Tunisian Republic
Ministry of Higher Education and Scientific Research
Tunis Business School

Advanced Programming Report

Submitted to

Tunis Business School

by

Yosser Fhal

TBS SmartHub

Defended on *??/05/2025* in front of:

Contents

Chapter 1

Introduction

In today's fast-paced academic environment, students juggle multiple responsibilities across various platforms. The fragmentation of academic information across different applications creates unnecessary stress and often leads to missed deadlines and communication gaps. The TBS SmartHub project addresses this critical pain point by providing a centralized mobile application specifically designed for Tunis Business School students.

Chapter 2

Project Overview

2.1 Problem Context

Students at Tunis Business School, like many university students worldwide, face the challenge of managing information scattered across multiple platforms. Course materials might be on one system, while schedules are on another, and communication with professors happens through yet different channels. This fragmentation creates several problems:

- Time wasted switching between applications
- Increased likelihood of missing important updates
- Difficulty in maintaining a comprehensive view of academic responsibilities
- Stress from information overload and disorganization

2.2 Solution Approach

The TBS SmartHub addresses these challenges by creating a unified platform with the following key features:

- Centralized dashboard for all academic information
- Integrated calendar with course schedules and events
- Direct communication with professors
- AI-powered chatbot for academic assistance
- Secure authentication system with role-based access
- Multi-language support for diverse student needs

Built using Kotlin and Android Studio, and integrated with Firebase for back-end services.

2.3 Project Objectives

The main objectives of the TBS SmartHub project are:

1. Provide students with a single access point for all academic information
2. Offer secure and flexible authentication options
3. Integrate AI tools for enhanced learning support
4. Support multiple languages for accessibility
5. Create an easily maintainable and extensible platform

Chapter 3

Implementation Details

3.1 Authentication System

The authentication system implements multiple login methods to provide flexibility while maintaining security:

- Email/password authentication
- Google Sign-In integration

The implementation uses Firebase Authentication as the backend service, with a local repository pattern to abstract the authentication details from the rest of the application.

```
class FirebaseAuthRepository {
// Current user state
private val \_currentUser = MutableStateFlow\<User?>(null)
val currentUser: StateFlow\<User?> = \_currentUser.asStateFlow()

// Sign in with email and password
suspend fun signIn(email: String, password: String): Result<User> {
// Implementation details
}

// Register new user
suspend fun register(email: String, password: String, name: String): Result<User> {
// Implementation details
}

// Sign in with Google
suspend fun signInWithGoogle(idToken: String): Result<User> {
// Implementation details
}
}
```

This approach ensures that authentication logic is centralized and can be easily modified or extended without affecting other parts of the application.

3.2 Dashboard Implementation

The dashboard serves as the central hub of the application, providing an at-a-glance view of the student's academic life:

- Personalized greeting with task summary
- Calendar events for the current day
- List of linked teachers for quick communication
- Upcoming events and deadlines

The implementation uses a combination of cards and lists to organize information in a visually appealing and easily scannable format.

```
@Composable
fun HomeScreen(navController: NavController) {
val currentUser by firebaseAuthRepository.currentUser.collectAsState()
val events by eventRepository.todayEvents.collectAsState()

LazyColumn {
item { GreetingCard() }
item { /\* Calendar implementation */ }
item { /* Teachers implementation */ }
item { /* Events implementation \*/ }
}
}
```

This modular approach allows each section to be developed and tested independently, while maintaining a cohesive overall experience.

3.3 AI Chatbot Integration

The AI chatbot feature provides students with immediate assistance for academic questions, leveraging the OpenAI API:

- Text-based conversation interface
- Image analysis capabilities for document understanding
- Conversation history management

The implementation includes fallback mechanisms to handle situations where the API is unavailable or not configured.

```
class ChatViewModel : ViewModel() {
private val openAIClient = OpenAIClient(System.getenv("OPENAI\_API\_KEY") ?: "")

fun sendTextMessage(content: String) {
if (openAIClient.hasValidApiKey()) {
// Get response from OpenAI
} else {
// Provide fallback response
}
}

// Similar pattern for image analysis
}
```

3.4 Calendar and Events System

The calendar system provides a structured view of academic schedules and events:

- Daily, weekly, and monthly views
- Integration with course schedules
- Event details including location and description
- Filtering capabilities by course or event type

The implementation uses a repository pattern to abstract the data source:

```
class EventRepository private constructor() {
private val allEvents = mutableListOf<Event>()
private val \_todayEvents = MutableStateFlow\<List<Event>>(emptyList())
val todayEvents: StateFlow\<List<Event>> = \_todayEvents.asStateFlow()

fun getEventsForDate(date: String): List<Event> { /*...*/ }
fun addEvent(event: Event): Boolean { /*...*/ }
fun updateEvent(event: Event): Boolean { /*...*/ }
fun deleteEvent(eventId: String): Boolean { /*...*/ }
}
```

This design allows for mock data usage in development and supports future integration with real backend systems.

Chapter 4

Future Enhancements

4.1 Technical Enhancements

Several improvements can be made to the technical foundation of TBS SmartHub to increase robustness and scalability:

- **Real-time Synchronization:** Integrate Firebase Firestore listeners to enable real-time updates for schedules, messages, and announcements.
- **Offline Support:** Incorporate Room Database for offline caching and seamless offline experience.
- **Push Notifications:** Implement Firebase Cloud Messaging to send important alerts such as exam reminders or announcements.
- **Performance Optimization:** Optimize performance using lazy loading and pagination for handling large data sets.
- **Automated Testing:** Expand test coverage with UI automation and integration tests for continuous quality assurance.

4.2 Feature Enhancements

Future iterations of the app could introduce new functionality to enhance student productivity and engagement:

- **Grade Tracking:** Securely store and visualize academic performance.
- **Document Repository:** Provide access to course materials and shared documents via cloud integration.
- **Peer Collaboration:** Include tools for managing group projects and study groups.
- **Campus Map:** Offer interactive navigation for university facilities.
- **Event Registration:** Enable sign-up for university events and track participation.

Chapter 5

Conclusion

The TBS SmartHub project was initiated with the objective of addressing the fragmentation of academic tools used by students at Tunis Business School. By creating a centralized, AI-powered mobile application, the project aimed to simplify access to academic resources, improve communication, and support student success.

Throughout the development process, the project emphasized a user-centric approach, focusing on seamless user experience, secure access, and intuitive navigation. The integration of modern technologies such as Jetpack Compose, Firebase, and OpenAI APIs allowed for rapid development while maintaining a high standard of quality and flexibility.