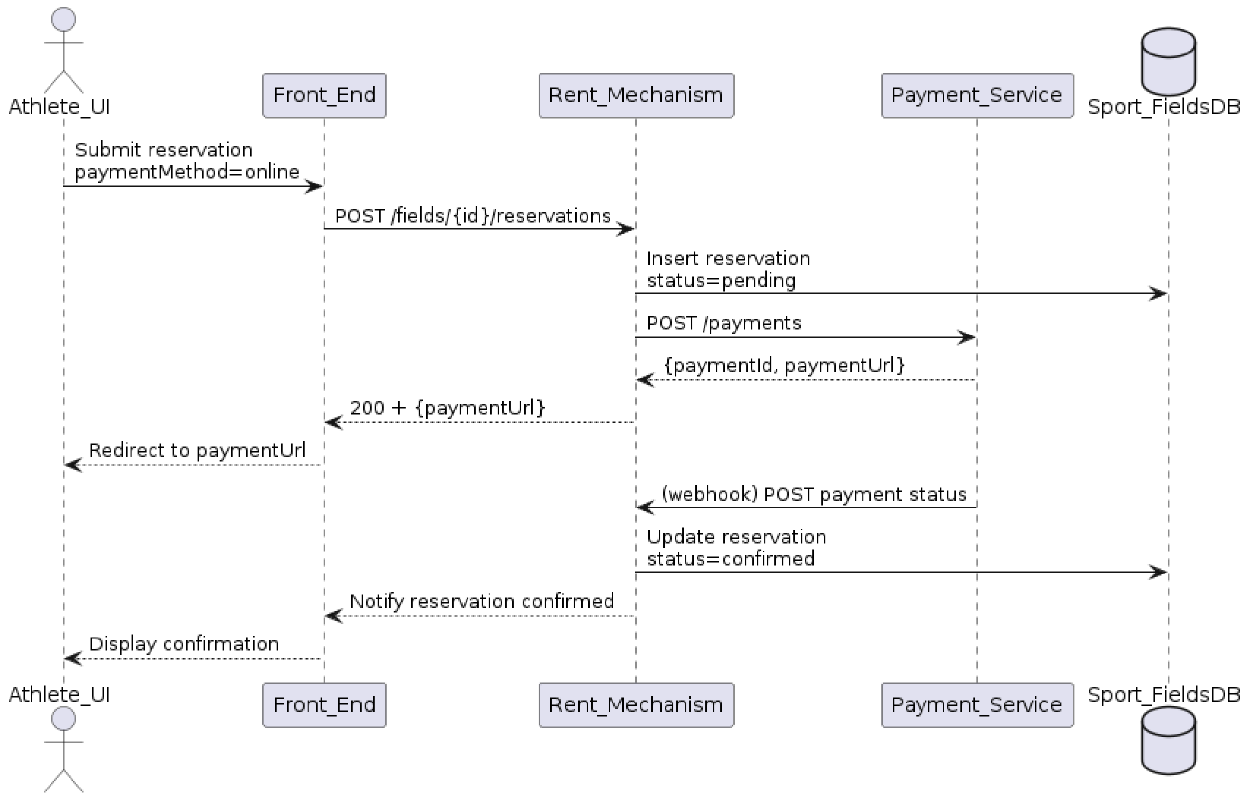
   { startTime, endTime, isPublic, paymentMethod, paymentMethodId }

4. Rent Mechanism:
    - Creates reservation with status `pending`.

If `paymentMethod=online`:
POST /api/v1/payments

- { reservationId, amount, currency, paymentMethod, paymentDetails }
  ->> returns `paymentId`, `paymentUrl`

5. Front-end redirects user to `paymentUrl` for authentication.
6. Payment Platform sends webhook or user polls:
   GET /api/v1/payments/{paymentId}
   ->> status `succeeded` or `failed`.
7. Upon success, Rent Mechanism updates reservation status to `confirmed`.
8. Athlete receives confirmation notification.



