BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT

THÀNH PHỐ HỒ CHÍ MINH

KHOA: CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

MÔN: LẬP TRÌNH DI ĐỘNG

LỚP: MOPR331279_23_2_04

ĐỀ TÀI: ỨNG DỤNG DI ĐỘNG VỀ LUYỆN TẬP CTF

GVHD: Lê Quang Thái

Sinh viên thực hiện:

Nguyễn Thắng Lợi_22162023

Lê Anh Khoa_22162016

TP. Hồ Chí Minh, tháng 05 năm 2024

Mục Lục

Phần 1: Tổng Quan Về Ứng Dụng

1.1 Lý do chọn đề tài

Nhóm lựa chọn thực hiện đề tài là "Ứng Dụng Di Động Về Luyện Tập CTF[1]". Ứng dụng được phát triển để cung cấp một nền tảng học tập trực tuyến, nơi người dùng có thể đọc các bài viết, xem video và làm các bài kiểm tra để ôn tập kiến thức. Ứng dụng này được xây dựng bằng cách sử dụng công nghệ React Native cho phía giao diện người dùng và Django REST framework cho phía server. Có hai mục đích chính mà nhóm muốn hướng đến khi bắt tay vào phát triển dự án nêu trên:

- Thứ nhất và cũng là quan trọng nhất: thực hiện dự án cuối kì dành cho môn học Lập Trình Di Động, đáp án các yêu cầu của môn học.
- Thứ hai, các sinh viên trong nhóm đều là thành viên của CLB An Toàn Thông Tin - HCMUTE ISC nên muốn xây dựng sản phẩm phục vụ cho hoạt động của CLB mà cụ thể ở đây là một ứng dụng học tập thông qua các bài viết học thuật về thi đấu CTF do CLB An Toàn Thông Tin - HCMUTE ISC soạn thảo và đăng tải.

1.2 Tính ứng dụng của dự án

Dự án sau khi hoàn thiện sẽ được triển khai thí điểm trong phạm vi nội bộ của nhóm phát triển nhằm kiểm tra các hoạt động và tìm ra bất thường nếu có của ứng dụng.

Xa hơn, nhóm phát triển sẽ triển khai ứng dụng trên phạm vi toàn hệ thống của CLB An Toàn Thông Tin và sử dụng phục vụ cho mục đích học tập của các thành viên.

1.3 Phương pháp phát triển

- Kiến trúc Client-Server:
    - Client: Phía client là một ứng dụng di động được xây dựng bằng React Native. Client chịu trách nhiệm giao diện người dùng, thu thập và gửi dữ liệu người dùng, và hiển thị thông tin từ server.

[1] CTF là viết tắt của Capture The Flag, một hình thức thi đấu phổ biến trong lĩnh vực An toàn Thông tin với nội dung cốt lõi là truy tìm các cờ (flag) thông qua khai thác các lỗ hổng bảo mật và giải quyết các thử thách kỹ thuật do ban tổ chức đặt ra.

- Server: Phía server là một API web được xây dựng bằng Django REST framework. Server chịu trách nhiệm xử lý logic nghiệp vụ, truy vấn cơ sở dữ liệu, và gửi dữ liệu về cho client.

- Phương pháp Lập trình Hướng Đối Tượng (OOP):
  - Class và Object: Các thực thể trong ứng dụng như User, Article, Section, và Quiz được mô hình hóa dưới dạng các class. Mỗi class có các thuộc tính và phương thức riêng để thao tác với dữ liệu.
  - Tái sử dụng và mở rộng: OOP cho phép tái sử dụng mã nguồn thông qua kế thừa và đa hình, giúp dễ dàng mở rộng tính năng mà không cần viết lại mã từ đầu.
- Django REST framework được sử dụng để xây dựng các API RESTful, tuân theo các nguyên tắc của REST (Representational State Transfer):
  - Stateless: Mỗi yêu cầu từ client tới server phải chứa tất cả thông tin cần thiết để server hiểu và xử lý yêu cầu đó. Server không lưu trạng thái phiên làm việc của client.
  - Resource-based: Mọi thứ đều được coi là một tài nguyên (resource), và mỗi tài nguyên được định danh duy nhất bằng một URI (Uniform Resource Identifier).
  - HTTP Methods: Các hành động trên tài nguyên được thực hiện thông qua các phương thức HTTP (GET, POST, PUT, DELETE).
- Phía client sử dụng React Hooks để quản lý trạng thái và vòng đời của các component một cách hiệu quả:
  - useState: Quản lý trạng thái cục bộ trong các component.
  - useEffect: Thực hiện các side-effect như gọi API, cập nhật DOM, và dọn dẹp các tài nguyên khi component được mount hoặc unmount.
  - useFocusEffect: Một hook đặc biệt từ React Navigation, giúp thực hiện các hành động khi màn hình được focus lại (như cập nhật dữ liệu mới).

Phần 2: Kết Cấu Của Ứng Dụng

2.1 Xây dựng frontend

2.1.1 Công nghệ sử dụng

React Native: Được sử dụng để phát triển ứng dụng di động đa nền tảng. React Native cho phép chúng ta viết mã nguồn JavaScript và xuất bản ứng dụng trên cả iOS và Android.

React Navigation: Thư viện điều hướng cho phép tạo ra các luồng màn hình phức tạp trong ứng dụng di động.

Axios: Thư viện HTTP Client giúp gửi các yêu cầu HTTP tới API server và xử lý phản hồi.

2.1.2 Cấu trúc mã nguồn

Components: Các thành phần giao diện người dùng được xây dựng dưới dạng các component độc lập, giúp dễ dàng tái sử dụng và bảo trì.

UI Components: Chứa các component nhỏ như Button, TextInput, ImageView.

