BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Chủ đề 1: Xây dựng ứng dụng Android quản lý lịch học, điểm thi, ngày thi sinh viên EAUT bằng Android Studio và ngôn ngữ lập trình Java

Sinh viên thực hiện	Lớp	Khóa
Nguyễn Trí Dũng	DCCNTT 13.10.16	13
Nguyễn Trung Chính	DCCNTT 13.10.16	13
Trần Văn Nam	DCCNTT 13.10.16	13
Vũ Văn Phong	DCCNTT 13.10.16	13
Đổng Trung Đức	DCCNTT 13.10.16	13

Bắc Ninh, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Nhóm: 6

Chủ đề 1: Xây dựng ứng dụng Android quản lý lịch học, điểm thi, ngày thi sinh viên EAUT bằng Android Studio và ngôn ngữ lập trình Java

STT	Sinh viên thực hiện	Mã sinh viên	Điểm bằng số	Điểm bằng chữ
1	Nguyễn Trí Dũng	20223155		
2	Nguyễn Trung Chính	20222999		
3	Trần Văn Nam	20222996		
4	Vũ Văn Phong	20222998		
5	Đổng Trung Đức	20222877		

CÁN BỘ CHẤM 1
(Ký và ghi rõ họ tên)

CÁN BỘ CHẤM 2
(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC CHỮ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	EAUT	East Asia University Technology
2	XML	Xtensible Markup Language
3	SDK	Software Development Kit
4	API	Application Programming Interface
5	APK	Android Package
6	HTTPS	HyperText Transfer Protocol Secure

LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, việc áp dụng công nghệ thông tin vào quản lý và tổ chức các hoạt động trong học tập đã trở thành nhu cầu thiết yếu. Là sinh viên của Trường Đại học Công nghệ Đông Á (EAUT), chúng em nhận thấy việc quản lý lịch học, điểm thi và ngày thi thường gây khó khăn cho nhiều bạn sinh viên do khối lượng thông tin lớn và việc truy cập thông tin không thuận tiện.

Dự án "Xây dựng ứng dụng Android quản lý lịch học, điểm thi, ngày thi sinh viên EAUT" được thực hiện nhằm áp dụng các kiến thức đã học vào thực tế, đồng thời cung cấp một giải pháp công nghệ giúp sinh viên quản lý hiệu quả thông tin học tập.

Ứng dụng hướng đến việc hỗ trợ sinh viên tra cứu lịch học, điểm thi và nhận thông báo về ngày thi một cách dễ dàng và nhanh chóng. Với giao diện thân thiện, dễ sử dụng, ứng dụng sẽ đồng bộ thông tin từ hệ thống nhà trường để đảm bảo độ chính xác, giúp tiết kiệm thời gian và nâng cao hiệu quả học tập.

Mục tiêu chính của dự án là xây dựng một ứng dụng Android với các tiêu chí rõ ràng: dễ sử dụng, giao diện trực quan và tính năng phù hợp với nhu cầu thực tế của sinh viên. Ứng dụng sẽ hỗ trợ sinh viên thực hiện các thao tác như:

- Xem lịch học hằng ngày, hàng tuần hoặc theo tháng.
- Tra cứu điểm thi một cách nhanh chóng.
- Nhận thông báo về ngày thi và các sự kiện quan trọng.
- Phạm vi của ứng dụng tập trung vào việc hỗ trợ các sinh viên thuộc EAUT, với thông tin được đồng bộ từ hệ thống dữ liệu của trường.

Ứng dụng được thiết kế dành riêng cho sinh viên Trường Đại học Công nghệ Đông Á (EAUT). Tuy nhiên, ứng dụng có thể mở rộng hoặc điều chỉnh để phù hợp với các đối tượng học sinh, sinh viên ở các cơ sở giáo dục khác trong tương lai.

Dự án được phát triển trên nền tảng Android Studio, sử dụng ngôn ngữ lập trình Java. Android Studio là môi trường phát triển tích hợp (IDE) mạnh mẽ dành cho lập trình ứng dụng Android, cung cấp nhiều công cụ và thư viện hữu ích. Chúng em áp dụng các công nghệ như:

- Firebase: Quản lý cơ sở dữ liệu và thông báo đẩy (push notification).
- XML: Thiết kế giao diện người dùng.
- SQLite: Lưu trữ thông tin ngoại tuyến trên thiết bị.

Những công nghệ này đảm bảo ứng dụng hoạt động hiệu quả, linh hoạt và thân thiện với người dùng.Với mong muốn tạo ra một sản phẩm hữu ích, chúng em cam kết đầu tư thời gian và công sức để hoàn thành dự án một cách tốt nhất.

Bảng phân công nhiệm vụ

Thành viên	Mã sinh viên	Nhiệm vụ chính	Ghi Chú
Nguyễn Trí Dũng	20223155	Xây dựng các giao diện XML. Xử lý api lấy thông tin người dùng và logic thông tin người dùng	
Nguyễn Trung Chính	20222999	Xử lý api login và logic đăng nhập	Nhóm trưởng
Trần Văn Nam	20222996	Xử lý api lịch học và logic lịch học	
Vũ Văn Phong	20222998	Xử lý api điểm học tập và logic điểm học tập	
Đổng Trung Đức	20222877	Xử lý api lịch thi và logic lịch thi	

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về phát triển ứng dụng di động

1.1.1 .Lý thuyết về nền tảng Android

Android là một hệ điều hành mã nguồn mở được phát triển bởi Google, dựa trên nhân Linux và được thiết kế dành riêng cho các thiết bị di động như điện thoại thông minh, máy tính bảng. Android cung cấp một môi trường phát triển mạnh mẽ với các công cụ hỗ trợ lập trình viên xây dựng và triển khai ứng dụng dễ dàng. Hệ điều hành này chiếm lĩnh thị phần lớn nhất trong lĩnh vực di động, nhờ sự linh hoạt và khả năng tùy biến cao.

1.1.2. Ưu và nhược điểm của nền tảng Android

❖ **Ưu điểm:**

- Mã nguồn mở: Android cho phép tùy chỉnh linh hoạt và không hạn chế.
- Cộng đồng lớn: Có một cộng đồng rộng lớn hỗ trợ lập trình viên với tài liệu, diễn đàn và tài nguyên phong phú.
- Tính phổ biến: Được sử dụng trên nhiều thiết bị với các cấu hình và giá cả khác nhau, từ cao cấp đến phổ thông.
- Kho ứng dụng phong phú: Google Play Store cung cấp hàng triệu ứng dụng phục vụ nhiều nhu cầu.

❖ **Nhược điểm:**

- Phân mảnh: Có rất nhiều phiên bản Android khác nhau, gây khó khăn trong việc đảm bảo ứng dụng hoạt động ổn định trên tất cả các thiết bị.
- Bảo mật: Do mã nguồn mở, Android dễ bị tấn công hơn nếu không có các biện pháp bảo mật phù hợp.
- Tài nguyên phần cứng: Một số ứng dụng có thể tiêu tốn nhiều tài nguyên hơn, dẫn đến giảm hiệu năng thiết bị.

1.2. Kiến thức cơ bản về Android

1.2.1. Giới thiệu về cấu trúc ứng dụng

Một ứng dụng Android bao gồm nhiều thành phần chính, mỗi thành phần có vai trò quan trọng trong việc xây dựng và vận hành ứng dụng. Mỗi phần đều có chức năng riêng nhưng phối hợp với nhau để tạo nên một ứng dụng hoàn chỉnh và hiệu quả :

- Activity: Thành phần giao diện chính, nơi người dùng tương tác trực tiếp.
- Service: Xử lý các công việc nền không yêu cầu giao diện.
- Broadcast Receiver: Nhận và xử lý các sự kiện hệ thống hoặc ứng dụng khác.
- Content Provider: Quản lý và chia sẻ dữ liệu giữa các ứng dụng.

1.2.2. Vòng đời Activity

Activity có vòng đời rõ ràng được quản lý bởi hệ thống Android, bao gồm các trạng thái chính như onCreate(), onStart(), onResume(), onPause(), onStop(), và

onDestroy(). Hiểu rõ vòng đời Activity giúp lập trình viên tối ưu hóa tài nguyên và xử lý các sự kiện đúng cách.

1.2.3. Quản lý bộ nhớ và giao diện người dùng

Android sử dụng bộ thu gom rác (Garbage Collector) để tự động quản lý bộ nhớ, nhưng lập trình viên cần tối ưu hóa mã nguồn, tránh rò rỉ bộ nhớ. Về giao diện, Android cung cấp các công cụ thiết kế trực quan như XML và hỗ trợ lập trình viên tạo giao diện đẹp mắt, tối ưu trên nhiều kích thước màn hình.

1.3. Tổng quan về công cụ và môi trường phát triển

1.3.1. Android Studio

Android Studio là môi trường phát triển tích hợp (IDE) chính thức do Google cung cấp, được xây dựng dựa trên IntelliJ IDEA. Với các tính năng toàn diện, Android Studio là công cụ không thể thiếu trong phát triển ứng dụng Android. Nó hỗ trợ lập trình viên với các tính năng nổi bật như:

Hình 1. 1 Android Studio

- Trình giả lập (Emulator): Kiểm thử ứng dụng trên nhiều thiết bị mô phỏng.
- Trình biên dịch Gradle: Quản lý thư viện và tài nguyên, tối ưu hóa quy trình phát triển.
- Giao diện kéo thả: Hỗ trợ thiết kế giao diện trực quan với XML.
- Phân tích hiệu năng: Công cụ kiểm tra và tối ưu hiệu suất ứng dụng.
- Tích hợp Git: Dễ dàng quản lý mã nguồn và làm việc nhóm.

1.3.2. Android SDK

Android SDK (Software Development Kit) là một bộ công cụ toàn diện được Google phát triển, cung cấp đầy đủ các thư viện, API, và công cụ hỗ trợ cần thiết để lập trình, kiểm thử, cũng như triển khai ứng dụng Android. Bộ công cụ này được thiết kế nhằm đáp ứng mọi nhu cầu của lập trình viên, giúp tối ưu hóa quá trình phát triển ứng dụng một cách dễ dàng và hiệu quả. SDK thường được tích hợp sẵn trong Android Studio, giúp phát triển ứng dụng dễ dàng và hiệu quả. Các thành phần chính của Android SDK bao gồm:

Hình 1. 2 Android SDK

- Thư viện và API: Hỗ trợ lập trình giao diện, kết nối mạng, xử lý dữ liệu.
- Trình biên dịch và gỡ lỗi: Chuyển mã nguồn thành APK và sửa lỗi.
- Trình giả lập: Mô phỏng thiết bị Android để kiểm tra ứng dụng.
- Hệ thống Gradle: Tự động hóa quá trình xây dựng và quản lý phụ thuộc.

1.3.3. Các công cụ hỗ trợ khác

Bên cạnh Android Studio và Android SDK, lập trình viên còn tận dụng nhiều công cụ hỗ trợ khác nhằm tối ưu hóa quá trình phát triển ứng dụng, cải thiện hiệu suất làm việc và đảm bảo chất lượng sản phẩm, chẳng hạn như:

- Git: Quản lý mã nguồn và làm việc nhóm.
- Firebase: Cung cấp dịch vụ đám mây như cơ sở dữ liệu, thông báo đẩy, và phân tích.
- ProGuard: Bảo vệ mã nguồn và tối ưu kích thước APK.
- Thiết bị vật lý: Kiểm thử ứng dụng trên thiết bị thực để đảm bảo hiệu năng.

1.4. Bảo mật trong phát triển ứng dụng di động

Các biện pháp bảo mật cho ứng dụng là tập hợp các phương pháp, kỹ thuật và công cụ được áp dụng nhằm bảo vệ ứng dụng khỏi các mối đe dọa tiềm tàng, đảm bảo an toàn cho dữ liệu người dùng, ngăn chặn truy cập trái phép và duy trì tính toàn vẹn của hệ thống. Việc tuân thủ các biện pháp bảo mật giúp ứng dụng an toàn, tạo niềm tin cho người dùng và bảo vệ dữ liệu của họ một cách hiệu quả.

- Mã hóa dữ liệu: Sử dụng mã hóa để bảo vệ dữ liệu nhạy cảm của người dùng.
- Quyền truy cập: Chỉ yêu cầu quyền cần thiết để giảm nguy cơ lạm dụng.
- Bảo vệ API: Sử dụng các cơ chế xác thực (API Key, OAuth) để bảo vệ API khỏi truy cập trái phép.
- Kiểm tra mã độc: Kiểm tra và quét ứng dụng để đảm bảo không chứa mã độc trước khi phát hành.
- Bảo vệ giao tiếp: Sử dụng HTTPS để bảo mật dữ liệu truyền tải giữa ứng dụng và máy chủ.

