ESSAY DESCRIPTION

CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT - 502071

SEMESTER 1 – ACADEMIC YEAR 2025 – 2026

Lecturer: Mai Van Manh

Last updated: Aug 17, 2025

Overview and Purpose

This group assignment is designed to help students explore extended topics in Flutter that are not covered directly in the course curriculum. The goal is to allow you to gain a deeper understanding of advanced concepts and tools, reinforcing your theoretical knowledge with hands-on experience. Each group will select one topic from a predefined list and will work on it comprehensively, both theoretically and practically.

Your task is to dive deeply into the theoretical aspects of your chosen topic, then develop an illustrative demo that exemplifies the key concepts. Finally, you will submit a comprehensive report, source code, a video presentation, and clear instructions for the graders to run your project.

Group Responsibilities

Each group (2-3 members) will:

- **Thoroughly research the chosen topic**: Your group must explore the theoretical background, best practices, and applications of the topic. This might involve studying relevant documentation, blogs, research papers, and other external resources.
- **Develop a working demo**: You will implement a small-scale to medium-scale demo project to showcase the theoretical concepts in practice. The demo should reflect the essence of the chosen topic and help explain its practical use.
- **Prepare a detailed report**: This report must follow the template provided by the Faculty and include the required sections (see below). Your report should provide a structured presentation of your research, demo, architecture, and results.
- **Collaborate effectively**: All team members should contribute equally to the project and video presentation.

ID	TOPIC'S NAME	TOPIC'S DESCRIPTION AND RECOMMENDATION These are just suggested guidelines; groups are encouraged to explore the topic more broadly and comprehensively beyond the information provided here
1	**State Management in Flutter: Comparative Analysis and Use Cases**	**Objective**: Explore and implement an advanced state management solution in Flutter, focusing on understanding its architecture, benefits, and practical use cases. **Key Focus**: • State Management Solutions: Choose one advanced state management solution (e.g. Riverpod, Bloc, or MobX) and dive deep into its architecture and principles. • Implementation: Develop a Flutter application using the chosen state management solution to address specific use cases. • Comparative Analysis: Understand how the chosen solution compares to other state management options in terms of performance, ease of use, and scalability. **Deliverables**: • A Flutter app demonstrating the implementation of the chosen state management solution with practical use cases. • A detailed analysis of the selected state management solution, including its architecture, implementation details, and a comparison with other state management approaches.

2	**Platform-Specific Code in Flutter: Writing Native Plugins**	**Objective**: Learn how to write platform-specific code in Flutter and how to create a Flutter plugin for Android and iOS. **Key Focus**: • Discuss the need for writing platform-specific code (native functionalities not supported by Flutter out-of-the-box). • Step-by-step guide to building a plugin that bridges Flutter with native code (e.g., accessing device sensors or native APIs). • Discuss plugin architecture (e.g., method channels) and how to ensure it works across Android and iOS. **Deliverables**: • A custom Flutter plugin that accesses a native feature. • A demo app utilizing the plugin on both platforms. • A report explaining plugin architecture, method channels, and performance implications.
3	**Internationalization (i18n) and Localization (l10n) in Flutter**	**Objective**: Explore how to add internationalization and localization support to Flutter apps for building multilingual applications. **Key Focus**: • Explain the concepts of i18n and l10n and their importance in building multi-language apps. • Investigate Flutter's intl package and best practices for managing translations. • Discuss how to handle language-specific formatting for dates, numbers, and currencies. **Deliverables**: • A multi-language demo app using the intl package with at least 3 different languages. • A report on challenges faced during internationalization and localization, and how they were solved.

4	**Flutter with Firebase: Authentication, Database, and Cloud Functions**	**Objective**: Explore the integration of Firebase services into Flutter applications, focusing on authentication, real-time databases, and cloud functions. **Key Focus**: • Investigate Firebase Authentication for user sign-in/sign-up and secure access to app resources. • Examine how to use Firebase's real-time database or Firestore for data persistence and retrieval. • Explore how to trigger and use Firebase Cloud Functions for backend logic. **Deliverables**: • A demo app that integrates Firebase Authentication, Firestore, and Cloud Functions. • A report explaining each Firebase feature used and any challenges encountered during integration.
5	**Flutter and Cloud Integration: Building Scalable Backend Solutions**	**Objective**: Explore how to integrate Flutter apps with cloud-based services to create scalable mobile solutions, focusing on cloud providers such as AWS, Google Cloud, and Firebase. **Key Focus**: • Investigate the best practices for integrating cloud storage, authentication, and serverless computing (using AWS Lambda, Google Cloud Functions, etc.) with Flutter. • Study how to manage and optimize network requests, handle errors, and implement caching strategies for efficient cloud data fetching. • Research cloud service SDKs available for Flutter and their limitations. **Deliverables**: • A cloud-connected app demonstrating authentication, real-time data syncing, or serverless functions integration. • A report detailing the architecture, cloud integration process, and potential scalability challenges.

