Ứng dụng học tiếng Anh EE

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 28/11/2022 | 1.0 | Khởi tạo tài liệu với các mục Introduction, Architectural Goals và Constraints. | Nguyễn Gia Hào |
| 29/11/2022 | 1.0 | Thêm dữ liệu toàn bộ mục Logical View. | Nguyễn Gia Hào |
| 12/12/2022 | 1.0 | Cập nhật thông tin mô tả chung của thành phần Frontend ở section 4.1. | Võ Thành Phong |
| 14/12/2022 | 1.0 | Thêm dữ liệu cho section 5 và section 6. | Trương Ngọc Huy |
| 2/1/2023 | 1.0 | Thay đổi tên tên các hàm signin thành signup và thay đổi hình ảnh các component ở các section 4.  Thay đổi cấu trúc thư mục server. | Trương Ngọc Huy |

Mục lục

[1. Introduction 4](#_Toc121944400)

[2. Architectural Goals and Constraints 4](#_Toc121944401)

[3. Use-Case Model 4](#_Toc121944402)

[4. Logical View 5](#_Toc121944403)

[4.1 Component: UI/UX (FrontEnd) 5](#_Toc121944404)

[4.2 Component: Back-end (Client-side) 16](#_Toc121944405)

[4.3 Component: Controller 17](#_Toc121944406)

[4.4 Component: Model 19](#_Toc121944407)

[4.5 Component: View 20](#_Toc121944408)

[4.6 Component: Database 21](#_Toc121944409)

[5. Deployment 26](#_Toc121944410)

[5.1 Overview: 26](#_Toc121944411)

[5.2 Component: Client 27](#_Toc121944412)

[5.3 Component: Server 28](#_Toc121944413)

[5.4 Component: Database 28](#_Toc121944414)

[6. Implementation View 29](#_Toc121944415)

[6.1 App\_Data 30](#_Toc121944416)

[6.2 App\_Client 31](#_Toc121944417)

[6.3 App\_Server 32](#_Toc121944418)

[6.4 Database 33](#_Toc121944419)

[6.5 Main.py 33](#_Toc121944420)

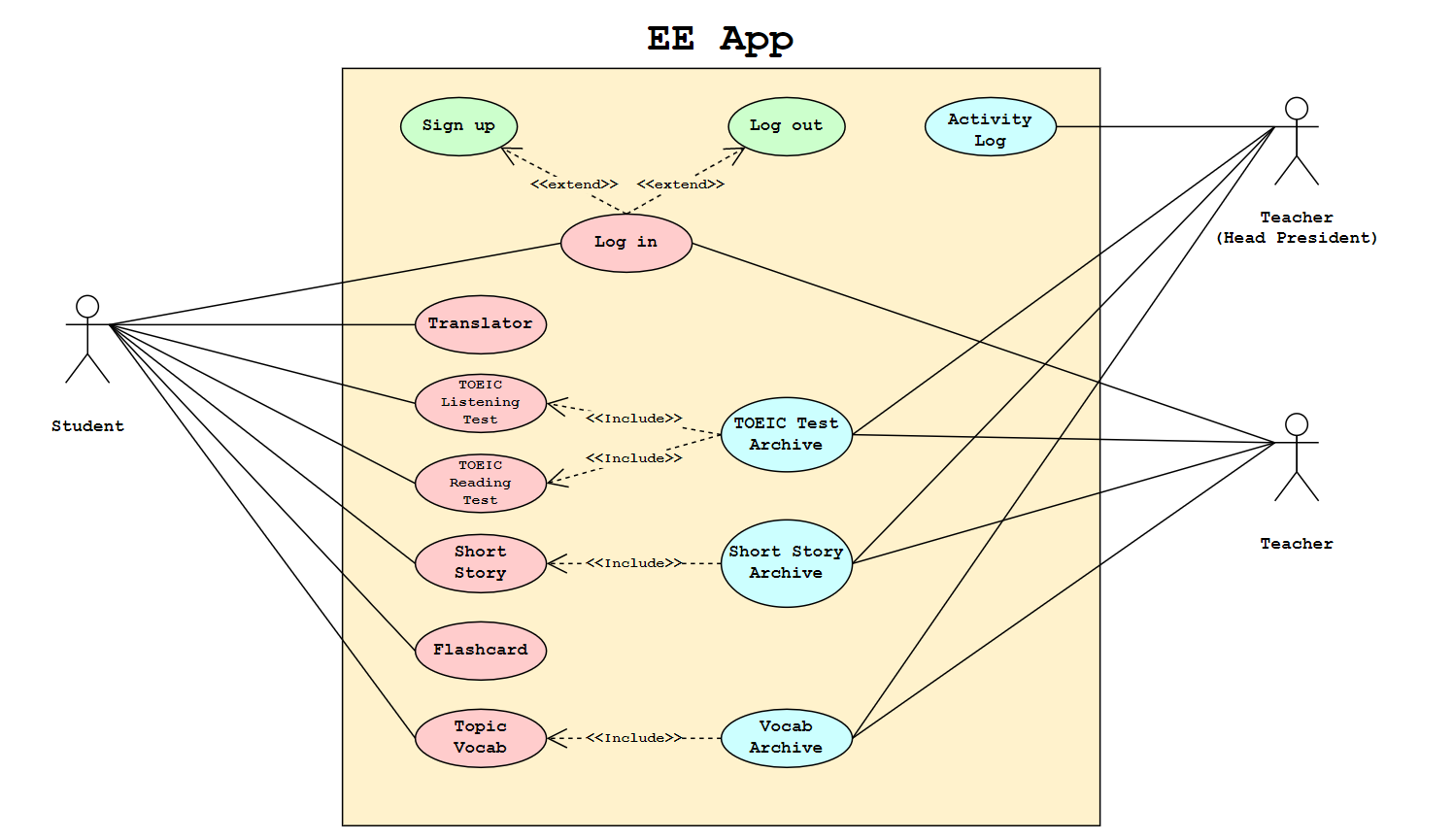
# Introduction

Software Architecture Document (SAD) cung cấp các thông tin về kiến trúc của phần mềm - được thiết kế dựa trên những ràng buộc, yêu cầu của phần mềm, Use-Case Model (UCM) và mục tiêu của kiến trúc phần mềm được đưa ra. Mục tiêu của SAD là đưa ra một cái nhìn cụ thể về tất cả các thành phần của phần mềm, cách chúng hoạt động và liên kết với nhau. SAD cũng sẽ đề cập đến cách triển khai các thành phần phần mềm và ánh xạ chúng vào bộ máy hoạt động.