Screen Components: Chứa các component lớn hơn đại diện cho các màn hình chính của ứng dụng, ví dụ: HomeScreen, QuizScreen, ArticleScreen.

Navigation: Định nghĩa các luồng điều hướng giữa các màn hình bằng React Navigation.

Services: Chứa các tệp mã nguồn liên quan đến việc gọi API, như cấu hình Axios và các hàm gọi API cụ thể.

Context: Sử dụng React Context để quản lý trạng thái toàn cục, ví dụ: AuthProvider để quản lý trạng thái xác thực người dùng.

2.1.3 Quy trình hoạt động

Hiển thị dữ liệu: Khi người dùng mở ứng dụng, các màn hình chính sẽ gửi yêu cầu HTTP tới backend để lấy dữ liệu. Sau đó, dữ liệu sẽ được hiển thị trên giao diện người dùng.

Xử lý sự kiện: Khi người dùng tương tác với giao diện (như chọn đáp án trong bài kiểm tra), ứng dụng sẽ cập nhật trạng thái cục bộ và có thể gửi yêu cầu tới backend để lưu trữ kết quả hoặc lấy dữ liệu mới.

Điều hướng: Khi người dùng điều hướng giữa các màn hình, React Navigation sẽ quản lý lịch sử điều hướng và chuyển đổi màn hình một cách mượt mà.

2.2 Xây dựng backend

2.2.1 Công nghệ sử dụng

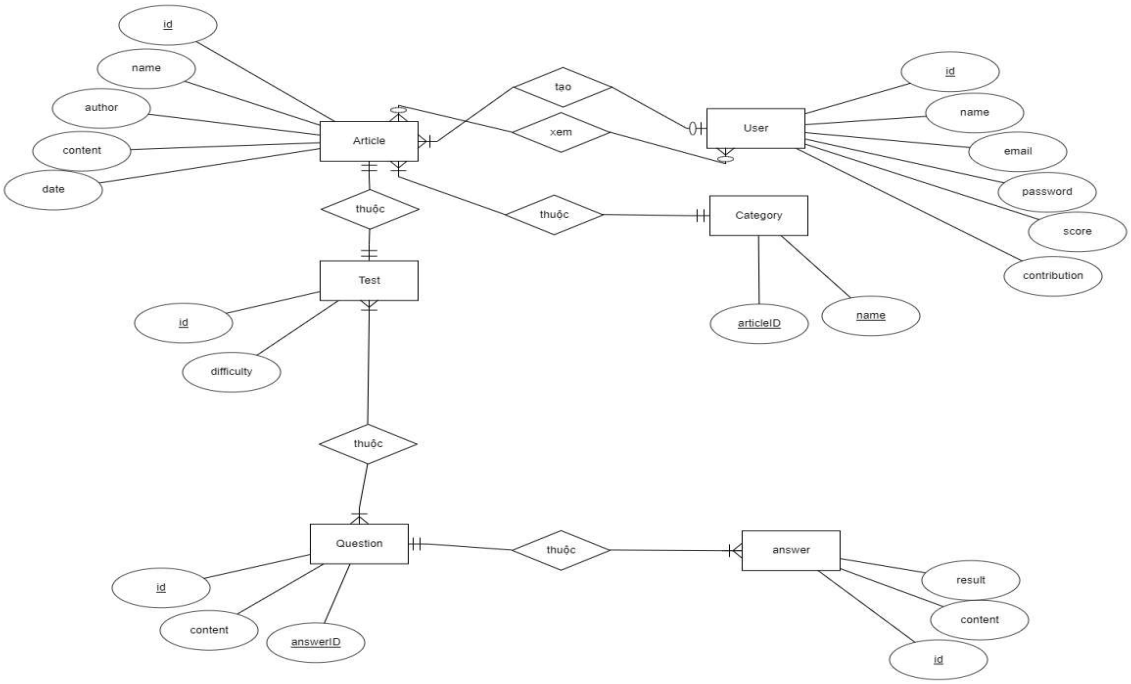
Django: Framework web mạnh mẽ được viết bằng Python, cung cấp các công cụ và thư viện cần thiết để xây dựng backend.

Django REST framework (DRF): Mở rộng của Django, giúp xây dựng các API RESTful một cách dễ dàng và hiệu quả.

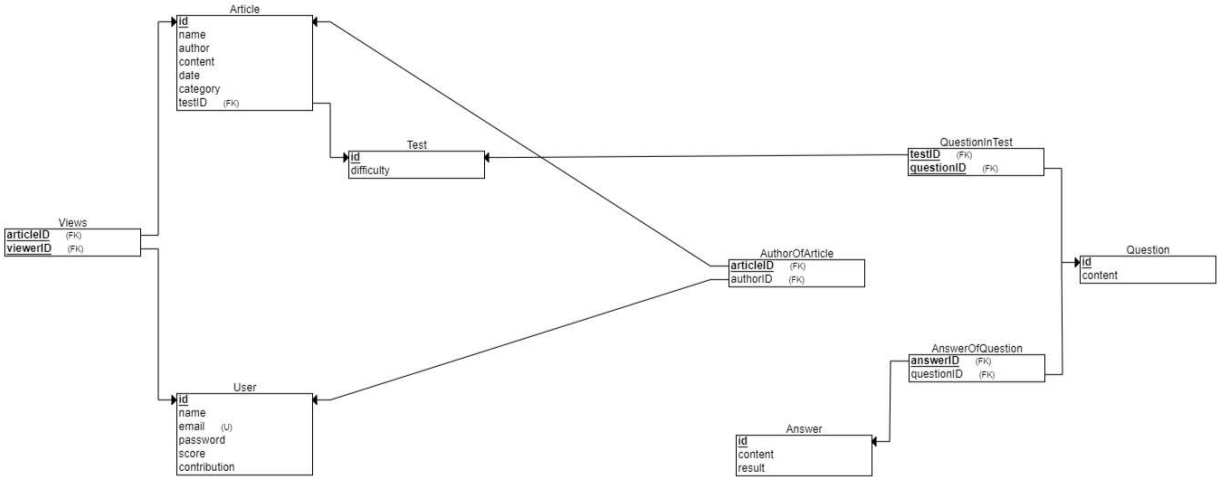
MySQL: Cơ sở dữ liệu để lưu trữ dữ liệu ứng dụng sử dụng trong môi trường phát triển và cả môi trường sản xuất.

