

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



DỰ ÁN CƠ SỞ LẬP TRÌNH
Hệ thống Đăng ký tín chỉ

Người hướng dẫn: **TS.Đặng Thiên Bình**

Sinh viên thực hiện:

Phạm Trung Hiếu	21TCLC_NHAT1	102210034
Châu Diễm Hoàng	21TCLC_NHAT1	102210036
Nguyễn Trương Anh Minh	21TCLC_NHAT1	102210040

Đà Nẵng, 12/2022

MỤC LỤC

Danh mục hình vẽ.....	2
1. GIỚI THIỆU ĐỀ TÀI.....	3
1.1. Đặt vấn đề.....	3
1.2. Mô tả bài toán.....	3
2. PHÂN TÍCH CHỨC NĂNG HỆ THỐNG.....	3
2.1. GET URL	4
2.2. POST URL	4
2.3. DELETE URL	5
3. THIẾT KẾ CẤU TRÚC DỮ LIỆU	5
4. PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG..	7
4.1. Cấu trúc hệ thống hướng đối tượng	7
4.1.1. Backend (SQLite3 + Django).....	8
4.1.1.1. Cài đặt các đối tượng.....	8
4.1.1.2. Cài đặt Database quản lí dữ liệu cho các đối tượng	13
4.1.2. Frontend (Angular)	16
4.2. Kết quả.....	17
4.2.1. Trang chủ và trang đăng nhập	17
4.2.2. Trang đăng ký học	21
4.2.3. Trang thời khoá biểu.....	23
4.2.4. Trang cá nhân	24
4.3. Nhận xét.....	24
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	25
TÀI LIỆU THAM KHẢO	25

DANH MỤC HÌNH VẼ

Hình 1. Sơ đồ cơ bản của hệ thống.....	5
Hình 2. Mô hình thiết kế MVC	6
Hình 3. Quan hệ giữa các đối tượng.....	8
Hình 4. Iobject.....	8
Hình 5. Lược đồ quan hệ của UserManager.....	10
Hình 6. Lược đồ quan hệ của ClassManager.....	12
Hình 7. Lược đồ quan hệ đầy đủ giữa các đối tượng (Backend).....	13
Hình 8. Lớp template Database cơ sở BaseDatabase	14
Hình 9. Hệ thống quản lí Database.....	15
Hình 10. Trang chủ.....	18
Hình 11. Trang đăng nhập	19
Hình 12. Trang đăng nhập lỗi.....	19
Hình 13. Trang chủ sau khi đăng nhập.....	20
Hình 14. Thanh điều hướng sau khi đăng nhập.....	21
Hình 15. Trang đăng ký tín chỉ khi chưa đăng ký	21
Hình 16. Trang đăng ký tín chỉ khi đăng ký.....	22
Hình 17. Trang đăng ký khi không tồn tại lớp đăng ký	23
Hình 18. Trang thời khoá biểu.....	23
Hình 19. Trang cá nhân	24

1. GIỚI THIỆU ĐỀ TÀI

1.1. Đặt vấn đề

Ứng dụng tiến bộ khoa học công nghệ và công nghệ thông tin vào hoạt động quản lý đào tạo, đăng ký học tín chỉ ở các trường đại học giúp bao quát toàn bộ hoạt động của trường một cách chính xác. Website đăng ký học tín chỉ như một “cánh tay đắc lực” hỗ trợ cán bộ lãnh đạo, quản lý nắm bắt đầy đủ và chi tiết những thông tin cần thiết trong mọi thời điểm của các sinh viên cũng như giảng viên trong quá trình học tập. Đồng thời, làm giảm những phiền hà, góp phần cải thiện chất lượng đào tạo, quản lý.

Nhất là trong thời buổi hiện nay, hầu hết các trường đại học, cao đẳng đều thực hiện học theo hình thức tín chỉ. Việc sử dụng website quản lý đào tạo tín chỉ sẽ giúp sinh viên chủ động trong việc tìm lớp, chọn môn và đăng ký học tín chỉ theo nhu cầu, giúp tối đa hóa hiệu quả dạy – học, tiết kiệm chi phí và thời gian cho người học – người dạy, giảm nhân lực giám sát tiếp nhận đăng ký lớp, tạo sự tiện lợi và nhanh chóng cho cả học viên và ban quản lý đào tạo.

Theo xu hướng đó, Đại học Bách Khoa – đại học Đà Nẵng cũng đang đào tạo theo tín chỉ. Việc đăng ký theo tín chỉ hiện nay được đại học Bách Khoa – đại học Đà Nẵng áp dụng tuy rất tốt nhưng vẫn còn có giao diện đơn giản, chưa hoàn thiện.

1.2. Mô tả bài toán

Bài toán đặt ra lúc này là dựa vào các kiến thức về các môn học ở kỳ 3, đặc biệt là *Lập trình hướng đối tượng*, tạo ra một website với giao diện đẹp và dễ sử dụng hơn và có các chức năng cơ bản của trang *Đăng ký tín chỉ* của trường đại học Bách Khoa – đại học Đà Nẵng.

Cụ thể, *Hệ thống đăng ký tín chỉ* có các chức năng sau:

- Hệ thống cho phép sinh viên đăng ký, huỷ đăng ký các học phần.
- Hệ thống giúp sinh viên biết được học phần nào mình đã đăng ký và học phần nào mình còn có thể đăng ký được.
- Hiện thị thời khoá biểu theo tuần, nhấn mạnh theo ngày để sinh viên có thể theo dõi thuận tiện trong quá trình học tập.

