

- Lab 6 - Flutter Data-Driven UI Development with Clean Architecture
 - Overview
 - 🎓 What You'll Learn
 - 🚀 Ready to Begin?
 - What's Already Implemented ✅
 - Navigation System
 - Data Loading & Business Logic
 - Available Data
 - 🚀 Lab 6 Challenge
 - What You Need to Implement 🎨
 - 1. Login Page
(lib/features/user_management/presentation/pages/login_page.dart)
 - 2. Student Profile Page
(lib/features/user_management/presentation/pages/student_profile_page.dart)
 - 3. Tutor Profile Page
(lib/features/user_management/presentation/pages/tutor_profile_page.dart)
 - 4. Admin Profile Page
(lib/features/user_management/presentation/pages/admin_profile_page.dart)
 - Design Resources 📖
 - Design Guidelines
 - Color Themes by Role
 - Data Structure Reference
 - 🛠️ Implementation Strategy
 - Phase 1: Build Common Widgets
 - Phase 2: Implement Pages
 - Phase 3: Polish & Test
 - 🎨 Design Tips
 - Testing ✅
 - Common Widgets 🧩
 - Files to Modify 📝
 - Widget Directory Structure
 - Do NOT Modify ❌
 - 🏆 Lab 6 Success Criteria
 - Technical Excellence

- [Design Excellence](#)
- [User Experience](#)
-  [Lab 6 Deliverables Checklist](#)
 - [Required Components](#)
 - [Bonus Points](#)

Lab 6 - Flutter Data-Driven UI Development with Clean Architecture

Overview

This is **Lab 6**, a continuation of [Lab 5 Flutter Fundamentals 2](#).

In Lab 5, you learned Flutter fundamentals and built basic widgets. Now in Lab 6, you will implement complete UI designs for a real-world application by combining your UI skills with data models - a crucial step in Flutter development.

This project provides a complete Flutter app structure with navigation and data loading, but **you need to implement the UI design** for all pages. The navigation system and data repositories are already implemented and working.

What You'll Learn

By completing Lab 6, you'll have a basic understanding of:

- **Full-stack Flutter development** (UI + Data integration)
- **Clean Architecture patterns** in practice
- **Component-driven development** with reusable widgets
- **Material Design implementation** at scale

Ready to Begin?

Below you have everything you need to complete the lab:

- Clear learning objectives and success criteria

- Target screenshots for reference
- Comprehensive implementation strategy
- All necessary data and architecture already provided

Remember: This lab bridges the gap between learning Flutter fundamentals and building real applications. Take your time, follow the phases, and don't hesitate to experiment with your designs! At any stage if you have any questions please ask.

What's Already Implemented

Navigation System

- **PageSelector:** Main navigation with tabs for Login, Student, Tutor, and Admin
- **Tab switching:** Fully functional navigation between different user roles
- **App structure:** Complete Flutter app setup with Material Design 3

Data Loading & Business Logic

- **Repository Pattern:** Complete implementation with data sources
- **JSON Data:** All sample data files are provided and working
- **User Management:** Student, Tutor, and Admin entities with proper data loading
- **System Analytics:** Admin dashboard data aggregation
- **Course Recommendations:** Academic feature implementation
- **Booking & Session Management:** Complete booking system
- **Review System:** Rating and review functionality

Available Data

- **Users:** 8 users (4 students, 3 tutors, 1 admin) with profile pictures
- **Sessions:** 5 tutoring sessions
- **Bookings:** 6 student bookings
- **Reviews:** Rating system with reviews
- **Courses:** Course catalog and recommendations



Lab 6 Challenge

This lab represents a significant step up from previous labs. While Lab 5 taught you the fundamentals of Flutter widgets and state management, Lab 6 challenges you to:

1. **Bridge the gap between UI and data** - You'll learn how to connect beautiful interfaces with real data models
2. **Think like a developer** - You'll work with existing architecture and understand how to consume data services
3. **Build production-ready interfaces** - You'll create UIs that handle real-world scenarios like loading states and empty data
4. **Master component reusability** - You'll build widgets that can be reused across different parts of the application

