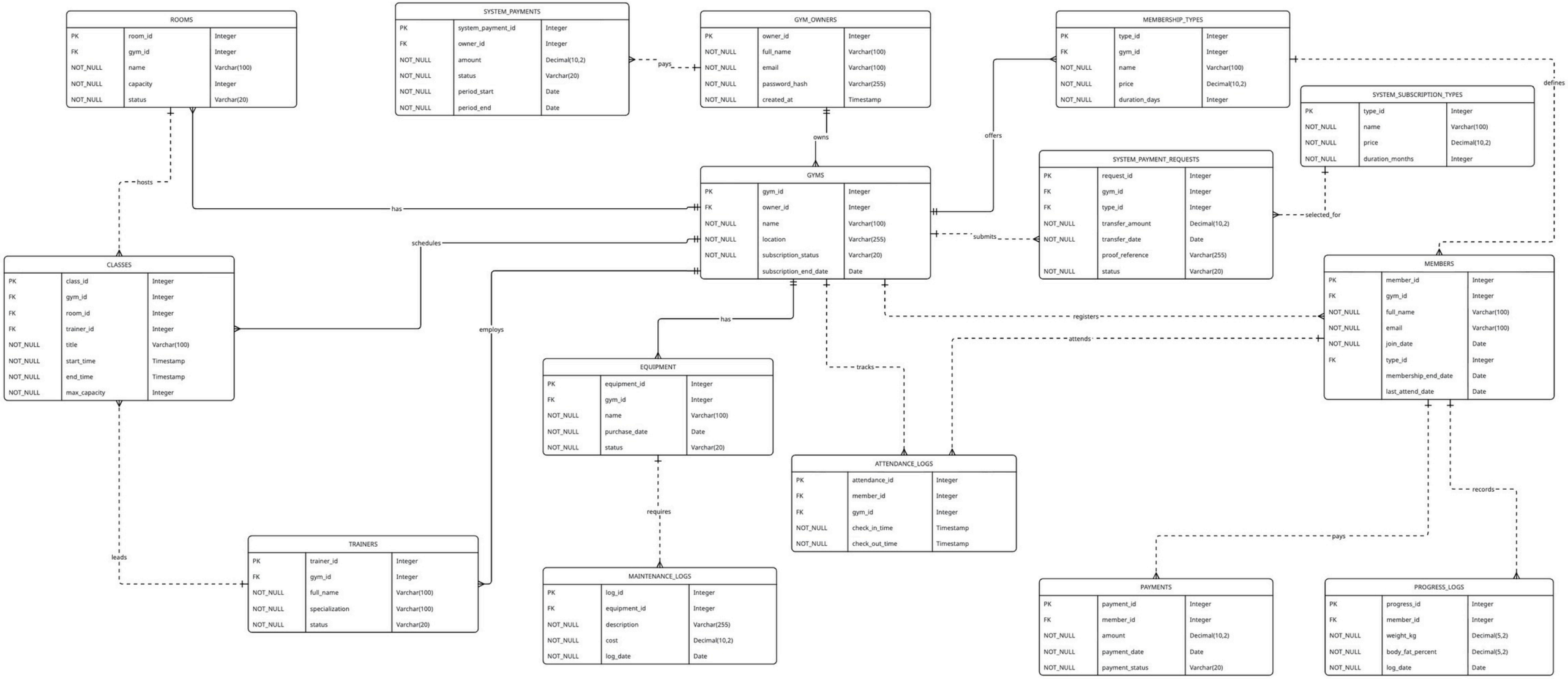
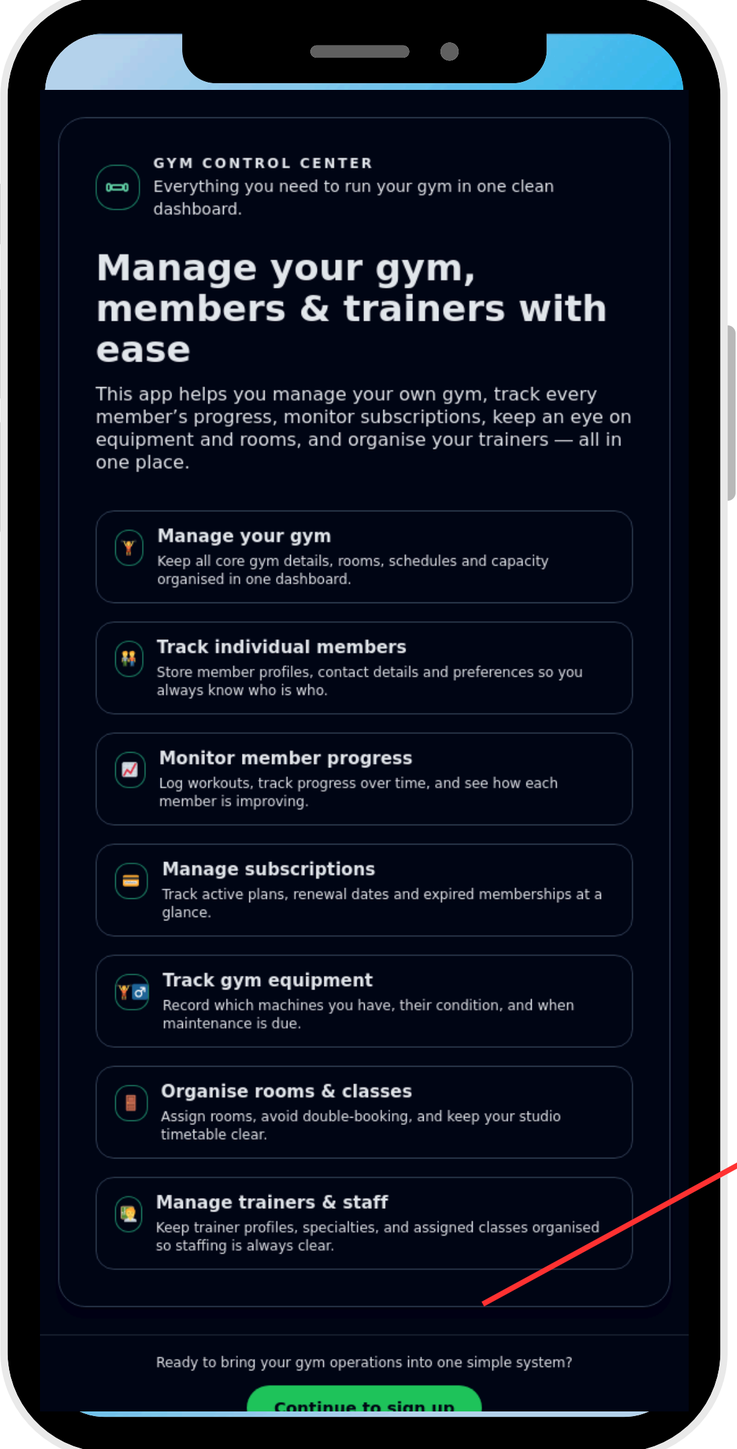

ROOMS
PK room_id Integer
FK gym_id Integer
NOT_NULL name Varchar(100)
NOT_NULL capacity Integer
NOT_NULL status Varchar(20)
SYSTEM_PAYMENTS
PK system_payment_id Integer
FK owner_id Integer
NOT_NULL amount Decimal(10,2)
NOT_NULL status Varchar(20)
NOT_NULL period_start Date
NOT_NULL period_end Date
pays
GYM_OWNERS
PK owner_id Integer
NOT_NULL full_name Varchar(100)
NOT_NULL email Varchar(100)
NOT_NULL password_hash Varchar(255)
NOT_NULL created_at Timestamp
owns
MEMBERSHIP_TYPES
PK type_id Integer
FK gym_id Integer
NOT_NULL name Varchar(100)
NOT_NULL price Decimal(10,2)
NOT_NULL duration_days Integer
offers
defines
SYSTEM_SUBSCRIPTION_TYPES
PK type_id Integer
NOT_NULL name Varchar(100)
NOT_NULL price Decimal(10,2)
NOT_NULL duration_months Integer
hosts
has
GYMS
PK gym_id Integer
FK owner_id Integer
NOT_NULL name Varchar(100)
NOT_NULL location Varchar(255)
NOT_NULL subscription_status Varchar(20)
subscription_end_date Date
submits
SYSTEM_PAYMENT_REQUESTS
PK request_id Integer
FK gym_id Integer
FK type_id Integer
NOT_NULL transfer_amount Decimal(10,2)
NOT_NULL transfer_date Date
proof_reference Varchar(255)
NOT_NULL status Varchar(20)
selected_for
schedules
CLASSES
PK class_id Integer
FK gym_id Integer
FK room_id Integer
FK trainer_id Integer
NOT_NULL title Varchar(100)
NOT_NULL start_time Timestamp
NOT_NULL end_time Timestamp
NOT_NULL max_capacity Integer
employs
has
registers
attends
MEMBERS
PK member_id Integer
FK gym_id Integer
NOT_NULL full_name Varchar(100)
NOT_NULL email Varchar(100)
NOT_NULL join_date Date
FK type_id Integer
membership_end_date Date
last_attend_date Date
EQUIPMENT
PK equipment_id Integer
FK gym_id Integer
NOT_NULL name Varchar(100)
NOT_NULL purchase_date Date
NOT_NULL status Varchar(20)
tracks
ATTENDANCE_LOGS
PK attendance_id Integer
FK member_id Integer
FK gym_id Integer
NOT_NULL check_in_time Timestamp
NOT_NULL check_out_time Timestamp
requires
records
leads
TRAINERS
PK trainer_id Integer
FK gym_id Integer
NOT_NULL full_name Varchar(100)
NOT_NULL specialization Varchar(100)
NOT_NULL status Varchar(20)
MAINTENANCE_LOGS
PK log_id Integer
FK equipment_id Integer
NOT_NULL description Varchar(255)
NOT_NULL cost Decimal(10,2)
NOT_NULL log_date Date
pays
PAYMENTS
PK payment_id Integer
FK member_id Integer
NOT_NULL amount Decimal(10,2)
NOT_NULL payment_date Date
NOT_NULL payment_status Varchar(20)
PROGRESS_LOGS
PK progress_id Integer
FK member_id Integer
NOT_NULL weight_kg Decimal(5,2)
NOT_NULL body_fat_percent Decimal(5,2)
NOT_NULL log_date Date

Apps

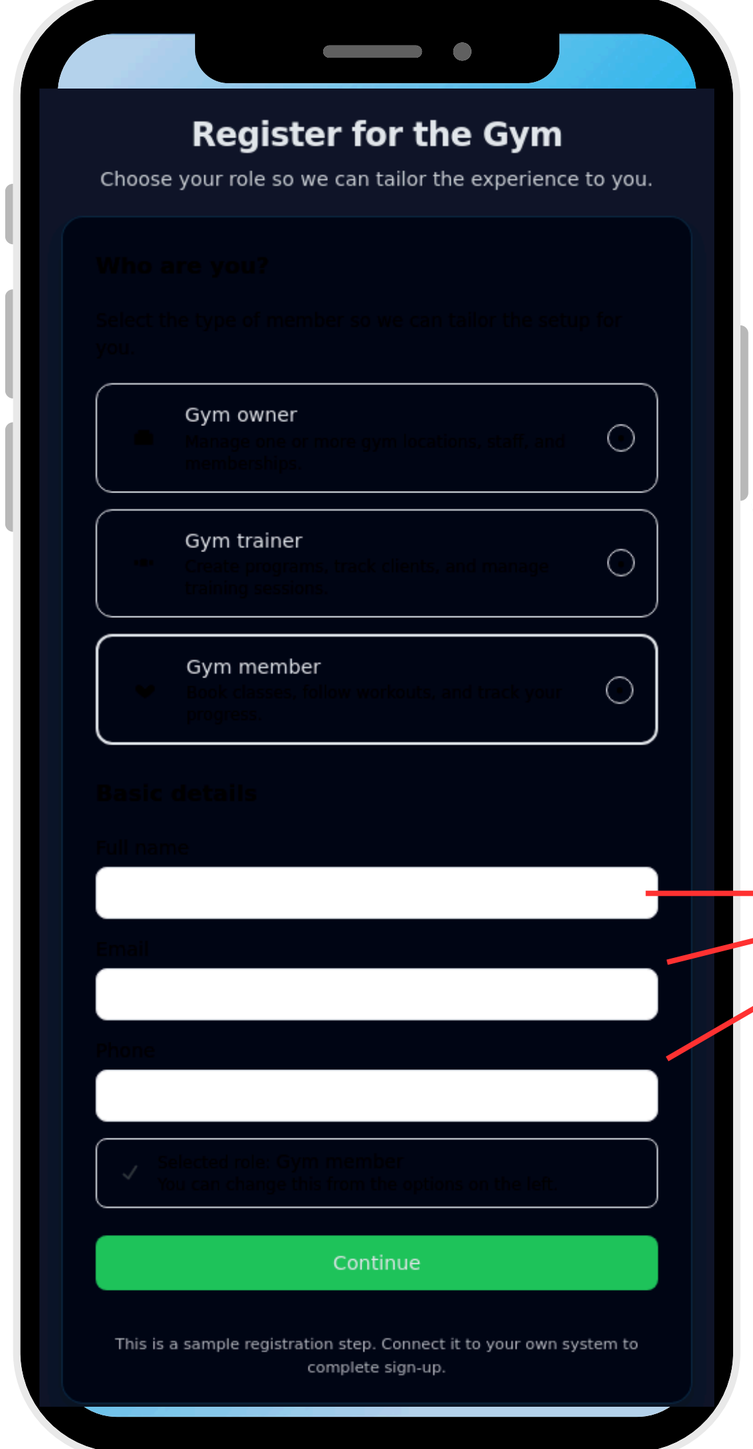


This is the first page when user uses the app for first time

It shows feature about this app.