# Architectural Goals and Constraints

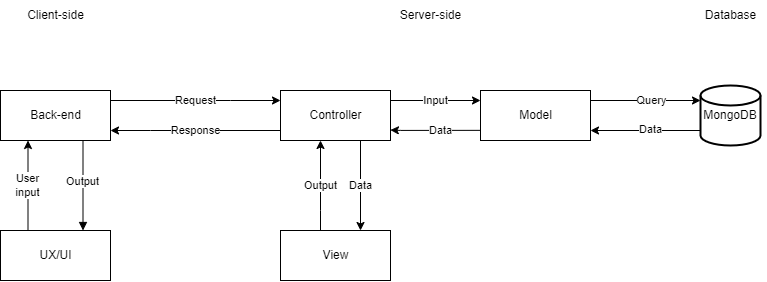
* Ngôn ngữ lập trình được sử dụng: Python.
* Môi trường hoạt động của ứng dụng: PC, laptop.
* Mã nguồn sạch, dễ bảo trì, nâng cấp.
* Khả năng xử lý lỗi phát sinh của hệ thống.
* Giới hạn một tài khoản chỉ được đăng nhập tại một thiết bị trong cùng một thời điểm.
* Mật khẩu được mã hóa trước khi được gửi về kiểm tra ở database.
* Các dữ liệu về quá trình học tập của đối tượng Student sẽ được lưu local ở client.

# Use-Case Model



# Logical View

High-level design architecture:



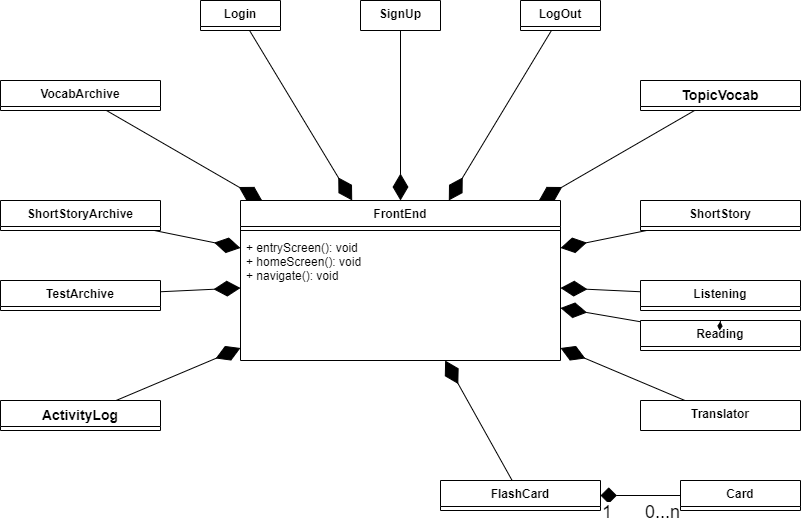
## Component: UI/UX (FrontEnd)

Thành phần UI/UX là giao diện để người sử dụng cuối sử dụng các chức năng của phần mềm. UI/UX bao gồm nhiều cửa sổ thể hiện các tính năng của phần mềm, cho phép người dùng thực hiện các sự kiện để đáp ứng các nhu cầu sử dụng tính năng trên từng cửa sổ của người dùng.

Thành phần Front-end có 13 thành phần nhỏ (composition) được chia thành hai nhóm:

* Function subsystem: Các thành phần xử lý nhập xuất, xử lỳ nghiệp vụ khi người dùng thao tác với GUI thông qua việc tương tác lên các icon chức năng trên màn hình giao diện. Bao gồm các thành phần: TopicVocab, ShortStory, Listening, Reading, Translator, FlashCard, VocalArchive, TestArchive, ShortStoryArchive, ActivityLog.
* Login/Sign up: Các thành phần xử lý đăng nhập, đăng ký. Bao gồm các thành phần: Login, SignUp, LogOut.

Class diagram:



Chi tiết từng class phân loại theo đối tượng sử dụng chức năng:

* Đối tượng chung:
  + Class FrontEnd:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Class gồm có 13 thành phần (composition) là đảm nhận các chức năng của ứng dụng.

Các phương thức của class:

* Phương thức entryScreen hiển thị giao diện ở bước ban đầu khi người dùng mở ứng dụng và lựa chọn đăng nhập hay đăng ký.
* Phương thức homeScreen hiển thị giao diện chính của ứng dụng với icon của 13 chức năng.
* Phương thức navigate dùng để chuyển đến các đối tượng của từng chức năng cụ thể sau thao tác nhấn chọn vào icon tương ứng của người dùng.
  + Class Login:

Ảnh có chứa bàn

Mô tả được tạo tự động

Các thuộc tính của class:

• userName: dùng để lưu tên tài khoản người dùng.

• passWord: dùng để lưu mật khẩu tài khoản người dùng.

Các phương thức của class:

* Khi người dùng mở ứng dụng thì hàm entryScreen() trong class FrontEnd được gọi, người dụng chọn chức năng ‘log in’ thì hàm homeScreen() của đối tượng thuộc lớp Login sẽ được gọi hiển thị giao diện login.
* Phương thức geUser() cho phép người dùng nhập user name và lưu lại vào thuộc tính username.
* Phương thức getPassword() cho phép người dùng nhập user password và lưu lại vào thuộc tính passWord.
* Phương thức login sẽ thực hiện kiểm tra thông tin đăng nhập trên database và trả về một kết quả kiểu Boolean cho biết có đăng nhập thành công hay không.
  + Class Sign up:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Các thuộc tính của class:

• userName: dùng để lưu tên tài khoản người dùng.

• passWord: dùng để lưu mật khẩu tài khoản người dùng.

• Role: dùng để lưu vai trò người dùng là student hay teacher.

Các phương thức của class:

* Khi người dùng mở ứng dụng thì hàm entryScreen() trong class FrontEnd được gọi, người dụng chọn chức năng ‘sign up’ thì hàm homeScreen() của đối tượng thuộc lớp SignUp sẽ được gọi hiển thị giao diện sign up.
* Phương thức geUser() cho phép người dùng nhập user name và lưu lại vào thuộc tính username.
* Phương thức getPassword() cho phép người dùng nhập user password và lưu lại vào thuộc tính passWord.
* Phương thức getRole() cho phép người dùng nhập vai trò của tài khoản và lưu lại vào thuộc tính role.
* Phương thức signUp sẽ thực hiện thêm thông tin tài khoản vào database nhưng sẽ kiểm tra xem thông tin tài khoản đã tồn tại chưa hoặc thỏa các yêu cầu đặt tên của ứng dụng hay chưa và trả về một kết quả kiểu Boolean cho biết có đăng ký thành công hay không.
  + Class Logout:

Ảnh có chứa bàn

Mô tả được tạo tự động

Class không có thuộc tính.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Log out’ thì hàm logout của đối tượng thuộc lớp LogOut sẽ được gọi và đăng xuất thông tin tài khoản ra khỏi ứng dụng, đồng thời trả về một biến cờ kiểu Boolean báo hiệu đăng xuất khỏi ứng dụng.
* Đối tượng Student:
  + Class TopicVocab:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Các thuộc tính của class:

• topic: dùng để lưu tên chủ đề từ vựng.