2.2.2 Tổ chức cơ sở dữ liệu

Thiết kế sơ đồ Thực thể - Mối quan hệ (ERD):



Ánh xạ ERD sang Lược đồ Quan hệ (Relational Schema):



Bảng auth_user: Bảng này được Django tự động tạo ra để lưu trữ thông tin người dùng. Đây là bảng cốt lõi trong mô hình xác thực và phân quyền của Django. Các trường chính của bảng auth_user bao gồm:

- id: Khóa chính, tự động tăng.
- username: Tên người dùng, duy nhất và không trùng lặp.

- password: Mật khẩu của người dùng, được mã hóa.
- email: Địa chỉ email của người dùng.
- first_name: Tên riêng của người dùng.
- last_name: Họ của người dùng.
- is_staff: Biến boolean, xác định liệu người dùng có quyền truy cập vào trang quản trị hay không.
- is_active: Biến boolean, xác định liệu tài khoản có đang hoạt động hay không.
- date_joined: Ngày và giờ người dùng đăng ký tài khoản.

Bảng authtoken_token: Bảng này được tạo ra khi sử dụng Django REST Framework's Token Authentication để lưu trữ các token xác thực cho người dùng. Các trường chính của bảng authtoken_token bao gồm:

- key: Khóa chính, token xác thực, thường là một chuỗi ngẫu nhiên duy nhất.
- user_id: Khóa ngoại liên kết đến bảng auth_user, xác định người dùng sở hữu token.
- created: Ngày và giờ token được tạo.

Bảng Test: Bảng này lưu trữ thông tin về các bài kiểm tra

- id: Khóa chính, tự động tăng.
- difficulty: Độ khó của bài kiểm tra, lưu dưới dạng chuỗi ký tự.

Bảng Articles: Bảng này lưu trữ thông tin về các bài viết.

- id: Khóa chính, định danh duy nhất của bài viết, kiểu UUID.
- name: Tên của bài viết.
- author: Khóa ngoại liên kết đến bảng User, xác định tác giả của bài viết.
- date: Ngày tạo bài viết, tự động điền khi bài viết được tạo.
- category: Thể loại của bài viết.
- test: Khóa ngoại liên kết đến bảng Test, xác định bài kiểm tra liên quan đến bài viết.

- total_views: Số lượt xem của bài viết.

Bảng Sections: Bảng này lưu trữ các phần của một bài viết, cho phép bài viết được chia thành nhiều phần với các loại nội dung khác nhau như văn bản, hình ảnh, và video.

- id: Khóa chính, định danh duy nhất của phần, kiểu UUID.
- article: Khóa ngoại liên kết đến bảng Articles, xác định bài viết chứa phần này.
- part_type: Loại phần (text, image, video).
- text: Nội dung văn bản của phần (nếu có).
- image: URL của hình ảnh (nếu có).
- video_url: URL của video (nếu có).
- created_at: Ngày tạo phần, tự động điền khi phần được tạo.
- position: Vị trí của phần trong bài viết.
- Phương thức save của bảng Sections sử dụng thư viện bleach để làm sạch nội dung văn bản, cho phép chỉ một số thẻ HTML nhất định.

Bảng AuthorOfArticle: Bảng này lưu trữ thông tin về tác giả của các bài viết, cho phép một bài viết có nhiều tác giả.

- id: Khóa chính, định danh duy nhất, kiểu UUID.
- article: Khóa ngoại liên kết đến bảng Articles.
- author: Khóa ngoại liên kết đến bảng User.

Bảng Question: Bảng này lưu trữ các câu hỏi.

- id: Khóa chính, tự động tăng.
- content: Nội dung của câu hỏi.

Bảng QuestionInTest: Bảng này lưu trữ mối quan hệ giữa các bài kiểm tra và câu hỏi.

- id: Khóa chính, tự động tăng.
- test: Khóa ngoại liên kết đến bảng Test.

- question: Khóa ngoại liên kết đến bảng Question.

Bảng Answer: Bảng này lưu trữ các đáp án cho các câu hỏi.

- id: Khóa chính, tự động tăng.
- content: Nội dung của đáp án.
- result: Biến boolean xác định đáp án đúng hay sai.
- question: Khóa ngoại liên kết đến bảng Question.

Bảng CustomUser: Bảng này lưu trữ thông tin mở rộng về người dùng.

- id: Khóa chính, định danh duy nhất, kiểu UUID.
- user: Khóa ngoại liên kết đến bảng User.
- score: Điểm số của người dùng.
- contribution: Số lượng bài viết mà người dùng đóng góp.
- rank: Thứ hạng của người dùng.
- avatar: URL của ảnh đại diện người dùng.

Bảng Comment: Bảng này lưu trữ bình luận của người dùng trên các bài viết.

- user: Khóa ngoại liên kết đến bảng User.
- article: Khóa ngoại liên kết đến bảng Articles.
- text: Nội dung của bình luận.
- created_at: Ngày tạo bình luận, tự động điền khi bình luận được tạo.