2. PHÂN TÍCH CHỨC NĂNG HỆ THỐNG

Chức năng của hệ thống sẽ được xử lý thông qua Backend. Phần này sẽ được thông qua các đường link và cụ thể là qua các API được lập trình sẵn để Frontend có thể lấy và gửi request về Backend.

Hệ thống sử dụng 3 request để xử lý chính, đó là GET, POST và DELETE.

- GET : được sử dụng để lấy thông tin từ sever theo URL đã cung cấp.
- POST : gửi thông tin tới sever thông qua các biểu mẫu HTTP.
- DELETE : xóa tài nguyên trên server.

2.1. GET URL

URL	Giải thích tham số	Mục đích
<code>/api/user/login?id=[id]&pass=[pass]</code>	<ul style="list-style-type: none">- Tham số <i>id</i>: tên tài khoản của sinh viên (mã số sinh viên)- Tham số <i>pass</i>: mật khẩu ứng với tài khoản sinh viên, mật khẩu khởi tạo mặc định cũng là mã số sinh viên.	Được sử dụng trong trang đăng nhập của hệ thống.
<code>/api/user/student</code>		Nhận thông tin mọi sinh viên. Được sử dụng để liệt kê sinh viên trong trang chủ.
<code>/api/user/student/:id</code>	Tham số <i>id</i> : mã số sinh viên	Nhận thông tin của sinh viên với mã số sinh viên là <i>id</i> . Được sử dụng trong trang cá nhân.
<code>/api/classection?sid=[studentid]</code>	Tham số <i>[studentid]</i> : mã số sinh viên.	Nhận thông tin lớp học phần mà sinh viên với mã số sinh viên là <i>sid</i> đã đăng ký.
<code>/api/classection?sid=[studentid]&mode=register</code>	Tham số <i>[studentid]</i> : mã số sinh viên.	Nhận thông tin lớp học phần mà sinh viên với mã số sinh viên là <i>sid</i> có thể đăng ký được.

2.2. POST URL

Hệ thống chỉ sử dụng một POST URL duy nhất với đường dẫn là:

`/api/classection/:id (body: sid=[studentid])`

URL này để thêm sinh viên có mã số sinh viên là *sid* vào lớp học phần có mã học phần là *id*. Phần body của URL là phần thông tin thêm cần gửi cho URL mà POST URL yêu cầu.

2.3. DELETE URL

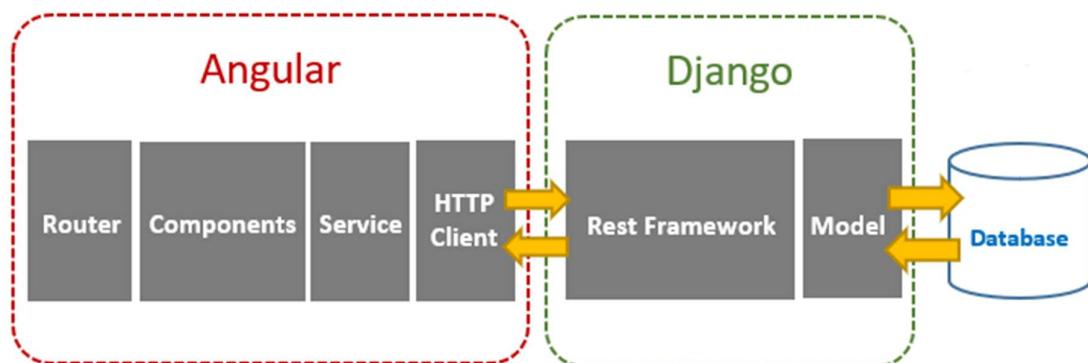
Hệ thống cũng chỉ sử dụng một DELETE URL duy nhất với đường dẫn là:

`/api/classection/:id (body: sid=[studentid])`

URL này để xóa sinh viên có mã số sinh viên là *sid* khỏi lớp học phần có mã học phần là *id*. Tương tự như POST URL, phần body của URL là phần thông tin thêm mà DELETE URL yêu cầu.

3. THIẾT KẾ CẤU TRÚC DỮ LIỆU

Hệ thống sẽ được chia làm 2 phần: Backend (cung cấp những API ở trên) và Frontend (sử dụng những API ở trên để xây dựng giao diện cho người dùng tương tác). Trong đề tài này, về phần Backend, SQLite3 sẽ được sử dụng cho phần cơ sở dữ liệu (Database), và hệ thống Backend chính sẽ được xây dựng từ Framework Django trong ngôn ngữ lập trình Python. Và phần Frontend sẽ được xây dựng từ Angular. REST Framework sẽ được sử dụng để làm phương thức giao tiếp giữa Django (Backend) và Angular (Frontend).



Hình 1. Sơ đồ cơ bản của hệ thống

Giới thiệu sơ về Django (Python)

Django là một **framework** bậc cao của Python có thể thúc đẩy việc phát triển phần mềm thần tốc và clean, thiết kế thực dụng. Được xây dựng bởi nhiều lập trình viên kinh nghiệm, Django tập trung lớn những vấn đề phát triển Web, bạn có thể phát triển

trang web của bạn mà không cần xây dựng từ những căn bản. Đặc biệt nó **free** và **open source**.