Join a Public Reservation

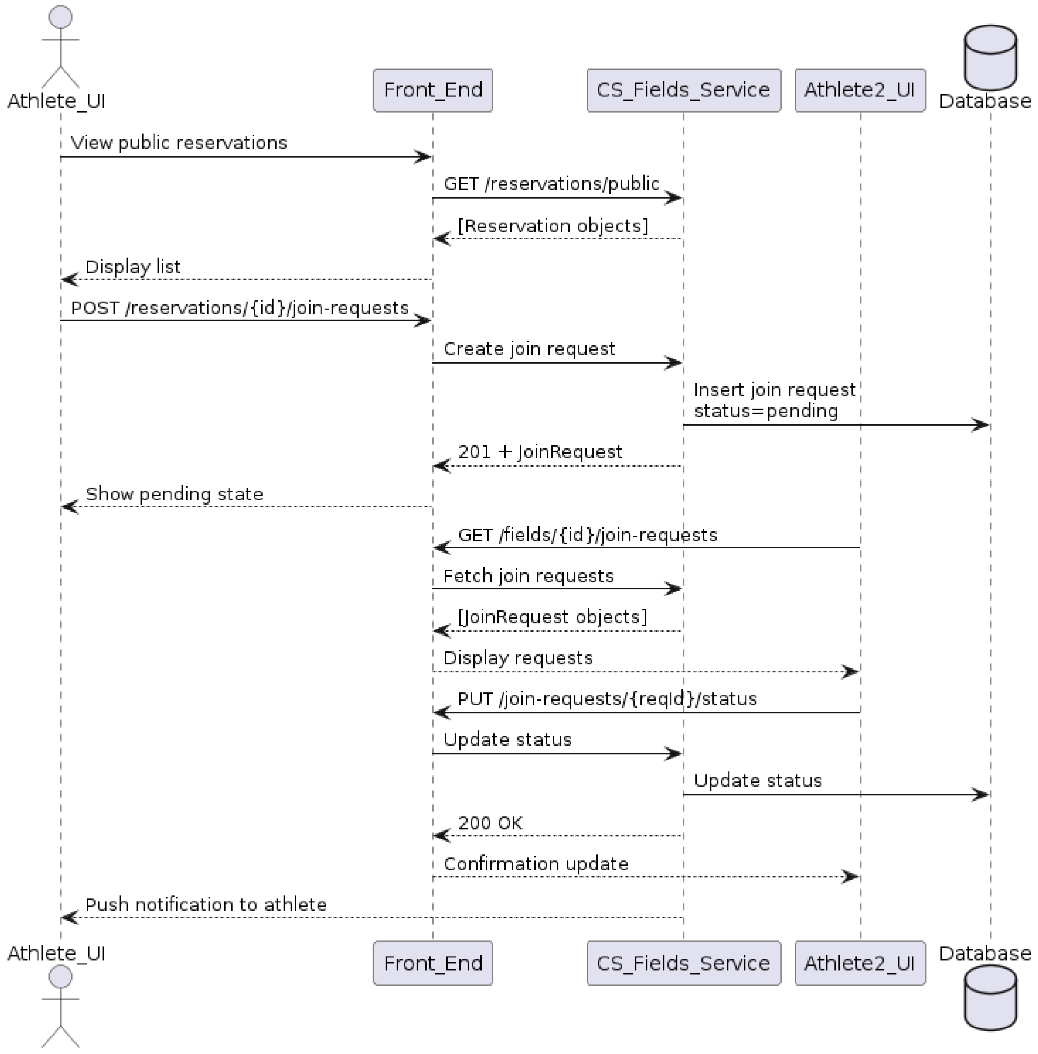
1. Athlete fetches public reservations:
   GET /api/v1/reservations/public

2. Athlete selects a reservation and requests to join:
   POST /api/v1/reservations/{reservationId}/join-requests
3. Centralize Sport Fields Service:
   - Creates `JoinRequest` with status `pending`.
4. Reservation owner reviews join requests:
   GET /api/v1/fields/{fieldId}/join-requests
5. Owner accepts or declines:

   PUT /api/v1/join-requests/{joinRequestId}/status

   { status: accepted | declined }

6. Athlete notified of outcome.

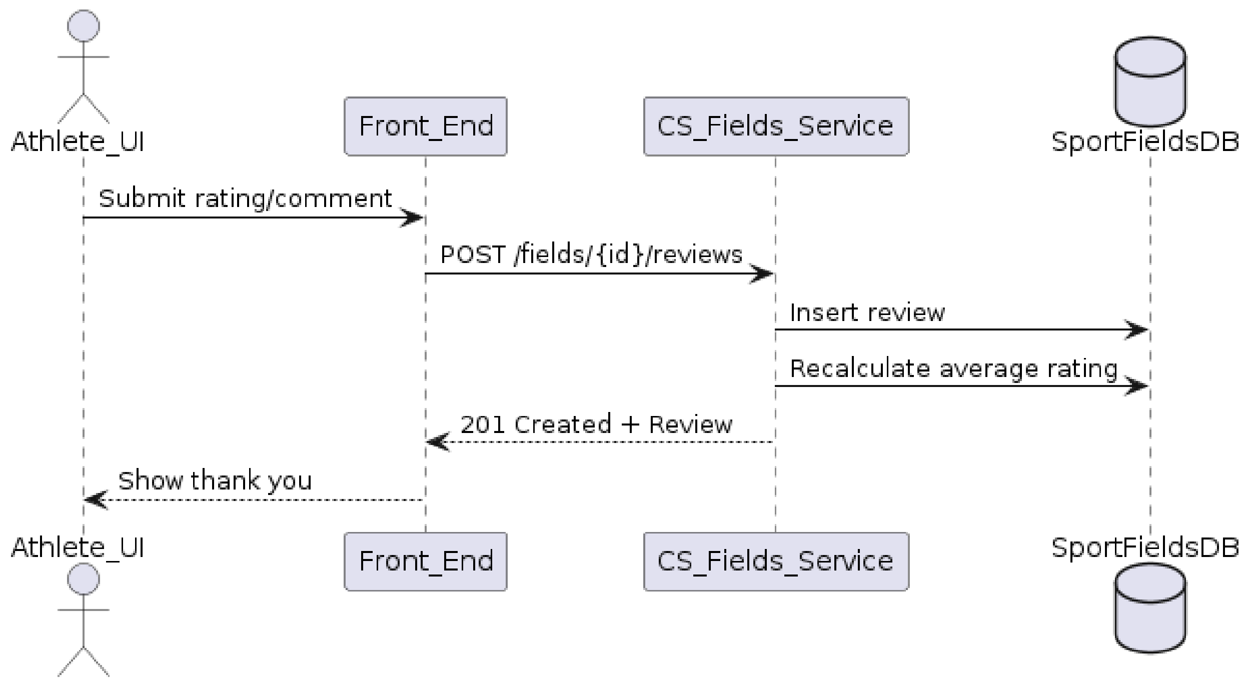


Leave a Review

1. After completed reservation, athlete navigates to field page.
2. Athlete submits review:

   POST /api/v1/fields/{fieldId}/reviews
   { rating, comment }

3. Centralize Sport Fields Service stores review and updates average rating.
4. Other users fetch reviews:



Register a New Field (Owner Flow)

1. Owner logs in and navigates to "Add Field".
2. Owner fills out field details (name, sport, location, pricePerHour, contactInfo).
3. Front-end calls:

   POST /api/v1/fields
   Content-Type: application/json
   { name, sport, location, pricePerHour, contactInfo }