2.2.3 Cấu trúc mã nguồn

Apps: Django tổ chức các thành phần thành các ứng dụng con (apps). Mỗi app có thể có mô hình, bộ điều khiển và bộ lọc riêng.

- CTF_App: Chứa các mô hình chính như User, Article, Section, Question, Answer, Test.

Models: Định nghĩa các mô hình dữ liệu, tương ứng với các bảng trong cơ sở dữ liệu.

- User: Mô hình người dùng, bao gồm thông tin xác thực và hồ sơ cá nhân.
- Article: Mô hình bài viết, bao gồm thông tin về tác giả, ngày xuất bản và nội dung.
- Section: Mô hình các phần của bài viết, bao gồm loại phần (text, image, video), nội dung và vị trí trong bài viết.
- Question: Mô hình câu hỏi trong bài kiểm tra, liên kết với các đáp án.
- Answer: Mô hình đáp án, bao gồm nội dung đáp án và xác định đáp án đúng hay sai.
- Test: Mô hình bài kiểm tra, bao gồm danh sách các câu hỏi và độ khó.

Serializers: Định nghĩa cách thức chuyển đổi giữa mô hình Django và JSON để trả về client.

Views: Chứa các logic xử lý yêu cầu HTTP và trả về phản hồi.

- Viewsets: Sử dụng Django REST framework viewsets để quản lý các hành động CRUD (Create, Read, Update, Delete) cho mỗi mô hình.

Urls: Định nghĩa các endpoint API, liên kết các URL tương ứng với các view có liên quan.

2.2.4 Quy trình hoạt động

Xác thực: Khi client gửi yêu cầu, backend sẽ xác thực người dùng thông qua token được gửi kèm trong header của yêu cầu HTTP.

Xử lý yêu cầu: Backend nhận yêu cầu, xác định view cần thiết và xử lý logic nghiệp vụ, bao gồm truy vấn cơ sở dữ liệu, tính toán và xử lý dữ liệu.

Trả về phản hồi: Sau khi xử lý xong, backend sẽ trả về dữ liệu cần thiết dưới dạng JSON, để client có thể hiển thị hoặc tiếp tục xử lý.

Phần 3: Các Chức Năng Và Cơ Chế Vận Hành Của Ứng Dụng

3.1 Các chức năng của ứng dụng

3.1.1. Quản lý Người Dùng

Đăng ký và Đăng nhập: Người dùng có thể đăng ký tài khoản mới hoặc đăng nhập vào tài khoản hiện có. Hệ thống sử dụng mô hình auth_user của Django để quản lý thông tin người dùng và xác thực. Khi đăng nhập thành công, một token xác thực được tạo và lưu trữ trong bảng authtoken_token, cho phép người dùng thực hiện các hành động được bảo vệ trong ứng dụng.

Thông tin Cá Nhân: Người dùng có thể xem và chỉnh sửa thông tin cá nhân của mình, bao gồm điểm số, thứ hạng và ảnh đại diện. Thông tin này được lưu trữ trong bảng CustomUser.

3.1.2. Quản lý Bài Viết

Tạo và Chỉnh sửa Bài Viết: Người dùng có thể tạo mới hoặc chỉnh sửa các bài viết. Mỗi bài viết bao gồm tiêu đề, nội dung, tác giả, ngày đăng, thể loại và tổng số lượt xem. Thông tin này được lưu trữ trong bảng Articles. Mỗi bài viết có thể chứa nhiều phần (sections) như văn bản, hình ảnh và video, được quản lý trong bảng Sections.

Hiển thị Bài Viết: Bài viết được hiển thị dưới dạng danh sách hoặc chi tiết. Khi người dùng truy cập vào một bài viết, thông tin từ bảng Articles và các phần từ bảng Sections được tải và hiển thị.

3.1.3. Quản lý Bình Luận

Thêm Bình Luận: Người dùng có thể thêm bình luận vào các bài viết. Bình luận bao gồm nội dung, người dùng viết bình luận và thời gian tạo. Thông tin này được lưu trữ trong bảng Comment.

Hiển thị Bình Luận: Bình luận được hiển thị dưới mỗi bài viết, cho phép người dùng tương tác và thảo luận về nội dung bài viết.

3.1.4. Quản lý Câu Hỏi và Bài Kiểm Tra

Tạo và Chỉnh sửa Câu Hỏi: Người dùng có thể tạo và chỉnh sửa các câu hỏi cho bài kiểm tra. Mỗi câu hỏi bao gồm nội dung và các câu trả lời, được quản lý trong bảng Question và Answer.

Quản lý Bài Kiểm Tra: Bài kiểm tra bao gồm nhiều câu hỏi và có độ khó khác nhau. Bài kiểm tra được liên kết với các bài viết thông qua bảng Test và QuestionInTest.

Thực hiện Bài Kiểm Tra: Người dùng có thể tham gia làm bài kiểm tra và nhận điểm số dựa trên kết quả. Điểm số được cập nhật và lưu trữ trong thông tin cá nhân của người dùng.

3.1.5. Quản lý Thể Loại và Lượt Xem

Phân loại Bài Viết: Bài viết được phân loại theo các thể loại khác nhau, giúp người dùng dễ dàng tìm kiếm và truy cập nội dung quan tâm.

Thống kê Lượt Xem: Mỗi bài viết có tổng số lượt xem được cập nhật mỗi khi người dùng truy cập. Thông tin này được lưu trữ trong trường total_views của bảng Articles.

3.2 Cơ chế vận hành

3.2.1. Mô hình Client-Server

Client (Frontend): Giao diện người dùng được xây dựng bằng React Native, cung cấp trải nghiệm người dùng mượt mà trên các thiết bị di động. Ứng dụng React Native giao tiếp với backend thông qua các API RESTful, gửi và nhận dữ liệu từ server.