Django được nhiều tài liệu, đặc biệt là Django Book giới thiệu rằng nó sử dụng mô hình MVC. Django có cách triển khai cách triển khai mô hình MVC hơi "dị". Bởi vì "C" chính là bản thân framework Django, và phần lớn các lập trình viên chỉ làm việc với Model, Template và View, nên Django thường được hiểu là sử dụng mô hình **MVT**.

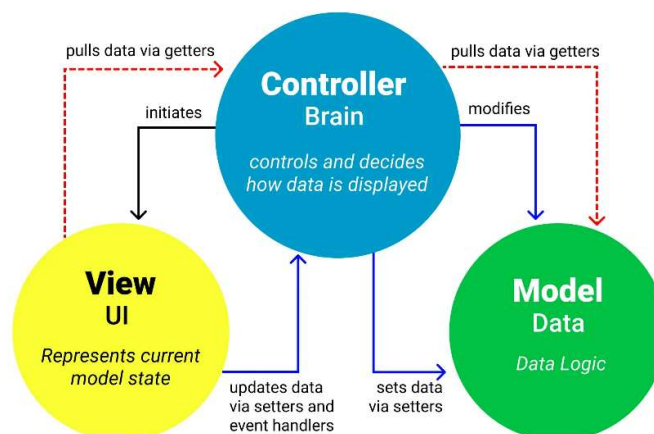
Trước khi giới thiệu về **MVT**, ta cần tìm hiểu sơ về mô hình thiết kế **MVC**.

MVC là viết tắt của Model - View - Controller. Là một mô hình thiết kế phần mềm thường được sử dụng để thiết kế phần mềm có giao diện người dùng. MVC tuy là một mô hình thiết kế phần mềm nhưng nó cũng được sử dụng rộng rãi trong web.

Mô hình MVC gồm 3 thành phần chính là:

Model	View	Controller
Là một dạng mẫu dữ liệu, có nhiệm vụ thao tác với cơ sở dữ liệu, nghĩa là nó sẽ chứa tất cả các hàm, các phương thức truy vấn trực tiếp với dữ liệu và controller sẽ thông qua các hàm, phương thức đó để lấy dữ liệu rồi gửi qua View.	Là các giao diện người dùng, có nhiệm vụ tiếp nhận dữ liệu từ controller là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh... nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.	Là các hành vi, hành động, xử lý của hệ thống đóng vai trò trung gian giữa Model và View. Nó có nhiệm vụ tiếp nhận yêu cầu từ client sau đó xử lý request, load model tương ứng và gửi data qua view tương ứng rồi trả kết quả về cho client.

MVC Architecture Pattern



Hình 2 Mô hình thiết kế MVC

Quay lại với mô hình MVT của Django, mô hình MVT cũng gồm 3 thành phần chính:

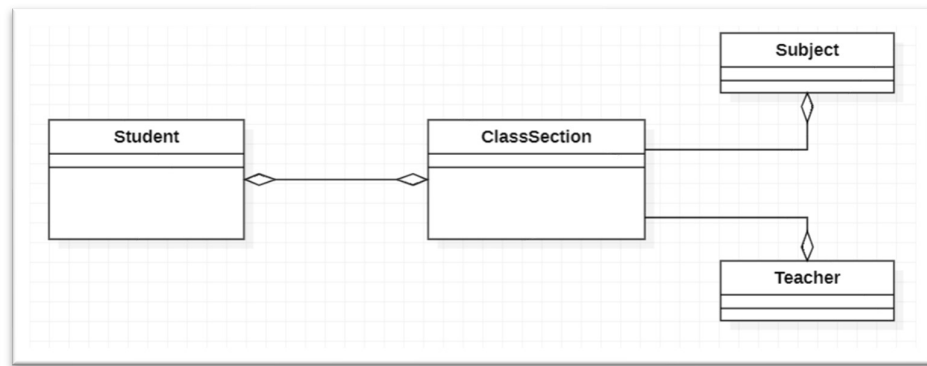
Model	Template	View
Là mô phỏng của dữ liệu. Nó không thực sự là dữ liệu, nhưng nó là một thể hiện của dữ liệu và là nơi để chúng ta thao tác với dữ liệu thật sự. Model cho phép chúng ta lưu dữ liệu vào DB và không cần hiểu những hoạt động sâu xa bên dưới. Hơn nữa, model cung cấp cho chúng ta cách thức thao tác với DB rất đơn giản, khiến cho một model có thể sử dụng với rất nhiều DB khác nhau.	Là những gì người dùng nhìn thấy. Nó là sự thể hiện của dữ liệu đối với người dùng. Nói một cách văn hoa, nó là sự thể hiện của Model. Trong ứng dụng Web, nó chính là những gì người dùng nhìn thấy trên trình duyệt. Với ứng dụng khác, nó là những gì họ nhìn thấy trên giao diện của ứng dụng. Ngoài ra, View còn cung cấp cho chúng ta phương thức để thu thập dữ liệu từ người dùng.	Dùng để điều khiển luồng thông tin dữ Model và View. Nó được sử dụng để cài đặt các login về việc lấy dữ liệu từ DB thông qua Model và chuyển sang View. Nó cũng là nơi xử lý những truy vấn từ người dùng thông qua View và thực hiện các logic khác: thay đổi View, cập nhật dữ liệu thông qua Model.