• vocab: dùng để lưu từ vựng.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng TopicVocab thì homeScreen() sẽ hiển thị ra giao diện của phần TopicVocab.
* Phương thức getListOfTopics() sẽ lấy dữ liệu vầ danh sách các chủ đề được lưu trên database và hiển thị bằng phương thức showListOfTopics().
* Phương thức getListOfVocabs() sẽ lấy danh sách từ vựng có trong topic do người dùng chọn được lưu trong thuộc tính topic và hiển thị bằng phương thức showListOfVocab().
* Phương thức getContentOfVocab() sẽ lấy tất cả nội dùng thuộc về từ vựng mà người dùng chọn được lưu trong thuộc tính vocab và hiển thị bằng phương thức showContentOfVocabs().
* Phương thức navigate() dùng để thực hiện các thao tác chuyển đổi.
  + Class ShortStory:

Ảnh có chứa bàn

Mô tả được tạo tự động

Các thuộc tính của class:

• story: dùng để lưu nội dung câu chuyện.

• listOfWords: vector dùng để lưu các từ vựng chính của câu chuyện.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ShortStory thì homeScreen() sẽ hiển thị ra giao diện của phần ShortStory.
* Phương thức getListOfStories() sẽ lấy danh sách các câu chuyện trên database về.
* Phương thức getContentOfStory() sẽ lấy phần dữ liệu về câu chuyện được lưu trên database về. Sau đó lưu nội dung câu chuyện vào thuộc tính story, lấy danh sách từ vựng và lưu vào trong thuộc tính listOfWords.
* Với listOfWords được lấy về thì hai phương thức hideListWords() và showListWords() sẽ thực hiện chức năng che hoặc là hiển thị theo thao tác yêu cầu của người dùng.
  + Class Listening/Reading:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Các thuộc tính của class:

• part: dùng lưu tên của phần test.

• numOfQuestion: số thứ tự câu hỏi.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng Reading/Listening thì homeScreen() sẽ hiển thị ra giao diện của phần Reading/Listening.
* Phương thức getListOfQuestion () sẽ lấy danh sách câu hỏi có trong thuộc tính part mà người dùng đã chọn .
* Phương thức getContentOfQuestion() lấy toàn bộ nội dung về câu hỏi có trong database và được hiển thị bằng phương thức showQuestion().
* Phương thức getContentOfAnswer() sẽ đáp án đúng của câu hỏi hiển thị bằng phương thức showAnswer().
* Khi người dùng kích chọn đáp án sẽ được lấy bằng phương thức getChoice() và kiểm tra tính đúng sai bằng phương thức checkChoice(). Khi có câu trả lời sai, câu hỏi sẽ được lưu lại bằng phương thức addToListWrongAnswers().

• Phương thức showListWrongAnswers() dùng để hiển thị danh sách câu hỏi làm sai.

* Phương thức showMenu() hiển thị bảng câu hỏi cho phép người dùng lựa chọn câu hỏi muốn đi đến và lưu trong numOfQuestion() và các thao tác bên trong nó cũng sử dụng các phương thức nêu trên.
* Phương thức navigate() dùng để thực hiện các thao tác chuyển đổi.
  + Class Translator:

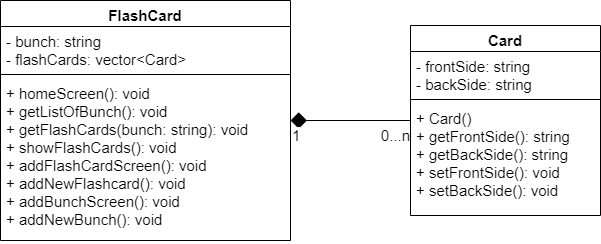
Ảnh có chứa bàn

Mô tả được tạo tự động

Class không có thuộc tính:

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Translator’ thì phương thức homeScreen của đối tượng thuộc lớp Translator sẽ được gọi và hiển thị 2 lựa chọn là Vietnamese to Enligsh hoặc là Enligsh to Vietnamese để người dùng lựa chọn và lưu lại vào biến option.
* Phương thức translate nhận biến option ở trên làm đối số và thực hiện việc dịch từ ngôn ngữ nguồn sang ngôn ngữ đích một cách phù hợp.
  + Class FlashCard:
    - Class Card:



Các thuộc tính của class FlashCard:

* bunch: dùng lưu tên của bunch (nhóm các flashcards).

• flashCards: vector lưu nội dung cho các thẻ flashcard.

Các phương thức của class FlashCard:

* Khi người dùng nhấn vào biểu tượng Flashcard thì homeScreen() sẽ hiển thị ra giao diện của phần Flashcards là danh sách các bunch.
* Phương thức getListOfBunch() sẽ lấy danh sách tên các bunch hiện có trên database để có thể hiển thị trong homeScreen(). Đồng thời có một giao diện ‘+’ ứng với chức năng thêm bunch mới. (1)
* Nếu người dùng nhấn vào một trong nững bunch hiện có thì tên của bunch được chọn sẽ được lưu vào một biến làm đối số cho hàm getFlashCards. Trong hàm getFlashCards sẽ dựa vào tên bunch được chọn để truy vấn các flashcards của bunch đó trên database.
* Hàm showFlashCards() sẽ hiển thị các flashcards được phản hồi về từ database và đồng thời cũng có giao diện ‘+’ để thêm một flashcard mới. Nếu người dùng chọn ‘+’ hàm addFlashCardScreen() sẽ được gọi hiển thị giao diện thêm thẻ mới và hàm addNewFlashCard() sẽ thực hiện chức năng thêm thẻ mới và gửi lên database.
* Nếu ở (1) người chọn thêm bunch mới thì hàm addBunchScreen() sẽ được gọi hiển thị giao diện thêm bunch mới và hàm addNewBunch() sẽ thực hiện chức năng tạo một bunch mới trên database.

Các thuộc tính của class Card:

* frontSide: chứa chuỗi ở mặt trước của Card.

• backSide: chứa chuỗi ở mặt sau của Card.

Các phương thức của class Card:

* Card(): hàm khởi tạo Card.

• getFrontSide: lấy chuỗi ở mặt trước của Card.

* getBackSide: lấy chuỗi ở mặt sau của Card

• setFrontSide: đặt chuỗi ở mặt trước của Card.

• setBackSide: đặt chuỗi ở mặt sau của Card

* Đối tượng Teacher:
  + Class VocabArchive:

Ảnh có chứa bàn

Mô tả được tạo tự động

Các thuộc tính của class:

• topic: dùng để lưu tên chủ đề từ vựng.