Server (Backend): Django REST Framework được sử dụng để xây dựng API, quản lý logic nghiệp vụ và giao tiếp với cơ sở dữ liệu. Server xử lý các

yêu cầu từ client, thực hiện các tác vụ như xác thực, truy xuất dữ liệu và cập nhật cơ sở dữ liệu.

3.2.2. Giao tiếp qua API RESTful

Các API RESTful cung cấp các endpoint cho các chức năng như đăng ký, đăng nhập, quản lý bài viết, bình luận và bài kiểm tra. Ví dụ, API /api/articles/ dùng để quản lý các bài viết, cho phép lấy danh sách bài viết, tạo mới, chỉnh sửa và xóa bài viết.

Các yêu cầu HTTP (GET, POST, PUT, DELETE) được sử dụng để tương tác với API. Mỗi yêu cầu được xử lý bởi các view trong Django, thực hiện các tác vụ tương ứng và trả về phản hồi dưới dạng JSON.

3.2.3. Quản lý Trạng thái và Dữ liệu

State Management: Trạng thái ứng dụng trên frontend được quản lý bằng cách sử dụng React hooks (useState, useEffect) và các context providers (AuthProvider) để chia sẻ trạng thái giữa các thành phần.

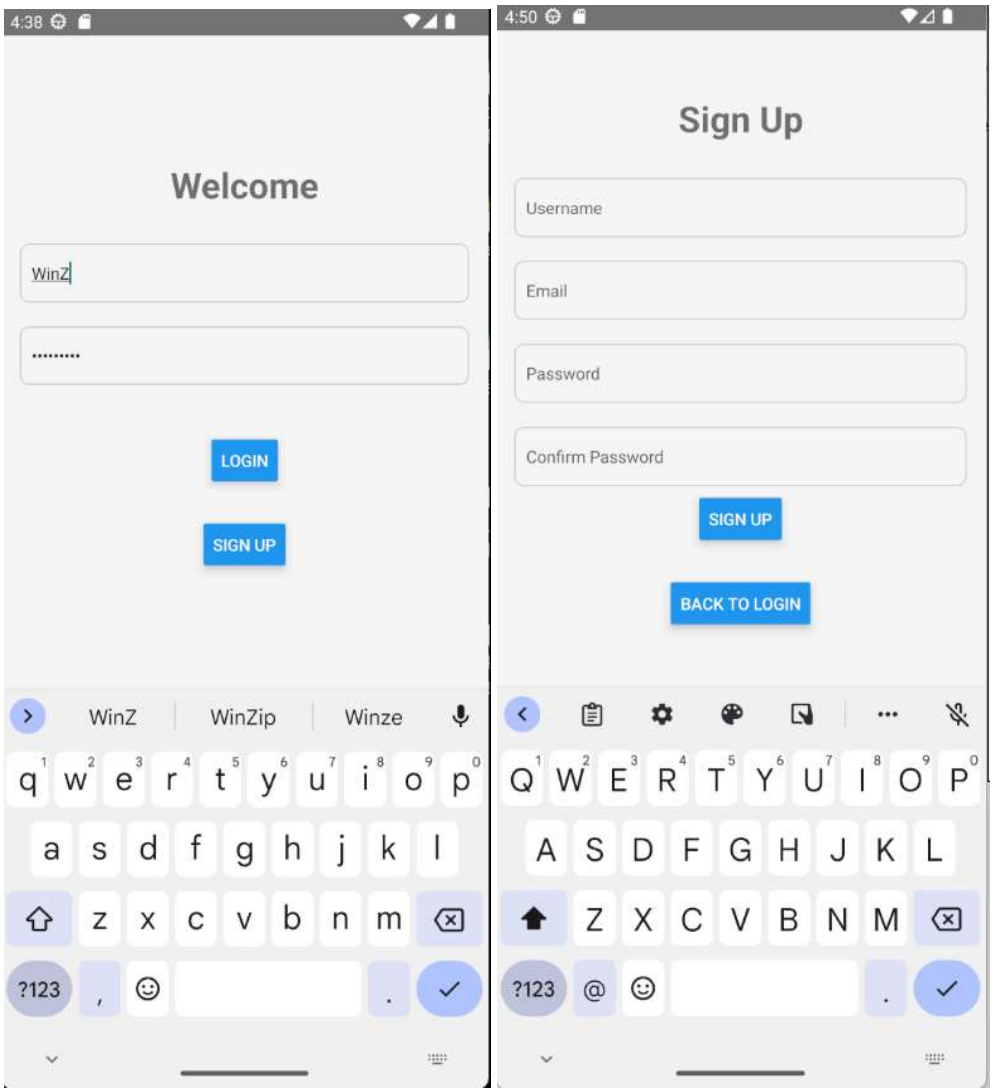
Dữ liệu: Dữ liệu từ server được lưu trữ trong cơ sở dữ liệu PostgreSQL. Django ORM (Object-Relational Mapping) được sử dụng để tương tác với cơ sở dữ liệu, cho phép truy vấn và thao tác dữ liệu một cách dễ dàng.