Chúng ta có thể hiểu nôm na là phần Template của mô hình MVT chính là phần View trong mô hình MVC, và phần View của MVT chính là phần Controller của MVC.

4. PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG

4.1. Cấu trúc hệ thống hướng đối tượng

Hệ thống đăng kí tín chỉ hoạt động xoay quanh 4 đối tượng chính: Sinh viên (Student), Lớp học phần (ClassSection), Môn học (Subject).



Hình 3. Quan hệ giữa các đối tượng

Các đối tượng này sẽ được cài đặt ở cả 2 phần: Backend và Frontend.

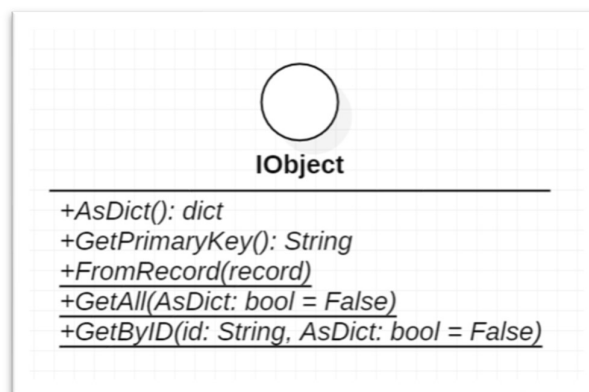
4.1.1. Backend (SQLite3 + Django)

Phần cài đặt sẽ được chia ra làm 2 phần: Cài đặt các lớp cho đối tượng và cài đặt hệ thống quản lý dữ liệu cho các đối tượng (database)

4.1.1.1. Cài đặt các đối tượng

Interface của các đối tượng: IObject

Để thuận tiện hơn trong việc tương tác với một đối tượng với các hệ thống khác, interface IObject sẽ được cài đặt như sau:



Hình 4. IObject

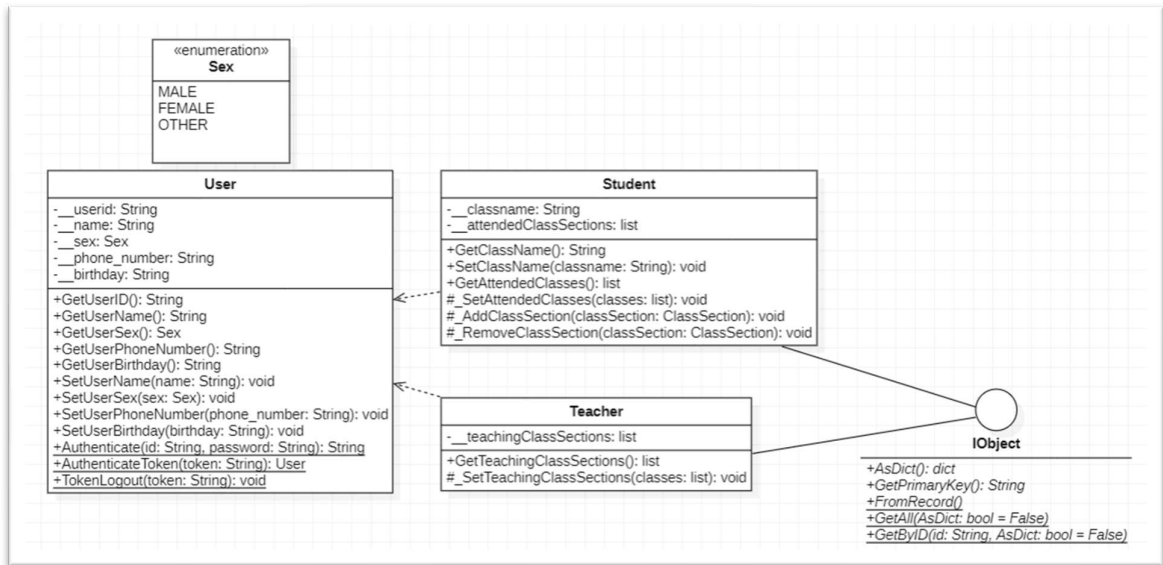
AsDict()

Phương thức này sẽ chuyển một đối tượng thành một từ điển, hay có thể coi là đối tượng JSON với Key là tên thuộc tính ở dạng xâu và Value là giá trị của thuộc tính đó. Việc chuyển đổi này sẽ giúp việc tương tác giữa Backend và Frontend dễ dàng hơn qua JSON.

<i>GetPrimaryKey()</i>	Phương thức trả về xâu được coi khoá chính của một đối tượng. Ở hệ thống này khoá chính của một đối tượng là ID của đối tượng đó.
<i>FromRecord(record)</i>	Phương thức trả về một đối tượng từ một bản ghi (record) được lấy từ Database. Tham số: record : bản ghi của đối tượng
<i>GetAll(AsDict: bool)</i>	Phương thức trả về một danh sách tất cả đối tượng được đọc từ Database. Tham số: ASDict : nếu bằng True, phương thức sẽ trả về danh sách các đối tượng ở dạng từ điển.
<i>GetByID(id, AsDict)</i>	Phương thức trả về một đối tượng được đọc từ Database theo id truyền vào. Tham số: id : ID của đối tượng cần lấy từ Database ASDict : nếu bằng True, phương thức sẽ trả về đối tượng ở dạng từ điển.