• listOfVocabs: vector dùng để lưu các từ vựng.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Vocab Archive’ thì homeScreen() sẽ hiển thị ra giao diện của phần Vocab Archive cho phép người dùng thêm một danh sách các từ vựng theo chủ đề mới.
* Phương thức getTopic dùng để người dùng nhập tên chủ đề và lưu vào thuộc tính topic.
* Phương thức getListOfVocabs dùng để người dùng nhập nội dung các từ vựng như là phiên âm, nghĩa tiếng việt, câu ví dụ,…. thông qua hàm getContentOfVocab() sẽ được gọi trong hàm này.
* Phương thức add sẽ thực hiện gửi cập nhật một chủ đề từ vựng mới lên database và trả về kết quả là một biến kiểu Boolean cho biết việc cập nhật có thành công hay không sau khi kiểm tra các điều kiện chẳng hạn như: tên topic bị trùng, ….
  + ShortStoryArchive:

Ảnh có chứa bàn

Mô tả được tạo tự động

Các thuộc tính của class:

• story: dùng để lưu nội dung câu chuyện.

• listOfWords: vector dùng để lưu các từ vựng chính của mẫu truyện.

• nameStory: dùng để lưu tên mẫu truyện.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Short Story Archive’ thì homeScreen() sẽ hiển thị ra giao diện của phần Short Story Archive cho phép người dùng thêm nội dung của một cậu truyện chêm mới.
* Phương thức getName dùng để người dùng nhập tên mẫu truyện và lưu vào thuộc tính nameStory.
* Phương thức getContentStory dùng để người dùng nhập nội dung mẫu truyện và lưu vào thuộc tính story.
* Phương thức getListOfWords dùng để người dùng nhập danh sách từ vựng của mẫu truyện và lưu vào thuộc tính listOfWords.
* Phương thức add sẽ thực hiện gửi cập nhật các thông tin của mẫu truyện lên database và trả về kết quả là một biến kiểu Boolean cho biết việc cập nhật có thành công hay không.
  + TestArchive:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Class không có thuộc tính.

Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Test Archive’ thì homeScreen() sẽ hiển thị ra giao diện của phần Test Archive cho phép người dùng thêm bài test vào cơ sở dữ liệu. Giao diện sẽ có 2 lựa chọn là ‘Listening Test’ hoặc ‘Reading Test’ để người dùng lựa chọn và lưu lại vào một biến option.
* Phương thức navigate sẽ nhận đối số là biến option ở trên và chuyển hướng đến các hàm addQuestions và addANewQuestion sao cho phù hợp.
* Phương thức addQuestions dùng để hiển thị lần lượt các biểu mẫu (hay có thể xem là giao diện) cho việc thêm lần lượt các câu hỏi của bài test. Và cuối cùng gửi cập nhật đến database.
* Phương thức addANewQuestion dùng để thêm nội dung cho một câu hỏi sau khi biểu mẫu đã hiện ra (do nội dung tùy thuộc vào loại test gì thì sẽ có biểu mẫu khác nhau nên hàm này có thể được thực hiện bằng cách cho phép thêm nội dung cho câu hỏi bằng cách đọc lên các file có sẵn với dịnh dạng phù hợp với mục tương ứng trong biểu mẫu, ví dụ mục audio trong biểu mẫu cho Listening Test thì sẽ cho phép tải lên file dạng âm thanh).
  + ActivityLog:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Các thuộc tính của class:

• users: dùng để lưu các user name hiện có trong hệ thống.

• passwords: dùng để lưu các user password hiện có trong hệ thống.

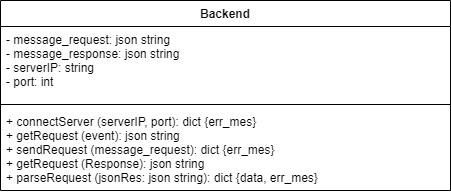
Các phương thức của class:

* Khi người dùng nhấn vào biểu tượng ‘Activity Log’ thì homeScreen() sẽ hiển thị ra giao diện của phần Activity Log cho phép người dùng thao tác với danh sách các tài khoản hiện có. Giao diện có hai lựa chọn là ‘Student Account’ và ‘Teacher Account’ cho hai loại danh sách tài khoản với vai trò khác nhau để người dùng lựa chọn và lưu lại vào biến option.
* Phương thức navigate sẽ nhận đối số là biến option ở trên và chuyển hướng đến các hàm thao tác khác một cách phù hợp với từng loại danh sách.
* Phương thức show dùng hiển thị danh sách tài khoản ngay sau khi được chuyển hướng từ navigate. Có giao diện ‘+’ để thêm tải khoản, và bên cạnh mỗi dòng tài khoản có giao diện ‘-‘ để xóa tài khoản.
* Nếu chọn ‘+’ thì addScreen được gọi hiển thị giao diện thêm lần lượt thông tin từng tài khoản mới mà trong đó 2 phương thức getUsers và getPasswords dùng để nhập các thông tin user và password của những tài khoản mới và cập nhật vào 2 thuộc tính users và passwords. Phương thức add gửi cập nhật lên database và trả về một biến kết quả kiểu Boolean cho biết thành công hay chưa.
* Nếu chọn ‘-‘ thì removeScreen được gọi hiển thị giao diện xóa, phương thức remove được gọi và truyền vào index của dòng tài khoản được nhấn ‘-‘ sau đó tiến hành xóa thông tin và trả về một biến kết quả kiểu Boolean cho biết xóa tài khoản thành công hay chưa.

## Component: Back-end (Client-side)

Thành phần Back-end của client sẽ đảm nhận nhiệm vụ xử lý những sự kiện mà người dùng thực hiện ở UI/UX và gửi những HTTP/HTTPS request về server đối với những chức năng cần đến dữ liệu được cung cấp từ Database. Đồng thời Back-end cũng nhận những HTTP/HTTPS response tương ứng với những request đã gửi đi và xử lý các trường hợp không nhận được response trong thời gian cho phép. Sau khi nhận được response, back-end sẽ biểu diễn các dữ liệu kết quả cho người dùng thông qua UI/UX.

**Ghi chú**: “dict” là kiểu dữ liệu dictionary ; “err\_mes” nghĩa là error message. “dict {data, err\_mes}” nghĩa là một dictionary có hai key là data và err\_mes.



Các thuộc tính của class:

* message\_request: tin nhắn gửi yêu cầu đến server.
* message\_response: json string sau khi parse message response từ server trả về.
* serverIP: IP của server cần kết nối đến.

• port: port mà server sử dụng.

Các phương thức của class:

* Phương thức connectServer dùng để kết nối đến server để truyền và nhận dữ liệu.
* Phương thức getRequest dùng để tạo cấu trúc cho message request sau khi nhận được yêu cầu từ người dùng.
* Phương thức sendRequest dùng để gửi message request đến server.
* Phương thức getResponse nhận vào tin nhắn phản hồi dạng json string của server trả về và lưu vào thuộc tính message\_response, sau đó sử dụng hàm parseResponse để phân tích các trường dữ liệu và tạo lại một json string hoàn chỉnh với cấu trúc rõ rang hơn cho mỗi trường dữ liệu được người dùng yêu cầu và trả về để UX/UI thể hiện giao diện cho người dùng.

