You are an expert web application developer specializing in React Native for Web using Expo, with deep knowledge of Vercel for deployment. Your task is to guide the full development and deployment of a high-quality, professional Minimum Viable Product (MVP) web application called "UCU Enavigator" for Uganda Christian University's Faculty of Engineering, Design and Technology. The app targets students and staff in three departments: Computing and Technology, Visual Art and Design, and Engineering. Focus on core objectives: enabling easy event discovery, navigation, sign-ups, and admin management while ensuring scalability, security, usability, and performance.
App Objectives and Requirements:- Core Functionality: - User authentication: Email/password login/sign-up restricted to @ucu.ac.ug domains, with role-based access (users vs. admins). - Event Discovery: Searchable list of events filtered by department, sorted by upcoming/past, with details like title, description, date/time, venue. - Navigation: Interactive Google Maps integration for campus (focus on Mukono campus) with GPS-based directions; fallback to placeholders if location denied. - Event Sign-Up: Embed Google Forms links for RSVPs. - Notifications: Browser-based reminders for events (1 hour before). - Admin Tools: Dashboard for creating/editing events (with verification workflow), approving pending events.- Scope Limitations: MVP only; no advanced features like AR navigation or social sharing until validated. Stick to faculty scope.- Tech Stack: - Frontend: React Native Web via Expo for cross-platform compatibility (web focus). - Backend: Supabase for auth, real-time database (events table), and storage. - Maps: Google Maps API (free tier). - Offline Support: Use AsyncStorage for caching events; disable writes (e.g., create/approve) offline with user-friendly alerts. - Connectivity: @react-native-community/netinfo for online/offline detection. - Other: Expo Location for geolocation, Expo Notifications for reminders.- Quality Standards: - Usability: Responsive design for desktop/mobile browsers; intuitive UI with Material Design principles. Include loading indicators, error alerts, and offline warnings. - Security: Validate inputs, use Supabase RLS for data access, secure API keys via environment variables. - Performance: Optimize for fast loads (code splitting, lazy loading); cache data to reduce API calls. - Accessibility: Follow WCAG guidelines (e.g., alt text for maps, keyboard navigation). - Error Handling: Comprehensive try-catch blocks, user-facing alerts for failures (e.g., network errors, invalid inputs). - Testing: Include unit tests (Jest) for services/auth, integration tests for screens, and manual testing for offline scenarios. - Deployment: Host on Vercel for seamless CI/CD; configure custom domain if needed, enable previews for branches.
Development Guidelines:- Start with project setup: Initialize Expo project, install dependencies from provided package.json.- Implement features iteratively: Auth → Event Listing → Details/Map → Admin → Offline/Notifications.- Use best practices: Modular code (separate services/models/screens), TypeScript for type safety, ESLint/Prettier for code style.- Ensure cross-browser compatibility (Chrome, Firefox, Safari).- Validate against objectives: After each milestone, test if app meets user needs (e.g., quick event navigation) and quality metrics (e.g., no crashes offline).- Output: Provide complete, updated code with comments. If refinements needed, explain changes.Generate the full webapp code, deployment steps for Vercel, and a quality checklist confirming objectives are met

set up the complete database schema and Supabase integration for the UCU Event Nav application. The database includes an events table with RLS policies for security, a profiles table with role-based access (user/admin), and proper triggers for auto-creating user profiles. The SQL scripts are ready to be executed to create all necessary tables, policies, and sample data, build the authentication system with email validation and role-based access,create the event discovery and listing page with search, filters, and sorting, using free tier,

removed all references to `NEXT_PUBLIC_GOOGLE_MAPS_API_KEY` from the codebase so that The event-map component now displays a simple location card with venue information and buttons to open Google Maps externally (no API key required). The documentation should be u been updated to remove all mentions of the Google Maps API key while keeping the app fully functional.

You are an expert in React Native Web development using Expo and Supabase. Revise the existing MVP web app "UCU Event Nav" to enable open sign-up and sign-in with any valid email address (remove @ucu.ac.ug restriction) and add a password reset feature.
Revision Objectives:- Allow broader access: Open auth to any email while keeping verification.- Add recovery: Implement password reset via Supabase email links.### Requirements:- Sign-Up/Sign-In: Eliminate domain check; retain email verification.- Password Reset: Add UI button on login screen; send reset link via Supabase.- UI Updates: Show "Forgot Password?" link; alerts for success/errors (e.g., "Reset link sent").- No changes to other features (events, admin, offline, notifications, maps).
Tech Guidelines:- Update AuthService.js: Remove domain validation; add resetPassword method.- Screens: Modify LoginScreen.js for reset UI/logic (e.g., TouchableOpacity link).- Supabase: Ensure email verification and reset enabled in dashboard.- Future-Proof: Comment removed restriction code.- Quality: Keep responsive UI, error handling, accessibility; test auth flows.### Guidelines:- Focus on auth revisions; build on existing code.- Practices: TypeScript; add comments for changes.- Output: Full updated code (package.json, services, screens); explain revisions; Vercel redeploy steps if impacted.

*
Prompt
:

Add

Department
-
Based

Admin

Access

Control
 to
UCU

Event

Nav
*

You
 are an expert
in

React

Native

Web

with

Expo
 and
Supabase
.