3.2.4. Bảo mật và Xác thực

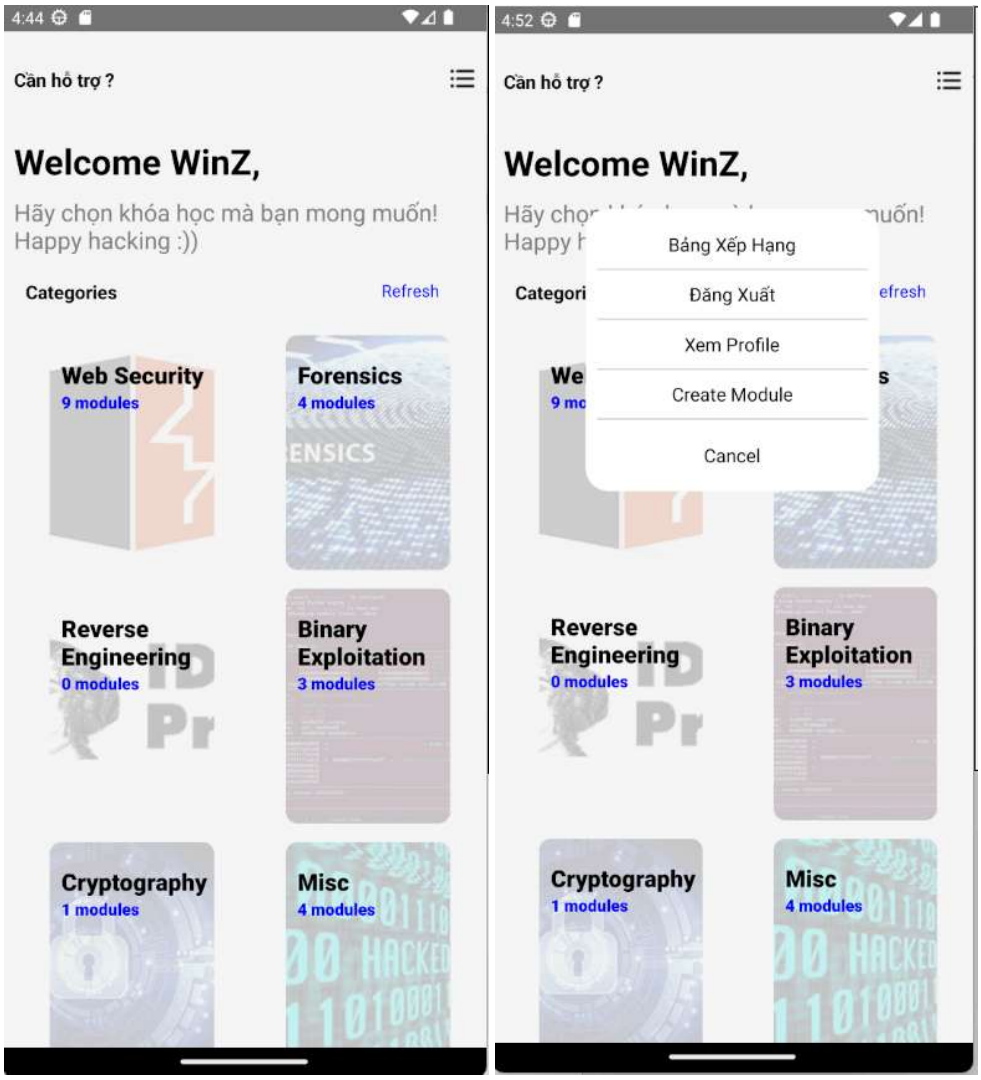
Xác thực bằng Token: Django REST Framework sử dụng token để xác thực người dùng. Mỗi lần người dùng đăng nhập, một token xác thực được tạo và lưu trữ. Các yêu cầu API từ client phải đính kèm token này trong header để được xác thực.

Bảo vệ Dữ liệu: Các biện pháp bảo mật như mã hóa mật khẩu, kiểm tra đầu vào và lọc HTML (sử dụng bleach) được áp dụng để bảo vệ dữ liệu người dùng và đảm bảo an toàn cho ứng dụng.