Next time if they are registered, they will go directly to homepage

This redirects to sign up page



The continue button redirection will base on the type they select

They can select what Type of user

This could have more box for inputs
(Too long for the slide)

```
create function register_gym_owner(_full_name character varying, _email character varying, _password_hash character varying, _phone character varying) returns text
    language plpgsql
```

after run:

```
Success: Owner registered with ID: 2
```

2	Johnny	Johnny@test.com	hashed	123456789	2025-11-26 03:36:28.672342

```
create function register_trainer(_gym_id integer, _full_name character varying, _email character varying, _password_hash character varying, _specialization character varying, _phone character varying) returns text
    language plpgsql
```

after run:

```
Success: Application sent to Muscle Factory BKK. Please wait for Owner approval.
```

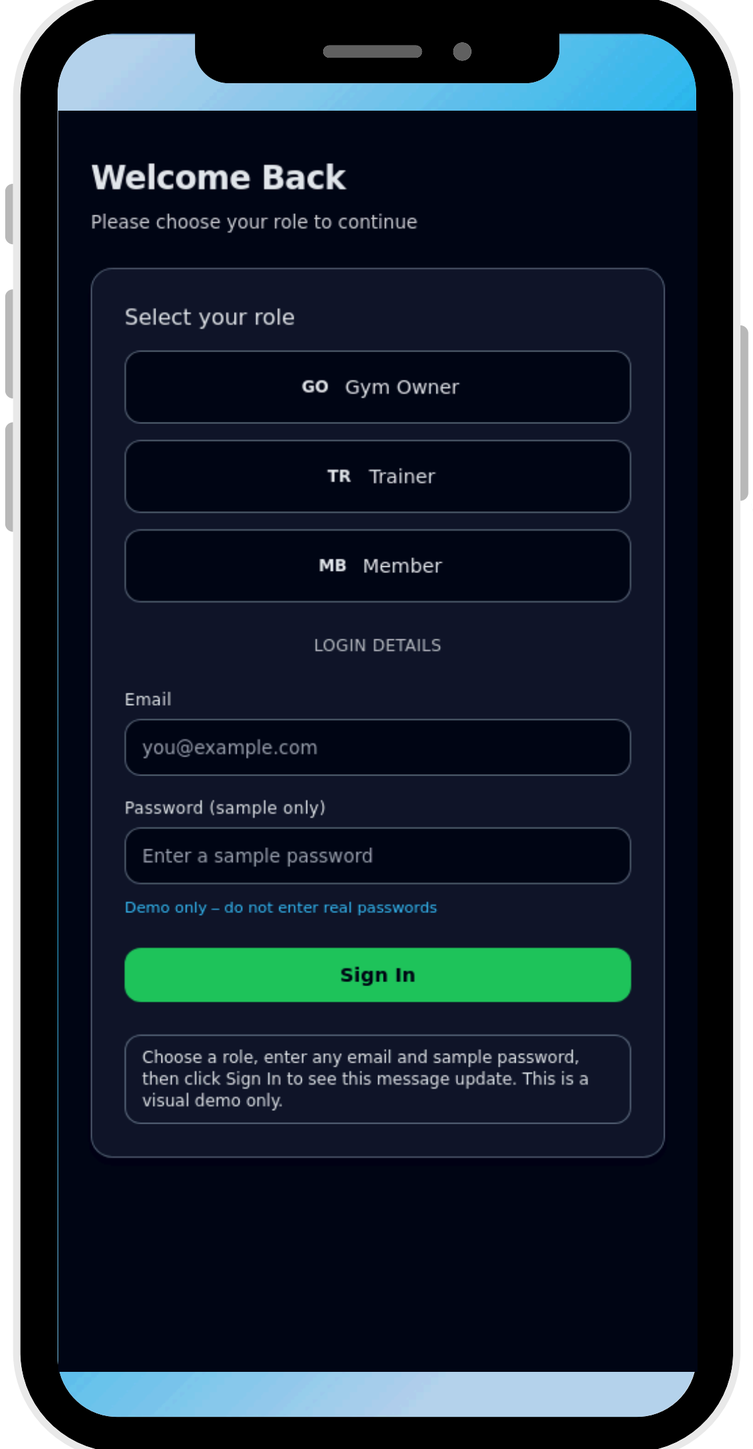
3	2	Tong	Tong@test.com	dfghj	Running	6237412374	Pending

```
create function register_member(_gym_id integer, _name character varying, _email character varying, _password_hash character varying, _phone character varying, _type_id integer) returns text
    language plpgsql
```

after run:

```
Success: Account created for Wea2. Please make a payment to activate membership.
```

4		wes2@test.com	123456789	2025-11-26	1	<null>	<null>	cscas2222

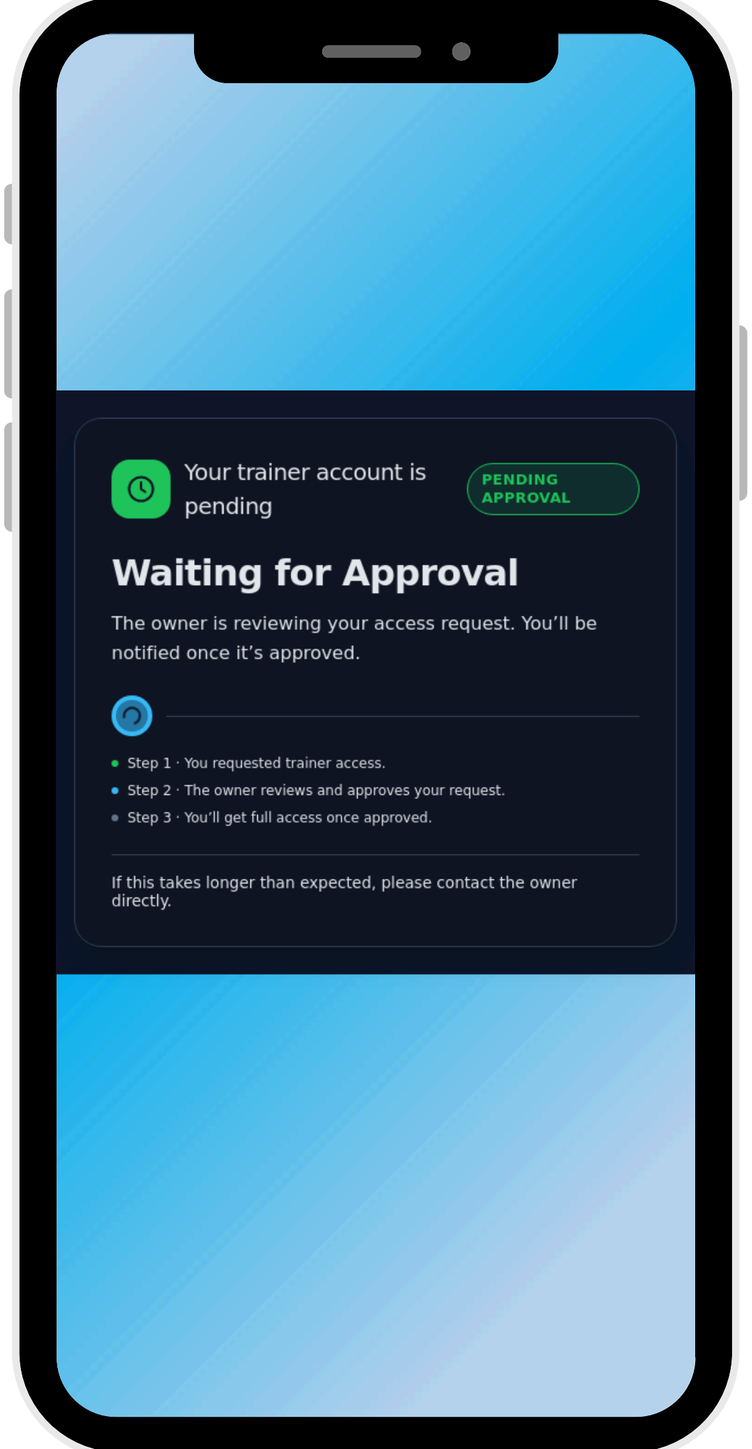


The sign in button redirection will base on the type they select

They can select what Type of user

The step of payment and approve will occur for trainer and member if they are here for first time

Approvement page for Trainer

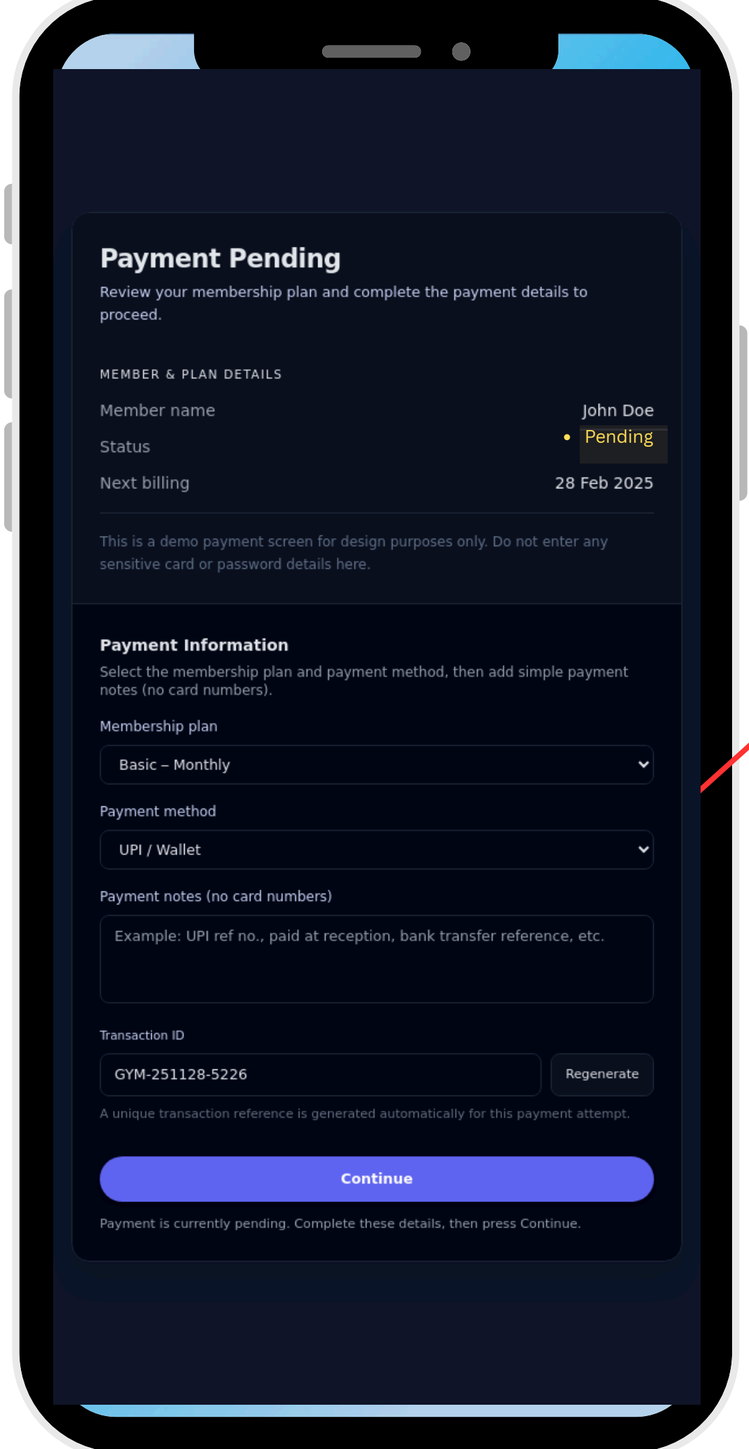



```
create function login_trainer(_gym_id integer, _email character varying, _password_hash character varying)
    returns TABLE(r_trainer_id integer, r_full_name character varying, r_status character varying,
r_message text)
    language plpgsql
```

after run and not approve:

