CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT - 502071

SEMESTER 1 – ACADEMIC YEAR 2024 – 2025

Lecturer: Mai Van Manh

*Version 1.1, Last updated on Sept **22**, 2024. Future updates will be shown here and highlighted in green*

FINAL PROJECT

SIMULATED EMAIL SERVICE APPLICATION

I. OVERVIEW

Your task is to develop an application that simulates an email service similar to Gmail. The goal is to create a functional email application using Flutter, either on the web or on the mobile. Since this is a simulated service, there is no need to use standard email protocols (IMAP/POP/SMTP). Instead, the application will use HTTP or sockets to exchange data between the client and the server. The application should only support internal communication — users can only send and receive emails with other users in the system.

All data should be stored on a database server (although you can use local databases such as SQLite if a database server is not possible), ensuring that users can access their accounts and emails across multiple devices. The server side can be implemented using any platform, such as using Firebase or you can develop your own back end servers using frameworks like ExpressJS, SpringBoot, ASP.NET, etc.

II. PROJECT REQUIREMENTS

1. Account Management

- Registration: Users register using their phone number (not email).
- Login: Secure login functionality.
- Password Management: Allow password changes and recovery.
- Two-step Verification: Option for users to enable/disable.
- Profile Management: View and update personal information, including profile pictures.

2. Compose and Send Email

- Basic Fields: Compose emails with fields like 'To', 'Subject', and 'Body'. Support for sending attachments.
- Advanced Editor: Include fields like 'CC', 'BCC' and integrate a WYSIWYG (What You See Is What You Get) editor for formatting.
- Reply and Forward: Reply to or forward emails to other users.
- Drafts: Autosave unsent emails as drafts.
- Email Actions: View metadata, assign labels, star, mark as read/unread, move to trash.

3. View Emails

- Organize Emails: Default folders like 'Inbox', 'Starred', 'Sent', 'Draft', and 'Trash'. Users can manage their labels but cannot alter these folders.
- Display Modes for email list: Support both basic (minimal details like sender and subject) and detailed views (includes previews, attachments, etc.).

4. Search Functionality

- Basic Search: Simple keyword-based search.
- Advanced Search: Utilize an advanced search interface with various filters (e.g., date range, attachments).

5. Label Management

- Manage Labels: List, add, remove, and rename labels.
- Label Assignment: Add or remove labels from emails.
- Filter by Label: Display emails based on selected labels.

6. Notifications for New Emails

- When the app is running, users should receive a notification or badge update in the app when they receive a new email.
- The notification should show the sender, subject, and time received.
- Realtime update: automatically display new emails in the inbox without requiring the user to refresh the page or reopen the screen.

8. Auto answer mode

- Auto Answer Mode allows users to automatically reply to all incoming emails with a predefined response. When enabled in the settings screen, any email received will trigger an automatic reply with the specified message, ensuring that the sender receives an immediate acknowledgment.
- Users can customize the response content and toggle this feature on or off based on their preferences.

7. Settings and User Preferences

- Users should be able to update basic preferences such as notification settings (e.g., turn off notifications).
- Users can also select a default font size, font family for the email text editor.
- Implement Dark Mode support, and allow users to switch between Light and Dark themes.
- Turn on/off auto answer mode.

8. Deployment

- Web Platform: Deploy the application to any hosting service (e.g., Firebase, AWS, Netlify) and provide a public URL for access.
- Mobile Platform (Android): Package the application as an APK file optimized for ARM64 architecture and ensure it runs smoothly on Android devices.

9. AI/Machine Learning Integration (plus points)

- Teams can integrate AI/ML/DL related features into the application and will receive bonus points. Each feature will be added 0.25 points and the total bonus points for this section will not exceed 1.0 point. Here are some suggestions for AI/ML/DL features:
  - Automatically label emails as spam
  - Language detection and translation
  - Suggest short answers
  - Any other useful AI-powered features

III. DETAIL SCORE (RUBRIC)

ID	FEATURES		1	2	3
		POINTS	0 PT	1/2 PTS	FULL POINTS
Account Management – 1.5 point					
1	Change password	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
2	Password recovery	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
3	Enable and use 2 steps verification	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.

4	View profile info and picture	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
5	Change profile info	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
6	Change profile image	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
Compose and Send Email – 2.5 points					
7	Send simple text email	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
8	Auto save as draft	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
9	Answer an email	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
10	Forward an email	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
11	Send email in CC and BCC	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
12	Advanced text editing with WYSIWYG	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.

13	sending and receiving attachments	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
14	Perform actions on an email (view meta-data, move to trash, mark read, assign labels)	0.5	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
15	Starred an email	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
Email Management and Settings – 4.0 points					
16	View emails in different categories (inbox, sent, draft, starred, trashed...)	0.5	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
17	View email list in basic view	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
18	View email list in detail view	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
19	Search email by keywords	0.5	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
20	Advanced searching	0.5	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
21	Label management	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.

22	Add/remove a label to an email	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
23	View email list by label type	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
24	Display notification	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
25	Realtime update inbox list	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
26	User setting screen	**0.5**	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
27	Auto answer	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use.	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
Others – 2.0 points					
28	UI and UX	**1.0**	UI and UX are either missing or poorly executed, resulting in a confusing and unattractive app	Basic UI and UX are present but inconsistent, with some usability issues and limited visual appeal.	UI and UX are well-designed, intuitive, and visually appealing, providing a seamless and engaging user experience.
29	Deployment web version	**0.5**	The web version is not deployed or accessible; no public URL is provided, and the app may have significant errors.	The web version is deployed and accessible but may have minor errors or issues. The public URL is provided but may not be fully functional.	The web version is successfully deployed to a hosting platform, runs smoothly without errors, and the public URL is provided and functional.