6	**Flutter and Artificial Intelligence (AI): Integrating Machine Learning Models**	**Objective**: Investigate how to design large, modular Flutter applications, focusing on separation of concerns, scalability, and maintainability. **Key Focus**: • Explore architectural patterns for breaking down a large Flutter app into modules (e.g., feature-based architecture, package splitting). • Study best practices for managing dependencies between modules, lazy-loading features, and reducing compile times. • Investigate how to structure a large app to ensure it can grow without becoming unmanageable (e.g., using micro frontends, plugin-based architecture). **Deliverables**: • A demo showing a modular Flutter application with multiple independent features or modules. • A report detailing the architectural choices, how dependencies were managed, and the pros/cons of the modular approach.
7	**Flutter DevOps: Automating Builds, Testing, and Deployments**	**Objective**: Investigate how to automate the building, testing, and deployment process for Flutter applications using modern DevOps practices. **Key Focus**: • Explore how to set up an automated CI/CD pipeline for Flutter apps, from automated testing to build creation and deployment to app stores (Google Play, Apple App Store). • Investigate tools such as Fastlane, GitHub Actions, Codemagic, and Bitrise for automating builds and releases. • Learn how to implement automated testing (unit, widget, integration) and ensure code quality during the build process. **Deliverables**: • A working CI/CD pipeline for a Flutter app, demonstrating automated testing and deployment. • A report explaining how the pipeline was set up, tools used, and the benefits of automating the development lifecycle.

8	**UI Automation Testing in Flutter: Implementing End-to-End Testing**	**Objective**: Implement and explore UI automation testing in Flutter to ensure user interface functionality and consistency. **Key Focus**: • Widget Testing: Utilize Flutter's widget testing framework for automating UI component tests. • Integration Testing: Use the integration_test package to simulate user interactions and validate app workflows. • Cross-Device Testing: Ensure UI consistency across various devices and screen sizes. **Deliverables**: • A Flutter app with automated widget and integration tests, including cross-device UI consistency. • A detailed report on UI testing strategies, implementation, and challenges.
9	**Integrating ChatGPT with Flutter: Building Conversational AI**	**Objective**: Explore how to integrate OpenAI's ChatGPT into a Flutter application to create a conversational AI feature. **Key Focus**: • API Integration: Use the OpenAI API to connect ChatGPT with a Flutter app. • User Interaction: Implement a chat interface that allows users to interact with ChatGPT in real-time. • Handling Responses: Process and display responses from ChatGPT in a user-friendly format. **Deliverables**: • A Flutter app with integrated ChatGPT for conversational interactions. • Detailed explanation of the integration process, user interaction design, and any challenges encountered.

Submission Requirements

Your final submission will consist of the following components:

1. **Complete Report (Word and PDF Format):** The report must be written according to the Faculty's template and must be professionally structured and free of grammatical or formatting errors. Ensure that you include appropriate references to external sources used in your research.
2. **Source Code and Libraries**
   - Complete source code: Provide all the source code and related files necessary to run your demo project. The code should be clean, well-documented, and adhere to coding best practices.
   - Dependency management: Include all necessary libraries or dependencies that are required to run the project. Make sure to specify these dependencies in a pubspec.yaml file.
   - Your submission must be organized into folders and include a README file (see 3.4) for clear instructions on how to set up and run the project.
3. **Product Presentation Video (or live presentation in class)**
   - Duration: Maximum 20 minutes.
   - Content: This video must include a presentation of the theoretical aspects, demo project, and its architecture, as well as an overview of how your project works in practice.
   - Each team member must participate in the presentation.
   - You can use Google Meet, Zoom, or any other tool to record your presentation.
   - The video should demonstrate the demo project, showing how it works, the technologies used, and explaining key parts of the code or architecture.
   - Make sure that the video is clear, concise, and well-organized. The grader should easily understand what your group has done and how it reflects the chosen topic.
4. **Readme.txt** - Instructions for Use and Testing
   - Setup instructions: Provide a step-by-step guide on how to set up the environment and run your project. Make sure the grader can follow these instructions easily and run your demo on their computer.
   - Include any system requirements (e.g., OS, Flutter version, additional dependencies).
   - Provide necessary commands for installing dependencies, running the project, and testing specific features.
   - Ensure that all setup instructions have been thoroughly tested.
   - The README should be clear, well-structured, and easy to follow.