CHƯƠNG II: KHẢO SÁT VÀ PHÂN TÍCH PHẦN MỀM

2.1. Khảo sát yêu cầu

2.1.1 .Mô tả yêu cầu người dùng

Ứng dụng được phát triển nhằm đáp ứng nhu cầu quản lý thông tin học tập của sinh viên EAUT, đặc biệt là việc theo dõi lịch học, điểm thi và ngày thi một cách thuận tiện và hiệu quả. Yêu cầu người dùng chính là các sinh viên có thể truy cập nhanh chóng thông tin học tập cá nhân và nhận thông báo về các sự kiện quan trọng. Ứng dụng cần có giao diện người dùng đơn giản, dễ sử dụng, và đặc biệt phải nhanh chóng trong việc xử lý và hiển thị dữ liệu. Cụ thể, người dùng cần các chức năng sau:

- Quản lý lịch học: Sinh viên có thể tra cứu và xem lịch học theo tuần, tháng, với các thông tin chi tiết về môn học, giảng viên, giờ học, phòng học.
- Theo dõi điểm thi: Người dùng có thể kiểm tra điểm thi cho từng môn học, điểm tổng kết và điểm trung bình học tập để theo dõi tiến trình học tập của mình.
- Thông báo lịch thi: Ứng dụng sẽ gửi thông báo nhắc nhở về lịch thi, giờ thi, và phòng thi. Ngoài ra, có thể cập nhật thay đổi về lịch thi khi có sự điều chỉnh từ nhà trường.

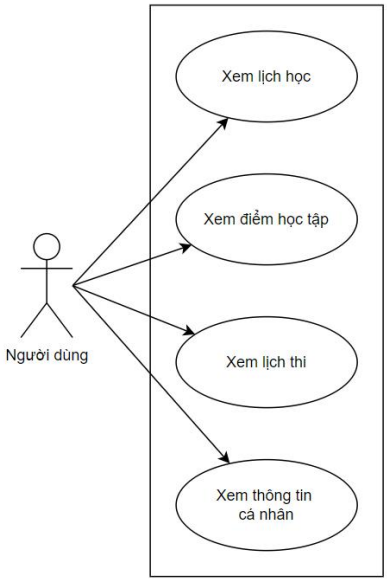
2.1.2. Chức năng chính của ứng dụng

Ứng dụng sẽ được thiết kế với các chức năng cơ bản và tiên tiến, nhằm đáp ứng đầy đủ nhu cầu quản lý học tập của sinh viên, giúp họ có thể theo dõi và tổ chức các công việc học tập một cách dễ dàng và hiệu quả. Những tính năng này không chỉ hỗ trợ việc quản lý lịch học, điểm thi mà còn cung cấp các công cụ hữu ích để sinh viên theo dõi tiến độ học tập và nhận thông báo về các sự kiện quan trọng trong học kỳ.Các chức năng này sẽ được xây dựng với mục tiêu giúp sinh viên dễ dàng quản lý công việc học tập của mình, giảm bớt sự lo lắng về các sự kiện quan trọng trong học kỳ. Sau đây là các tính năng cảu ứng dụng:

- Lịch học: Hiển thị chi tiết lịch học của sinh viên theo tuần, tháng, với đầy đủ thông tin về môn học, giờ học, phòng học, và giảng viên.
- Điểm thi: Cho phép sinh viên theo dõi điểm thi cho từng môn học, giúp cập nhật kết quả thi nhanh chóng và chính xác.
- Ngày thi: Cung cấp thông tin về ngày thi, giờ thi, phòng thi và các thông báo quan trọng liên quan đến kỳ thi.
- Thông báo đẩy: Gửi thông báo về các thay đổi trong lịch học, lịch thi và cập nhật điểm thi mới. Sinh viên sẽ được thông báo ngay lập tức qua các thông báo đẩy của ứng dụng.

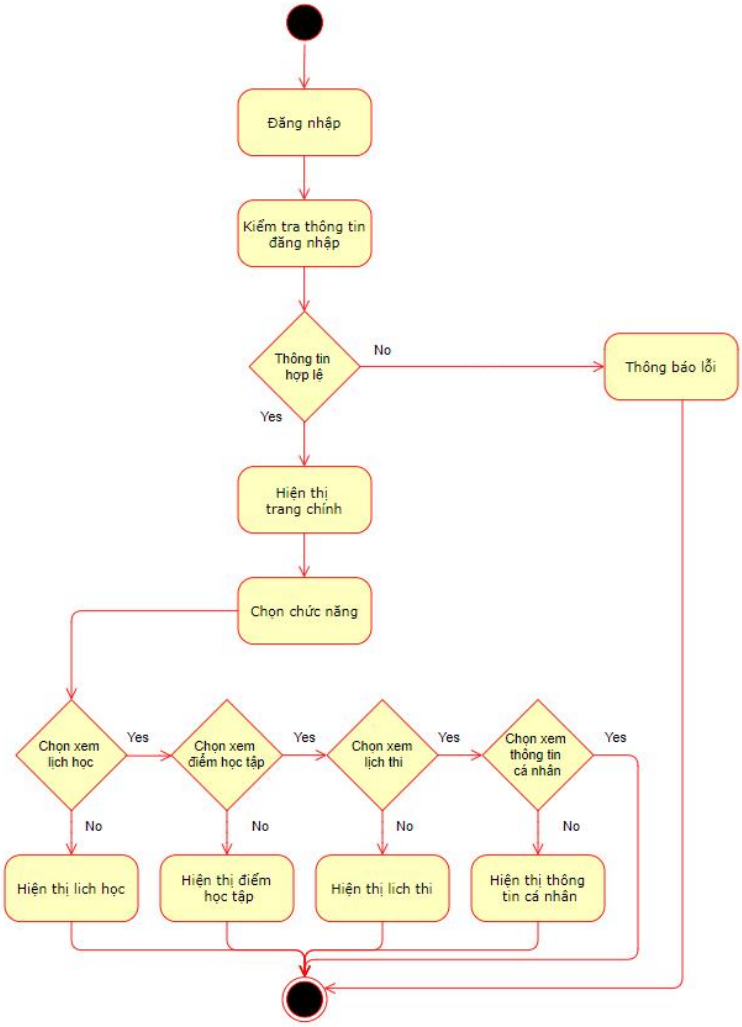
2.2. Phân tích hệ thống

2.2.1. Sơ đồ use case



Hình 2. 1 Sơ đồ use case

2.2.2. Sơ đồ hoạt động (Activity Diagram)



Hình 2. 2 Sơ đồ hoạt động

2.3. Đặc tả chức năng

2.3.1 Chi tiết các chức năng

- Lịch học: Sinh viên có thể xem lịch học theo tuần, tháng, với các chi tiết về môn học, giờ học, giảng viên và phòng học. Các môn học có thể được thêm, sửa hoặc xóa dễ dàng.
- Điểm thi: Sinh viên có thể kiểm tra điểm thi của mình cho từng môn học và theo dõi tiến trình học tập. Hệ thống sẽ tự động tính toán điểm trung bình và đưa ra đánh giá học tập.
- Ngày thi: Ứng dụng cung cấp thông tin về các kỳ thi sắp tới, bao gồm thời gian, địa điểm và phòng thi, giúp sinh viên chuẩn bị tốt cho kỳ thi.
- Thông báo đẩy: Ứng dụng sẽ gửi thông báo về các thay đổi trong lịch học, kết quả thi mới và các thông báo quan trọng từ trường. Sinh viên sẽ nhận thông báo ngay lập tức.

2.3.2 Yêu cầu khác

- Hiệu suất: Ứng dụng phải hoạt động mượt mà, không gặp sự cố khi xử lý các tác vụ phức tạp, và có thể quản lý lượng dữ liệu lớn từ nhiều môn học và điểm thi.
- Bảo mật: Đảm bảo rằng tất cả thông tin cá nhân của người dùng, bao gồm điểm thi và lịch thi, đều được mã hóa và bảo vệ khỏi các truy cập trái phép. Các biện pháp bảo mật như xác thực người dùng và mã hóa dữ liệu sẽ được sử dụng để đảm bảo an toàn.
- Tính thân thiện: Giao diện người dùng cần phải dễ sử dụng và trực quan, với khả năng truy cập nhanh chóng vào các chức năng chính như lịch học, điểm thi và thông báo lịch thi. Ứng dụng sẽ được thiết kế sao cho sinh viên có thể sử dụng mà không gặp khó khăn, dù là người mới sử dụng smartphone hay những người có kinh nghiệm.

2.4. Điểm mạnh và điểm yếu

Các ứng dụng này đều có giao diện người dùng thân thiện, hỗ trợ quản lý lịch học hiệu quả. Một số ứng dụng còn có tính năng giúp sinh viên theo dõi bài tập và nhiệm vụ.

Tuy nhiên, các ứng dụng này chưa hỗ trợ đầy đủ các nhu cầu của sinh viên EAUT như thông báo lịch thi, điểm thi, hoặc đồng bộ với hệ thống dữ liệu của trường học. Điều này khiến sinh viên phải sử dụng nhiều ứng dụng khác nhau để quản lý thông tin học tập.

Ứng dụng của chúng em sẽ kết hợp tất cả các chức năng cần thiết, từ quản lý lịch học, điểm thi, đến thông báo lịch thi, giúp sinh viên tiết kiệm thời gian và công sức.

CHƯƠNG III: THIẾT KẾ PHẦN MỀM

3.1. Thiết kế kiến trúc ứng dụng

Mô hình MVVM giúp tổ chức và quản lý mã nguồn hiệu quả bằng cách tách biệt các thành phần giao diện, logic nghiệp vụ, và dữ liệu. Với MVVM, View và Model được kết nối thông qua ViewModel, giúp giảm sự phụ thuộc giữa các thành phần, từ đó cải thiện khả năng bảo trì và mở rộng ứng dụng.

(Model-View-ViewModel) là một mô hình kiến trúc được sử dụng rộng rãi trong phát triển ứng dụng Android hiện đại. Mô hình này phân tách ứng dụng thành bốn thành phần chính:

Bảng 3. 1 Bảng cấu trúc Model

Tên class	**Chức năng**
ExamScheduleModel	Lưu trữ thông tin lịch thi
GradeModel	Lưu trữ thông tin điểm học tập
ScheduleModel	Lưu trữ thông tin lịch học
UserDatabase	Quản lý thông tin người dùng

Bảng 3. 2 Bảng cấu trúc DAO

Tên class	**Chức năng**
BaseActivity	Quản lý giao diện chính và điều hướng giữa các fragment.
ExamScheduleActivity	Hiện thị lịch thi
GradesActivity	Hiện thị điểm học tập
ProfileActivity	Hiện thị thông tin cá nhân
ScheduleActivity	Hiện thị lịch học

Bảng 3. 3 Bảng cấu trúc Adapter

Tên class	**Chức năng**
ExamScheduleAdapter	Hiển thị thông tin lịch thi trong danh sách
GradeAdapter	Hiển thị thông tin điểm học tập trong danh sách
ScheduleAdapter	Hiển thị thông tin lịch học trong danh sách

- Chức năng của DatabaseHelper là quản lý cơ sở dữ liệu SQLite cho ứng dụng. Lớp này thực hiện các nhiệm vụ như: Quản lý cơ sở dữ liệu SQLite, bao gồm việc tạo bảng và nâng cấp cơ sở dữ liệu khi cần thiết. Bên cạnh đó cũng có vai trò: Đảm bảo việc lưu trữ và truy xuất dữ liệu người dùng một cách hiệu quả, bao gồm các thông tin quan trọng như mã sinh viên và mật khẩu.
- Chức năng của ApiHandler là quản lý các yêu cầu và phản hồi từ API. Lớp này thực hiện các chức năng chính sau:

- Kiểm tra đăng nhập với API: Gửi yêu cầu đăng nhập tới API và nhận phản hồi về trạng thái đăng nhập của người dùng.
- Lấy thông tin người dùng: Gửi yêu cầu lấy thông tin người dùng từ API dựa trên mã sinh viên.
- Lấy lịch thi và điểm thi: Gửi yêu cầu lấy lịch thi và điểm thi từ API.
- Gửi các yêu cầu GET tới API: Xử lý các yêu cầu và phản hồi từ API, sử dụng AsyncTask để thực thi không đồng bộ.

Vai trò của ApiHandler:

- Quản lý giao tiếp giữa ứng dụng và server.
- Cung cấp các phương thức để lấy dữ liệu từ server (như thông tin người dùng, lịch thi, điểm thi) và truyền kết quả cho các callback.
- Xử lý lỗi và thông báo cho người dùng nếu có sự cố trong việc kết nối với API.