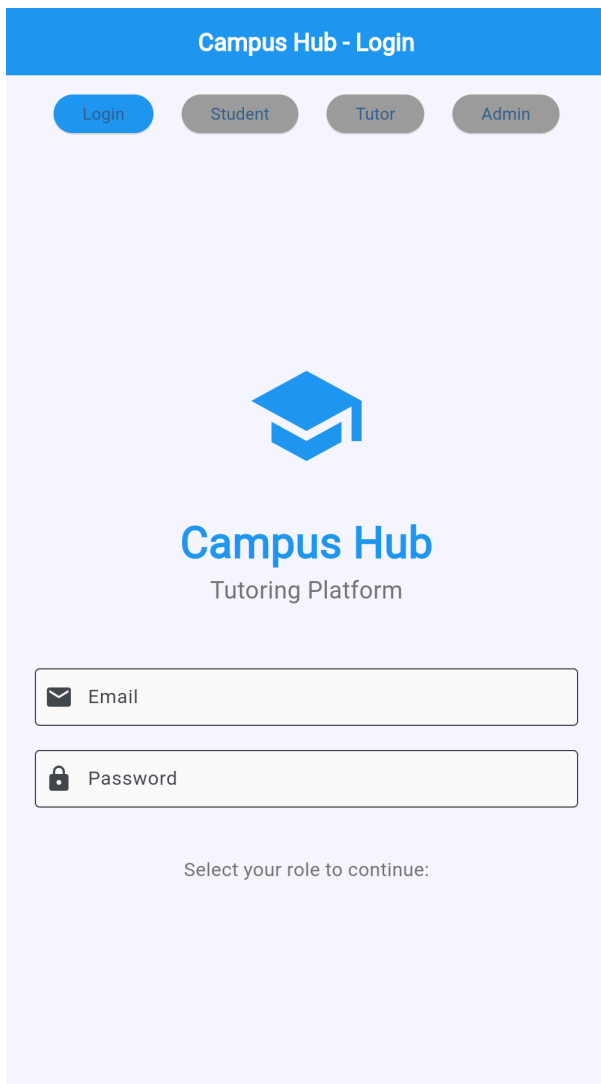
This is your first taste of **full-stack Flutter development** - where UI meets data!

What You Need to Implement

1. Login Page

(**lib/features/user_management/presentation/pages/login_page.dart**)

Current State: Shows placeholder text



The image shows a login screen for 'Campus Hub - Login'. At the top, there is a blue header bar with the text 'Campus Hub - Login'. Below the header, there are four buttons: 'Login' (blue), 'Student' (grey), 'Tutor' (grey), and 'Admin' (grey). In the center, there is a blue icon of a graduation cap. Below the icon, the text 'Campus Hub' is displayed in a large blue font, followed by 'Tutoring Platform' in a smaller grey font. Below this, there are two input fields: 'Email' (with an envelope icon) and 'Password' (with a lock icon). At the bottom, there is a text prompt: 'Select your role to continue:'.