User (bao gồm Student và Teacher)

Lớp User là lớp cơ sở của 2 lớp Student và Teacher, lưu trữ và quản lí những thông tin cơ bản của người dùng trên hệ thống.



Hình 5. Lược đồ quan hệ của UserManager

Lớp User lưu trữ thuộc tính ID, họ tên, giới tính, số điện thoại, ngày sinh của người dùng. Các thuộc tính đều có mức truy cập là private và đều có các cặp phương thức Get, Set cho mỗi thuộc tính ở mức truy cập public (Trừ phương thức Set cho ID).

Ngoài ra lớp User còn có các phương thức static phục vụ cho thao tác với người dùng:

*Authenticate(id,
password)*

Phương thức nhận ID của người dùng và mật khẩu từ người dùng nhập từ Frontend truyền xuống, sau đó kiểm tra ID và mật khẩu có khớp hay không trong Database.

- Nếu ID và mật khẩu khớp, phương thức trả về một xâu có độ dài 256 kí tự làm login token để lưu vào Database và vào kho lưu trữ cục bộ của trình duyệt (local browser storage) của người dùng.

- Nếu ID và mật khẩu không khớp, phương thức trả về một xâu rỗng.

Tham số:

- *id*: ID của người dùng nhập vào

- *password*: mật khẩu của người dùng nhập vào

<i>AuthenticateToken(token)</i>	<p>Phương thức nhận token được truyền xuống từ local browser storage của người dùng và kiểm tra xem token có tồn tại trong Database hay không.</p> <ul style="list-style-type: none"> - Nếu token tồn tại trong Database, phương thức sẽ trả về ID người dùng tương ứng với login token. - Nếu token không tồn tại, hàm sẽ trả về xâu rỗng. <p>Tham số <i>token</i>: login token trong local browser storage của người dùng.</p>
<i>TokenLogout(token)</i>	<p>Phương thức nhận token được truyền xuống từ local browser storage của người dùng và kiểm tra xem token có tồn tại trong Database hay không.</p> <ul style="list-style-type: none"> - Nếu token tồn tại trong Database, phương thức sẽ xóa token đó khỏi Database. <p>Tham số <i>token</i>: login token trong local browser storage của người dùng.</p>

Lớp Student kế thừa trực tiếp từ lớp User, và định nghĩa các phương thức trong Interface IObject. Ngoài ra lớp Student có thêm thuộc tính *__classname* (Tên lớp sinh hoạt), *__attendedClassSections* (Danh sách lớp học phần đã đăng kí) ở mức truy cập private và cặp phương thức Get, Set cho nó ở mức truy cập public.

Ngoài ra, lớp Student còn có thêm các phương thức dùng để phục vụ cho các thao tác của sinh viên đối với lớp học phần:

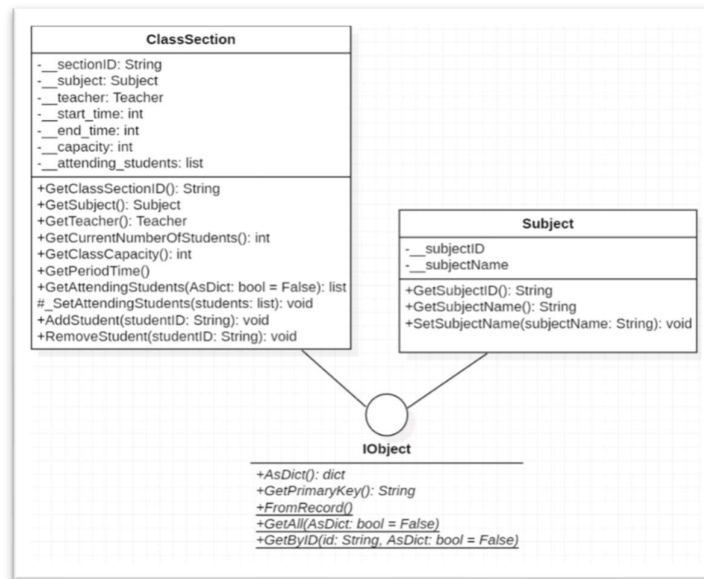
<i>_AddClassSection(classSection)</i>	<p>Phương thức thêm một lớp học phần vào danh sách lớp học phần đã đăng kí của đối tượng sinh viên (Student).</p> <p>Tham số <i>classSection</i>: đối tượng lớp học phần</p>
<i>_RemoveClassSection(classSection)</i>	<p>Phương thức loại bỏ một lớp học phần ra khỏi danh sách lớp học phần đã đăng kí của đối tượng sinh viên (Student).</p>

Tham số *classSection*: đối tượng lớp học phần

Tương tự, lớp Teacher được kế thừa trực tiếp từ lớp User và định nghĩa các phương thức trong Interface IObject, ngoài ra có thêm thuộc tính *__teachingClassSections* (Lớp học phần đang dạy) và cặp phương thức Get, Set cho nó.