- HomeActivity là một màn hình chính trong ứng dụng MyStudyEAUT. Đây là nơi người dùng sẽ bắt đầu tương tác với các chức năng chính của ứng dụng. Màn hình này cung cấp thông tin cơ bản về người dùng và các tùy chọn để điều hướng đến các phần khác của ứng dụng như lịch học, điểm thi, lịch thi, và thông tin cá nhân. Chức năng chính:

- Hiển thị giao diện chính của ứng dụng: Giao diện này cung cấp các nút điều hướng đến các tính năng khác của ứng dụng như lịch học, điểm thi, lịch thi và thông tin cá nhân.
- Lấy thông tin người dùng: Lấy tên người dùng từ cơ sở dữ liệu local và cập nhật giao diện bằng cách gọi API để lấy thông tin người dùng.
- Chuyển đổi giữa các màn hình (fragments): Sử dụng BaseActivity để thay thế các fragment khi người dùng chọn một mục trong menu.

Vai trò của HomeActivity:

- Quản lý giao diện chính của ứng dụng: Là màn hình chính mà người dùng tương tác đầu tiên.
- Xử lý việc lấy thông tin người dùng và cập nhật giao diện: Hiển thị tên người dùng hoặc "Guest" nếu không tìm thấy tên.
- Điều hướng người dùng đến các chức năng khác: Chuyển đến các màn hình lịch học, điểm thi, lịch thi và thông tin cá nhân khi người dùng nhấn các nút tương ứng.

- LoginActivity là màn hình đăng nhập trong ứng dụng MyStudyEAUT. Đây là nơi người dùng nhập thông tin tài khoản (mã sinh viên và mật khẩu) để xác thực và truy cập vào ứng dụng. Chức năng của LoginActivity:

- Xác thực thông tin đăng nhập (mã sinh viên và mật khẩu) của người dùng.
- Kiểm tra thông tin đăng nhập từ cơ sở dữ liệu cục bộ, nếu không có, gửi yêu cầu đến API để xác thực.
- Đăng nhập thành công sẽ chuyển đến màn hình chính của ứng dụng (BaseActivity).

Vai trò của LoginActivity:

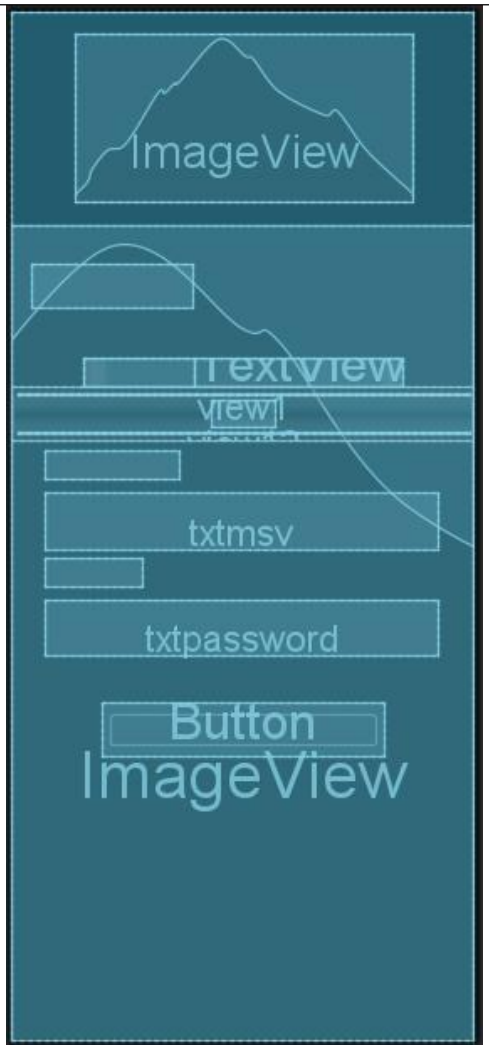
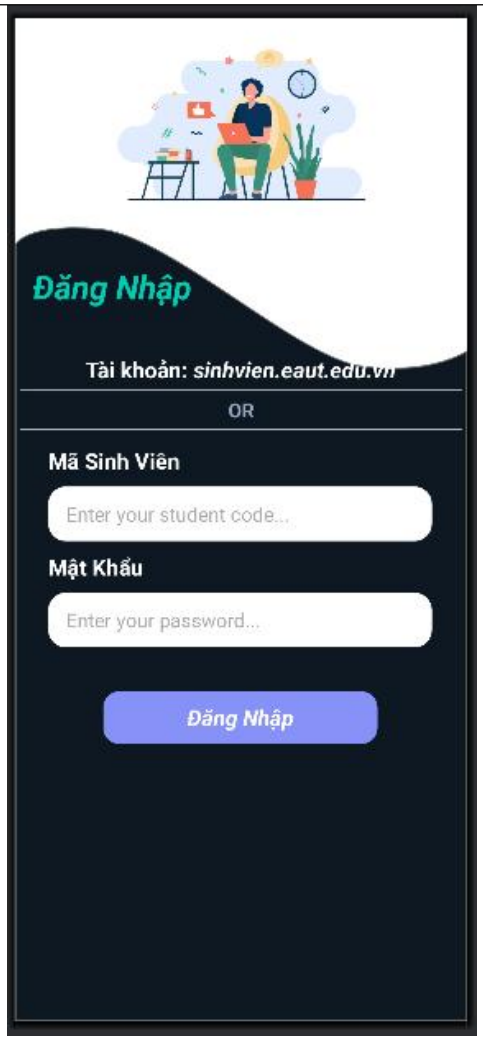
- Là cửa ngõ đầu tiên để người dùng truy cập vào ứng dụng.
- Đảm bảo chỉ những người dùng có thông tin hợp lệ mới có thể sử dụng ứng dụng.

3.2. Thiết kế giao diện người dùng

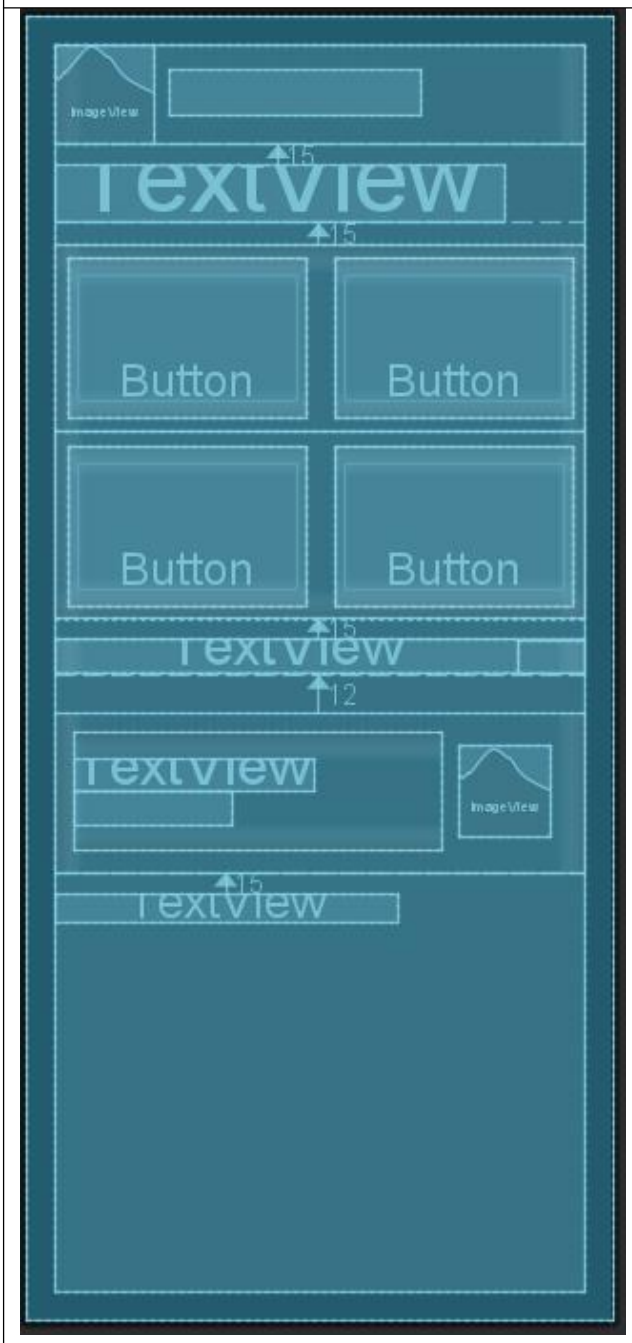
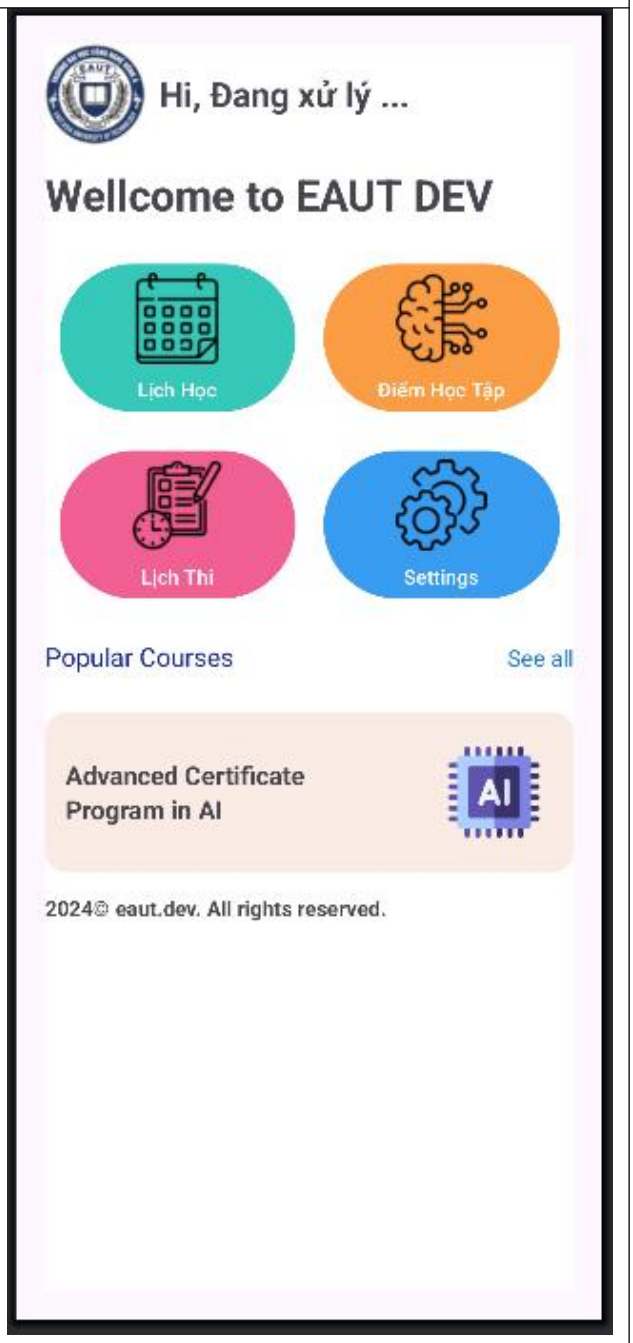
3.2.1. Wireframes và mockups

Trong ứng dụng này, chúng ta có tổng cộng 7 wireframes và Mockups chính, mỗi cái đại diện cho một phần chức năng quan trọng trong hệ thống. Các wireframes ban đầu được thiết kế để thể hiện cấu trúc cơ bản và luồng người dùng, trong khi các Mockups là những giao diện chi tiết hơn, thể hiện cách thức người dùng sẽ tương tác với ứng dụng. Các wireframes và Mockups này bao gồm: màn hình đăng nhập (login), màn hình chính (home), cơ sở dữ liệu (base), lịch thi (examSchedule), bảng điểm (grades), lịch học (schedule), và hồ sơ người dùng (profile). Mỗi wireframe và Mockup đều đóng vai trò quan trọng trong việc giúp người dùng dễ dàng tương tác và sử dụng ứng dụng. Sau đây là chi tiết các giao diện wireframes và Mockups được sử dụng.