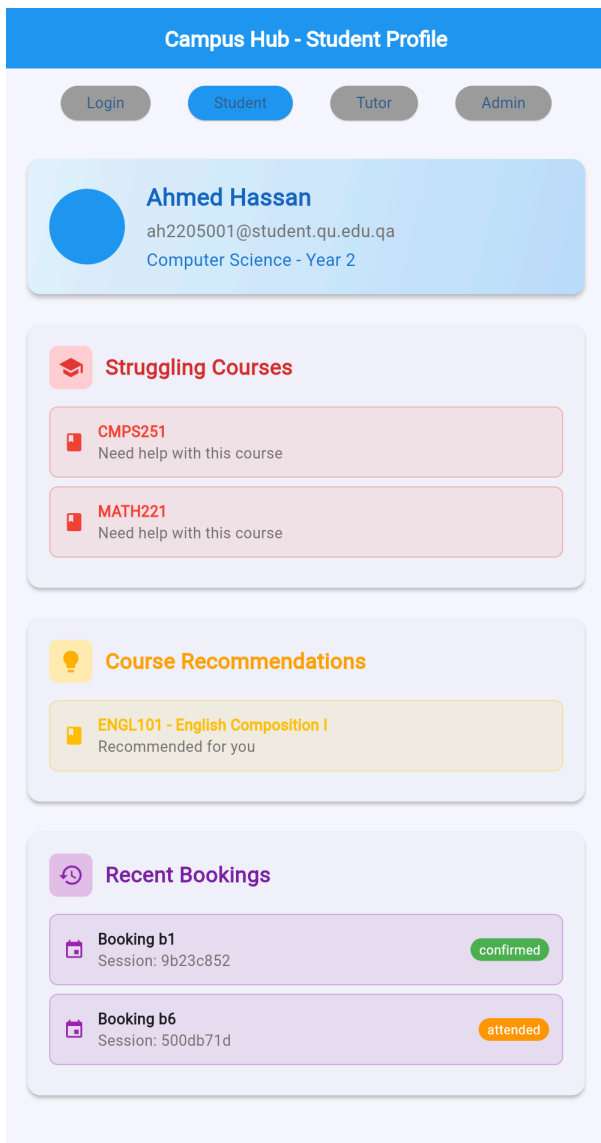
Requirements: Replicate the login screen design above with email/password fields, role selection, and Campus Hub branding.

Available Data: No user data needed (login form only)

2. Student Profile Page

(lib/features/user_management/presentation/pages/student_profile_page.dart)

Current State: Shows basic data in plain text



Requirements: Replicate the student profile design above showing profile header, struggling courses, course recommendations, and booking history.

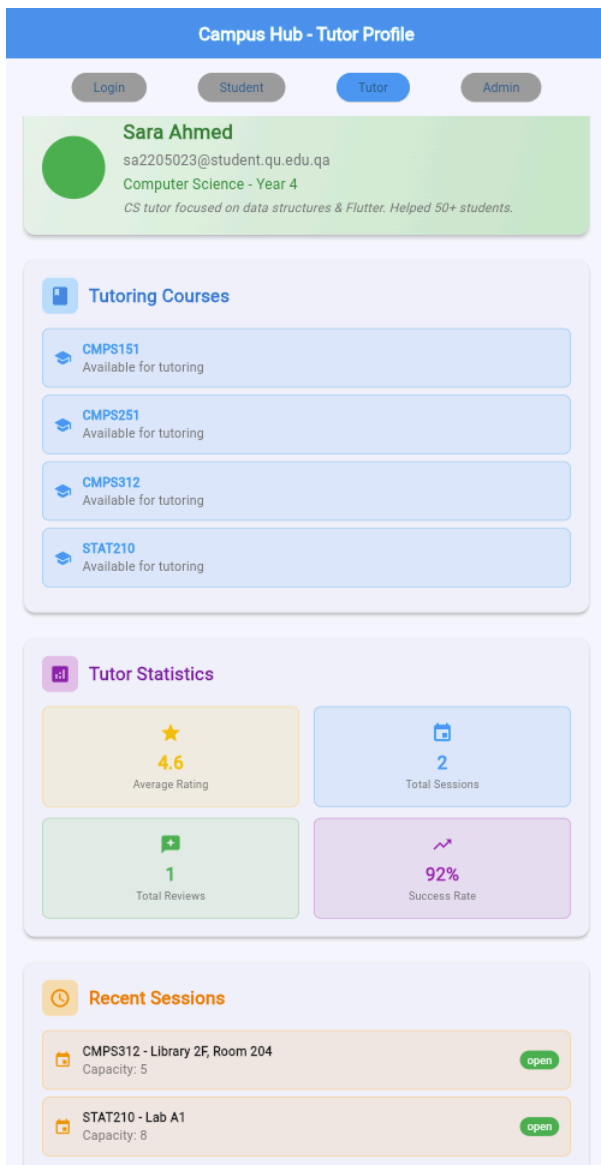
Available Data:

- **student:** Student object with profile information
- **courseRecommendations:** List **<Course>** - recommended courses
- **studentBookings:** List **<Booking>** - student's booking history

3. Tutor Profile Page

(**lib/features/user_management/presentation/pages/tutor_profile_page.dart**)

Current State: Shows basic data in plain text



Requirements: Replicate the tutor profile design above showing profile header, tutoring courses, statistics dashboard, and recent sessions.