3	Tong	Pending	Login Failed: Account is pending approval

Payment page for member

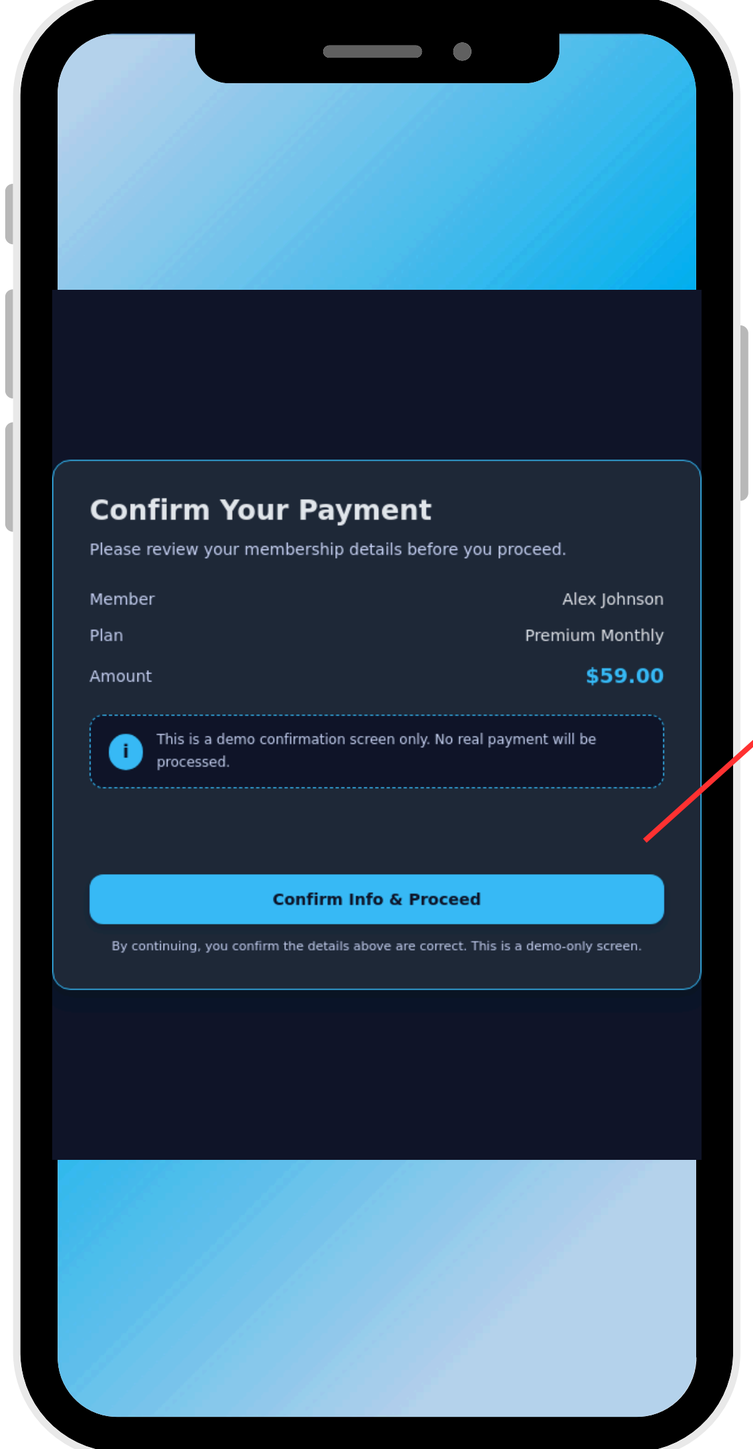


This will do

```
create function create_member_payment(_gym_id integer, _member_id integer, _plan_type_id integer,
_payment_method character varying, _transaction_id character varying) returns text
    language plpgsql
```

```
Success: Payment #2 created for Bronze ($1099.00). Status: Pending.
```

2	4	1099.00	2025-11-26 07:31:43.644454		567	Pending



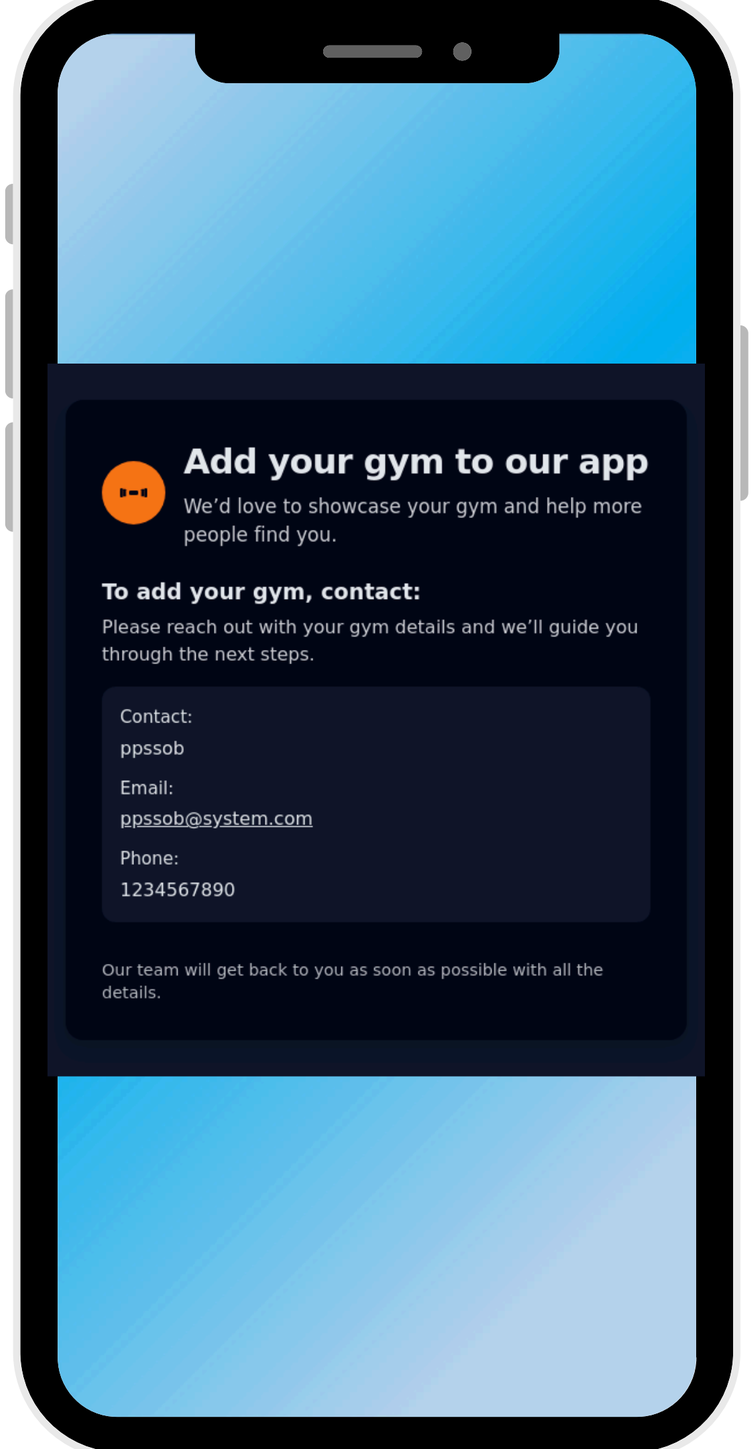
This will do

```
create function complete_payment_and_renew(_payment_id integer) returns text
    language plpgsql
```

```
Success: Payment verified ($1099.00) and membership extended.
```

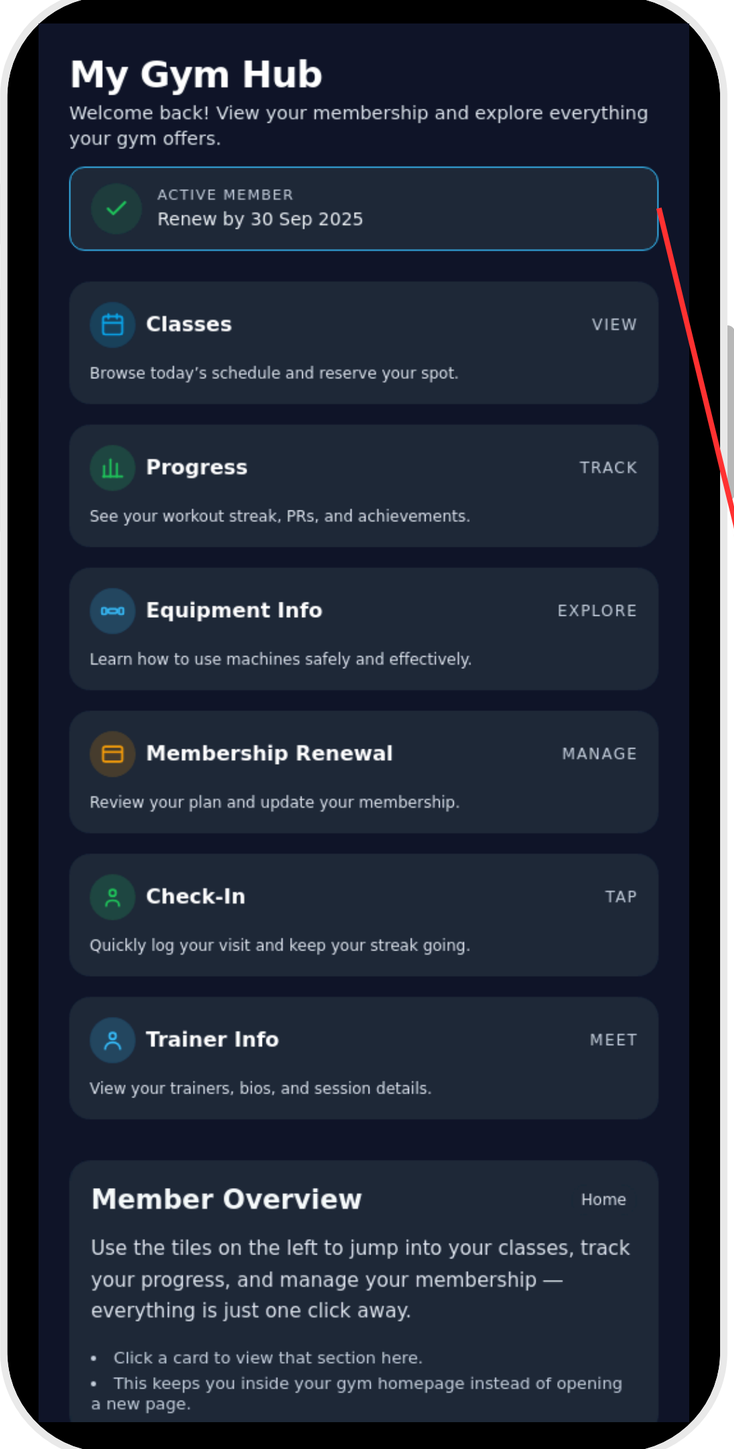
2	4	1099.00	2025-11-26 07:31:43.644454		567	Paid

Owner add gym contact page



This will appear when the owner first sign in or they dont have any own in the database