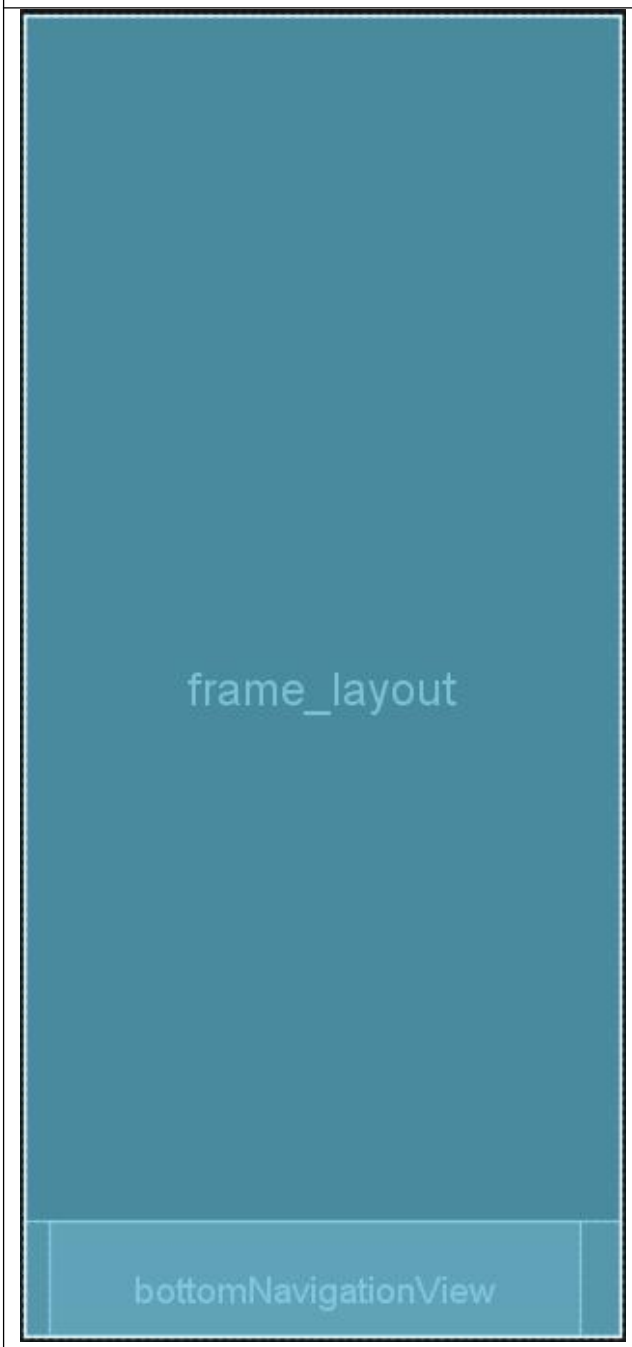
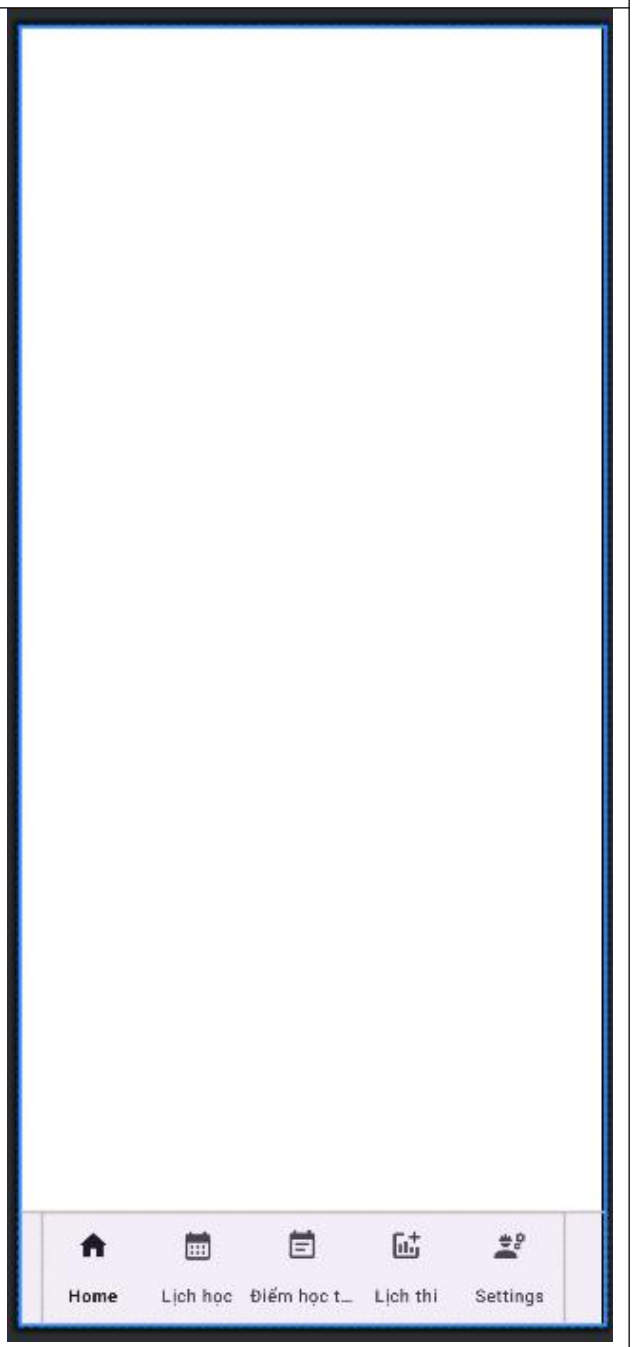
➢ Login