Subject và ClassSection

Hai lớp Subject và ClassSection lần lượt lưu thông tin của môn học và lớp học phần, được cài đặt theo lược đồ sau:



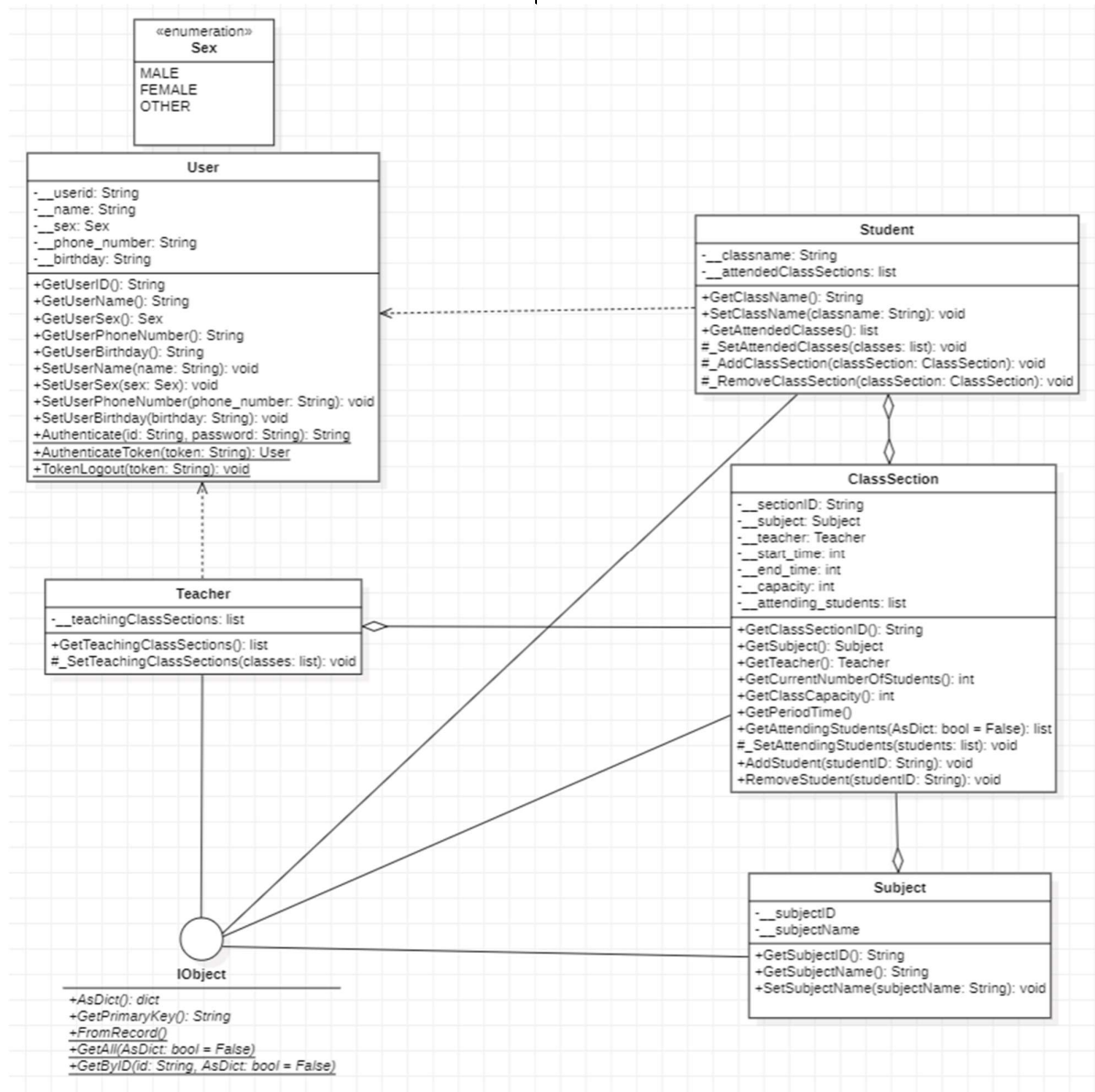
Hình 6. Lược đồ quan hệ của ClassManager

Lớp Subject được cài đặt theo Interface IObject của hệ thống hiện tại chỉ có 2 thuộc tính cơ bản ở mức truy cập private: *__subjectID* và *__subjectName* lần lượt là ID môn học và tên môn học. Theo sau đó là phương thức Get cho *__subjectID* và cặp phương thức Get, Set cho *__subjectName*.

Lớp ClassSection cũng được cài đặt theo Interface IObject, gồm các thuộc tính lưu trữ ID, môn học, giảng viên, thời gian biểu, số lượng sinh viên tối đa, danh sách sinh viên của lớp học phần. Ở lớp này chỉ có những phương thức Get cho mỗi thuộc tính (Trừ thuộc tính danh sách sinh viên có thêm hàm Set), ngoài ra còn có thêm các phương thức để đối tượng lớp học phần tương tác với lớp Student:

GetCurrentNumberOfStudents() | Hàm trả về số lượng sinh viên hiện tại đã đăng kí của lớp học phần.