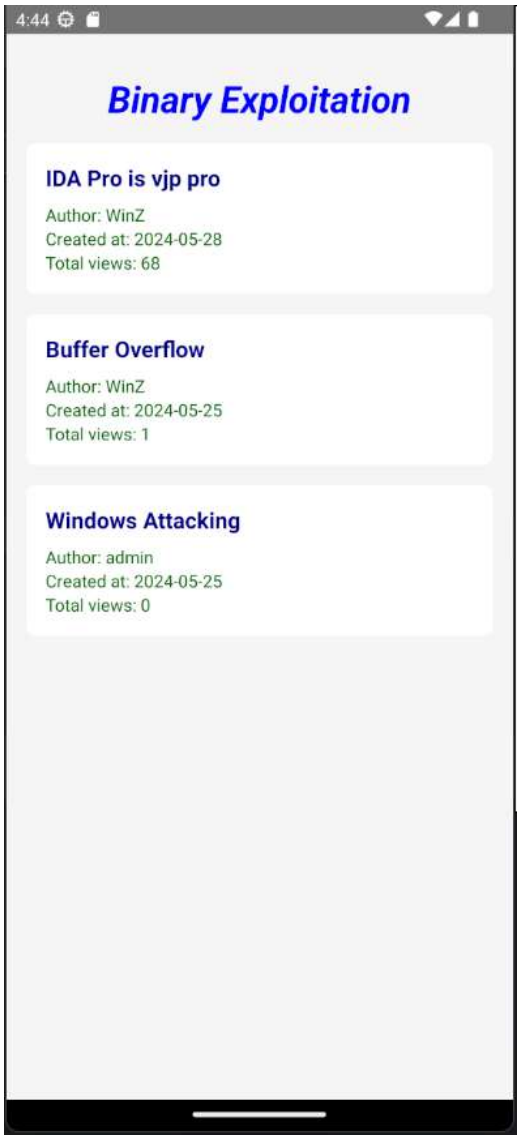
3.3 Các màn hình của ứng dụng



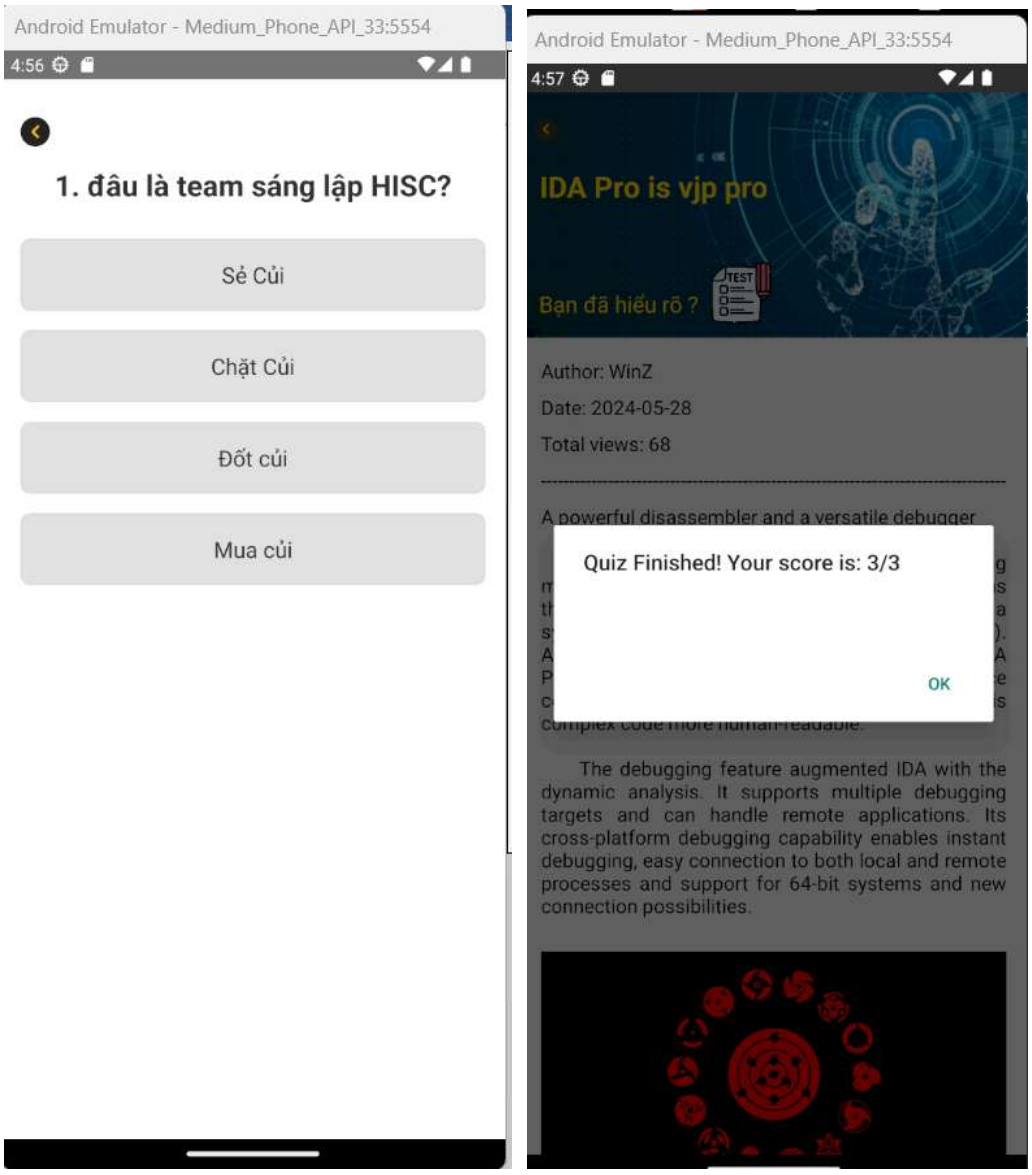
Màn hình đăng nhập và đăng ký



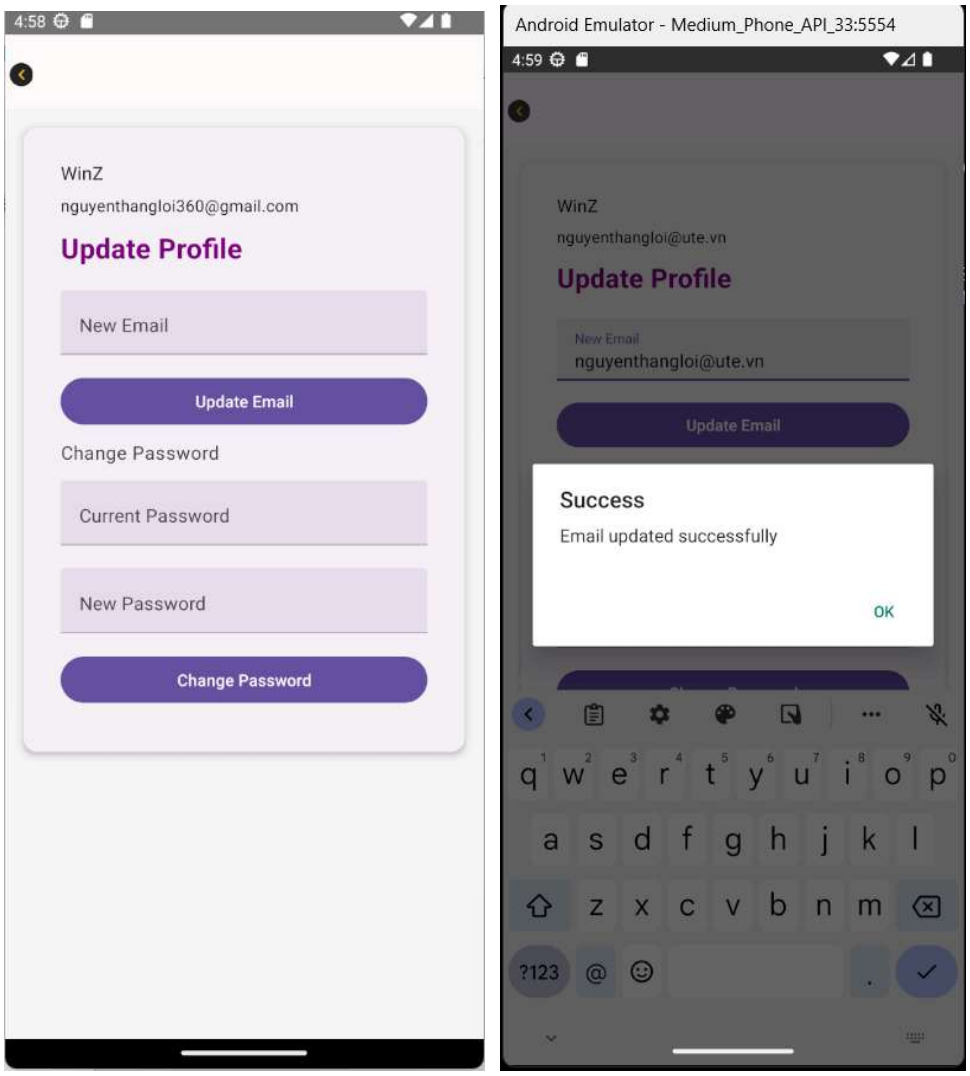
Trang chủ sau khi đăng nhập và khi ấn menu góc phải trên