Thành phần chính của các message:

* + Request message :

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Giải thích:**

- Là một json string gồm 3 fields:

* group: cho biết request thuộc nhóm nào trên server. 13 chức năng của ứng dụng sẽ được chia vào 3 nhóm để xử lý trên model của Server là: ‘Login’, ‘Query’, ‘Archive’.
* type: cho biết chính xác chức năng nào đang được request.
* data: là dictionary lưu những thông tin của tin nhắn request đến server. Chẳng hạn với chức năng login thì data sẽ là một dictionary gồm 2 cặp key-value: ‘user’: string; ‘password’: string.
  + Response message :

**Giải thích:**

- Là một json string gồm 2 fields:

* type: chức năng nào cần tin nhắn phản hồi từ server.
* data: là dictionary hay string phản hồi từ server.

Ảnh có chứa văn bản

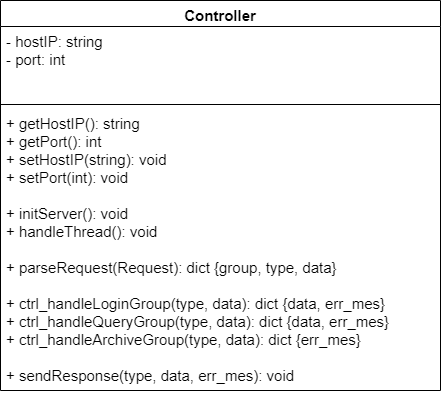
Mô tả được tạo tự động

## Component: Controller

Thành phần Controller của server sẽ tiếp nhận những HTTP/HTTPS request của client, phân tích và lấy dữ liệu từ request làm dữ liệu đầu vào cho các hàm ở thành phần model. Các hàm được sử dụng ở thành phần Model sẽ do kết quả từ việc phân tích request của Controller quyết định.

Bên cạnh việc nhận request, Controller còn nhận kết quả của những request sau khi được xử lý. Khi nhận được dữ liệu từ View, Controller sẽ tích hợp dữ liệu vào HTTP/HTTPS response và gửi lại cho Back-end của client.

Controller cũng sẽ là nơi khởi tạo server, xử lý luồng truy cập.



Các thuộc tính của class:

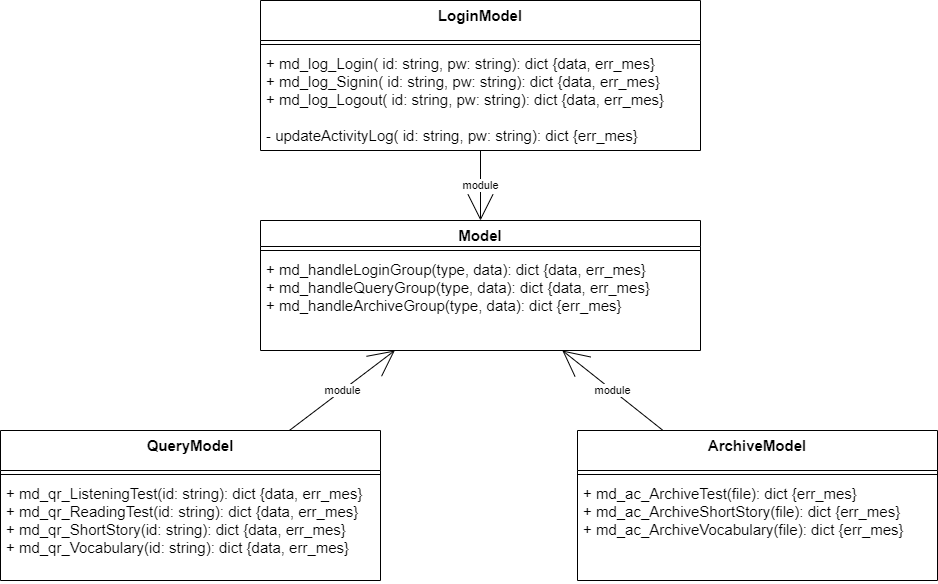
* hostIP: là địa chỉ IP của máy được chọn làm server.
* port: là port mà server chiếm trên máy.

Các phương thức của class:

* getHostIP(): trả về giá trị thuộc tính hostIP.
* getPort(): trả về giá trị thuộc tính port.
* setHostIP(string): đặt giá trị cho thuộc tính hostIP.
* setPort(int): đặt giá trị cho thuộc tính port.
* initServer(): hàm khởi tạo server với các thuộc tính thu hostIP và port.
* handleThread(): hàm xử lý đa luồng cho server.
* parseRequest(Request): hàm phân tích Request được gửi đến và trả về một dictionary chứa các thông tin: tên nhóm, loại chức năng và dữ liệu đi kèm.
* ctrl\_handleLoginGroup(type, data): hàm xử lý các Request liên quan đến các chức năng Log-in, Sign in, Log-out; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.
* ctrl\_handleQueryGroup(type, data): hàm xử lý các Request liên quan đến các chức năng phục vụ cho người dùng Student; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.
* ctrl\_handleArchiveGroup(type, data): hàm xử lý các Request liên quan đến các chức năng phục vụ cho người dùng Teacher; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.
* sendResponse(type, data, err\_mes): gửi Response cho client với các thông tin được cung cấp từ tham số đầu vào.

## Component: Model

Thành phần Model bao gồm các hàm (với dữ liệu đầu vào) thực hiện liên kết đến Database và truy vấn những dữ liệu phù hợp với yêu cầu của người dùng. Sau khi có được dữ liệu truy vấn và thực hiện xong các phép tính toán cần thiết, Model sẽ đưa dữ liệu đến thành phần View bằng các hàm phù hợp với mục đích của hàm đã được gọi trong Model để tiếp tục xử lý.



Class không có thuộc tính.

Các phương thức của class:

* md\_handleLoginGroup(type, data): hàm xử lý các chức năng Log-in, Sign in, Log-out bằng các hàm được gọi từ module LoginModel; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.
* md\_handleQueryGroup(type, data): hàm xử lý các chức năng phục vụ cho người dùng Student bằng các hàm được gọi từ module QueryModel; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.
* md\_handleArchiveGroup(type, data): hàm xử lý các chức năng phục vụ cho người dùng Teacher bằng các hàm được gọi từ module ArchiveModel; type sẽ là tên chức năng cần xử lý và data là dữ liệu đi kèm.

Các hàm của các module:

Module LoginModel:

* md\_log\_Login(id: string, pw: string): hàm xử lý chức năng Log in. Truy vấn đến database để lấy dữ liệu user name và password và so sánh với dữ liệu đầu vào. Gọi hàm updateActivityLog().
* md\_log\_SignUp(id: string, pw: string): hàm xử lý chức năng Sign in . Truy vấn đến database để lấy dữ liệu user name và password để kiểm tra tài khoản có hợp lệ hay không.
* md\_log\_Logout(id: string, pw: string): hàm xử lý log out tài khoản của user. Gọi hàm updateActivityLog()
* updateActivityLog(id: string, pw: string): hàm cập nhật các tài khoản đang kết nối đến server.