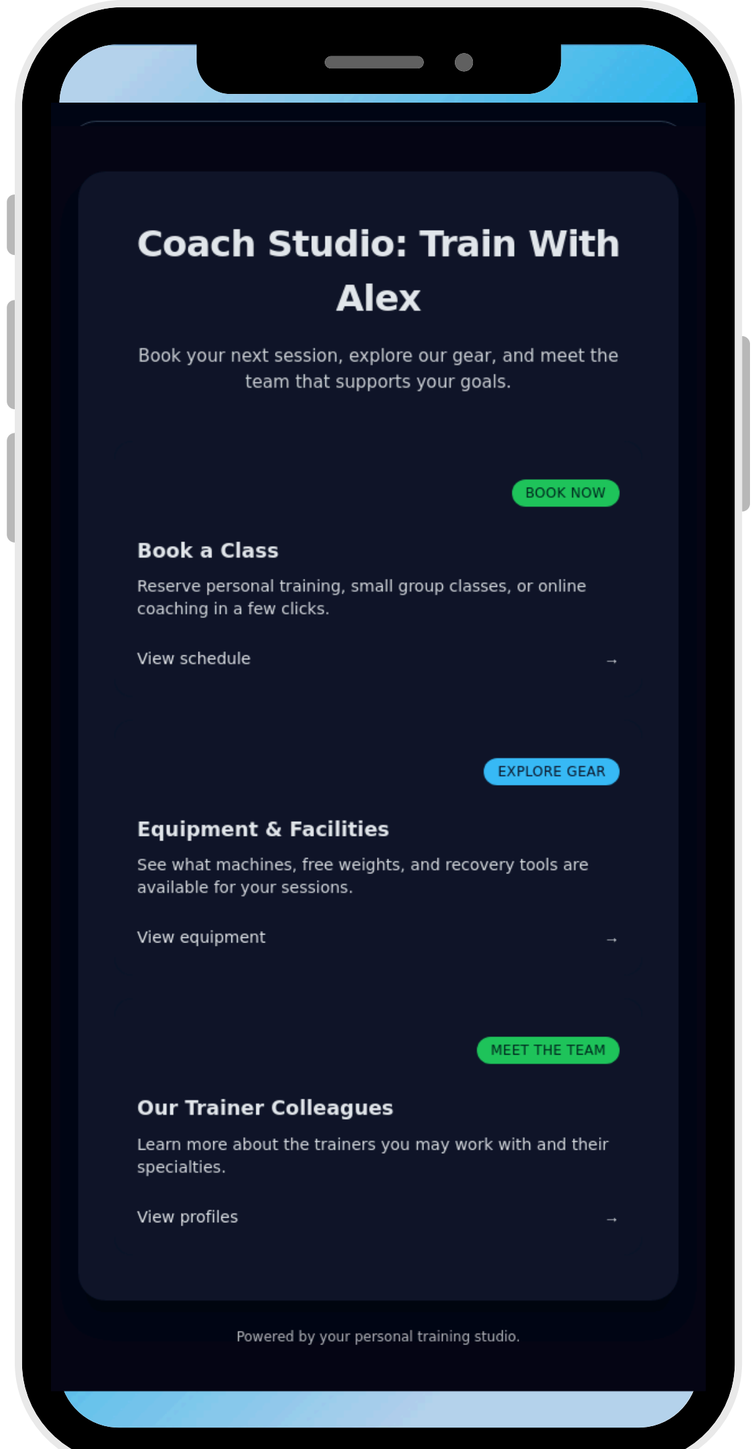
Member home page



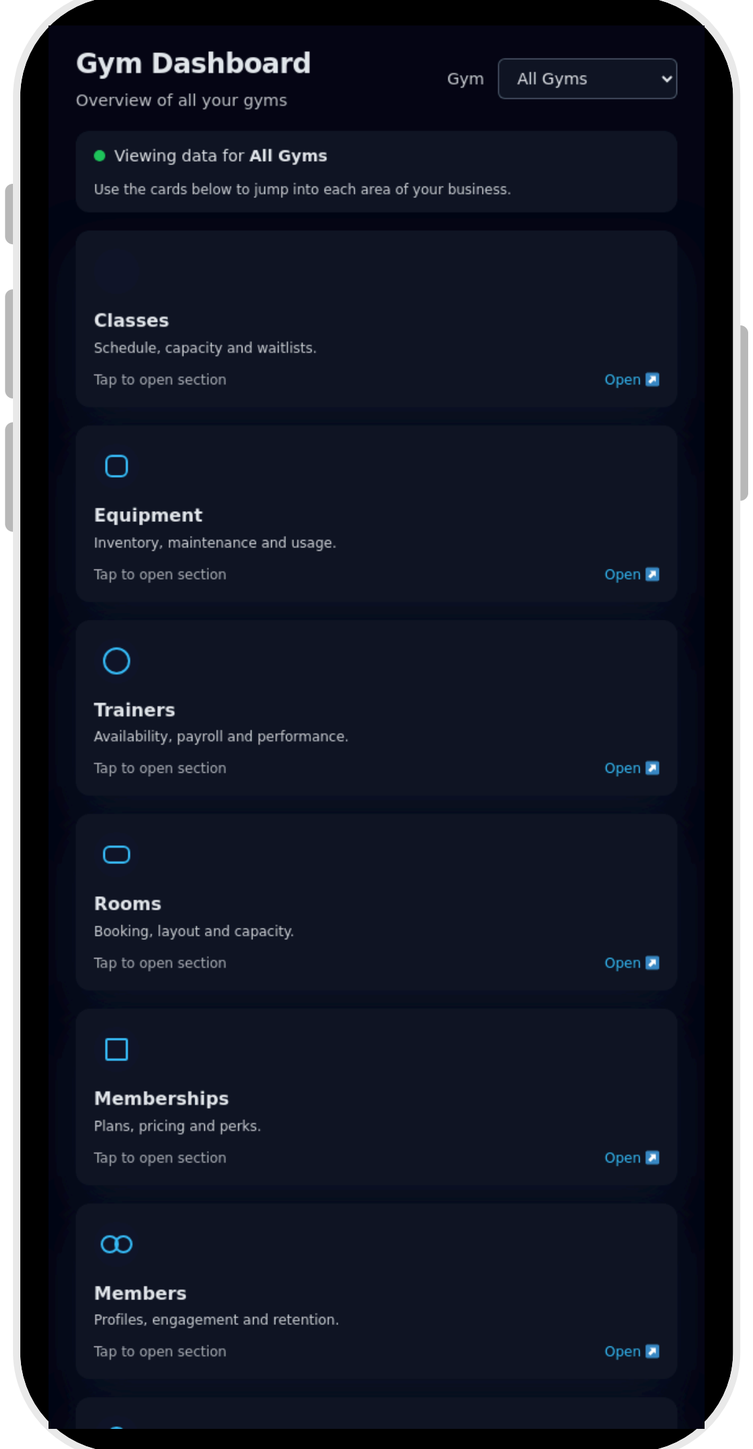

```
create function check_membership_status(_member_id integer) returns text
    language plpgsql
```

This should show something like this

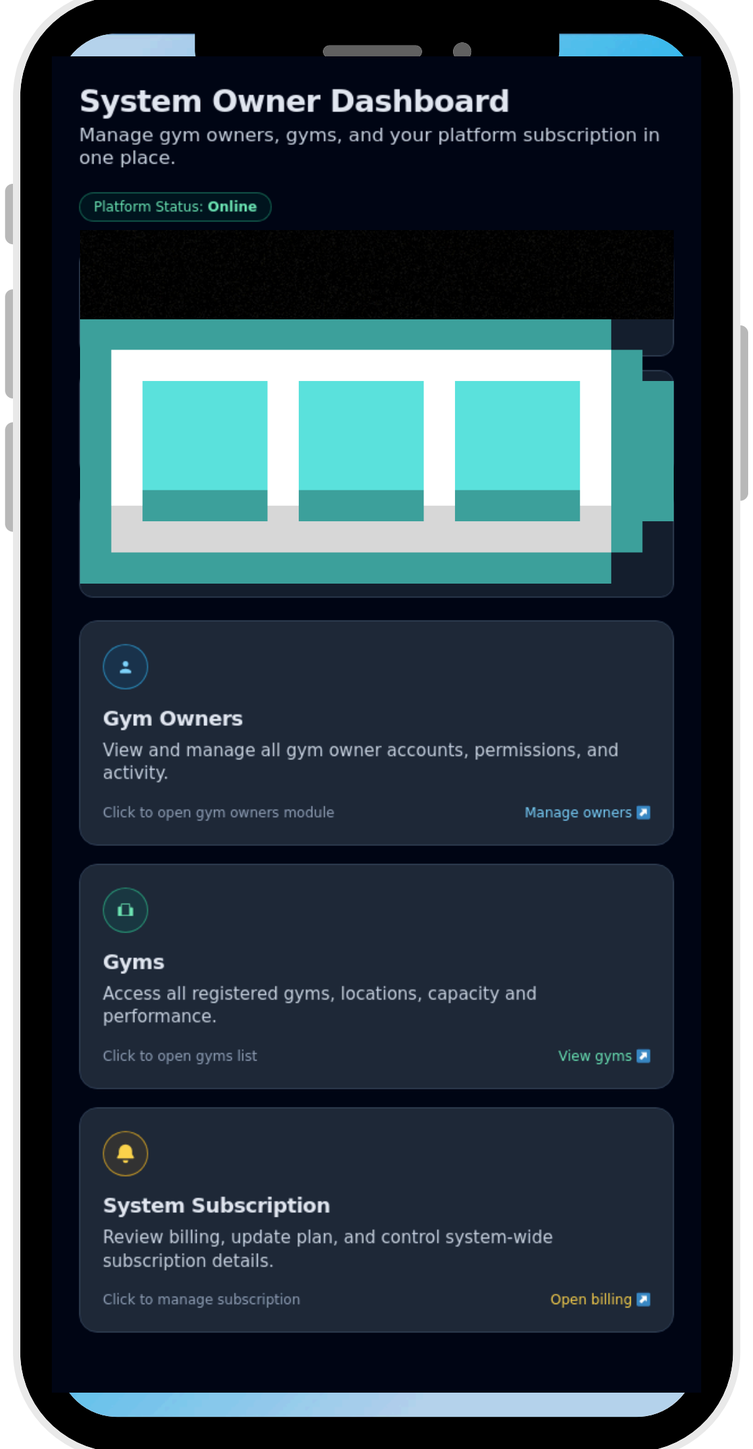
```
Member: Wea2 | Status: ACTIVE until 2025-12-26
```



Trainer home page

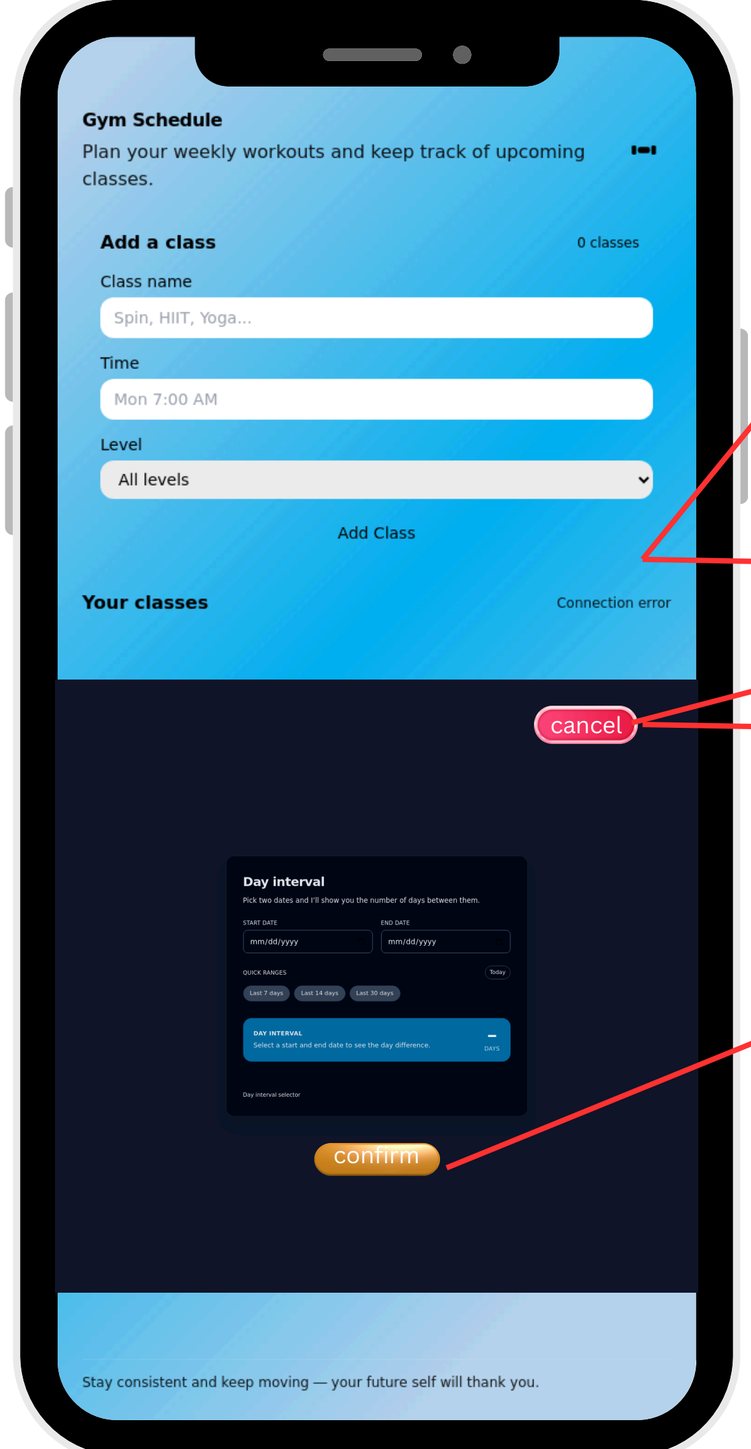


Owner home page



System Owner home page

Class page




```
create function schedule_class(_gym_id integer, _room_id integer, _trainer_id integer, _title character varying, _start_time timestamp without time zone, _duration_minutes integer) returns text
    language plpgsql
```

3	2	2	3	nothing	2015-12-12 14:30:00.000000	2015-12-12 15:10:00.000000	20

```
Success: "nothing" scheduled from 2015-12-12 14:30:00 to 2015-12-12 15:10:00
```

Visible for trainer and gym owner

```
create function cancel_class(_gym_id integer, _class_id integer) returns text
    language plpgsql
```