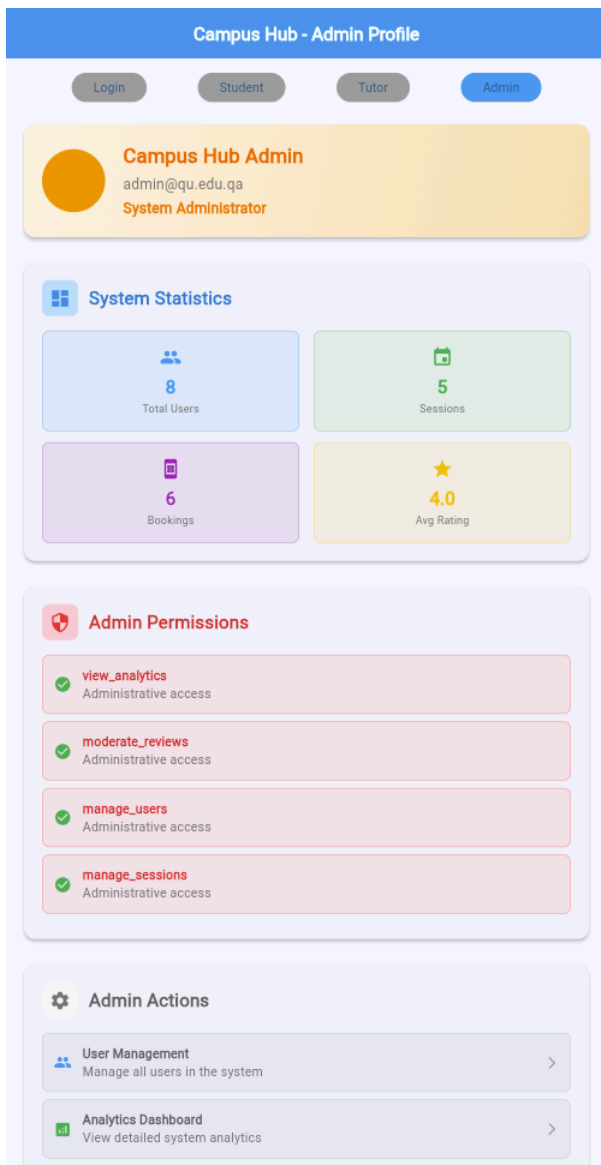
Available Data:

- **tutor**: Student object (with role=tutor) containing tutor information
- **tutorSessions**: List **<Session>** - sessions created by this tutor
- **tutorReviews**: List **<Review>** - reviews for this tutor

4. Admin Profile Page

(lib/features/user_management/presentation/pages/admin_profile_page.dart)

Current State: Shows basic data in plain text



Requirements: Replicate the admin profile design above showing profile header, system statistics dashboard, permissions, and admin actions.

Available Data:

- **admin**: Admin object containing admin information and permissions
- **systemAnalytics**: Map<String, dynamic> containing:
 - **systemAnalytics['users']['totalUsers']** - total number of users
 - **systemAnalytics['sessions']['totalSessions']** - total number of sessions
 - **systemAnalytics['bookings']['totalBookings']** - total number of bookings
 - **systemAnalytics['reviews']['averageRating']** - average rating across all reviews

Design Guidelines

- **File:** `DESIGN_GUIDELINES.md`
- **Contains:** Complete color palette, icon system, UI patterns, spacing system
- **Icons:** All Material Icons used in the app
- **Colors:** Primary colors, variations, role-based themes
- **UI Patterns:** Card design, avatar design, button styles, text styles

Color Themes by Role

- **Student:** Blue theme (`Colors.blue` variations)
- **Tutor:** Green theme (`Colors.green` variations)
- **Admin:** Orange theme (`Colors.orange` variations)

Data Structure Reference

For detailed data structure examples, check the JSON files in `assets/json/` directory:

- `users_sample.json` - Complete user data structure
- `courses_catalog.json` - Course information
- `sessions_sample.json` - Session data
- `bookings_sample.json` - Booking information
- `reviews_sample.json` - Review and rating data