Wireframes	Mockups
	
Hình 3. 1 Wireframes login	*Hình 3. 2 Mockups login*

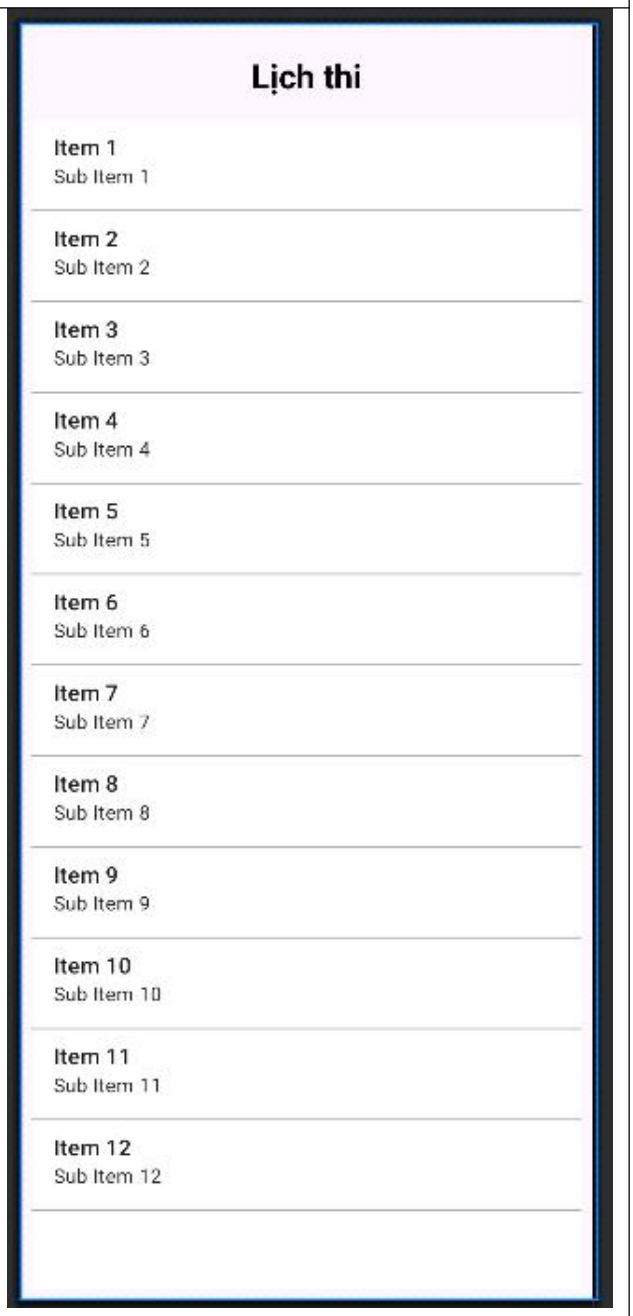
➢ Home

Wireframes	Mockups
	
Hình 3. 3 Wireframes home	*Hình 3. 4 Mockups home*

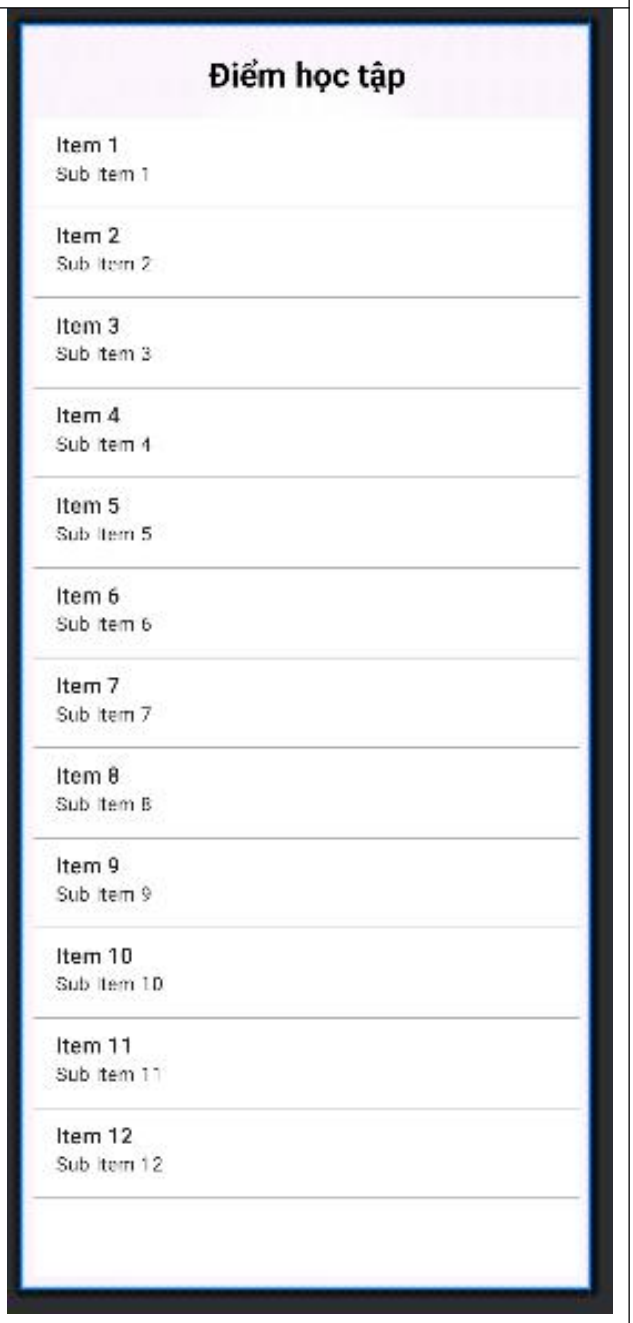
- Base

Wireframes	Mockups
	
Hình 3. 5 Wireframes base	*Hình 3. 6 Mockups base*

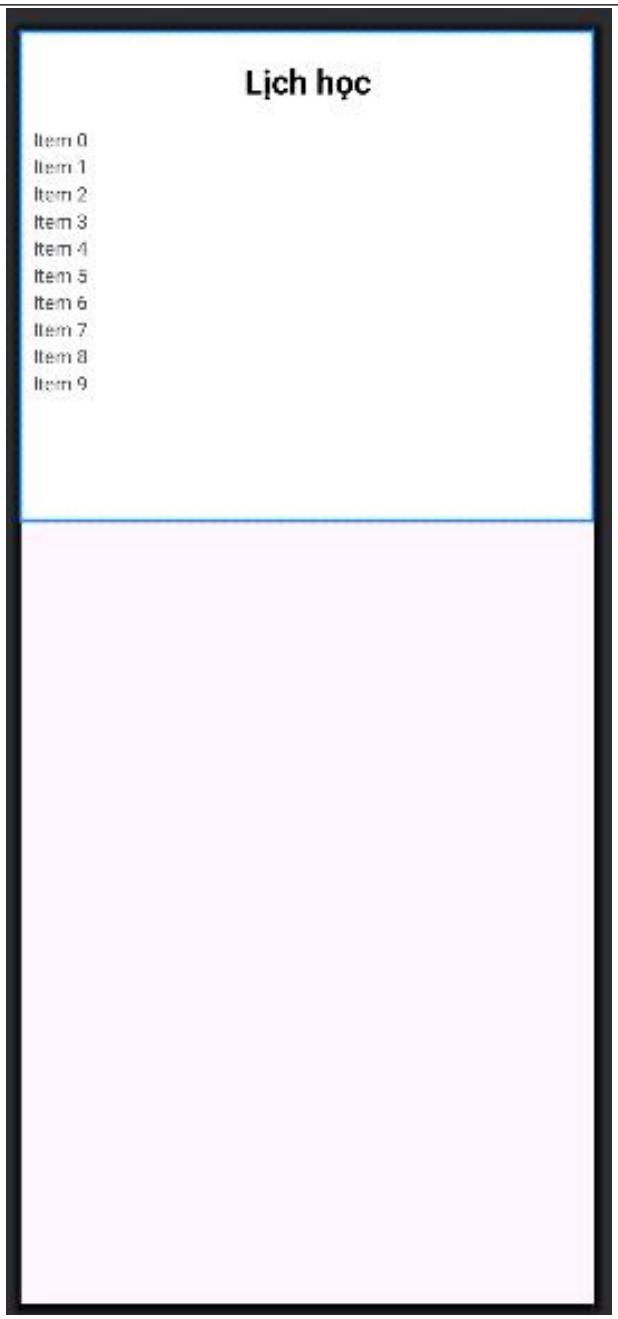
- ExamSchedule

Wireframes	Mockups
	
Hình 3. 7 Wireframes examSchedule	*Hình 3. 8 Mockups examSchedule*

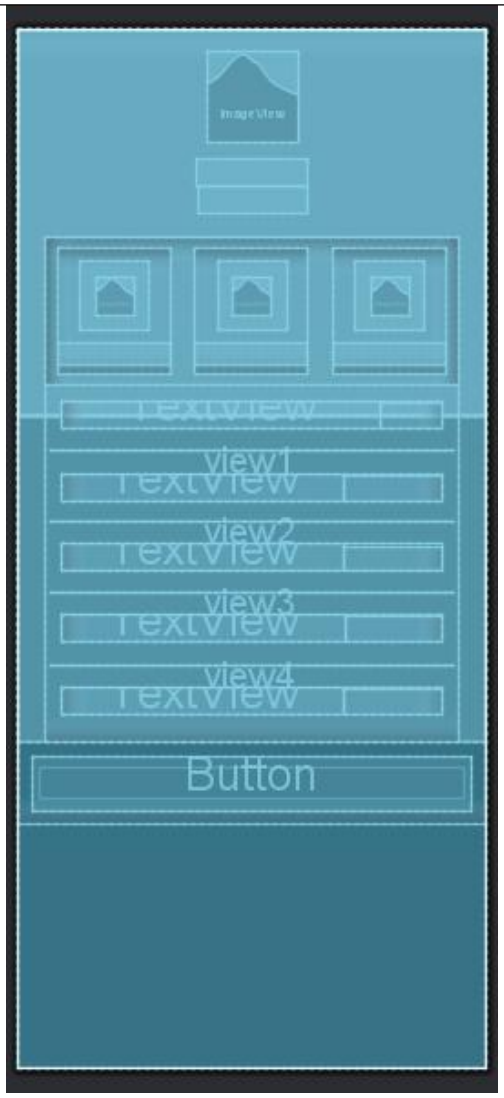
➢ Grades

Wireframes	Mockups
	
Hình 3. 9 Wireframes grades	Hình 3. 10 Mockups grades

➢ Schedule

Wireframes	Mockups
	
Hình 3. 11 Wireframes schedule	Hình 3. 12 Mockups schedule

➢ Profile

Wireframes	Mockups
	
Hình 3. 13 Wireframes profile	Hình 3. 14 Mockups profile

3.2.2. Điều hướng giữa các màn hình

Điều hướng (Navigation) giữa các màn hình là rất quan trọng trong việc đảm bảo trải nghiệm người dùng mượt mà và trực quan. Có thể sử dụng Navigation Component của Android để thực hiện điều này, giúp điều hướng giữa các màn hình trở nên dễ dàng và dễ bảo trì.

Home Activity: Là màn hình chính, từ đó người dùng sẽ điều hướng đến các màn hình khác.

Bottom Navigation: Đây là hai lựa chọn phổ biến cho điều hướng. Bottom Navigation sẽ là lựa chọn phù hợp hơn trong trường hợp này, vì nó giúp người dùng dễ dàng truy cập vào các phần chính của ứng dụng.

Điều hướng đơn giản giữa các màn hình:

HomeActivity → ScheduleActivity(Lịch học)

HomeActivity → GradesActivity(Điểm thi)

HomeActivity → ExamScheduleActivity(Ngày thi)

HomeActivity → ProfileActivity(Thông tin cá nhân)

Ví dụ mã Java để điều hướng từ MainActivity đến ScheduleActivity và đến các Activity khác:

```
if (item.getItemId() == R.id.home) {
    replaceFragment(new HomeActivity());
} else if (item.getItemId() == R.id.lichhoc) {
    replaceFragment(new ScheduleActivity());
} else if (item.getItemId() == R.id.diemhoctap) {
    replaceFragment(new GradesActivity());
} else if (item.getItemId() == R.id.lichthi) {
    replaceFragment(new ExamScheduleActivity());
} else if (item.getItemId() == R.id.settings) {
    replaceFragment(new ProfileActivity());
}
```

3.2.3. Màu sắc và phong cách thiết kế (UI/UX)

Về màu sắc, chúng em đã lựa chọn một bảng màu tươi sáng và hài hòa để mang lại cảm giác dễ chịu và chuyên nghiệp cho người dùng. Các màu sắc được sử dụng trong ứng dụng không chỉ giúp tạo điểm nhấn mà còn hỗ trợ tăng cường khả năng tương tác, giúp người dùng dễ dàng nhận diện các thành phần quan trọng. Bảng màu bao gồm các gam màu tươi sáng như xanh ngọc, cam và hồng, kết hợp với các sắc thái trung tính để cân bằng và tạo sự thống nhất trong toàn bộ giao diện. Ví dụ 1 trong những màu tiêu biểu được liệt kê ra sau:

Nền: Màu trắng (#FFFFFF)

Nút chính: Màu xanh dương (#1B83FE)

Nút phụ: Màu xanh lục đậm (#018786)

Nút bấm 1 trong home: Màu xanh ngọc (#36C8B8)

Nút bấm 2 trong home: Màu cam sáng (#FB9D45)

Nút bấm 3 trong home: Màu hồng tươi (#F25F92)

Nút bấm 4 trong home: Màu xanh dương tươi sáng(#389DF1)

Về phong cách thiết kế, chúng em hướng đến một giao diện tối giản, dễ sử dụng, nhưng vẫn đầy đủ tính năng và hiện đại. Các yếu tố thiết kế được tối ưu hóa để mang lại trải nghiệm người dùng mượt mà và trực quan, với các đường nét rõ ràng, bố cục hợp lý và không gian trắng để tạo sự thoải mái cho mắt người dùng. Màu sắc, hình ảnh và biểu tượng được lựa chọn kỹ lưỡng, phù hợp với mục đích của từng chức năng, đồng thời tạo nên một không gian trực quan, dễ hiểu và dễ tiếp cận.

Material Design: Android cung cấp Material Design là một phong cách thiết kế giao diện hiện đại với các yếu tố như bóng đổ, chuyển động mượt mà và các hiệu ứng ấn tượng. Bạn nên sử dụng các thành phần Material Design như CardView, RecyclerView, FloatingActionButton, v.v.

Font chữ: Sử dụng các font chữ dễ đọc và hiện đại. Một số font chữ phổ biến trên Android là Roboto, Noto, hoặc Google Fonts.

Responsive UI: Đảm bảo giao diện của bạn có thể hiển thị tốt trên nhiều kích thước màn hình khác nhau, từ điện thoại nhỏ đến máy tính bảng.

3.3. Thiết kế cơ sở dữ liệu

Ứng dụng sử dụng một cơ sở dữ liệu đơn giản với một bảng duy nhất, users, để lưu trữ thông tin người dùng. Bảng này bao gồm hai cột chính: mã sinh viên (student_code) và mật khẩu (password), dùng để xác thực người dùng khi đăng nhập vào hệ thống. Với thiết kế này, ứng dụng đảm bảo việc quản lý thông tin đăng nhập hiệu quả và dễ dàng.

Bảng 3. 4 Bảng User

Tên cột	**Kiểu dữ liệu**	**Mô tả**
id	INTEGER (PK)	Khóa chính, tự động tăng (Auto Increment)
student_code	TEXT	Mã sinh viên (dùng để đăng nhập)
password	TEXT	Mật khẩu của người dùng

3.4. Thiết kế API

Trong phần thiết kế API, ứng dụng sử dụng các API RESTful để giao tiếp với server, thực hiện các thao tác như đăng nhập, lấy thông tin người dùng, lịch học và lịch thi. Cụ thể, ứng dụng gửi yêu cầu HTTP GET đến các endpoint của server, truyền tham số cần thiết như mã sinh viên và mật khẩu. Sau khi nhận phản hồi từ server, ứng dụng xử lý dữ liệu JSON nhận được và cập nhật giao diện người dùng tương ứng.

Các API chính được sử dụng bao gồm:

3.4.1. API Đăng nhập

Để thực hiện đăng nhập, ứng dụng sử dụng một API RESTful với phương thức GET. URL của API được cấu hình động, trong đó mã sinh viên và mật khẩu sẽ được mã hóa trước khi gửi đến server để đảm bảo an toàn. Đoạn mã API đăng nhập:

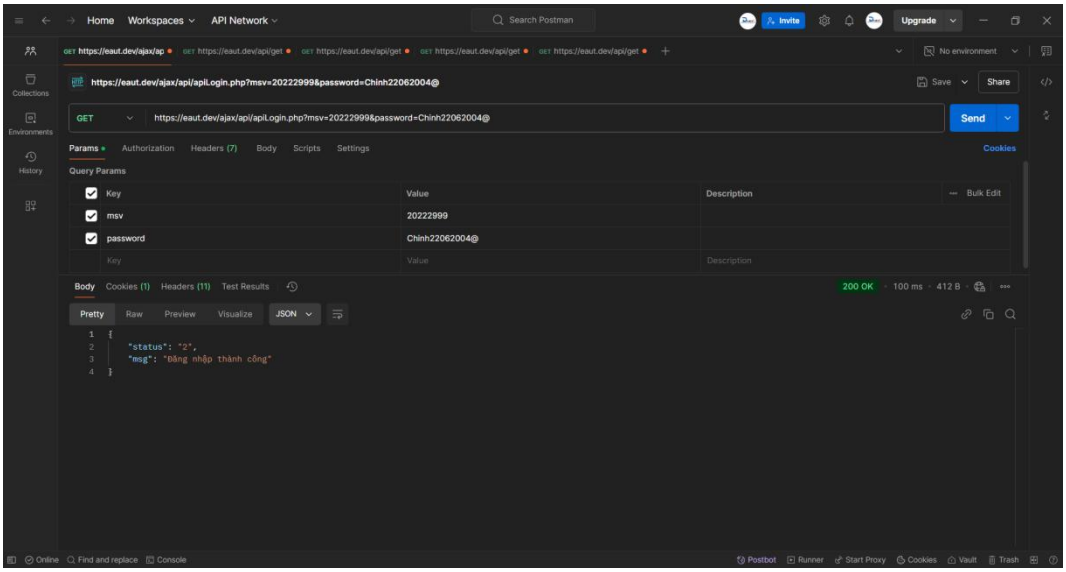
```
String apiUrl = "https://eaut.dev/ajax/api/apiLogin.php?msv="
+ URLEncoder.encode(studentCode, "UTF-8") + "&password="
                    + URLEncoder.encode(password, "UTF-8");
```

Giải thích:

- studentCode: Mã sinh viên (msv) nhập vào bởi người dùng.
- password: Mật khẩu người dùng đã nhập.

- URLEncoder.encode(): Được sử dụng để mã hóa các tham số trong URL, đảm bảo rằng các ký tự đặc biệt (như dấu cách, dấu &, ...) sẽ được chuyển đổi thành dạng mã hóa URL hợp lệ.
- URL sau khi được mã hóa sẽ được gửi đến API trên server với các tham số msv (mã sinh viên) và password (mật khẩu).

Ví dụ 1: Trong ví dụ này, chúng ta tạo một URL cho API đăng nhập, trong đó mã sinh viên và mật khẩu được mã hóa bằng URLEncoder.encode() để đảm bảo chúng an toàn khi được gửi qua mạng.



Hình 3. 15 JSON trả về đăng nhập

3.4.2. API lấy thông tin người dùng

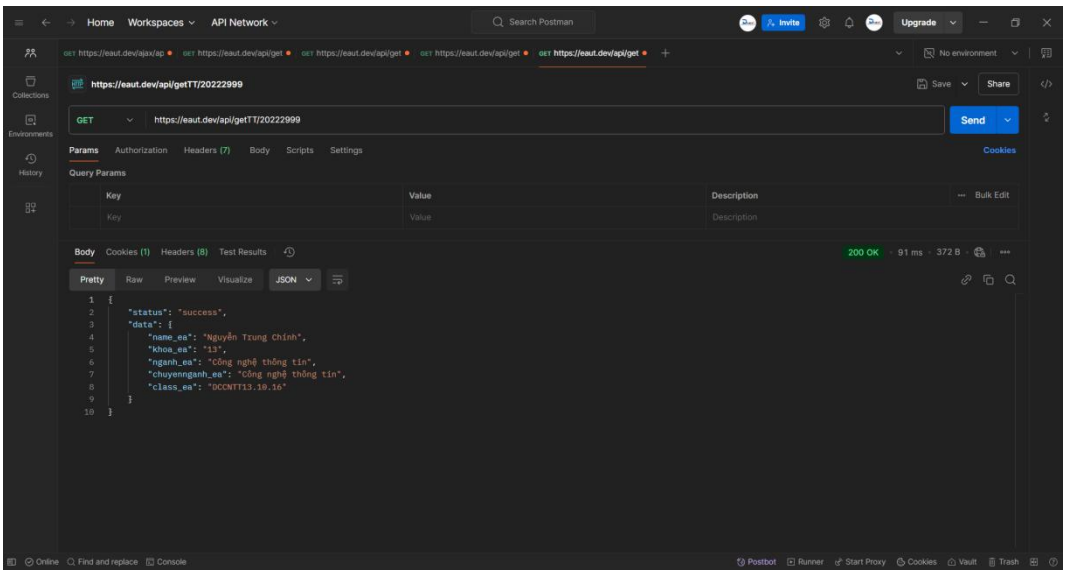
Để lấy thông tin chi tiết của người dùng, ứng dụng sử dụng API GET từ server. Đường dẫn API này yêu cầu mã sinh viên (username) và trả về thông tin cá nhân của người dùng đó.Đoạn mã API lấy thông tin người dùng:

```
String apiUrl = "https://eaut.dev/api/getTT/" + URLEncoder.encode(username, "UTF-8");
```

Giải thích:

- username: Mã sinh viên, được lấy từ cơ sở dữ liệu của ứng dụng sau khi người dùng đăng nhập thành công.
- URLEncoder.encode(username, "UTF-8"): Mã hóa username để đảm bảo rằng nó được truyền tải an toàn qua URL (đặc biệt là khi có ký tự đặc biệt hoặc khoảng trắng).
- URL này sẽ được sử dụng để gửi yêu cầu đến server, nơi dữ liệu thông tin người dùng sẽ được trả về dưới dạng JSON.
- Server sẽ xử lý yêu cầu và trả về thông tin của sinh viên tương ứng với username.