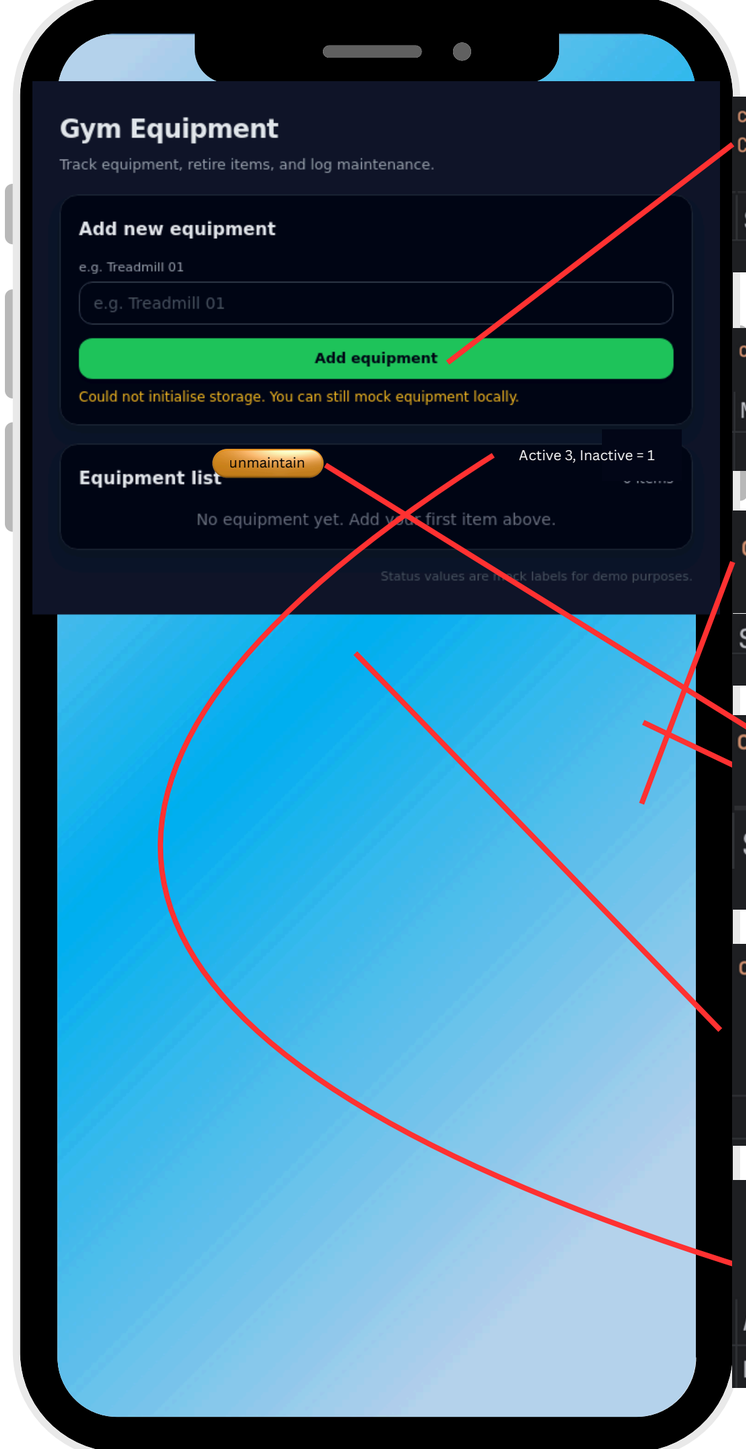
```
Success: Class "nothing" has been cancelled.
```

Visible for trainer and gym owner

```
create function get_gym_schedule(_gym_id integer, _start_date timestamp without time zone, _end_date timestamp without time zone)
    returns TABLE(r_class_id integer, r_title character varying, r_start_time timestamp without time zone, r_end_time timestamp without time zone, r_trainer_name character varying, r_room_name character varying, r_max_capacity integer)
    language plpgsql
```

5	nothing	2027-12-12 14:30:00.000000	2027-12-12 15:10:00.000000	Tong	Blank room

Equipment Page




```
create function add_equipment(_gym_id integer, _name character varying, _purchase_date date DEFAULT CURRENT_DATE) returns text
    language plpgsql
```

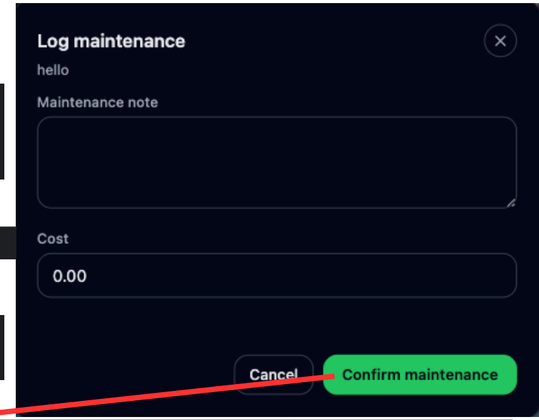
```
Success: "qsedr" added to inventory (ID: 2)
```

2	2	qsedr	2024-12-12	<null>	Active

```
create function perform_maintenance(_equipment_id integer, _notes text, _cost numeric) returns text
    language plpgsql
```

```
Maintenance recorded. qsedr is now marked as INACTIVE.
```

2	2	qsedr	2024-12-12	2025-11-26	Inactive



Only gym owner is able to edit all of this

```
create function finish_maintenance(_equipment_id integer) returns text
    language plpgsql
```

```
Success: qsedr is now ACTIVE and ready for use.
```

2	2	qsedr	2024-12-12	<null>	Active

```
create function retire_equipment(_gym_id integer, _equipment_id integer) returns text
    language plpgsql
```

```
Success: qsedr is now marked as Retired.
```

2	2	qsedr	2024-12-12	2025-11-26	Retired

```
create function get_equipment_status(_gym_id integer)
    returns TABLE(r_equipment_id integer, r_name text, r_last_maintenance date, r_status text)
    language plpgsql
```

1	Dumbbell 15kg	2025-11-25	Active
2	qsedr	2025-11-26	Retired

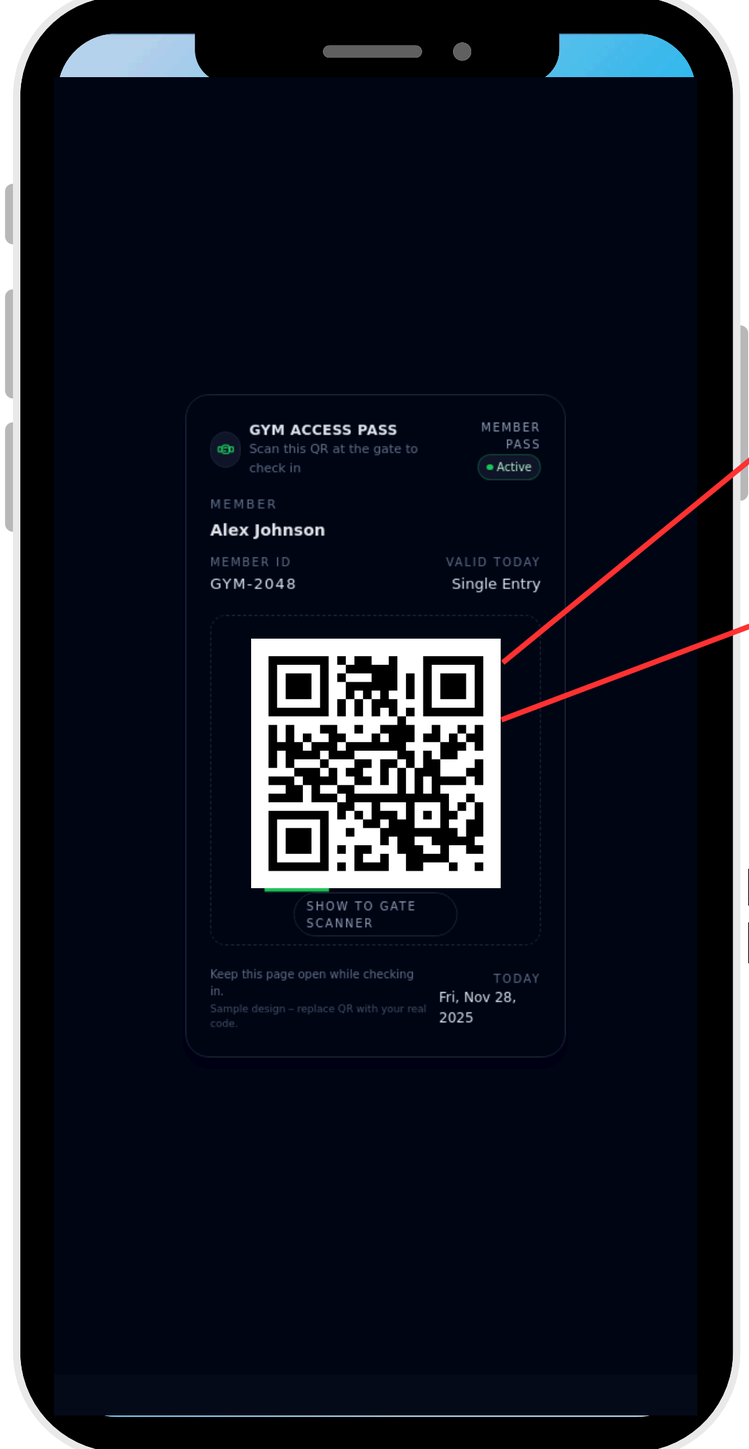
```
create function get_equipment_summary(_gym_id integer)
    returns TABLE(status_type text, count bigint)
    language plpgsql
```

Active	1
Retired	1

```
create function get_equipment_due_for_maintenance(_gym_id integer)
    returns TABLE(r_equipment_id integer, r_name character varying, r_last_maintenance date)
    language plpgsql
```

equipment that has not been checked for 30 days since last maintenance

Check in page



U can scan it but i don’t recommended

After tap at the gym gate or at front

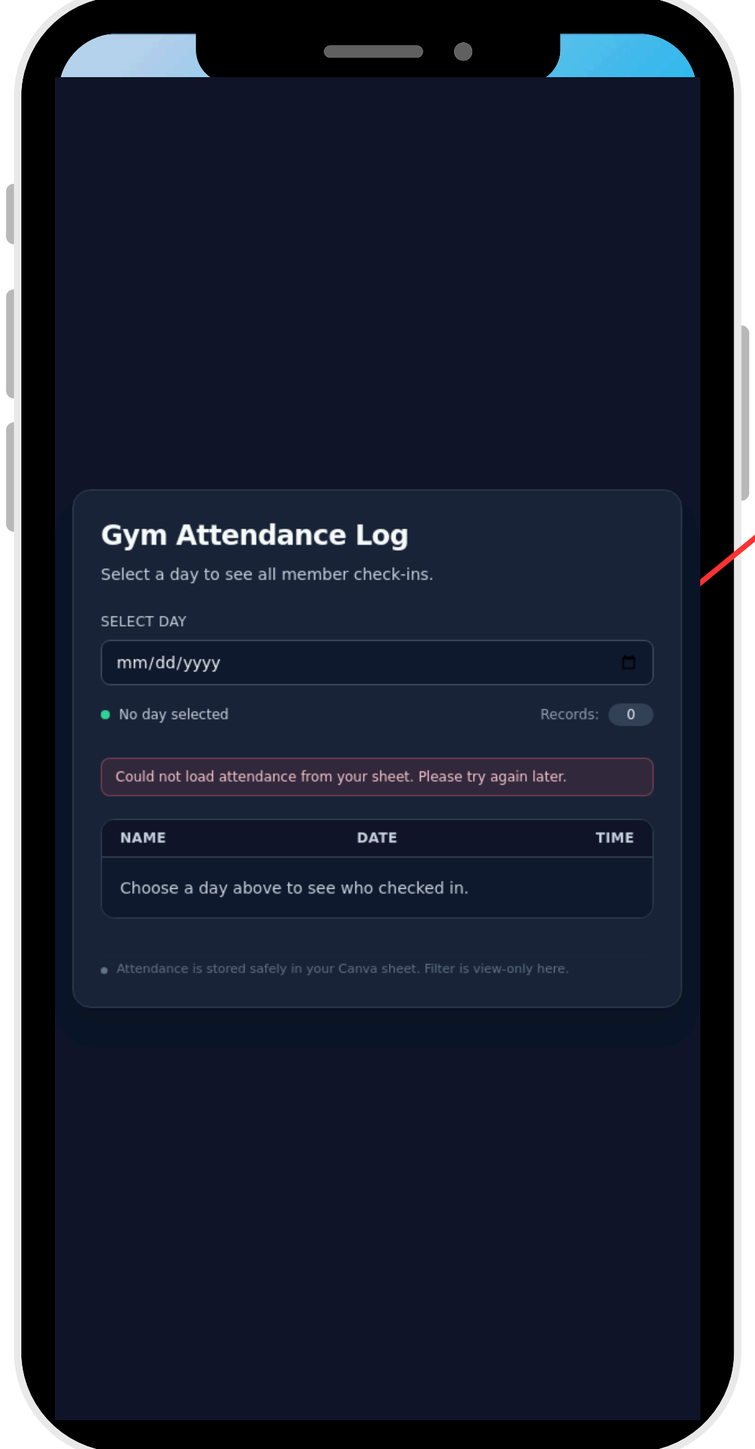
```
create function check_in_member(_member_id integer, _gym_id integer) returns text
    language plpgsql
```

```
Success: Welcome Wea2! Check-in recorded.
```

wes2@test.com	123456789	2025-11-26	1	2025-12-26	2025-11-26 14:03:04.359715	cscas2222