<i>AddStudent(studentID)</i>	<p>Thêm một đối tượng sinh viên vào lớp học phần.</p> <p>Tham số <i>studentID</i>: ID của sinh viên</p>
<i>RemoveStudent(studentID)</i>	<p>Xoá một đối tượng sinh viên ở trong lớp học phần.</p> <p>Tham số <i>studentID</i>: ID của sinh viên</p>

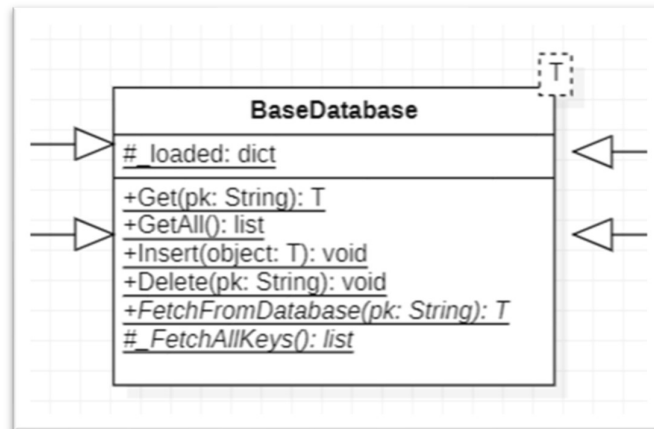


Hình 7. Lược đồ quan hệ đầy đủ giữa các đối tượng (Backend)

4.1.1.2. Cài đặt Database quản lý dữ liệu cho các đối tượng

Sau khi cài đặt các lớp cho các đối tượng chính, ta cần có một hệ thống Database để lưu trữ thông tin cho các đối tượng và phục vụ cho các đối tượng đó thông qua

Interface IObject. Nhận thấy rằng, tất cả các đối tượng IObject ở trên đều tương tác với Database thông qua khoá chính của đối tượng và có các thao tác giống nhau, do đó ta chỉ cần định nghĩa một lớp template Database cho các lớp Database của các đối tượng trên. Lớp template Database mà hệ thống sử dụng là lớp BaseDatabase được cài đặt như sau:



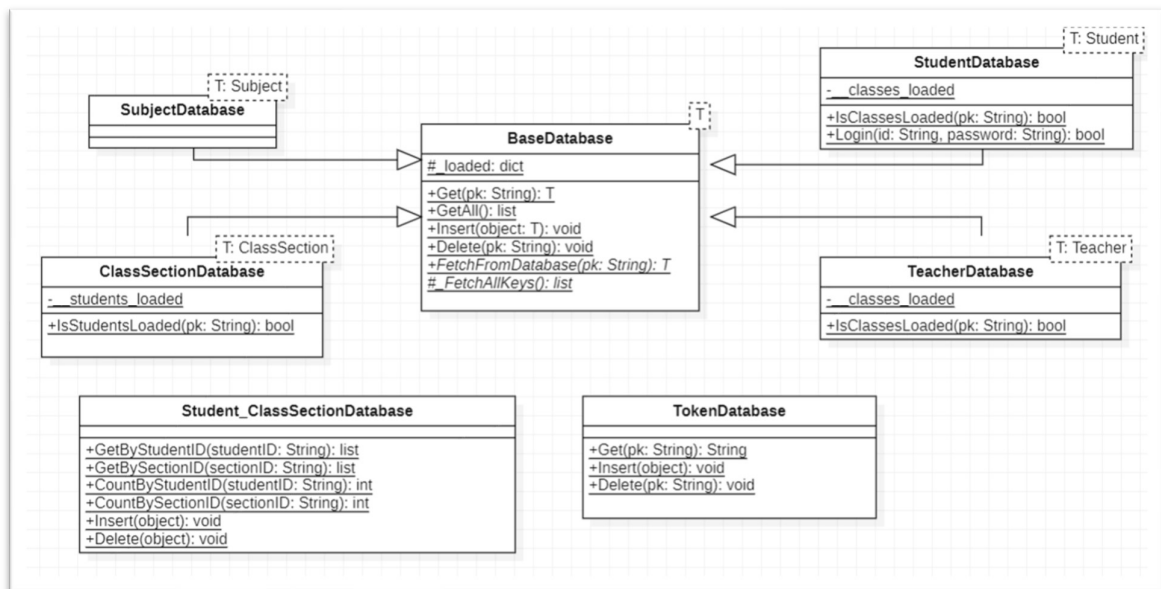
Hình 8. Lớp template Database cơ sở BaseDatabase

Lớp BaseDatabase gồm có duy nhất một thuộc tính `_loaded` ở mức truy cập protected. Thuộc tính `_loaded` là một từ điển có Key là khoá chính của đối tượng được định nghĩa trên Interface IObject, và Value là chính đối tượng đó. Thuộc tính `_loaded` dùng để lưu trữ các đối tượng đã được tải lên thông qua truy vấn đến Database của SQLite3. Các thao tác truy vấn từ Frontend sẽ được thực hiện trên các đối tượng được lưu trữ trong `_loaded` thông qua các phương thức static:

<i>Get(pk)</i>	<p>Phương thức trả về đối tượng có khoá chính là xâu <i>pk</i> được truyền vào từ <code>_loaded</code>. Nếu ở <code>_loaded</code> phương thức <code>FetchFromDatabase()</code> sẽ được gọi để tải đối tượng từ Database SQLite3.</p> <p>Tham số:</p> <p><i>pk</i>: khoá chính của đối tượng</p>
<i>GetAll()</i>	<p>Phương thức trả về danh sách các đối tượng có trong Database SQLite3.</p>
<i>Insert(object)</i>	<p>Chèn một đối tượng vào <code>_loaded</code>.</p> <p>Tham số:</p> <p><i>object</i>: đối tượng được chèn</p>

