Any queries about the assignment, feel free to reach out to Nandan – 8073031940 / nandan@vegam.co

Admin Dashboard: Users Page

Internship Take-Home Assignment

Duration: 3 Days **Tech Stack:** React, TypeScript, Material React Table, MSW

Introduction

You've been given a **partially built** admin dashboard. Your task is to:

1. **Fix 3 bugs** in the existing code
2. **Complete 3 incomplete features**
3. **Build 2 new features** from scratch

This tests your ability to work on a real codebase - reading existing code, debugging issues, and extending functionality.

Getting Started

```
# Install dependencies
npm install

# Initialize MSW (required for mock API)
npx msw init public --save

# Start development server
npm run dev
```

Open http://localhost:5173 to see the app.

What's Already Built

Component	Status
Project setup (Vite, TypeScript, MUI)	Complete
MSW mock API with 100 users	Complete
Material React Table integration	Complete (with bugs)
Metadata-driven columns	Complete (with bugs)
Pagination	Complete (with bugs)
Status filter	Complete

Search input	Incomplete
Snackbar notifications	Complete

Part 1: Bug Fixes (Required)

Bug #1: Table Doesn't Refresh After Status Update

Symptom: When you click activate/deactivate, the snackbar shows success, but the table doesn't update. You have to refresh the page to see the change.

Location: `src/hooks/useUsers.ts`

Hint: Look at the `useUpdateUserStatus` hook. Compare with how other apps handle React Query cache invalidation after mutations.

Bug #2: Groups Column Shows "[object Object]"

Symptom: The "Groups" column should show group names as chips, but instead shows "[object Object]" for each group.

Location: `src/components/tables/DynamicGrid.tsx`

Hint: Look at the `renderCellByType` function, specifically the `chiplist` case. The groups array contains objects with a `groupName` property.

Bug #3: Pagination Not Synced with URL

Symptom: - When you change pages, the URL doesn't update - When you filter by status, the URL doesn't update - When you refresh the page, pagination resets to page 1

Location: `src/pages/UsersPage/UsersPage.tsx`

Expected Behavior: - URL should show: `/users?page=2&status=active` - Refreshing should preserve the current page and filters

Hint: Use `useSearchParams` from React Router to update URL when pagination/filters change.

Part 2: Complete Features (Required)

Feature #1: Debounced Search

Current State: Search triggers API call on every keystroke.

What to Do: - Use the provided `useDebounce` hook in `src/hooks/useDebounce.ts` - Add 300ms debounce to search input - Only fetch when user stops typing

Location: `src/pages/UsersPage/UsersPage.tsx`

Feature #2: Loading Skeleton

Current State: Table shows a generic loading spinner.

What to Do: - Add a proper loading skeleton/placeholder for the table - Should show placeholder rows that match the table structure - Use MUI Skeleton component

Location: `src/pages/UsersPage/UsersPage.tsx` or create a new component

Feature #3: Optimistic UI for Status Toggle

Current State: When toggling user status, you wait for API response before seeing the change.

What to Do: - Immediately update the UI when button is clicked - Show the new status right away (don't wait for API) - If API fails, revert to original status and show error

Location: `src/hooks/useUsers.ts` and/or `src/pages/UsersPage/UsersPage.tsx`

Hint: React Query's `useMutation` has `onMutate`, `onError`, and `onSettled` callbacks for optimistic updates.

Part 3: New Features (Required)

Feature #1: Actions Column with Status Toggle

Current State: Actions column exists but styling could be improved.

What to Do: - Add proper hover states - Add confirmation dialog before deactivating (optional but recommended) - Ensure accessibility (proper aria labels, keyboard navigation)

Feature #2: Error Handling

Current State: Basic error alert if API fails.

What to Do: - Add proper error boundary component - Show user-friendly error messages - Add retry button for failed requests - Handle network errors gracefully

---

Bonus Features (Optional)

Choose one or more for extra credit:

Bonus A: Persist State in URL/localStorage

- Save column visibility preferences
- Remember sort order

Bonus B: Unit Tests

- Test the `useDebounce` hook
- Test the `DynamicGrid` column renderer

Bonus C: Role-Based UI

- Show different actions based on user role
- Add visual indicators for admin users

Column Metadata Reference

```
[
  {"key":"name","header":"Name","type":"string","pinned":"left","width":220,"sorting":true},
  {"key":"email","header":"Email","type":"string","width":260,"sorting":true},
  {"key":"status","header":"Status","type":"badge","width":120},
  {"key":"createdAt","header":"Joined","type":"date","format":"YYYY-MM-DD","width":140},
  {"key":"groups","header":"Groups","type":"chiplist","width":280}
]
```

API Endpoints (Mocked)

Endpoint	Method	Description
`/api/users?page=&pageSize=&query=&status=`	GET	Fetch paginated users
`/api/users/:id`	PATCH	Update user status

Submission Checklist

- ☐ All 3 bugs fixed
- ☐ Debounced search working
- ☐ Loading skeleton added
- ☐ Optimistic UI implemented
- ☐ Error handling improved
- ☐ Separate git commits for each fix/feature
- ☐ README updated with changes
- ☐ Deployed to Vercel/Netlify
- ☐ (Optional) Bonus features

Submission Format

1. **GitHub Repo** - Private repo, share access or Public repo
2. **Live Demo** - Vercel or Netlify URL
3. **README** - Updated with your changes
4. **Video -** Short video or screenshots

Tips for Success

1. **Read the existing code first** - Understand how things work before changing
2. **Follow existing patterns** - Look at how similar things are implemented
3. **Commit often** - One commit per fix/feature
4. **Test thoroughly** - Make sure fixes don’t break other things
5. **Ask questions** - If something is unclear, reach out

Good luck! We look forward to reviewing your submission.