3	4	2	2025-11-26 14:03:04.359715	<null>

Check in page

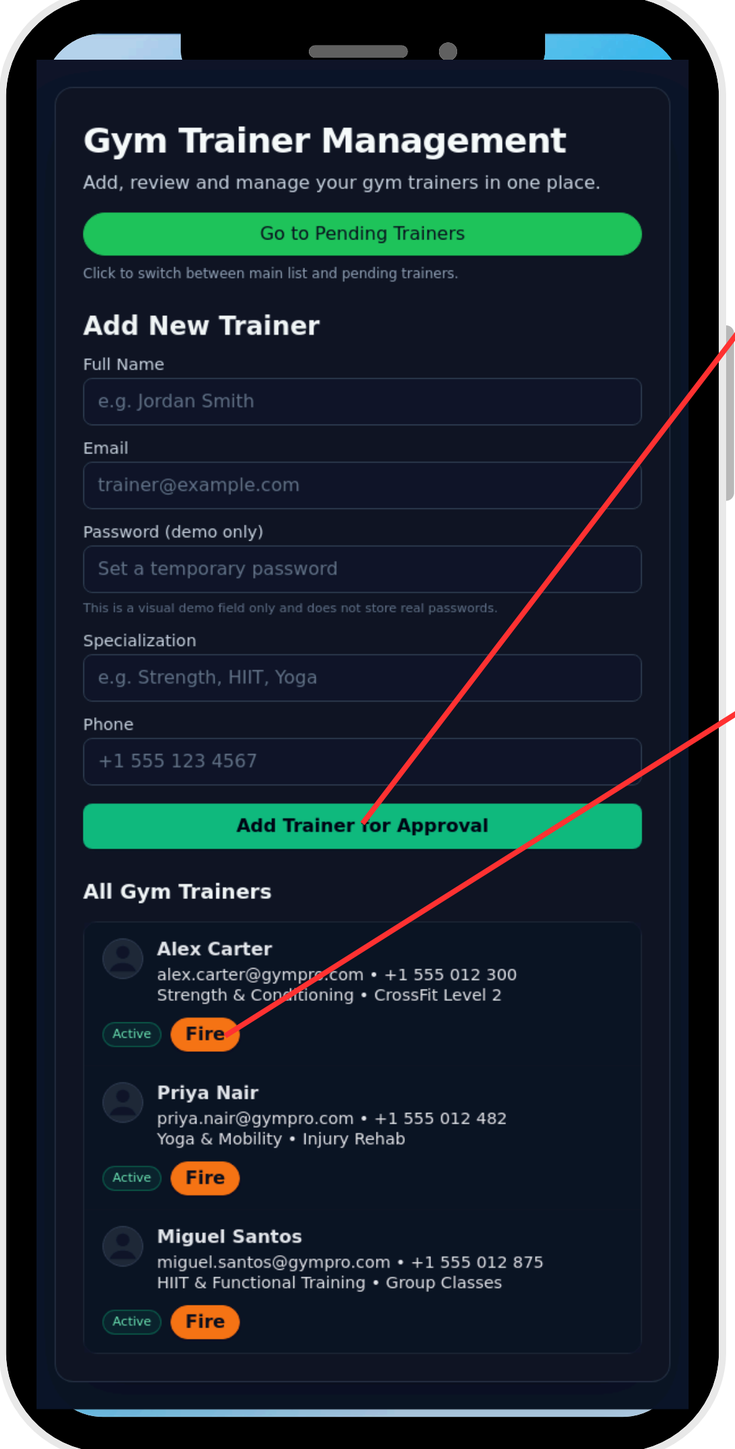


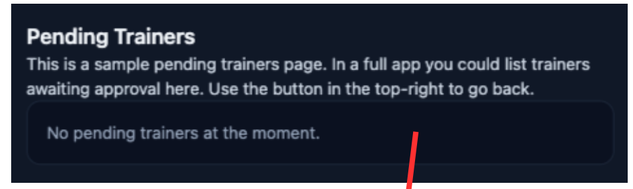

```
create function get_daily_attendance_log(_gym_id integer, _date date DEFAULT CURRENT_DATE)
    returns TABLE(r_member_name character varying, r_check_in_time timestamp without time zone, r_status
character varying)
    language plpgsql
```

Wea2	2025-11-26 14:03:04.359715	Active

Trainer page

Only gym owner can edit the trainer page





The gym owner can do add trainer without approve

```
create function add_trainer(_gym_id integer, _full_name character varying, _email character varying,
_password_hash character varying, _specialization character varying, _phone character varying) returns
text
    language plpgsql
```

```
create function terminate_trainer(_gym_id integer, _trainer_id integer) returns text
    language plpgsql
```

```
Success: Trainer Guga is now removed.
```

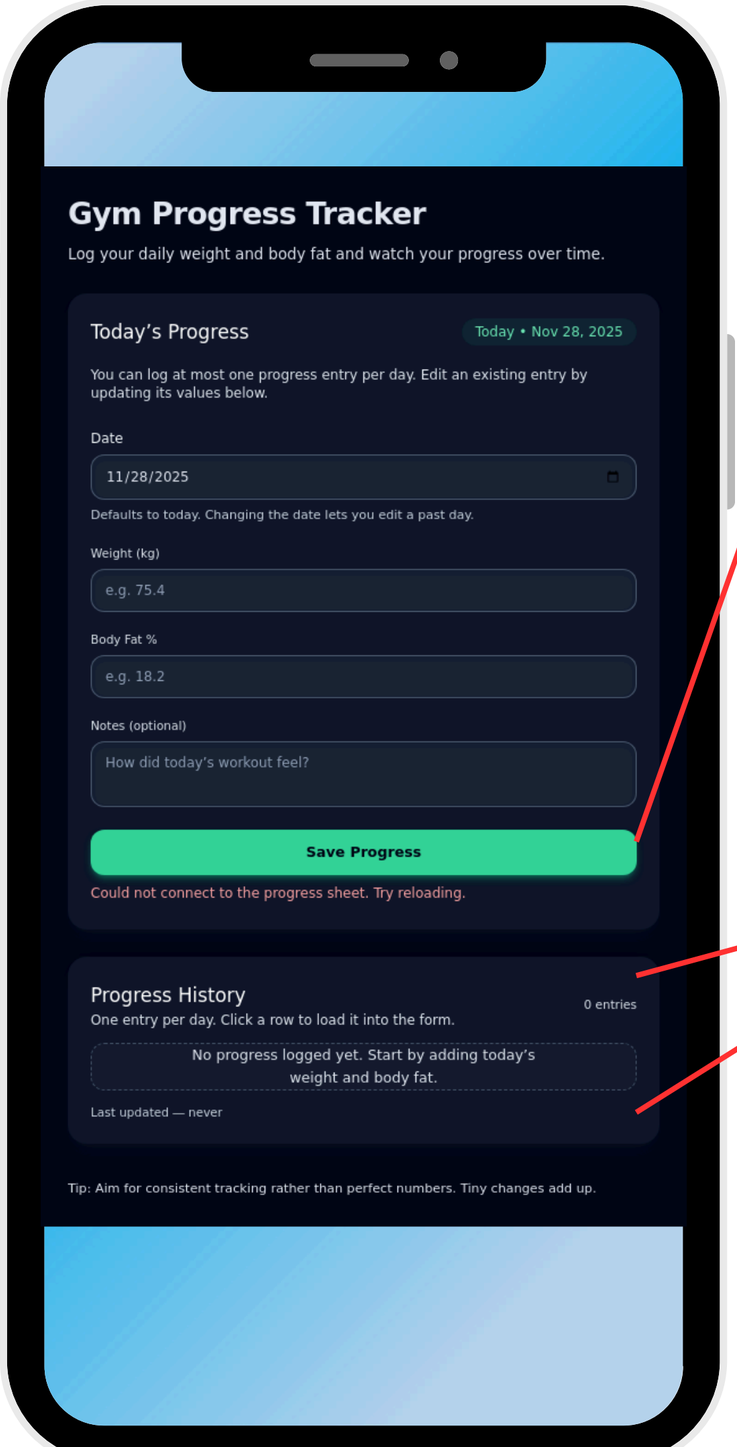
This give the list of registered trainer

```
create function get_pending_trainers(_owner_id integer)
    returns TABLE(gym_name character varying, trainer_id integer, trainer_name character varying,
specialization character varying, applied_date timestamp without time zone)
    language plpgsql
```

There will be approve button

```
create function approve_trainer(_owner_id integer, _trainer_id integer) returns text
    language plpgsql
```

Progress page




```
create function log_progress(_member_id integer, _weight numeric, _body_fat numeric) returns text
    language plpgsql
```

```
Success: Progress for Wea2 updated for today.
```

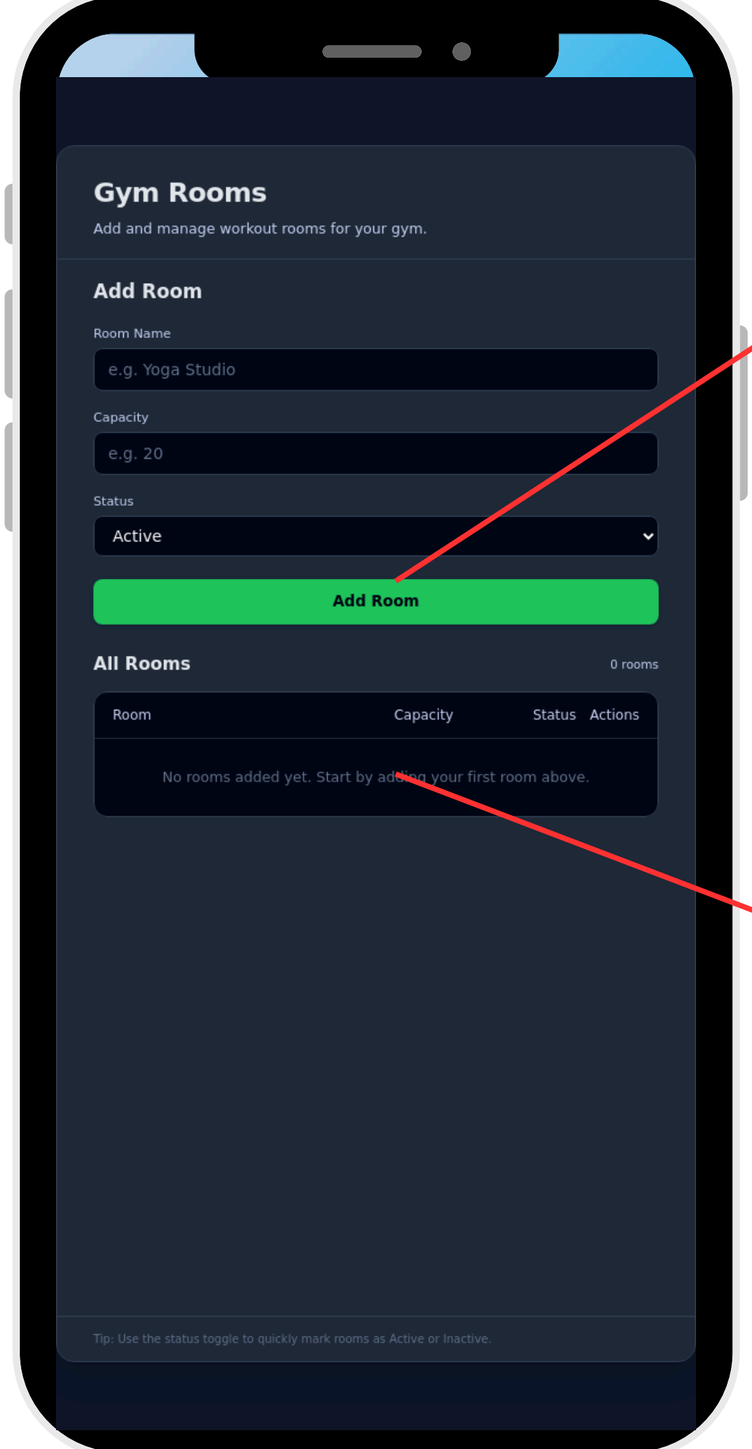
1	4	59.00	13.0	2025-11-26

```
create function get_member_progress(_gym_id integer, _member_id integer)
    returns TABLE(log_date date, weight_kg numeric, body_fat_percent numeric)
    language plpgsql
```

The progress cant be deleted and if want to update just summit another form

2025-11-26	59	13

Room Page (Owner)




```
create function add_room(_gym_id integer, _name character varying, _capacity integer) returns text
    language plpgsql
```