30	Teamwork (github insights)	0.5	No evidence of teamwork on GitHub; contributions are sporadic or missing, with no early or regular commits.	Some teamwork is visible with occasional commits, but work distribution is uneven, and there are delays or a lack of early contributions.	Evidence of effective teamwork with balanced work distribution, early and regular commits, and proactive collaboration throughout the project.

The above description is only relative in nature and cannot list detailed instructions for each feature on correct or incorrect implementation. However, during grading, features must be relatively well-developed to be eligible for full points. Groups must be active in referring to related applications and applying their daily app usage experience in their assignments. For examples:

- Login Feature:
  - Poor approach: When logging in to use the app, nothing unusual happens; however, if the app is closed and reopened, users are prompted to log in again.
  - Better approach: After successful login, the app must save data to mark that the user has logged in, so in subsequent app openings, users can use the app immediately without needing to log in again (unless the session has expired).
- When changing the user avatar:
  - Poor approach: You have to close and reopen the screen to see the results.
  - Better approach: The new avatar will be updated automatically as soon as it is changed, and other users will see this update too.
- When sending attachments:
  - Poor approach: After picking the files, the user is unable to review files before sending.
  - Better approach: Both sender and recipient can view files directly inside the app (for some popular formats)

IV. OUTPUT REQUIREMENTS

- Required submission components include:
  - The "**source**" folder: Contains the entire source code of the Flutter application, web server, and API if applicable, along with relevant database files. Ensure that this source code can be compiled and run on the teacher's computer. The project needs to be "cleaned" to remove unnecessary content before submission and to reduce the size of the compressed file.
  - The "**bin**" folder: This folder should contain executable files for at least two platforms, including Android (APK) and Windows (EXE)/or Web platform. Teachers will use these files for installation on their real/virtual machines for assessment. If necessary, teachers can choose to build the source code from scratch.

- Introduction video "**demo.mp4**": A team representative should record a screen presentation showcasing the group's application, highlighting ALL features based on self-assessment. The video, with a minimum resolution of 1080p, should have clear and audible sound without theoretical explanations.
- The "**git**" folder: Contains screenshots that demonstrate collaboration between team members on the github or gitlab repo. Multiple screenshots may be submitted, as long as there is evidence of effective teamwork with balanced work distribution, early and regular commits, and proactive collaboration throughout the project. Evidence of teamwork must clearly indicate that the project duration from the start to the present is at least one month, during which each member must have at least two commits per week.
- **Readme.txt** file: Provide all necessary information for the evaluation process, such as project building and running instructions, URL + server login information (if applicable), and usernames/passwords for accounts with pre-loaded data for assessment. Include any relevant notes on building, running, and using the application for teachers to reproduce the project. If your team implements some optional features (which get extra points), it should be clearly stated in the readme file.
- **The "Bonus"** folder**:** The bonus folder should include a description of the extra features your team implemented for additional points, along with evidence. Organize the information clearly, concisely, and convincingly.
- **Rubrik.docx:** This file lists the 30 required features for the project. Teams should self-assess their completion level in this file. The instructor will provide this file at the submission time. The file will also include the public URL to the web application and any required username/password for login.

- Organize all the above contents into a folder named **id1_fullname1_id2_fullname2**, then compress this folder in ZIP format with the same name, e.g., id1_fullname1_id2_fullname2.zip. A team representative should submit this file on the online learning system as instructed by the course instructor.
- Teachers do not accept sumissions via email, only elearning is accepted.

IV. IMPORTANT NOTES

- If teams develop their own backend (instead of using third-party services like Firebase), they will receive an additional 0.5 points.
- The Final Project must be implemented using a **Flutter** project using the **Dart** programming language. Groups are allowed to use any libraries within the framework of a Flutter project.
- Teams may use online storage services such as Firebase or equivalent services, or set up their own web server.

- If your group submits a project that is not relevant, it will not be graded, and the entire group will receive a score of 0. For example, if your group obtains the source code of another email client application for some reason, and only a few features are related to the description in this requirement, while the majority are completely unrelated, the project will be given 0 point.
- The Essay is entirely independent of the Final Project. Therefore, all team members must participate in both the Essay and the Final Project. The Essay will be assessed by the lab instructor, while the Final Project will be assessed by the theoretical instructor.
- Groups are prohibited from sharing code with each other, obtaining source code from the internet, and must take responsibility for protecting their group's source code. Groups with similar source code (verified by specialized software) or code found online, even if only in part, will receive a score of 0 for all members, regardless of which group shared or received the code.
- Failure to submit source code will result in the entire team receiving a score of 0.
- If the team does not fill out and submit the **rubric.docx** file, the submission will not be graded.
- Failure to deploy the web version to public hosting or submit the apk/exe file will result in the entire team receiving a score of 0.
- In case the submission is missing or there are no specific instructions on how to run the project, and the lecturer has tried his best but still cannot run the project on his computer, leading to no way to check the results, the whole group will also receive 0 points.
- Deductions will also apply in the following situations:
    - Late submission: 1-day late deducts 1 point. Submissions late by 1 second to less than 1 day are considered 1 day late.
    - Complex project configuration without specific instructions for instructors to compile and run the program: Deduction of 2 points.
    - Submit the entire project without performing a clean to remove unnecessary files: 0.5 point.
    - Failure to provide necessary grading information, such as missing usernames/passwords, incorrect file naming, or not submitting required content: 1.0 point.