Ví dụ 2: Trong ví dụ này, chúng ta tạo một URL để truy cập API lấy thông tin, trong đó tên người dùng (username) được mã hóa bằng URLEncoder.encode() để đảm bảo không có ký tự đặc biệt gây lỗi khi gửi yêu cầu đến server.



Hình 3. 16 JSON trả về thông tin người dùng

3.4.3. API lấy lịch học

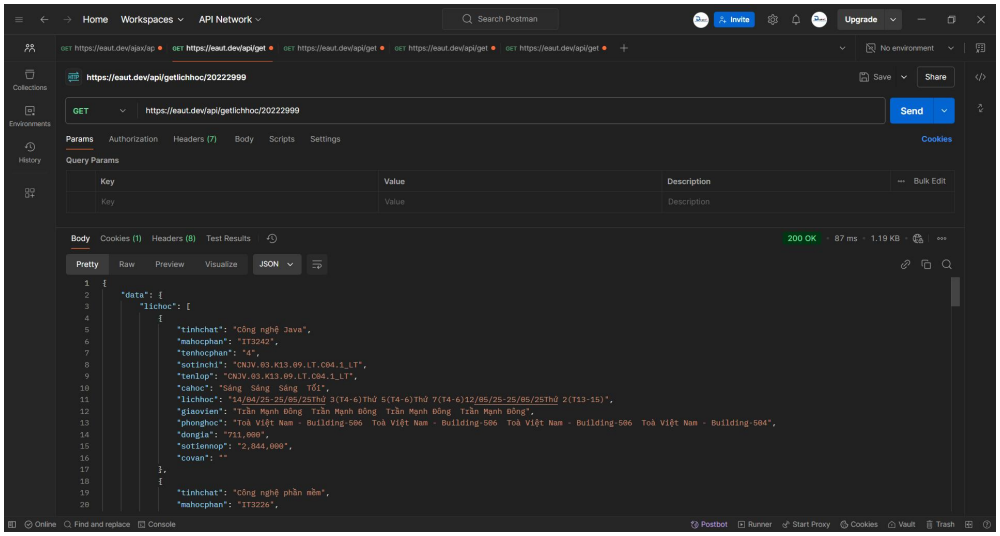
Để lấy thông tin lịch học của sinh viên, ứng dụng sử dụng API GET từ server. Đường dẫn API này yêu cầu mã sinh viên (username) và trả về thông tin lịch học của sinh viên đó. Đoạn mã API lấy lịch học:

```
String apiUrl = "https://eaut.dev/api/getlichhoc/" + username;
```

Giải thích:

- username: Mã sinh viên, được lấy từ cơ sở dữ liệu sau khi người dùng đăng nhập.
- URL này sẽ gửi yêu cầu đến server với username của sinh viên, server sẽ xử lý và trả về kết quả lịch học dưới dạng JSON.
- Dữ liệu lịch học sẽ được sử dụng để hiển thị lên giao diện người dùng, giúp sinh viên theo dõi thời khóa biểu của mình.

Ví dụ 3: Để lấy lịch học của người dùng, chúng ta sử dụng một URL API đơn giản, trong đó tên người dùng (username) được thêm trực tiếp vào đường dẫn API. Điều này cho phép hệ thống truy vấn thông tin lịch học của sinh viên một cách nhanh chóng.



Hình 3. 17 JSON trả về lịch học

3.4.3. API lấy lịch thi

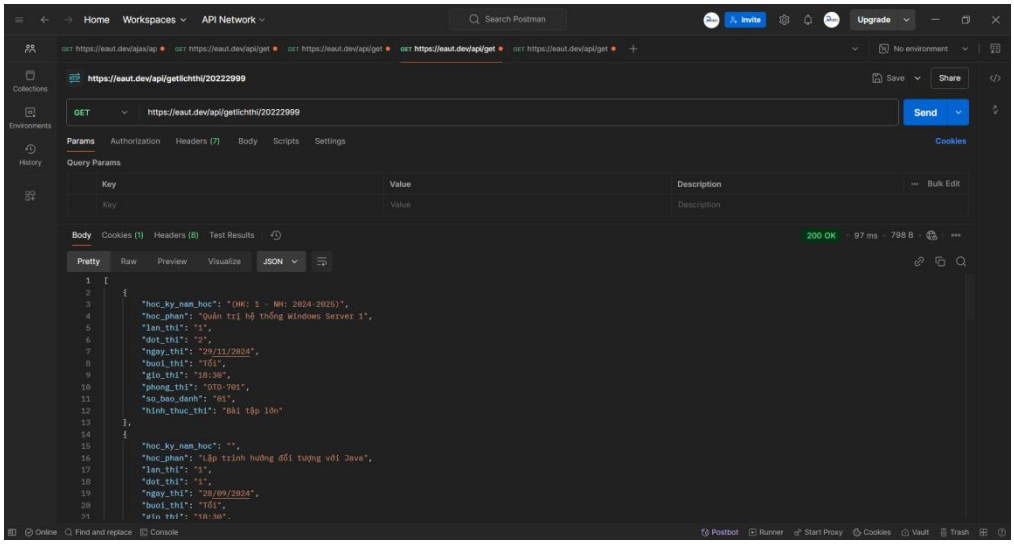
Để lấy lịch thi của sinh viên, ứng dụng sử dụng API GET từ server. Đường dẫn API này yêu cầu mã sinh viên (username) để trả về thông tin lịch thi cụ thể của người đó. Đoạn mã API lịch thi:

```
String apiUrl = "https://eaut.dev/api/getlichthi/" + username;
```

Giải thích:

- username: Mã sinh viên, được lấy từ cơ sở dữ liệu của ứng dụng sau khi người dùng đăng nhập thành công.
- URL này sẽ được sử dụng để gửi yêu cầu đến server, nơi dữ liệu lịch thi sẽ được trả về dưới dạng JSON.
- Server sẽ xử lý yêu cầu và trả về thông tin lịch thi cho sinh viên tương ứng với username.

Ví dụ 4: API URL được tạo ra bằng cách ghép tên người dùng vào phần cuối của URL. Khi đó, hệ thống có thể truy vấn lịch thi của sinh viên dựa trên tên người dùng đã nhập.



Hình 3. 18 JSON trả về lịch thi

3.4.3. API lấy điểm học tập

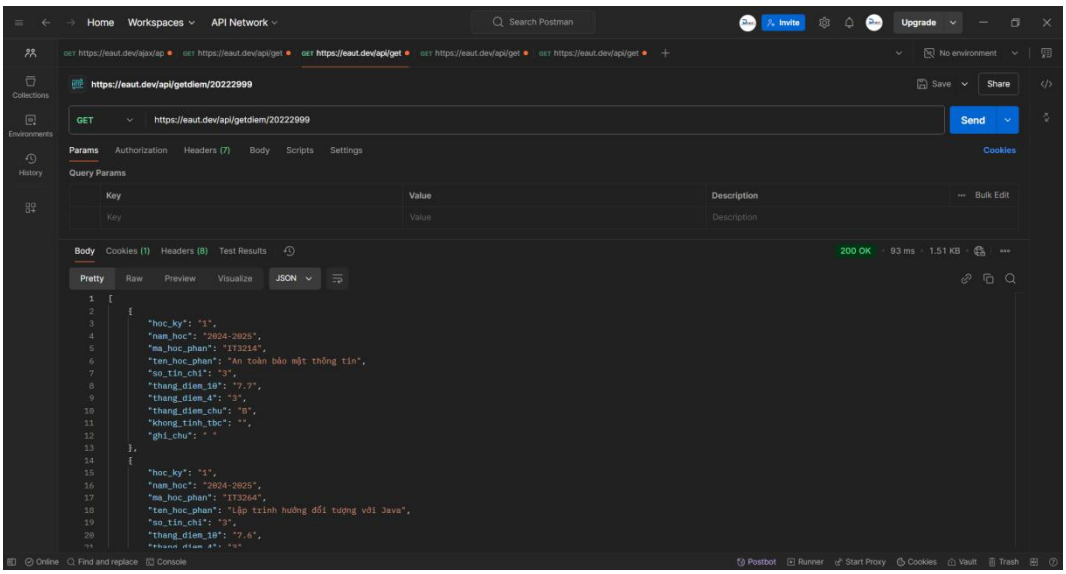
Để lấy thông tin điểm học tập của người dùng, ứng dụng sử dụng API GET từ server. Đường dẫn API này yêu cầu mã sinh viên (username) và trả về thông tin điểm của sinh viên đó. Đoạn mã API lấy điểm học tập:

```
String apiUrl = "https://eaut.dev/api/getdiem/" + username;
```

Giải thích:

- username: Mã sinh viên, được lấy từ cơ sở dữ liệu sau khi người dùng đăng nhập.
- URL này sẽ gửi yêu cầu đến server với username của sinh viên, server sẽ xử lý và trả về kết quả điểm học tập dưới dạng JSON.
- Dữ liệu điểm sẽ được sử dụng để hiển thị lên giao diện người dùng, giúp sinh viên kiểm tra điểm của mình.

Ví dụ 5: Đoạn mã này tạo ra một URL để truy vấn thông tin điểm học tập của sinh viên, trong đó tên người dùng (username) được thêm vào cuối URL, giúp lấy được dữ liệu điểm của sinh viên một cách nhanh chóng và chính xác.



Hình 3. 19 JSON trả về điểm học tập

CHƯƠNG IV: CÀI ĐẶT VÀ KIỂM THỬ

4.1. Cài đặt ứng dụng

4.1.1. Quá Trình Cài Đặt

Để phát triển ứng dụng Android, bạn cần cài đặt Android Studio. Đây là công cụ chính thức của Google để phát triển ứng dụng Android. Các bước cài đặt như sau:

Bước 1: Cài đặt Android Studio

- Tải xuống Android Studio: Truy cập trang web chính thức của Android Studio tại android.dev và tải phiên bản phù hợp với hệ điều hành của bạn (Windows, macOS, hoặc Linux).
- Cài đặt Android Studio:

+ Mở tệp cài đặt và làm theo hướng dẫn trên màn hình.

+ Trong quá trình cài đặt, Android Studio sẽ yêu cầu bạn cài đặt một số phần mềm hỗ trợ như Android SDK, Android Virtual Device (AVD) cho việc giả lập Android, và Java Development Kit (JDK).

- Cấu hình Android Studio: Sau khi cài đặt thành công, mở Android Studio và thực hiện cấu hình ban đầu. Android Studio sẽ tự động cài đặt các công cụ cần thiết (SDK, NDK, v.v.) nếu chúng chưa có.

Tạo một Project mới: Khi đã cài đặt xong Android Studio, bạn có thể tạo một Project mới cho ứng dụng Android của mình bằng cách chọn Start a new Android Studio project.

Bước 2: Cài đặt trên Thiết Bị Android giả lập: Bạn có thể sử dụng giả lập (Android Emulator). Android Studio cung cấp AVD Manager để tạo và chạy một giả lập Android trên máy tính.

Bước 3: Cài Đặt Ứng Dụng Lên Thiết Bị

- Biên dịch và chạy ứng dụng: Trong Android Studio, nhấn nút Run (hình tam giác xanh) để biên dịch và triển khai ứng dụng lên thiết bị di động hoặc giả lập.
- Kiểm tra kết quả: Sau khi quá trình biên dịch hoàn tất, ứng dụng sẽ tự động chạy trên thiết bị. Bạn có thể kiểm tra ứng dụng của mình trên giao diện của điện thoại hoặc giả lập.