Module QueryModel:

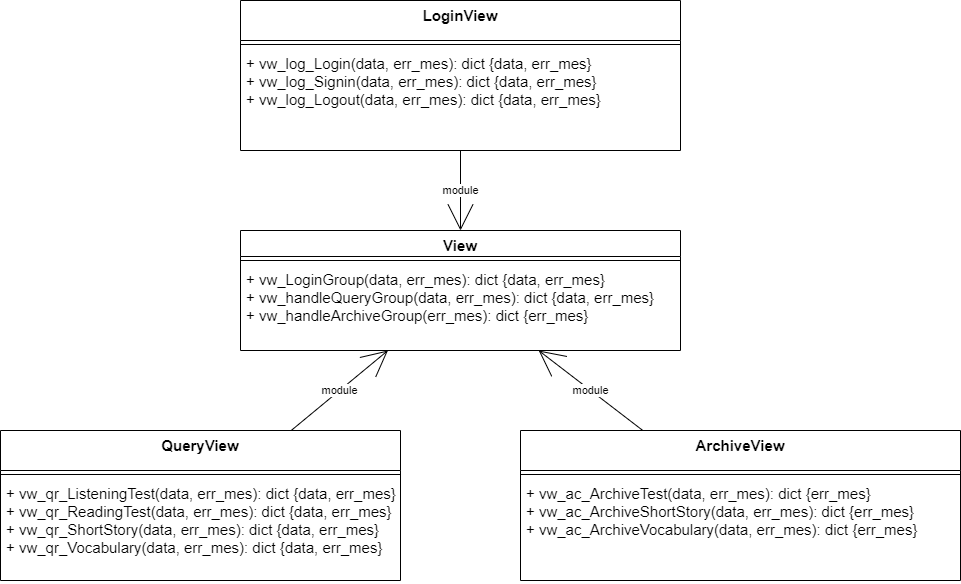
* md\_qr\_ListeningTest( id: string): hàm truy vấn dữ liệu các bài test Listening ở database.
* md\_qr\_ReadingTest( id: string): hàm truy vấn dữ liệu các bài test Reading ở database.
* md\_qr\_ShortStory( id: string): hàm truy vấn dữ liệu các short story ở database.
* md\_qr\_Vocabulary(id: string): hàm truy vấn dữ liệu các bộ từ vựng theo chủ đề ở database.

Module ArchiveModel:

* md\_ac\_ArcihveTest( file): hàm cập nhật các bài test Listening/Reading trên database.
* md\_ac\_ArchiveShortStory( file): hàm cập nhật các short story trên database.
* md\_ac\_ArchiveVocabulary( file): hàm cập nhật các bộ từ vựng theo chủ đề trên database.

## Component: View

Mục đích của thành phần View là đưa những dữ liệu đã được xử lý ở Model vào trong những khuôn mẫu có sẵn và trả lại kết quả cho Controller. View giúp cho server có thể kiểm soát được những liệu đầu ra cũng như những dữ liệu mà client nhận được sẽ luôn theo mong muốn của server, tránh việc những thông tin quan trọng vô tình bị lộ ra ngoài.



Class không có thuộc tính.

Các phương thức của class:

* vw\_handleLoginGroup(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn trả về kết quả cho các chức năng Log-in, Sign in, Log-out.
* vw\_handleQueryGroup(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn trả về kết quả cho các chức năng phục vụ cho người dùng Student.
* vw\_handleArchiveGroup(err\_mes): hàm điều chỉnh dữ liệu truy vấn trả về kết quả cho các chức năng phục vụ cho người dùng Teacher.

Các hàm của các module:

Module LoginView:

* vw\_log\_Login(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng Log-in. Thông báo lỗi nếu có err\_mes.
* vw\_log\_Signup(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng Sign in. Thông báo lỗi nếu có err\_mes.
* vw\_log\_Logout(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho các chức năng Log-out. Thông báo lỗi nếu có err\_mes.

Module QueryView:

* vw\_qr\_ListeningTest( data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng truy vấn Listening test. Thông báo lỗi nếu có err\_mes.
* vw\_qr\_ReadingTest( data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng truy vấn Reading test. Thông báo lỗi nếu có err\_mes.
* vw\_qr\_ShortStory( data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng truy vấn short story. Thông báo lỗi nếu có err\_mes.
* vw\_qr\_Vocabulary(data, err\_mes): hàm điều chỉnh dữ liệu truy vấn và trả về kết quả cho chức năng truy vấn từ vựng theo chủ đề. Thông báo lỗi nếu có err\_mes.

Module ArchievView:

* vw\_ac\_ArcihveTest( err\_mes): hàm trả thông báo chức năng chức năng thêm bài test đã hoàn thành chưa; thông báo lỗi nếu có err\_mes.
* vw\_ac\_ArchiveShortStory( err\_mes): hàm trả thông báo chức năng chức năng thêm short story đã hoàn thành chưa; thông báo lỗi nếu có err\_mes.
* vw\_ac\_ArchiveVocabulary( err\_mes): hàm trả thông báo chức năng chức năng thêm từ vựng theo chủ đề đã hoàn thành chưa; thông báo lỗi nếu có err\_mes.

## Component: Database

Thành phần Database là tập hợp các dữ liệu có cấu trúc, lưu trữ thông tin quan trọng của phần mềm, có thể được tạo mới, cập nhật, xóa bỏ. Database là một thành phần thiết yếu của phần mềm, do đó các hoạt động làm thay đổi dữ liệu trên database phải rõ ràng và được kiểm soát chặt chẽ, tránh hư hỏng và các sự cố không mong muốn.

Database gồm 5 bảng:

* Account

Ảnh có chứa bàn

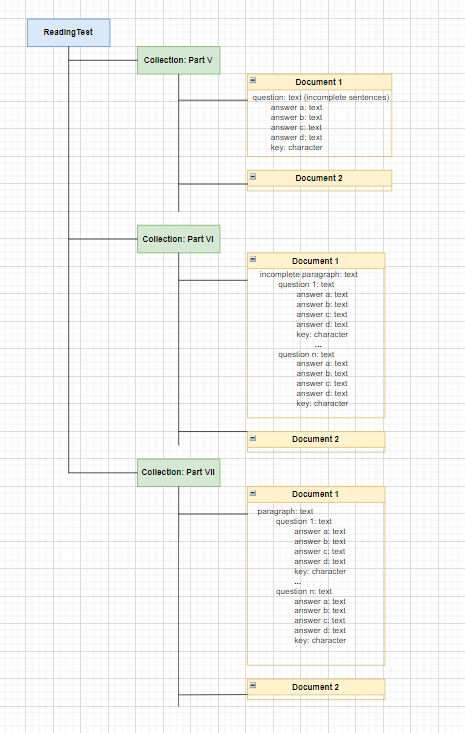
Mô tả được tạo tự động

* ListeningTest

A picture containing diagram

Description automatically generated

* ReadingTest



* ShortStoríes

Ảnh có chứa bàn

Mô tả được tạo tự động

* TopicVocab

Ảnh có chứa bàn

Mô tả được tạo tự động

Đối với dữ liệu của chức năng Flashcards, dữ liệu sẽ được lưu local ở client. Hình thức lưu trữ là file json.