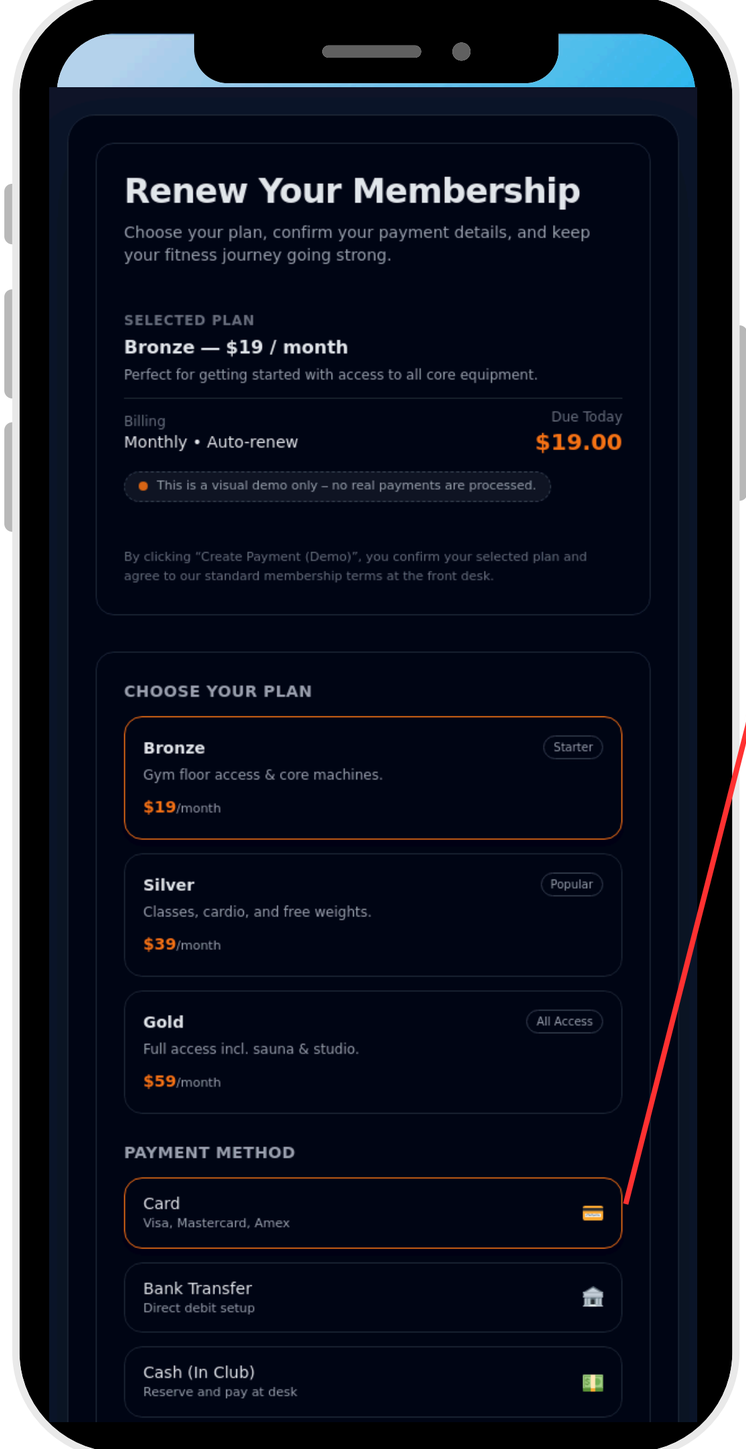
No delete here as we only set to inactive

```
create function set_room_status(_room_id integer, _new_status character varying) returns text
    language plpgsql
```

```
Success: Room "Yoga" is now set to Inactive
```

room_id	gym_id	name	capacity	status
2	2	Blank room	40	Available
1	2	Yoga	20	Inactive

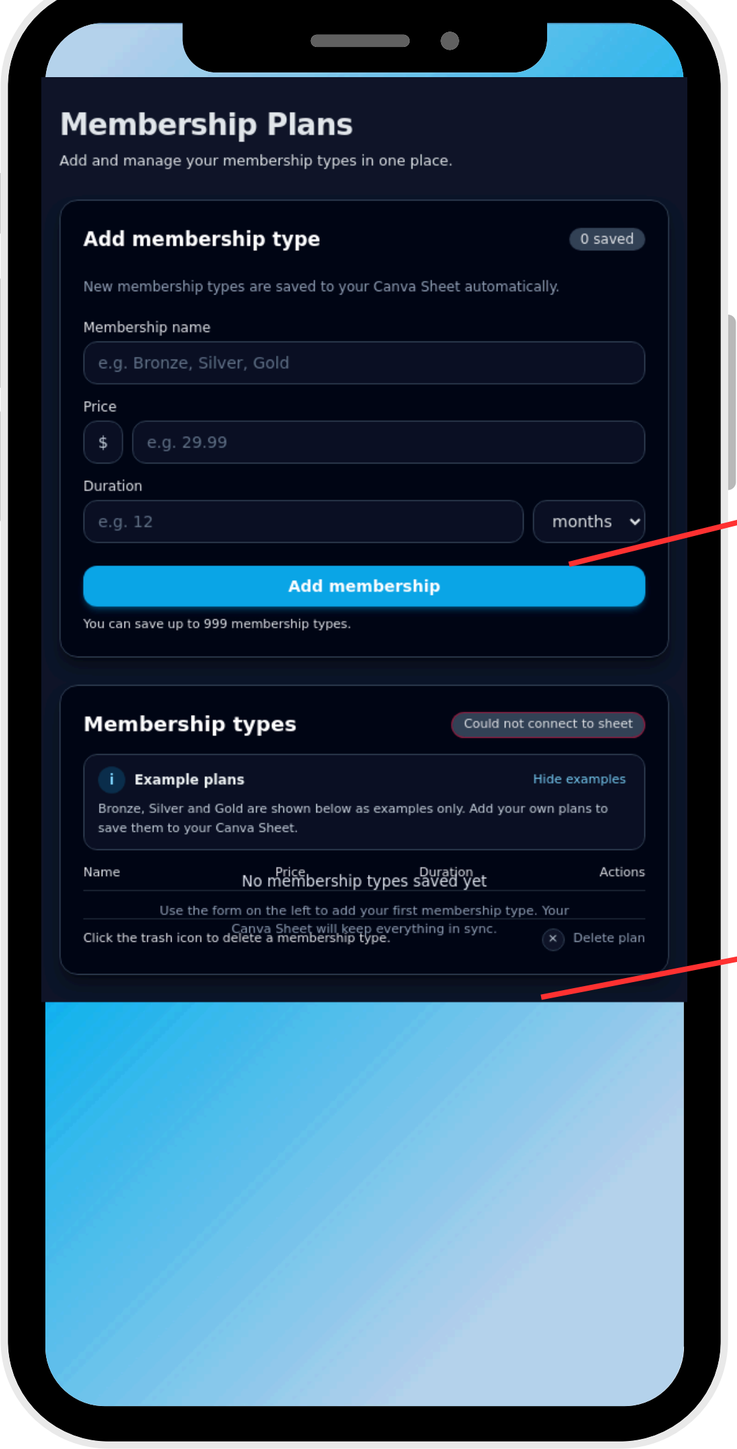
Membership renewal page for Member



after typing in all info then there is the button to continue which trigger this

```
create function create_member_payment(_gym_id integer, _member_id integer, _plan_type_id integer,
_payment_method character varying, _transaction_id character varying) returns text
    language plpgsql
```

Membership page for Owner

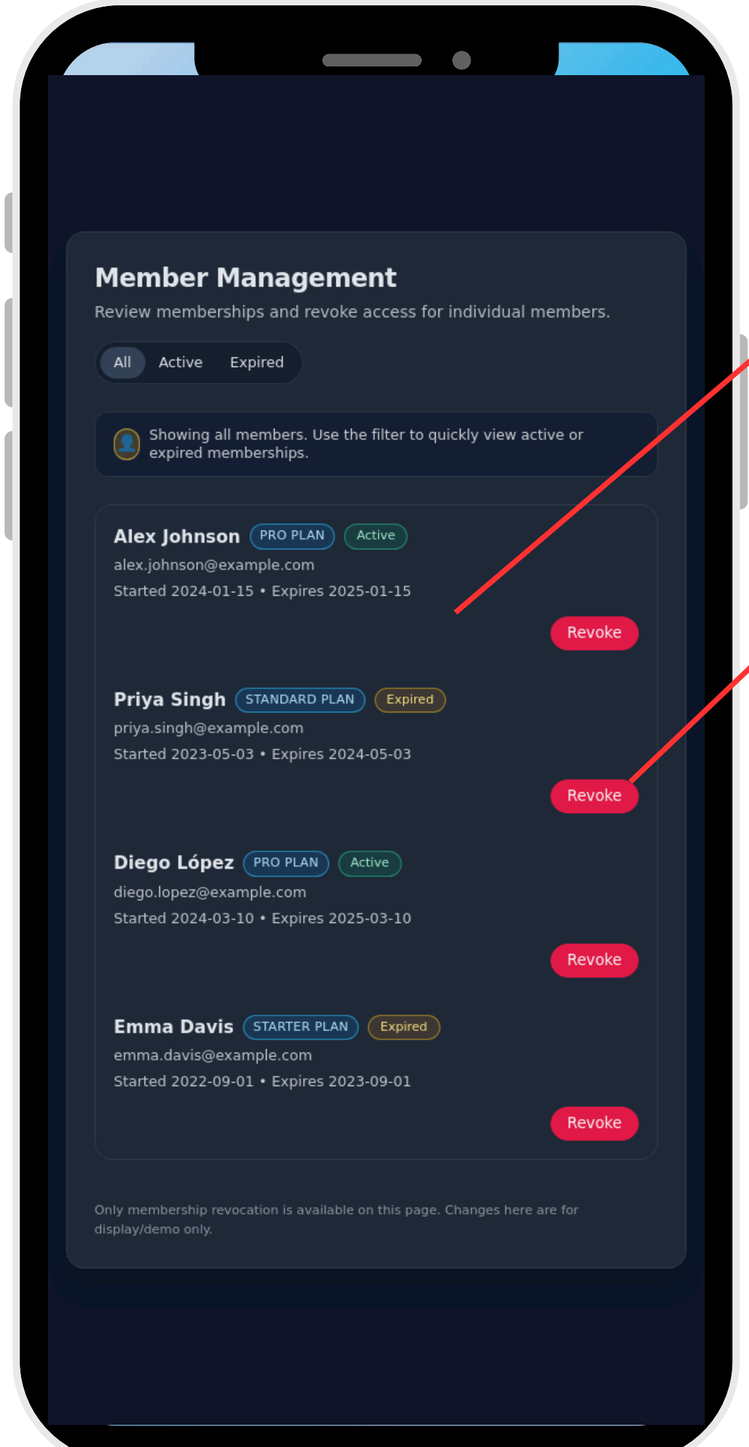



```
create function add_membership_type(_gym_id integer, _name character varying, _price numeric,
_duration_days integer) returns text
    language plpgsql
```

```
create function get_gym_membership_types(_gym_id integer)
    returns TABLE(r_type_id integer, r_name character varying, r_price numeric, r_duration_days integer)
    language plpgsql
```

1	Bronze	1099	30

Member page for Owner



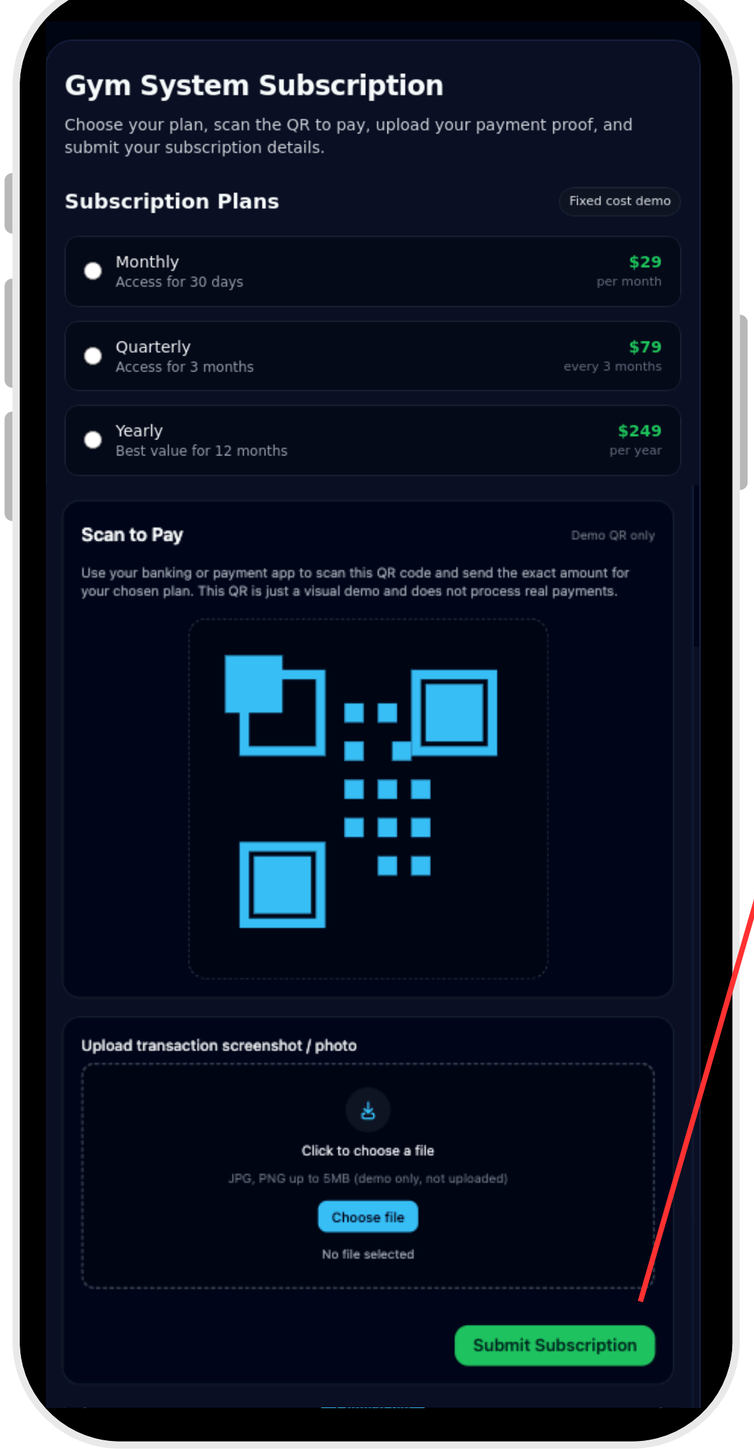
This list all the member and their status

```
create function get_all_expirations(_gym_id integer)
    returns TABLE(r_member_id integer, r_name character varying, r_expires date, r_status text)
    language plpgsql
```