4:44
Binary Exploitation
IDA Pro is vjp pro
Author: WinZ
Created at: 2024-05-28
Total views: 68
Buffer Overflow
Author: WinZ
Created at: 2024-05-25
Total views: 1
Windows Attacking
Author: admin
Created at: 2024-05-25
Total views: 0

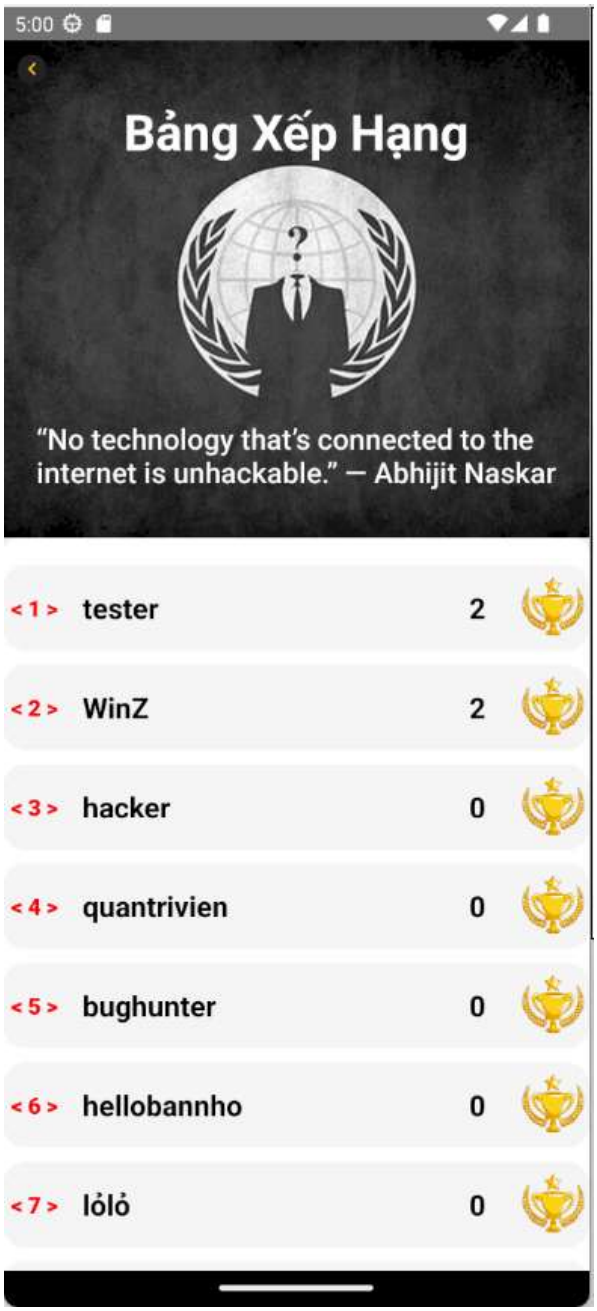
Màn hình xem danh sách các bài viết theo thể loại và chi tiết về bài viết



Màn hình làm bài kiểm tra ôn tập kiến thức



Màn hình profile, đổi email và mật khẩu



Màn hình bảng xếp hạng

Phần 4: Phân Chia Công Việc

Về vấn đề phân chia công việc cụ thể cho mỗi thành viên, nhóm tiến hành tạo repo Github với 3 branches để thuận tiện quản lý source code, lần lượt là: Main (nơi chứa source code chính thức của dự án, cần thống nhất cả nhóm để commit và push code), Loi (nơi lưu trữ source code của thành viên Thắng Lợi), Khoa (nơi lưu trữ source code của thành viên Anh Khoa). Dự án được tiến hành thực hiện từ đầu tháng 05/2024.

Họ tên	Công việc	Tỷ lệ hoàn thành
Nguyễn Thắng Lợi	- Xây dựng kế hoạch phát triển ứng dụng - Thiết kế cơ sở dữ liệu - Phát triển backend với Django - Kiểm thử ứng dụng - Viết báo cáo	100%
Lê Anh Khoa	- Xây dựng kế hoạch phát triển ứng dụng - Thiết kế cơ sở dữ liệu - Phát triển frontend với React Native - Kiểm thử ứng dụng - Viết báo cáo	100%