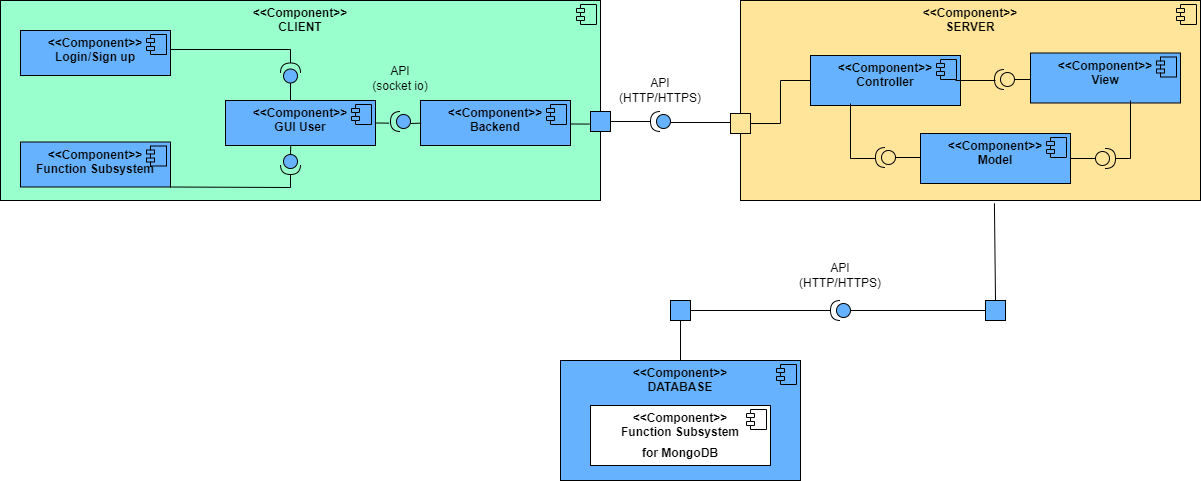
Ảnh có chứa văn bản, thiết bị điện tử

Mô tả được tạo tự động

Một flashcard trong một bunch.

# Deployment

## Overview:



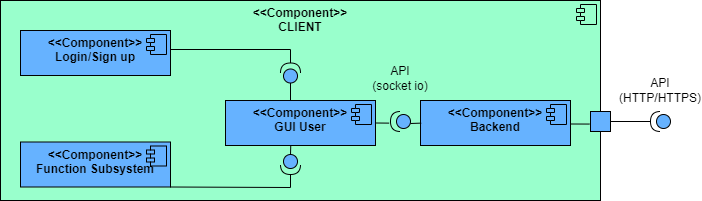
Deployment Diagram gồm 3 component lớn (3 node): CLIENT, SERVER, và DATABASE.

Mỗi node sẽ có trong đó các component nhỏ khác như là các subsystem hay các lớp chức năng khác.

Một số ký hiệu trong diagram:

|  |  |
| --- | --- |
|  | Component |
|  | Component A sử dụng interfeace từ component B |
| Ảnh có chứa văn bản, cửa shoji  Mô tả được tạo tự động | Component A giao tiếp với component B thông qua API X, X có thể là http, https, socket io, …. |

## Component: Client

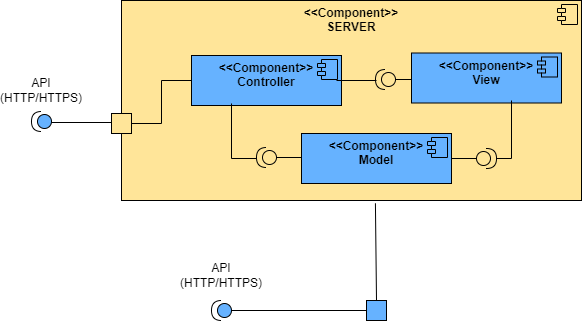


Chứa các component con như sau:

* GUI user: thành phần UX/UI dùng để giao tiếp với người dùng.
* Function subsystem: Các hàm xử lý nhập xuất, xử lỳ nghiệp vụ khi người dùng thao tác với GUI thông qua việc tương tác lên các icon chức năng trên màn hình giao diện.
* Login/Sign up: Lớp xử lý đăng nhập, đăng ký. Mục đích tách riêng khỏi Function subsystem là do chức năng này cần hiện thị một màn hình riêng biệt trước khi vào giao diện sử dụng chính và cần truy xuất đến cơ sở dữ liệu ngay từ đầu.
* Backend: Lớp này nhận các nội dung request từ Login/Sign up hoặc Function subsystem và tiếp tục xử lý nghiệp vụ, tổng hợp thành cấu trúc tin nhắn request chung mà nhóm đã trình bày ở section 4 của SAD, cuối cùng gửi đến server.
* Giao tiếp giữa các component: GUI user sử dụng interface được cung cấp từ Login/ Sign up và Function subsystem để có thể chuyển các thao tác của người dùng thành nội dung request sau đó chuyển nội dung đến backend bằng cách giao tiếp qua API socket io.

Tin nhắn request gửi từ Backend đến server và Backend nhận tin response từ server thông qua API HTTP/HTTPS, nghĩa là Component CLIENT giao tiếp với component SERVER qua API HTTP/HTTPS.

## Component: Server

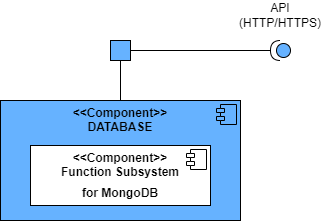


Chứa các component con như sau:

* Controller: Thành phần Controller của server sẽ tiếp nhận những HTTP/HTTPS request của client, phân tích và lấy dữ liệu từ request làm dữ liệu đầu vào cho các hàm ở thành phần model.
* Model: Thành phần Model bao gồm các hàm (với dữ liệu đầu vào) xử lý nghiệp vụ thực hiện liên kết đến Database và truy vấn những dữ liệu phù hợp với yêu cầu của người dùng.
* View: View là đưa những dữ liệu đã được xử lý ở Model vào trong những khuôn mẫu có sẵn và trả lại kết quả cho Controller.
* Controller sẽ sử dụng các interface trực tiếp từ model hoặc từ view đối với các dữ liệu cần định dạng nhất định, View sẽ sử dụng interface từ Model để tạo định dạng cho dữ liệu.

Component SERVER sẽ truy vấn đến component DATABASE thông qua API HTTP/HTTPS.