Implement

```
 role
-
based authentication where only
3
 specific department administrators can create events
,

while
 all other
users

(
students
)
 can only view events
.

###
Objectives
:

-

*
Fixed

Admin

Access
*
:

Only

3
 pre
-
defined department admin emails can access admin dashboard
-

*
Admin

Privileges
*
:

Exclusive
 event creation
,
 editing
,
 and deletion rights
-

*
Student

Access
*
:

All
 other verified emails can sign up but only view events
```

-

*
Separate

Dashboards
*
:

Admin
 panel vs
Student
 browsing
interface

###
Requirements
:

*
Hardcoded

Admin

Whitelist
*
:

javascript
const

DEPARTMENT_ADMINS

=

[

'admin1@ucu.ac.ug'
,

// Department 1 Admin

'admin2@ucu.ac.ug'
,

// Department 2 Admin

'admin3@ucu.ac.ug'

// Department 3 Admin

]
;

*
Database

Schema

(
Supabase

)
*
:

-

Add
 role column to profiles
table

(
values
:

'admin'
 or
'student'
)

-

Add
 department
column

(
optional
,

for
 admin tracking
)

-

Only
 these
3
 emails
get

'admin'
 role
;
 all others
default
 to
'student'

*
Authentication

Flow
*
:

-

*
Sign
-
Up
*
:

```
Check
 email against
DEPARTMENT_ADMINS
 array

-

If
 match
:

Assign
 role
=
'admin'


-

If
 no match
:

Assign
 role
=
'student'

-

*
Sign
-
In
*
:

Fetch
 user role
from
 profiles table
-

*
Dashboard

Routing
*
:


-

Admins
 →
AdminDashboard

(
with

Create

Event
```

button
)

-

Students
→
StudentDashboard

(
view
-
only
)

*
Admin

Dashboard

Features

(
Exclusive
)
*
:

-

"Create New Event"

button

(
primary action
)

-

Event
management list
with

Edit
/
Delete
options
-

Event
preview before publishing
-

Optional
:

Event
analytics
/
attendance tracking
*
Student

Dashboard

Features
*
:

-

Browse
 all published events
-

Search
/
filter events
-

View
 event details
-

RSVP
/
notification
features

(
if
 exists
)

-

*
No
*
 create
/
edit
/
delete
 capabilities
*
Authorization

Implementation
*
:

*
AuthService
.
js
 additions
*
:

javascript
// Check if email is department admin

const

isAdmin

```
=

(
email
)

=>

DEPARTMENT_ADMINS
.
includes
(
email
.
toLowerCase
(
)
)
;

// Assign role during signup

const

assignRole

=

(
email
)

=>

isAdmin
(
email
)

?

'admin'

:

'student'
;

// Get user role method

getUserRole
(
)

// Returns 'admin' or 'student'

*
Protected

Routes
*
:
```

-

Wrap

AdminDashboard

in
 role check component
-

Redirect
 non
-
admins attempting to access
/
admin
-

Show

"Access Denied"

for
 unauthorized attempts
*
UI

Conditional

Rendering
*
:

-

Navigation
 menu shows different options per role
-

Admin
:

"Dashboard"
,

"Create Event"
,

"Manage Events"

-

Student
:

"Browse Events"
,

"My Events"
,

"Notifications"

*

Supabase

Security

(
RLS

Policies
)
*
:

sql
--

Events
 table policies
--

Students
 can only
SELECT

CREATE

POLICY

"Students view events"

ON
 events

FOR

SELECT

TO
 authenticated

USING

(
true
)
;

--

Only
 these
3
 admins can
INSERT
/
UPDATE
/
DELETE

CREATE

POLICY

"Admins manage events"

```
ON
 events

FOR

ALL

TO
 authenticated

USING

(

    auth
.
email
(
)

IN

(


'admin1@ucu.ac.ug'
,


'admin2@ucu.ac.ug'
,


'admin3@ucu.ac.ug'


)


)
;

*
Security

Layers
*
:

1.

*
Frontend
*
:

Hide
 admin
UI

from

students
```

(
UX
)

2.

*
Service

Layer
*
:

Role
 validation before
API
 calls
3.

*
Backend

(
Supabase

RLS
)
*
:

Enforce
 at database
level

(
critical
)

###
File

Modifications
:

*
New
/
Updated

Files
*
:

-
 config
/
admins
.
js
:

Export

DEPARTMENT_ADMINS

array
-

AuthService
.
js
:

Add
role assignment logic
-

AdminDashboard
.
js
:

Create
event form
+
management
interface

-

StudentDashboard
.
js
:

View
-
only event browsing
-

ProtectedRoute
.
js
:

Role
-
based route guard component
-
supabase
/
migrations
/
:

Add
role column
+

RLS
policies
*
LoginScreen
.
js
changes
*
:

- After successful login, fetch role
- Navigate to AdminDashboard or StudentDashboard based on role
- Display role-specific welcome message

Deliverables:

- SQL migration for role column and RLS policies
- Hardcoded admin list in config file
- Separate dashboard components
- Role-checking middleware/guards
- Updated navigation structure
- Testing documentation for:

- Admin

login →
Create

Event
access

-

Student
login →
View
-
only access

-

Non
-
admin attempting admin
routes

(
should fail
)

###
Error

Handling
:

-

Clear
messages
:

"Access denied - Admin privileges required"

-

Graceful
redirects
from
unauthorized routes
-

Log
unauthorized access
attempts

(
optional
)

###
Maintenance

Notes
:

-

To

```
 add
/
remove admins
:

Update

DEPARTMENT_ADMINS
 array
+
 redeploy
-

Consider
 moving to database table
if
 admin list changes frequently
-

Document
 which department each admin email represents,
```