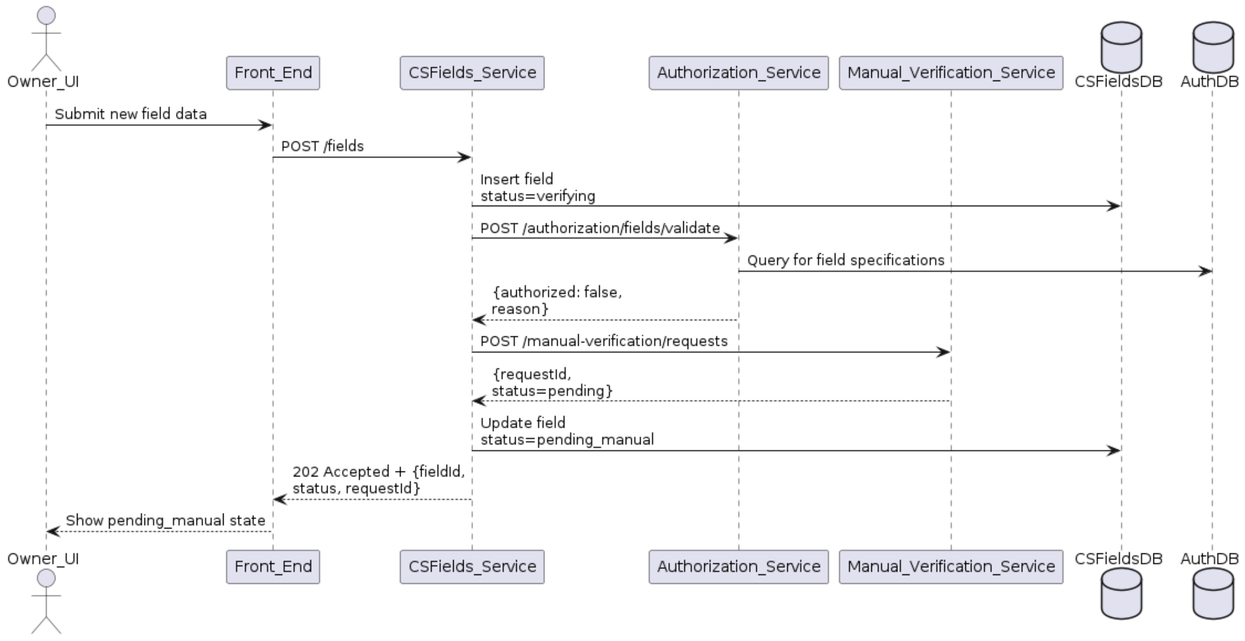
4. Centralize Sport Fields Service:
   - Inserts record with status `verifying`.

Calls Authorization Service:
POST /api/v1/authorization/fields/validate
{ <field payload> }

5. If authorized:
    - Updates Field status to `active`.
    - Responds `201 Created` with Field object.
6. If not authorized:

   Creates Manual Verification request:
   POST /api/v1/manual-verification/requests
   { fieldId, ownerId, payload, reason }

    - Sets Field status to `pending_manual`.
    - Responds `202 Accepted` with fieldId, status, requestId.
7. Owner sees field in pending/active state accordingly.


Owner_UI
Front_End
CSFields_Service
Authorization_Service
Manual_Verification_Service
CSFieldsDB
AuthDB
Submit new field data
POST /fields
Insert field status=verifying
POST /authorization/fields/validate
Query for field specifications
{authorized: false, reason}
POST /manual-verification/requests
{requestId, status=pending}
Update field status=pending_manual
202 Accepted + {fieldId, status, requestId}
Show pending_manual state