Bước 4: Cài đặt APK trên thiết bị Android

Ngoài việc sử dụng Android Studio, bạn cũng có thể tạo APK và cài trực tiếp lên thiết bị Android:

- Tạo APK: Chọn Build > Build APK trong Android Studio.
- Chuyển APK sang thiết bị: Sau khi APK đã được tạo, bạn có thể sao chép nó vào thiết bị Android hoặc gửi qua email, USB, hoặc bất kỳ phương tiện nào khác.
- Cài đặt APK: Trên điện thoại, tìm đến tệp APK và nhấn để cài đặt. Nếu bạn chưa bật tính năng cài đặt từ nguồn không xác định, vào Settings > Security, bật Install unknown apps cho trình duyệt hoặc ứng dụng mà bạn sử dụng.

4.1.2. Công Nghệ và Thư Viện Đã Sử Dụng

Ngôn Ngữ Lập Trình: Java là ngôn ngữ chính được sử dụng để phát triển ứng dụng Android trong bài tập này. Android Studio hỗ trợ Java, và là một ngôn ngữ phổ biến cho các ứng dụng di động, cung cấp tính linh hoạt cao và khả năng xử lý mạnh mẽ.

Android SDK (Software Development Kit): SDK là bộ công cụ phát triển phần mềm chính thức của Android, bao gồm các thư viện, công cụ và tài liệu cần thiết để phát triển ứng dụng Android. Trong SDK có các thư viện UI, API để kết nối với cơ sở dữ liệu, xử lý các tác vụ đa nhiệm, và nhiều tính năng khác.

Android Jetpack

Jetpack là một bộ thư viện hỗ trợ phát triển ứng dụng Android nhanh chóng và dễ dàng. Nó bao gồm các thư viện như:

- Room: Dành cho lưu trữ dữ liệu trong cơ sở dữ liệu SQLite.
- LiveData và ViewModel: Dành cho việc quản lý trạng thái và dữ liệu ứng dụng.
- Navigation: Hỗ trợ di chuyển giữa các màn hình trong ứng dụng.

RecyclerView là một thư viện trong Android giúp hiển thị danh sách dữ liệu động với hiệu suất cao. Được sử dụng trong các màn hình như Lịch học, Điểm thi, và Ngày thi để hiển thị các môn học, điểm thi và ngày thi.

Material Design là bộ nguyên tắc thiết kế của Google, được sử dụng để tạo giao diện người dùng đẹp mắt và dễ sử dụng. Trong Android Studio, bạn có thể sử dụng các thành phần giao diện người dùng Material như Buttons, TextFields, CardView để tạo ra giao diện sinh động và dễ tương tác.

SQLite (hoặc Room Database)

SQLite hoặc Room Database (tầng trừu tượng của SQLite) là các giải pháp lưu trữ dữ liệu trong ứng dụng Android. Bạn có thể sử dụng các công nghệ này để lưu trữ thông tin về môn học, điểm thi, và ngày thi.

Glide là thư viện mã nguồn mở giúp tải và hiển thị hình ảnh từ URL hoặc tài nguyên nội bộ trong ứng dụng Android một cách dễ dàng và hiệu quả. Nếu ứng dụng có hình ảnh (ví dụ, hình ảnh môn học), bạn có thể sử dụng Glide.

Retrofit là thư viện dành cho việc kết nối với các API RESTful và dễ dàng xử lý dữ liệu từ các dịch vụ web (nếu cần thiết). Nếu bạn muốn mở rộng ứng dụng với việc lấy dữ liệu từ một API server (ví dụ: lấy thông tin lịch học từ hệ thống trường học trực tuyến), Retrofit sẽ giúp bạn dễ dàng giao tiếp với backend.

Đoạn mã bạn của bạn cung cấp là một phần của cấu hình dependencies trong tệp build.gradle của ứng dụng Android. Đây là danh sách các thư viện mà dự án của bạn phụ thuộc vào, giúp ứng dụng có thể sử dụng các tính năng và chức năng từ các thư viện bên ngoài.

a, Thư viện Android Cơ Bản

implementation libs.appcompat: Cung cấp hỗ trợ cho giao diện người dùng hiện đại (Material Design).

implementation libs.material: Cung cấp các thành phần giao diện người dùng dựa trên Material Design.

implementation libs.activity: Thư viện hỗ trợ cho việc quản lý hoạt động và vòng đời của các Activity.

implementation libs.constraintlayout: Thư viện cho phép bạn sử dụng ConstraintLayout - một layout mạnh mẽ giúp bạn thiết kế giao diện linh hoạt và tối ưu.

b, Thư viện Kiểm Thử

testImplementation libs.junit: Thư viện JUnit giúp bạn viết và thực thi các bài kiểm tra đơn vị.

androidTestImplementation libs.ext.junit: Thư viện mở rộng JUnit để thực hiện kiểm thử UI trên Android.

androidTestImplementation libs.espresso.core: Thư viện Espresso để kiểm thử giao diện người dùng trong ứng dụng Android.

c, Thư viện Mạng

implementation 'com.squareup.retrofit2:retrofit:2.9.0': Retrofit là một thư viện phổ biến để giao tiếp với API RESTful.

implementation 'com.squareup.retrofit2:converter-gson:2.9.0': Bộ chuyển đổi để Retrofit có thể làm việc với dữ liệu JSON (được chuyển đổi thành các đối tượng Java).

implementation 'com.squareup.okhttp3:okhttp:4.9.0': OkHttp là thư viện mạng dùng để thực hiện các yêu cầu HTTP và tối ưu hóa việc xử lý kết nối.

d, Thư viện AndroidX

implementation 'androidx.appcompat:appcompat:1.5.1': Cung cấp các tính năng và sự tương thích ngược cho các phiên bản Android cũ hơn.

implementation 'com.google.android.material:material:1.7.0': Thư viện Material Design từ Google.

implementation 'androidx.constraintlayout:constraintlayout:2.1.4': Hỗ trợ sử dụng ConstraintLayout trong ứng dụng Android.

e, Thư viện Jetpack Compose

Jetpack Compose là một thư viện UI hiện đại từ Google, giúp xây dựng giao diện người dùng declarative.

implementation("androidx.compose.ui:ui:1.4.3"): Cung cấp các thành phần UI cho Jetpack Compose.

implementation("androidx.compose.ui:ui-tooling:1.4.3"): Thư viện hỗ trợ thiết kế giao diện và kiểm thử UI trong Jetpack Compose.

implementation("androidx.compose.foundation:foundation:1.4.3"): Cung cấp các thành phần cơ bản trong Jetpack Compose.

implementation("androidx.compose.material:material:1.4.3"): Cung cấp các thành phần UI theo Material Design cho Compose.

implementation("androidx.compose.material:material-icons-core:1.4.3"): Thư viện hỗ trợ sử dụng các biểu tượng của Material Design.

implementation("androidx.compose.material:material-icons-extended:1.4.3"): Hỗ trợ sử dụng các biểu tượng mở rộng của Material Design.

implementation("androidx.activity:activity-compose:1.7.2"): Tích hợp Jetpack Compose vào Activity.

implementation("androidx.constraintlayout:constraintlayout-compose:1.0.1"): Sử dụng ConstraintLayout với Jetpack Compose.

implementation("androidx.navigation:navigation-compose:2.4.0"): Cung cấp điều hướng trong Jetpack Compose.

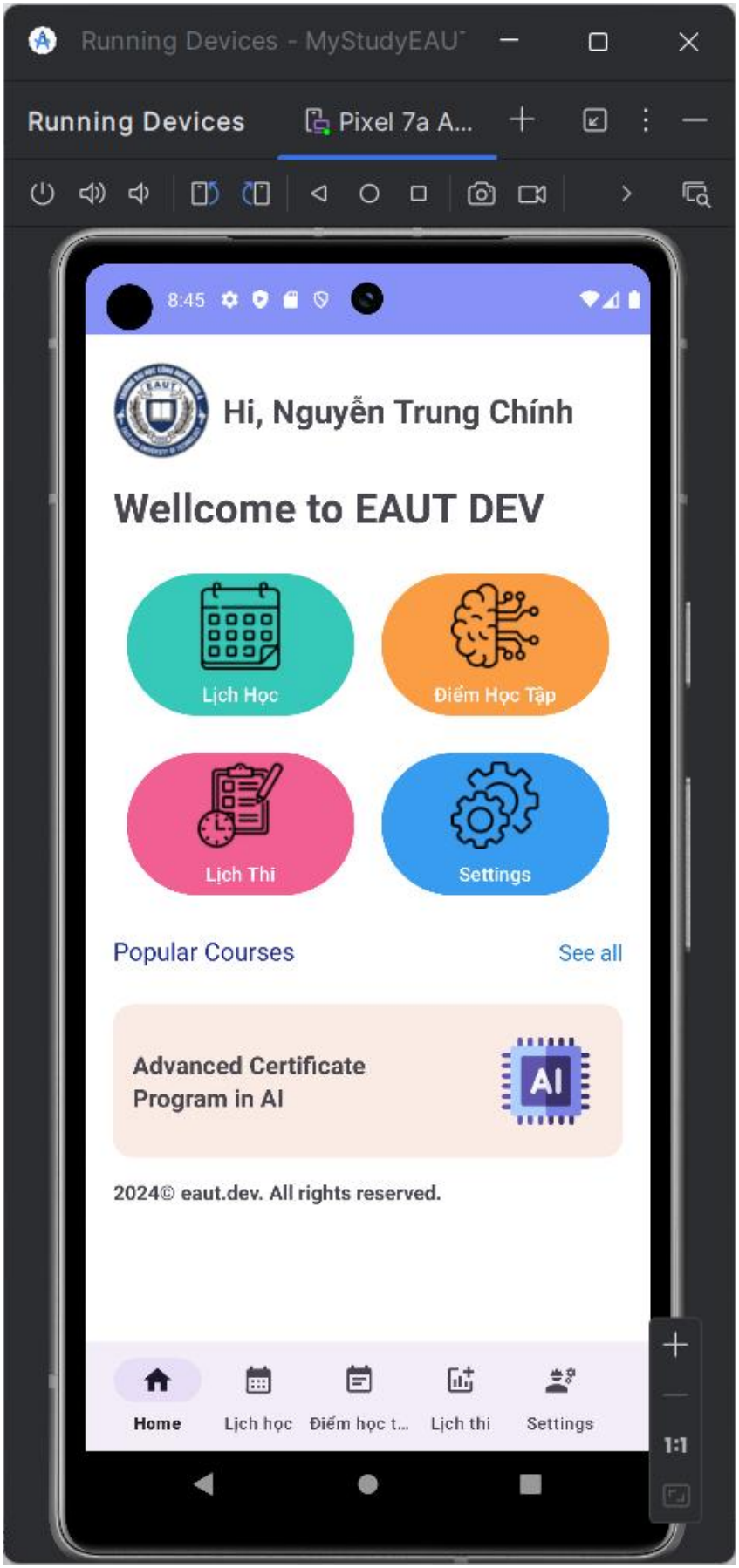
implementation("androidx.compose.ui:ui-tooling-preview:1.4.2"): Hỗ trợ kiểm thử UI và xem trước giao diện trong Compose.

Công nghệ chính của ứng dụng bao gồm Java, Android SDK, RecyclerView cho giao diện động, SQLite hoặc Room Database cho lưu trữ dữ liệu, và có thể sử dụng Firebase nếu cần backend trực tuyến. Các thư viện như Material Design sẽ hỗ trợ việc tạo giao diện đẹp mắt, trong khi Jetpack sẽ giúp bạn tối ưu mã nguồn và quản lý ứng dụng dễ dàng hơn.