CRITERIA	POINT	0	25%	50% - 75%	Full Score
REPORT	**6.0**				
Theoretical Understanding	2.0	No relevant theoretical content provided.	Superficial understanding, missing key concepts, lacks detail and depth.	Covers basic theoretical aspects but lacks clarity or detail in certain sections.	Thorough and in-depth explanation of all key theoretical aspects, demonstrating a strong understanding.
Topic Relevance	1.0	No connection to the chosen topic.	The topic is mentioned but lacks focus and depth, with significant irrelevant sections.	Somewhat focused on the topic, but with gaps or distractions from the main subject.	Completely relevant, with a clear and focused discussion on the chosen topic, fully addressing key points.
Comparative Analysis	1.0	No comparison of different approaches.	Comparison is mentioned but is vague or incorrect, with little insight into differences.	Comparison covers different approaches but lacks depth or is only partially correct.	Detailed and insightful comparison of different approaches, highlighting their strengths and weaknesses effectively.
Report Presentation*	2.0	No report or highly incomplete report.	Report is basic, lacks structure, or has significant formatting or content issues.	Mostly complete but contains some formatting errors, lacks detail in certain sections, or has inconsistent clarity.	Well-organized and professionally formatted report, follows required structure, and provides clear explanations and logical flow throughout.
Demo/Presentation	**4.0**	This section won't be evaluated unless it includes a **demo video with sound** and comprehensive **source code with instructions**			

Implementation of Demo	1.0	No demo or completely non-functional.	Demo is incomplete or non-functional, with significant errors.	Demo is functional but incomplete, missing key features, or has major bugs.	Fully functional demo, meeting all requirements and accurately reflecting the theoretical work.
Architecture Presentation	1.0	No architecture presented or explained.	Architecture is presented but is unclear, inaccurate, or too simple.	Basic architecture is presented but lacks depth, explanation, or clarity in some parts.	Detailed architecture is presented clearly, demonstrating strong design principles and how the system is structured.
Illustration of Use Case	1.0	No use case demonstrated.	Use case is illustrated minimally, without a clear connection to the topic.	Use case is demonstrated but lacks completeness or depth, with some unclear areas.	The use case is fully demonstrated, with clear, well-explained examples that showcase the application of the theoretical topic.
Presentation and Clarity of Demo	1.0	No demo presentation or explanation.	Demo is presented poorly with little clarity, missing important details.	Demo is explained but some parts are unclear, or there are missing elements.	Demo is presented clearly, with all parts explained thoroughly, demonstrating a solid understanding of the topic and how the demo was implemented.

* Refer to the attached file **Report templates and guidelines 2024.zip** for the regulations and essential guidelines to craft a well-written report. A well-written report should avoid the following mistakes (the list is not exhaustive)

- The cover page was presented carelessly, with serious errors such as: using a substandard logo, incorrect instructor information, incorrect group member information, incorrect topic name.
- Missing necessary appendix pages as required. Not having enough chapters (or contents) as required such as: theoretical survey, requirement analysis, system design, presentation of results with necessary analysis/comparison, conclusion chapter of the report.
- Spelling errors; inconsistent font, color, and font size; failure to use appropriate margins and line spacing; overuse of bullet points in presentation.
- The report has a lot of text but lacks images, diagrams, tables, and charts to illustrate and make the content easier for readers to understand. This also shows the possibility of abusing AI tools to write reports.

- Lack of investment in presenting images, tables, and diagrams: for example, images that are too large or too small, fonts/background colors that are difficult to read, screenshots of redundant content, leading to a poorly presented report.
- Too much mention of source code while lacking description or explanation of the system, the operating principles of the components, or how the team implemented and deployed the system, makes it difficult for readers to understand what the team is working on, how they did it, what the results were, or what good things can be learned from this topic.

Late Submission Penalty: For every day that an assignment is submitted late, **one point** will be deducted from the total possible score. Please note that even 1 second late will be considered as one full day late.

Teamworking Policy

To ensure effective collaboration and equitable distribution of work, the following guidelines apply to team-based projects:

- **Team Size**: Each team must consist of 2-3 members.
- **Single-Member Teams:** Teams with only one member will receive a **0.5-point deduction**. This policy encourages teamwork and reflects the expectations for group collaboration.
- **Exception:** If a team member drops out of the course or is unable to continue working on the project due to extenuating circumstances, the remaining member(s) should notify the teacher immediately. In such cases, the point deduction and score cap may be waived at the instructor's discretion, depending on the situation.

Points may aslo be deducted in the following cases:

- The report or video presentation is not in English.
- Not all team members are present in the video presentation, unless previously approved.
- The demo video has poor audio or video quality, making it difficult to understand or follow.
- There is an uneven distribution of work among team members.
- The submission does not adhere to the specified format or is missing required information, such as:
    - Missing or incomplete team member details.
    - Insufficient descriptive information, such as failing to include instructions for running the source code or not providing the necessary admin account credentials to access the application.