CS404: Applied Mobile Engineering

Applied Mobile Engineering & Systems Architecture

NIT Jalandhar

14-Day Program

Course Description

This intensive 14-day engineering track provides a comprehensive foundation in modern mobile application development with Flutter and Dart. Students engage with production-grade mobile engineering concepts including cross-platform rendering architectures, state management, network protocols, cloud infrastructure, and deployment pipelines. The curriculum bridges theoretical computer science with applied systems design, culminating in the deployment of production-ready mobile applications.

Learning Objectives

Upon successful completion of this course, students will be able to:

- Design and implement cross-platform mobile applications using Flutter's declarative UI framework
- Apply object-oriented programming principles and asynchronous programming patterns in Dart
- Architect scalable client-server systems with RESTful APIs and real-time WebSocket communication
- Integrate cloud-based backend services including NoSQL databases and authentication systems
- Implement modern UI/UX design principles including responsive layouts and adaptive theming
- Deploy production applications with proper security, obfuscation, and performance optimization
- Utilize version control systems and collaborate on open-source software projects
- Integrate machine learning APIs and understand inference with large language models

Course Schedule

Day	Topics & Concepts
1	**Flutter Architecture & Foundations:** Introduction to Flutter framework and its advantages over native development; rendering pipeline including Skia and Impeller engines; compilation strategies (JIT vs AOT); graphics APIs (OpenGL, Vulkan, Metal); Dart language fundamentals; Flutter project structure and widget composition; understanding the widget, element, and render object trees; coordinate systems; Material Design components including MaterialApp, Scaffold, and Container; flex layouts with Row, Column, and Expanded widgets. *Assignment: Personal information card UI*
2	**Object-Oriented Programming & State Management:** Core OOP concepts including classes, objects, and inheritance in Dart; distinction between StatelessWidget and StatefulWidget; managing application state with setState(); hot reload vs hot restart development workflow; package management with pubspec.yaml; pseudo-random number generation using dart:math library. *Assignment: Interactive coin flipper application*
3	**Navigation & Dynamic Lists:** Stack data structure and its application in navigation; Navigator API for multi-screen applications; understanding BuildContext and context propagation; screen transitions with Navigator.push() and pop(); Dart list data structures; implementing scrollable lists with ListView; performance optimization through ListView.builder for lazy rendering; visual separators with ListView.separated; widget extraction and composition patterns. *Assignment: Multi-screen game interface*
4	**Advanced Layouts & UI/UX Design Principles:** Grid-based layouts with GridView, GridView.builder, and GridView.count; foundational UI/UX principles including the 8-point grid system; anti-aliasing for visual clarity; fat finger rule for touch targets; safe area handling for notched displays; typographic scale hierarchy; 60/30/10 color distribution rule; effective use of whitespace; integrating external packages from pub.dev; Google Fonts package integration; semantic versioning conventions; recommended typography choices; implicit animations with AnimatedContainer; explicit animations with AnimationController; Dart mixins and SingleTickerProviderStateMixin. *Assignment: Spotify home screen clone*
5	**Network Programming & Asynchronous Operations:** Client-server architecture fundamentals; Internet protocols and HTTP specification (RFC); request-response paradigm; HTTP/1.1 protocol analysis; domain name system and IP addressing (IPv4/IPv6); port allocation and network sockets; local server deployment demonstration; RESTful API design principles; HTTP methods and status codes; http package for network requests; browser developer tools for debugging; concurrency with threads; Future-based asynchronous programming; async/await syntax; Dart event loop mechanics; FutureBuilder for reactive UI. *Assignment: Integration with dummyjson.com API*

6 **User Input & Data Persistence:** Text input with TextField widgets; managing input with TextEditingController; form validation using TextFormField; focus management with FocusNode; form state management with Form and GlobalKey; computer memory hierarchy; data persistence strategies; implementing local storage with shared_preferences package. *Assignment: To-do application with persistent storage*