Implementation Strategy

Phase 1: Build Common Widgets

1. **ProfileAvatar** → **StatItem** → **ListItem** → **InfoCard** → **ProfileHeader**

Phase 2: Implement Pages

2. **Login Page** (form design) → **Student Profile** (basic data) → **Tutor Profile** (statistics) → **Admin Profile** (complex data)

Phase 3: Polish & Test

3. Responsive design and error handling

Design Tips

- Use **Card** widgets with elevation 3-4 and 12px border radius
- Apply role-based theming: Student (blue), Tutor (green), Admin (orange)
- Handle loading states and empty data gracefully

Testing

1. **Run:** `flutter run`
2. **Test navigation:** Switch between tabs
3. **Verify data:** Check all data loads correctly
4. **Test responsiveness:** Different screen sizes

Common Widgets

Available in `lib/features/user_management/presentation/widgets/`:

1. **ProfileAvatar** - Avatar with network image/fallback
2. **InfoCard** - Card with header icon and title
3. **StatItem** - Statistics with icon, value, label
4. **ListItem** - List item with consistent styling
5. **ProfileHeader** - Complete header with avatar and user info

 **All widgets show placeholders** - implement based on TODO comments in each file.

Implementation Order: ProfileAvatar → StatItem → ListItem → InfoCard → ProfileHeader

Files to Modify

Focus on implementing the UI in these files:

- `lib/features/user_management/presentation/pages/login_page.dart`
- `lib/features/user_management/presentation/pages/student_profile_page.dart`
- `lib/features/user_management/presentation/pages/tutor_profile_page.dart`
- `lib/features/user_management/presentation/pages/admin_profile_page.dart`

Widget Directory Structure

```
lib/features/user_management/presentation/  
├── widgets/                # Common reusable widgets  
│   ├── profile_avatar.dart # Avatar widget  
│   ├── info_card.dart      # Card widget  
│   ├── stat_item.dart      # Statistics widget  
│   ├── list_item.dart      # List item widget  
│   └── profile_header.dart # Profile header widget  
└── pages/                  # Page implementations  
    ├── login_page.dart  
    ├── student_profile_page.dart  
    ├── tutor_profile_page.dart  
    └── admin_profile_page.dart
```

Do NOT Modify ❌

Do NOT modify these files:

- Navigation system (`page_selector.dart`)
- Data repositories and entities
- JSON data files
- Repository implementations
- Domain models and contracts



Lab 6 Success Criteria

A successful Lab 6 implementation should demonstrate:

Technical Excellence

1. **Data Integration Mastery** - All data displays correctly from repositories
2. **Widget Architecture** - All common widgets implemented and reusable
3. **State Management** - Proper handling of loading, empty, and error states
4. **Code Organization** - Clean, maintainable code following Flutter best practices

Design Excellence

5. **Visual Consistency** - Follows design guidelines and maintains consistent patterns
6. **Role-Based Theming** - Proper blue/green/orange theming throughout
7. **Material Design** - Implements Material Design 3 principles correctly
8. **Responsive Design** - Works seamlessly on different screen sizes

User Experience

9. **Intuitive Navigation** - Clear, logical flow between different user roles
10. **Data Presentation** - Information is clearly organized and easy to understand
11. **Error Handling** - Graceful handling of edge cases and empty states
12. **Performance** - Smooth scrolling and responsive interactions



Lab 6 Deliverables Checklist

Required Components

- ☐ **Login Page** - Complete form with validation and role selection
- ☐ **Student Profile** - Full profile with courses and bookings
- ☐ **Tutor Profile** - Dashboard with statistics and sessions
- ☐ **Admin Profile** - Comprehensive analytics dashboard
- ☐ **Common Widgets** - All 5 widgets fully implemented

Bonus Points

- ☐ **Advanced Styling** - Custom themes and advanced Material Design
- ☐ **Performance Optimization** - Efficient widget rebuilding
- ☐ **Accessibility** - Proper semantic labels and screen reader support

Good luck with Lab 6 - you've got this! 🎯