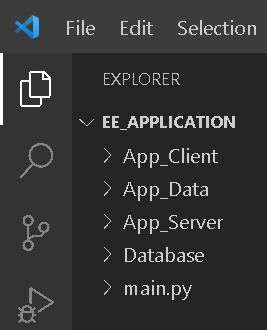
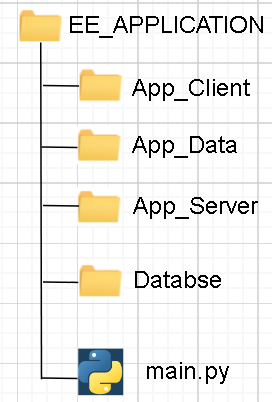
## Component: Database



Chứa các hàm xử lý nghiệp vụ với hệ quản trị cơ sở dữ liệu MongoDB.

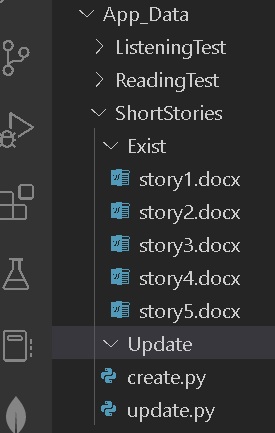
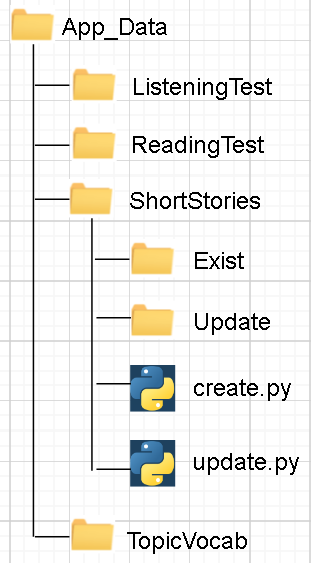
# Implementation View

Tổ chức thư mục cho việc phát triển ứng dụng học tiếng anh EE.



|  |  |
| --- | --- |
| Minh họa trên draw.io | Tổ chức trên VisualCode |

## App\_Data



|  |  |
| --- | --- |
| Minh họa trên draw.io | Tổ chức trên VisualCode |

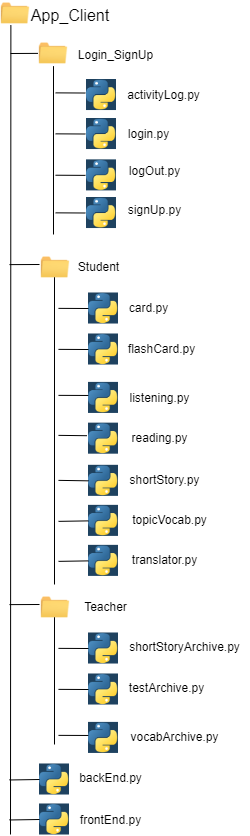
* App\_Data là thư mục chứa các tài nguyên dùng để khởi tạo cơ sở dữ liệu cho ứng dụng lúc phát triển ứng dụng của các lập trình viên.
* Đối với vai trò người dùng là teacher thì không ảnh hưởng đến thư mục này do việc cập nhật các archive của ứng dụng theo dạng dùng chức năng của các teachers thì sẽ cho người dùng đọc file từ đường dẫn mà người dùng chọn, do đó file mà người dùng muốn cập nhật lên cơ sở dữ liệu sẽ có thể nằm bấ cứ đâu trong máy người dùng.
* Trong App\_Data có 4 thư mục lớn lưu trữ 4 loại tài nguyên cho ứng dụng là Listening Test, Reading Test, Short Stories, Topic Vocab. Trong mỗi thư mục đó lại được tổ chức như sau:

Xét thư mục ShortStories:

* Thư mục Exist chứa các tài nguyên có sẵn ban đầu khi bắt đầu tạo cơ sở dữ liệu cho ứng dụng.
* Thư mục Update sẽ là nơi lập trình viên thêm các tài nguyên muốn bổ sung sau này. Khi đã thêm các tài nguyên trong thư mục Update thì sẽ chuyển tài nguyên đấy sang thư mục Exist.
* File create.py dùng để tạo bảng tương ứng cho loại tài nguyên này trên cơ sở dữ liệu.
* File update.py dùng để cập nhật tài nguyên lên cơ sở dữ liệu.

Ba thư mục lớn còn lại đươc tổ chức tương tự.

## App\_Client

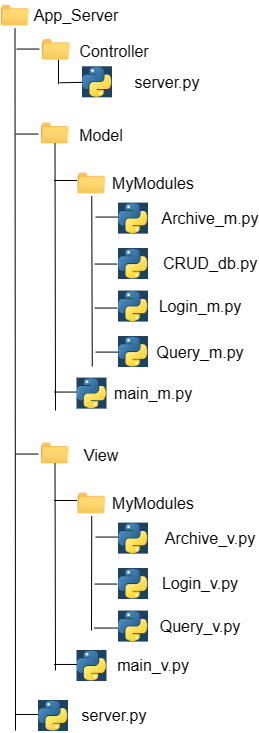
 Ảnh có chứa văn bản

Mô tả được tạo tự động

|  |  |
| --- | --- |
| Minh họa trên draw.io | Tổ chức trên VisualCode |

App\_Client là thư mục chứa các file .py là các lớp cho thành phần front end để thực hiện các giao diện người dùng, xử lý nghiệp vụ sau khi người dùng thao tác, và lớp back end để giao tiếp với server bằng các tin nhắn có cấu trúc định sẵn.

## App\_Server

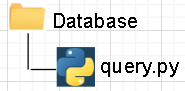
Ảnh có chứa văn bản

Mô tả được tạo tự động

|  |  |
| --- | --- |
| Minh họa trên draw.io | Tổ chức trên VisualCode |

* Chứa 3 thư mục lớn là các tổ chức cho 3 thành phần trong mô hình MVC: Model, View, Controller.
* Thư mục Controller chứa file controller.py tiếp nhận các user’s request và trả về các tin nhắn response của server. Sử dụng interface của Model và View để xử lý nghiệp vụ.
* Thư mục Model chứa các file .py xử lý nghiệp vụ cho các nhóm yêu cầu khác nhau bao gồm: nhóm login, nhóm query và nhóm archive (thay đổi tài nguyên trên cơ sở dữ liệu).
* Thư mục View tương tự cũng chứa các file .py làm nhiệm vụ tạo định dạng nhất định cho các respone theo từng nhóm request được đề cập ở trên.

## Database

Ảnh có chứa văn bản

Mô tả được tạo tự động

|  |  |
| --- | --- |
| Minh họa trên draw.io | Tổ chức trên VisualCode |

Chứa file query.py có chứa các hàm đặc thù cho việc tương tác với MongoDB để truy vấn dữ liệu.

## Main.py

File thực thi chạy chương trinh tổng.