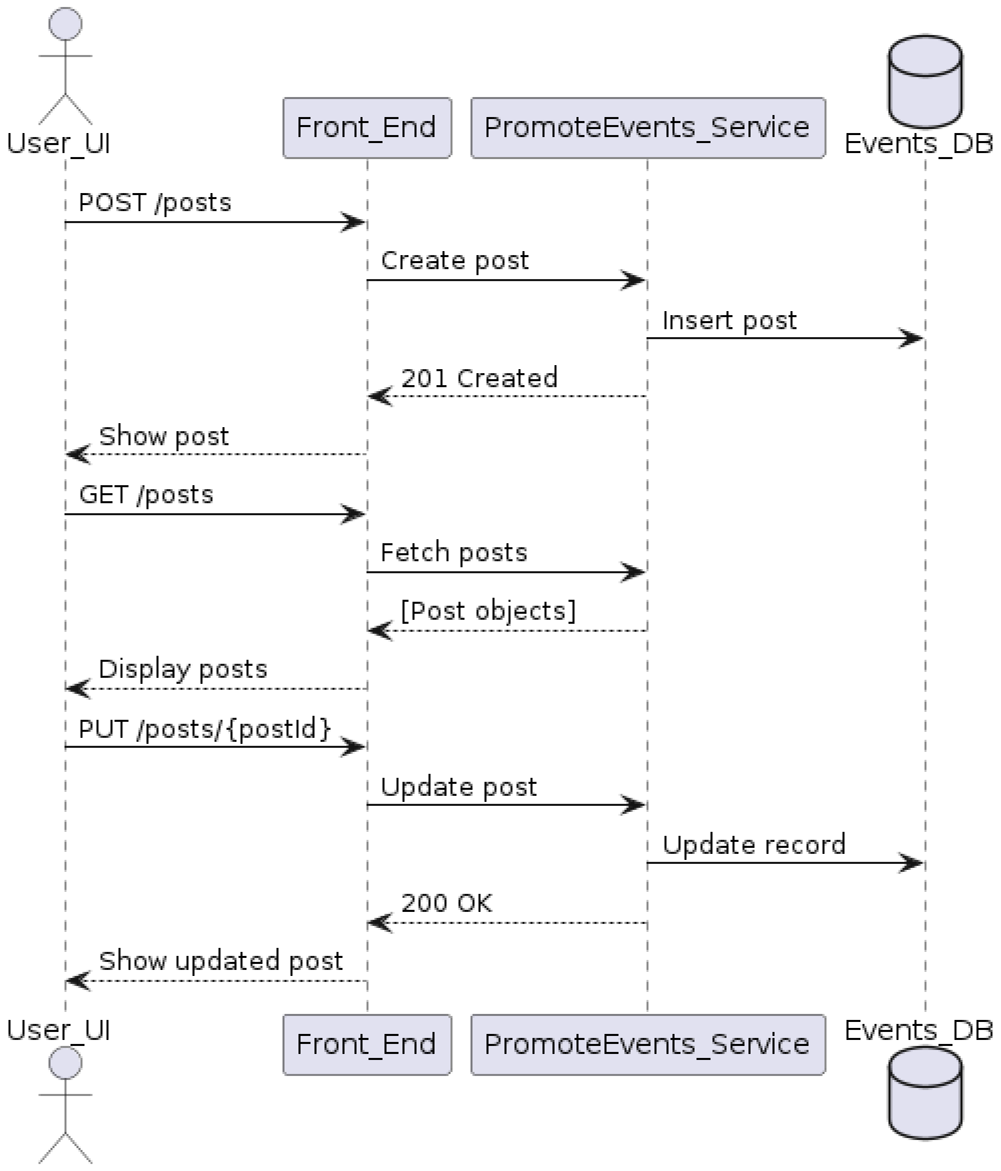
Create and Manage Event Posts

1. Any user navigates to "Create Event".
2. User fills out post details (title, description, eventTime, location).
3. Front-end calls:
   POST /api/v1/posts { title, description, eventTime, location }
4. Promote Events Service stores post.

5. Users view posts:
   GET /api/v1/posts
6. Post author edits or deletes via:

   PUT /api/v1/posts/{postId}

   DELETE /api/v1/posts/{postId}


User_UI
Front_End
PromoteEvents_Service
Events_DB
POST /posts
Create post
Insert post
201 Created
Show post
GET /posts
Fetch posts
[Post objects]
Display posts
PUT /posts/{postId}
Update post
Update record
200 OK
Show updated post