Set the status of that user to be revoke and they camt login and checkin into gym

```
create function revoke_membership(_gym_id integer, _member_id integer, _reason text) returns text
    language plpgsql
```

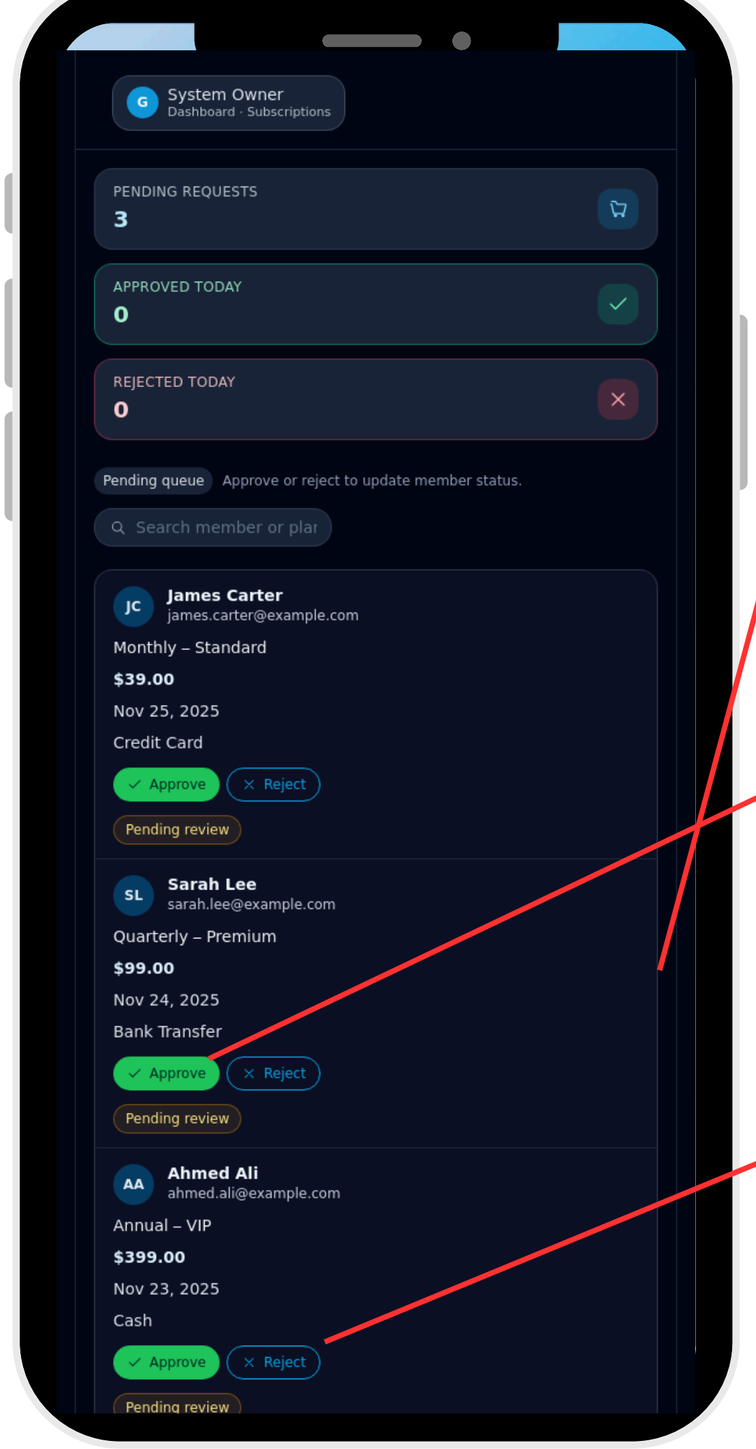
System subscription page for Owner




```
create function submit_payment_proof(_gym_id integer, _plan_type_id integer, _proof_ref character varying) returns text
    language plpgsql
```

```
'Success: Payment proof submitted for ' || _plan_name || ' ($' || _official_price || '). Please wait for Admin approval.';
```

System subscription page for System Owner (To approve)

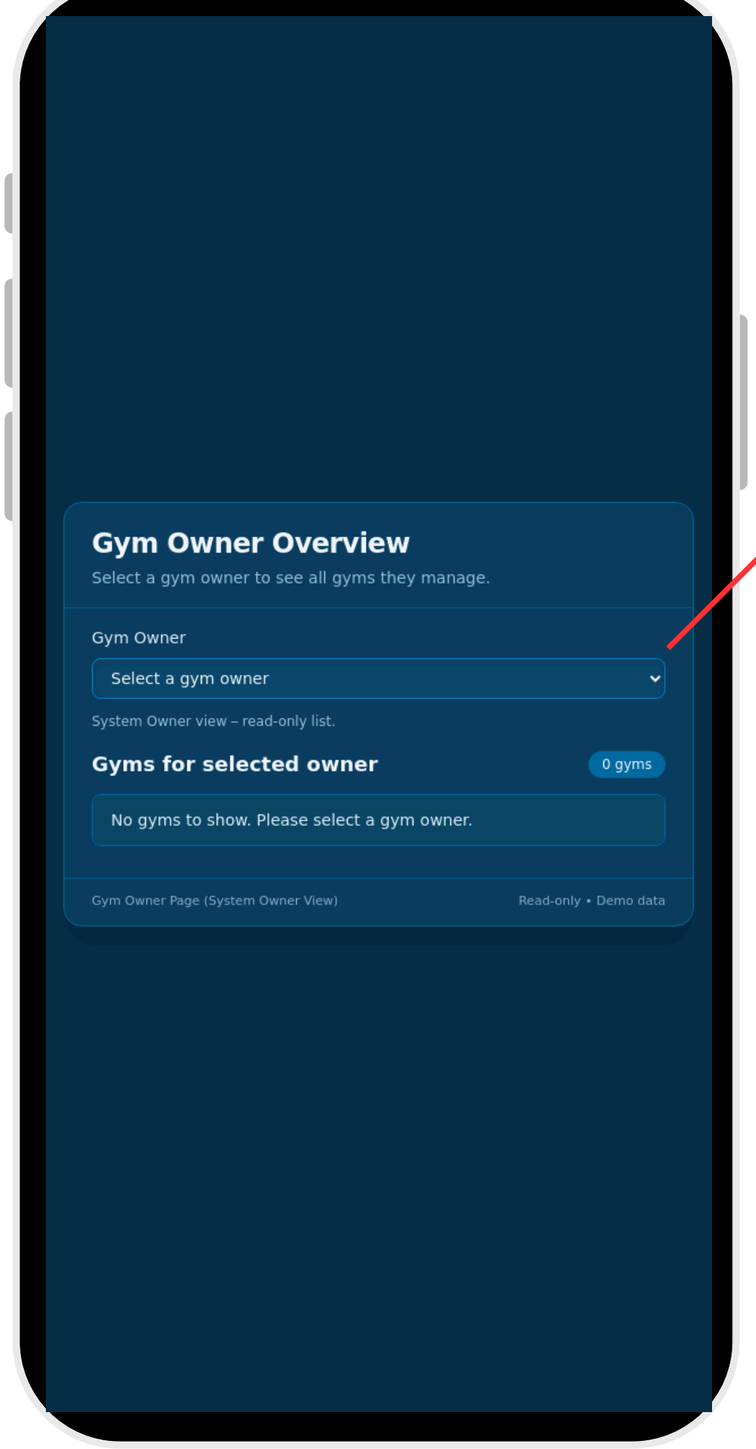



```
create function get_pending_payment_requests()
    returns TABLE(r_request_id integer, r_gym_name character varying, r_owner_name character varying,
r_plan_name character varying, r_amount numeric, r_proof_ref character varying, r_date timestamp without
time zone)
    language plpgsql
```

```
create function approve_payment_request(_request_id integer) returns text
    language plpgsql
```

```
create function reject_payment_request(_request_id integer, _reason text) returns text
    language plpgsql
```

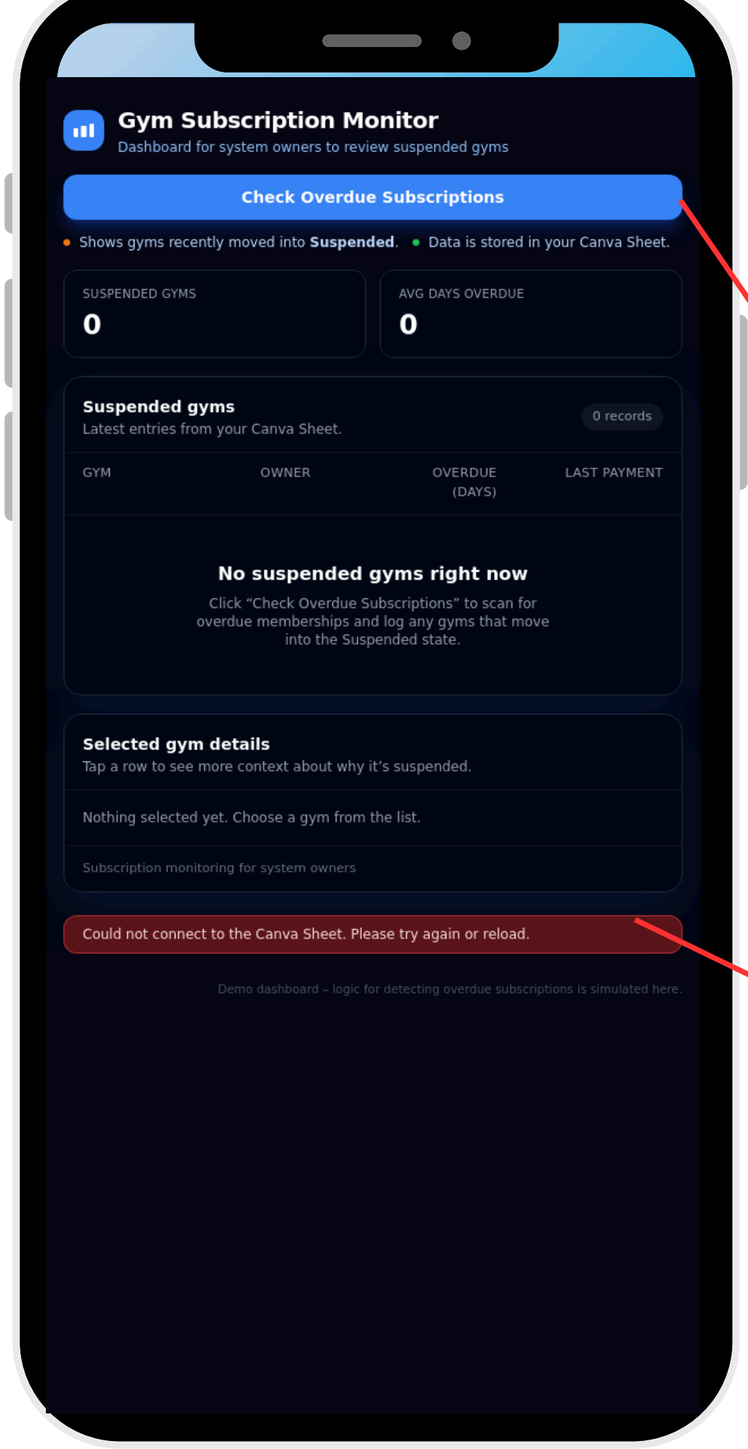
Gym owner page for System Owner




```
create function get_owners_gyms(_owner_id integer)
    returns TABLE(r_gym_id integer, r_name character varying, r_location character varying, r_status
character varying)
    language plpgsql
```

2	Muscle Factory BKK	Bangkok	Active
3	Power Gym	Bangkok	Active

Gym for System Owner




```
create function check_overdue_subscriptions()
    returns TABLE(gym_name character varying, expired_on date)
    language plpgsql
as
$$
BEGIN
    UPDATE gyms
    SET subscription_status = 'Suspended'
    WHERE subscription_end_date < CURRENT_DATE
      AND subscription_status = 'Active';

    RETURN QUERY
    SELECT name, subscription_end_date
    FROM gyms
    WHERE subscription_status = 'Suspended';
```

after click on the gym on the list there should be a button to archive gym

```
create function admin_archive_gym(_gym_id integer) returns text
    language plpgsql
```