7 **Theming & Cloud Backend Services:** Application-wide theming with ThemeData; color scheme generation from seed colors; implementing light and dark mode with persistent user preference; Backend-as-a-Service (BaaS) platforms; comparison of BaaS vs traditional servers; vertical and horizontal scaling architectures; Cloud Firestore NoSQL database; document-oriented data models; Firebase security rules; Firebase project setup and CLI tools; Firestore read and write operations. *Assignment: Anonymous confession box with cloud storage*

8 **Real-time Communication & Authentication:** WebSocket protocol fundamentals; HTTP 101 Switching Protocols; WebSocket bidirectional communication architecture; Dart Streams for reactive programming; StreamBuilder for real-time UI updates; comparing stateless HTTP vs stateful WebSockets; authentication mechanisms and JSON Web Tokens (JWT); Firebase Authentication integration; user registration and login flows. *Assignment: Authentication system with login, signup, and logout*

9 **Integrated Application Development:** Live coding session building anonymous confession application; advanced concepts including cryptographic hashing for security; dropdown widgets and form controls; integrating previously learned concepts into cohesive application architecture.

10 **Monetization & Version Control:** In-app purchases, subscriptions, and ad-supported revenue models; advertising identifiers (GAID/IDFA); supply-side platforms (SSP) and demand-side platforms (DSP); ad exchange auctions (first-price vs Vickrey); advertisement formats including banner, interstitial, and rewarded ads; Android manifest and iOS info.plist configuration; Android test ad unit IDs; Google Mobile Ads SDK integration; banner ad implementation; version control fundamentals; Git and GitHub workflows; commit message conventions; collaboration patterns; personal access tokens; branching strategies; forking and pull requests; contributing to open-source projects.

11 **Platform Integration & Deep Linking:** Knowledge transfer equation ($K = i \times c$); social sharing with share_plus package; platform channels and method codecs for native integration; URL launching across platforms; deep linking architecture; custom URI schemes; app links package; Android intent filters; deep link configuration and testing workflows.

12	**Large Language Models & API Integration:** Introduction to large language models (LLMs); recurrent neural networks (RNNs) and transformer architectures; vector embeddings and semantic representations; GPU types and CUDA/Tensor cores; numerical precision formats (FP16, INT8); model inference pipelines; memory bandwidth considerations; quantization techniques for model compression; tokenization strategies; running models locally with Ollama; Groq Language Processing Units (LPUs); HTTP POST requests; API testing with cURL and Postman. *Assignment: Gemini API integration for conversational AI*
13	**Advanced Concurrency & Streaming:** Dart Streams and async generators; streaming responses from Groq API; concurrency vs parallelism; race conditions and deadlocks; Isolates and the actor model for parallel computation; preventing UI thread blocking during heavy computation; Foundation library's compute() function; inter-isolate communication with SendPort and ReceivePort; implementing compute-intensive tasks in isolates.
14	**Deployment & Production Optimization:** Application icon generation with flutter_launcher_icons; development dependencies vs runtime dependencies; splash screen implementation with flutter_native_splash; modifying application display names; environment variable management with .env files and flutter_dotenv; code optimization through tree shaking and R8 compiler; code obfuscation for intellectual property protection; building Android APK and AAB (Android App Bundle) formats; ABI splitting for size optimization; introduction to Gradle build system; Android package name modification; Play App Signing infrastructure; cryptographic key generation and application signing for distribution.

Required Materials

- Laptop with Flutter SDK installed (Windows, macOS, or Linux)
- Android Studio or Visual Studio Code with Flutter extensions
- Git version control system
- Firebase account (free tier)
- GitHub account for version control and collaboration

Academic Integrity

Students are encouraged to collaborate and learn from peers, but all submitted work must represent individual effort and understanding. Proper attribution must be given when utilizing external code libraries, API documentation, or open-source resources.