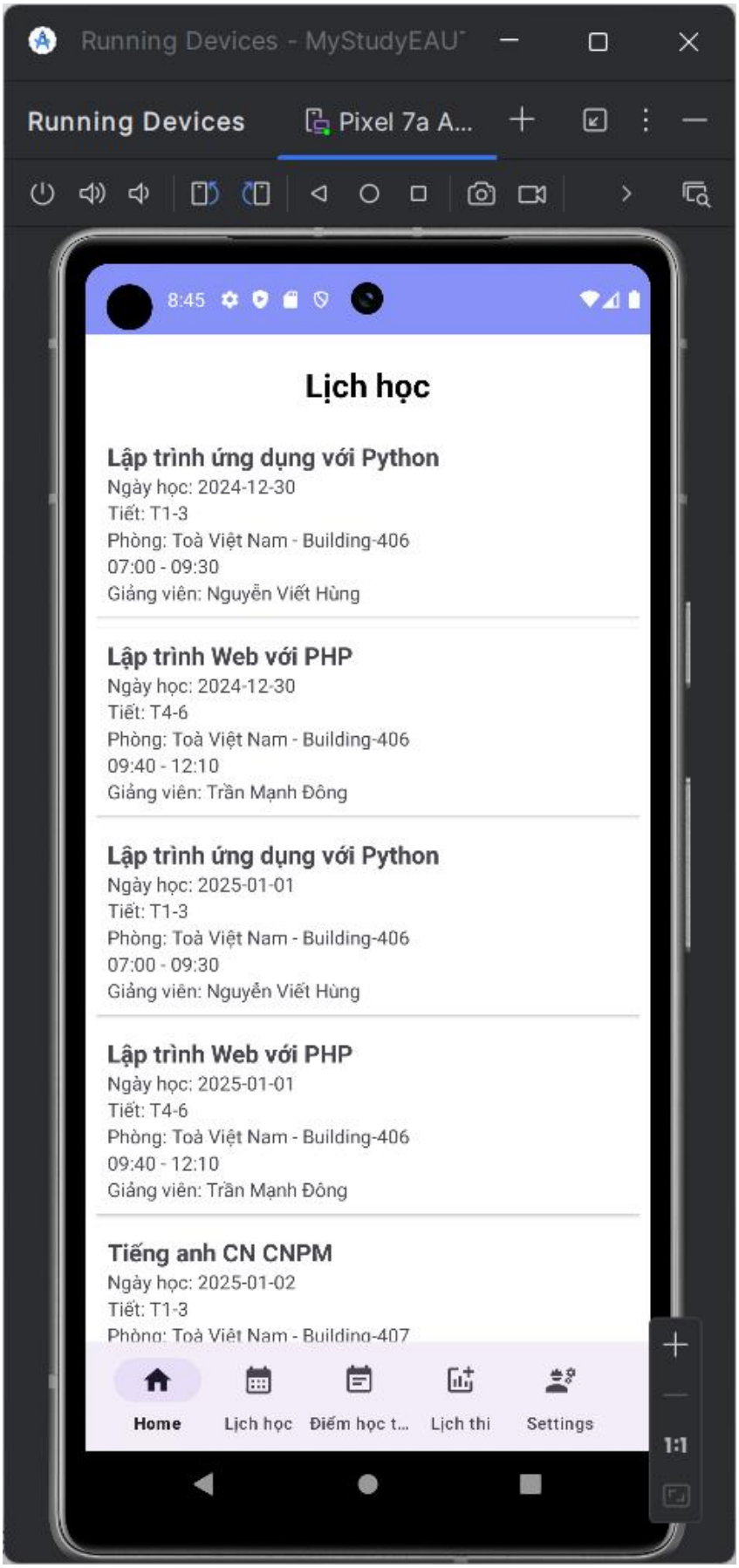
Các thư viện trong phần dependencies đã được sử dụng để xây dựng ứng dụng Android của bạn có thể giúp bạn tối ưu hóa phát triển giao diện, làm việc với API, kiểm thử và lưu trữ dữ liệu. Việc quản lý phiên bản của các thư viện rất quan trọng để đảm bảo tính tương thích và bảo mật cho ứng dụng của bạn. Các gợi ý trên giúp bạn mở rộng và cải tiến ứng dụng của mình, bao gồm việc hỗ trợ các tính năng mới và nâng cao hiệu suất.

4.2. Giao diện ứng dụng EAUT DEV

Hình 4. 1 Giao diện đăng nhập



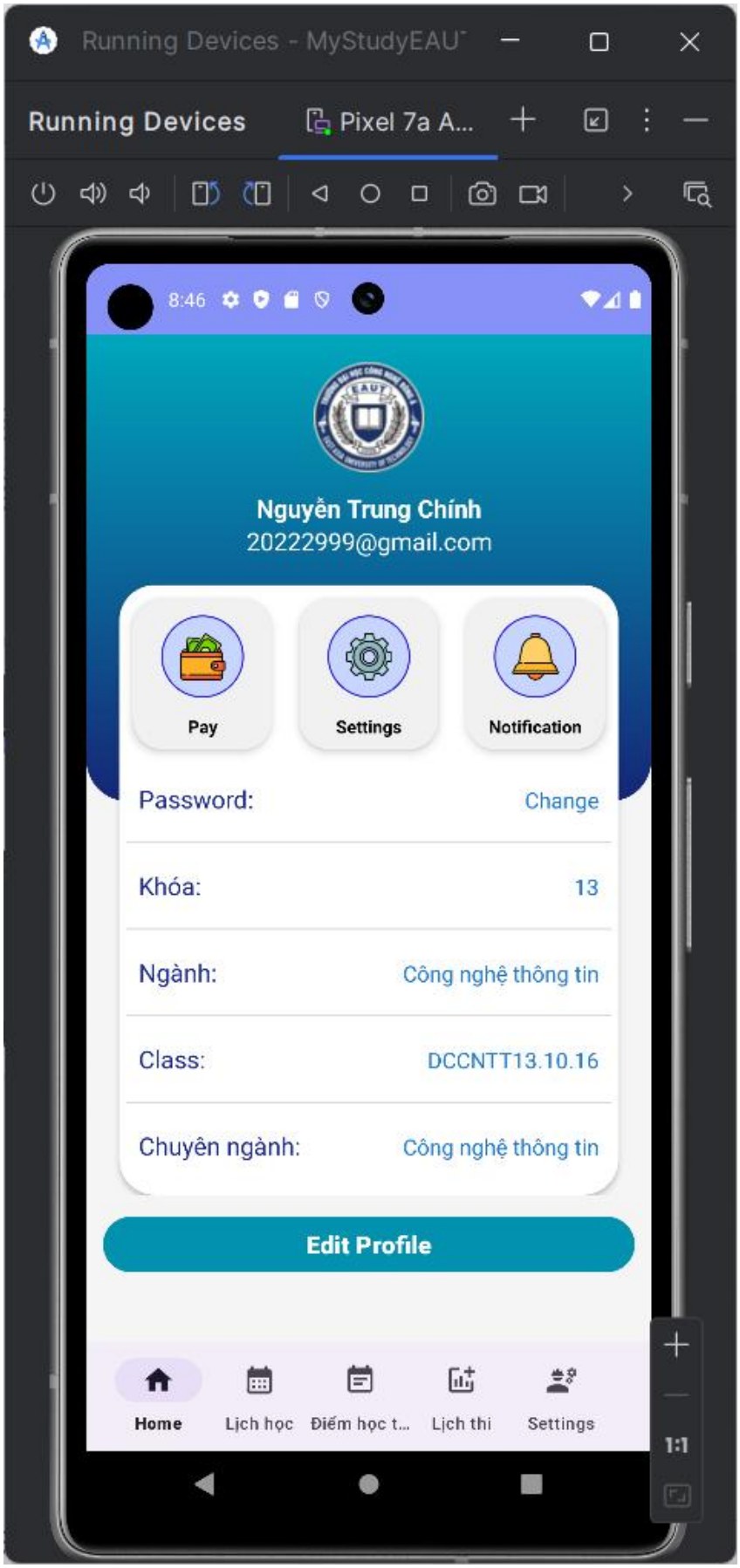
Hình 4. 2 Giao diện home



Hình 4. 3 Giao diện lịch học

Hình 4. 4 Giao diện điểm học tập

Hình 4. 5 Giao diện lịch thi



Hình 4. 6 Giao diện profile người dùng

KẾT LUẬN

Tóm tắt kết quả đạt được

Dự án "Xây dựng ứng dụng Android quản lý lịch học, điểm thi, ngày thi sinh viên EAUT" đã hoàn thành với những kết quả đáng khích lệ. Ứng dụng đã tích hợp đầy đủ các chức năng chính, bao gồm quản lý lịch học, theo dõi điểm thi, thông báo ngày thi, và quản lý thông tin cá nhân cho sinh viên.

Giao diện của ứng dụng được thiết kế đơn giản, dễ sử dụng, phù hợp với đối tượng sinh viên. Các tính năng như thông báo đẩy và đồng bộ hóa dữ liệu giúp sinh viên không bỏ lỡ bất kỳ sự kiện quan trọng nào trong quá trình học tập. Qua đó, ứng dụng đã giúp sinh viên EAUT quản lý tốt hơn các công việc học tập của mình, tiết kiệm thời gian và nâng cao hiệu quả học tập.

Thách thức và giải pháp

Trong suốt quá trình phát triển, nhóm gặp phải một số thách thức như vấn đề đồng bộ hóa dữ liệu từ hệ thống trường học, sự phức tạp trong việc thiết kế giao diện sao cho đơn giản nhưng đầy đủ chức năng, và đảm bảo ứng dụng hoạt động ổn định trên các thiết bị Android khác nhau.

Để giải quyết các vấn đề này, nhóm đã nghiên cứu và ứng dụng các giải pháp như sử dụng API đồng bộ hóa dữ liệu với hệ thống trường học, tối ưu hóa mã nguồn và giao diện người dùng để phù hợp với nhiều thiết bị, đồng thời thực hiện kiểm thử kỹ lưỡng để đảm bảo tính ổn định và hiệu suất của ứng dụng.

Bài học kinh nghiệm

Qua quá trình thực hiện dự án, nhóm đã rút ra được nhiều bài học quý giá. Đầu tiên là việc lập kế hoạch chi tiết và phân công công việc rõ ràng ngay từ đầu là yếu tố quan trọng giúp dự án được thực hiện suôn sẻ.

Thứ hai, việc giao tiếp hiệu quả trong nhóm và chủ động tìm hiểu, học hỏi về công nghệ mới là yếu tố giúp nhóm vượt qua nhiều khó khăn trong quá trình phát triển. Cuối cùng, việc thử nghiệm và nhận phản hồi từ người dùng thực tế đã giúp cải thiện rất nhiều chất lượng của ứng dụng.

Hạn chế và hướng phát triển tương lai

Mặc dù ứng dụng đã hoàn thành và đáp ứng được yêu cầu ban đầu, nhưng vẫn còn một số hạn chế cần khắc phục. Một trong số đó là khả năng hỗ trợ đa nền tảng, hiện tại ứng dụng chỉ hoạt động trên hệ điều hành Android. Trong tương lai, nhóm dự định phát triển ứng dụng trên các nền tảng khác như iOS để mở rộng phạm vi người dùng.

Ngoài ra, ứng dụng cũng cần cải tiến thêm về tính năng cá nhân hóa, như tạo các báo cáo thống kê chi tiết về quá trình học tập của sinh viên, và cải thiện hệ thống bảo mật để đảm bảo thông tin người dùng luôn được bảo vệ tốt nhất.

TÀI LIỆU THAM KHẢO

1. Android Developers. (n.d.). Building your first Android app. Retrieved November 26, 2024, from https://developer.android.com/training/basics/firstapp
2. Meyers, S. (2014). Java for Android development. O'Reilly Media.
3. Sách hướng dẫn chi tiết về lập trình Java dành riêng cho phát triển ứng dụng Android.
4. Morrison, R. (2017). Android Programming: The Big Nerd Ranch Guide. Big Nerd Ranch.
5. Google Developers. (n.d.). Saving data in Android. Retrieved November 26, 2024, from https://developer.android.com/training/data-storage
6. Sharma, A. (2019). Master Android development with Java: Build real-time apps. Packt Publishing.
7. Yaroch, R. (2016). Android Application Development for Java Programmers. Addison-Wesley.
8. Chaudhary, S. (2018). Android Application Development with Kotlin and Java. Wiley.
9. McDonald, D. (2015). Beginning Android 4 Application Development. Wiley.
10. Dwyer, M. (2015). Android development with Java. Cambridge University Press.
11. Narula, V. (2020). Android Studio 4.0 Development Essentials. Packt Publishing.
12. OpenAI. (2024). ChatGPT: Assistant for mobile app development. OpenAI. Retrieved November 26, 2024, from https://chat.openai.com
13. Studocu. (n.d.). Xây dựng ứng dụng quản lý sinh viên. Retrieved November 26, 2024, from https://www.studocu.com/vn/document/truong-dai-hoc-thu-dau-mot/cong-nghe-thong-tin/xay-dung-ung-dung-quan-ly-sinh-vien/42925923
14. Express Magazine. (2024, October 10). Lập trình Android: Ứng dụng quản lý sinh viên. Retrieved November 26, 2024, from https://expressmagazine.net/development/1688/lap-trinh-android-ung-dung-quan-ly-sinh-vien
15. Lehuutrung1412. (2024, March 5). Student Management. Retrieved November 26, 2024, from https://github.com/lehuutrung1412/StudentManagement
16. Sharecode. (n.d.). Ứng dụng Android quản lý sinh viên. Retrieved November 26, 2024, from https://sharecode.vn/source-code/ung-dungandroi-quan-ly-sinh-vien-1007.htm
17. CodeGym. (n.d.). Tài liệu lập trình Android cơ bản (PDF). Retrieved November 26, 2024, from https://codegym.vn/blog/tai-lieu-lap-trinh-android-co-ban-pdf/