<i>Delete(pk)</i>	<p>Xoá một đối tượng có khoá chính là <i>pk</i> ra khỏi <i>_loaded</i>.</p> <p>Tham số:</p> <p><i>pk</i>: khoá chính của đối tượng bị xoá</p>
<i>FetchFromDatabase(pk)</i>	<p>Phương thức trực tiếp thực hiện truy vấn lên Database của SQLite3 và tải đối tượng vào <i>_loaded</i>. Sau đó trả về đối tượng đó.</p> <p>Tham số:</p> <p><i>pk</i>: khoá chính của đối tượng</p>
<i>_FetchAllKeys()</i>	<p>Phương thức trả về danh sách tất cả các khoá chính của đối tượng có trong Database của SQLite3.</p>

Sau khi cài đặt xong lớp Database cơ sở BaseDatabase, các lớp Database của từng đối tượng chỉ cần kế thừa lớp BaseDatabase và định nghĩa các phương thức truy vấn *FetchFromDatabase(pk)* và *_FetchAllKeys()* để có các chức năng thao tác trên Database của SQLite3.



Hình 9. Hệ thống quản lý Database

Đối với các lớp StudentDatabase, TeacherDatabase, ClassSectionDatabase, đối tượng của mỗi lớp sẽ được tải theo 2 chế độ:

1. Chỉ tải lên đối tượng

2. Tải lên đối tượng và các đối tượng ở lớp có quan hệ với đối tượng được tải.

Việc tải lên đối tượng theo 2 chế độ này hạn chế được việc sử dụng tài nguyên bộ nhớ quá nhiều, Database chỉ tải lên những đối tượng liên quan chỉ khi được truy vấn tới (như là khi gọi phương thức `GetAttendedClasses()` của đối tượng `Student` đang được tải lên ở chế độ 1, lúc này đối tượng `Student` sẽ được chuyển sang chế độ tải lên thứ 2 và các đối tượng lớp học phần lúc này mới được tải lên từ Database).

Để phục vụ cho việc tải lên đối tượng theo 2 chế độ, tại các lớp `StudentDatabase`, `TeacherDatabase`, `ClassSectionDatabase` có thêm các thuộc tính đánh sách như: `__classes_loaded`, `__student_loaded` dùng để lưu khoá chính của các đối tượng đã được tải lên theo chế độ 2 và các phương thức `IsClassesLoaded()`, `IsStudentsLoaded()` để kiểm tra xem đối tượng đã được tải lên theo chế độ 2 hay chưa.

Hiện tại Database còn khá nhỏ nên việc xoá đi những đối tượng ở `_loaded` để tiết kiệm bộ nhớ chưa được thực hiện.

Ngoài những lớp Database của các đối tượng, còn có lớp Database: `Student_ClassSectionDatabase` (dùng để lưu 2 khoá ngoại: ID của `Student` và ID của lớp học phần, thể hiện việc đăng kí lớp học phần của sinh viên) và `TokenDatabase` (dùng để lưu những login token của người dùng).

4.1.2. Frontend (Angular)

```
export interface Class {  
  sectionID: string;  
  subjectName: string;  
  teacherName: string;  
  startTime: number;  
  endTime: number;  
  current: number;  
  capacity: number;  
}
```

- `sectionID`: kiểu xâu - mã lớp học phần.
- `subjectName`: kiểu xâu - tên lớp học phần.

```
export interface Student {  
  studentId: Number;  
  name: String;  
  sex: Boolean;  
  class: String;  
  phoneNumber: String;  
  birthday: String;  
}
```

- `studentId`: kiểu số - mã số sinh viên.
- `name`: kiểu xâu - họ và tên sinh viên.

- <i>teacherName</i> : kiểu <i>xâu</i> - tên giáo viên đứng lớp học phần.	- <i>sex</i> : kiểu <i>đúng/ sai</i> - giới tính sinh viên, với 1 (<i>true</i>) là nữ và 0 (<i>false</i>) là nam.
- <i>startTime</i> : kiểu <i>số</i> - thứ, tiết bắt đầu của lớp học phần.	- <i>class</i> : kiểu <i>xâu</i> – tên lớp sinh hoạt của sinh viên.
- <i>endTime</i> : kiểu <i>số</i> - thứ, tiết kết thúc của lớp học phần.	- <i>phoneNumber</i> : kiểu <i>xâu</i> – số điện thoại của sinh viên.
- <i>current</i> : kiểu <i>số</i> - số lượng sinh viên đã đăng ký học lớp học phần này.	- <i>birthday</i> : kiểu <i>xâu</i> – ngày tháng năm sinh của sinh viên.
- <i>capacity</i> : kiểu <i>số</i> - số lượng sinh viên tối đa có thể đăng ký lớp học phần này.	

```
export interface Subject {
    subjectID: number;
    subjectName: string;
}
```

- *subjectID*: kiểu *số* - mã lớp học phần.
- *subjectName*: kiểu *xâu* - tên lớp học phần.

```
export interface Teacher {
    teacherID: number;
    name: string;
}
```

- *teacherID*: kiểu *số* - mã giảng viên.
- *name*: kiểu *xâu* - họ và tên giảng viên.

4.2. Kết quả

4.2.1. Trang chủ và trang đăng nhập



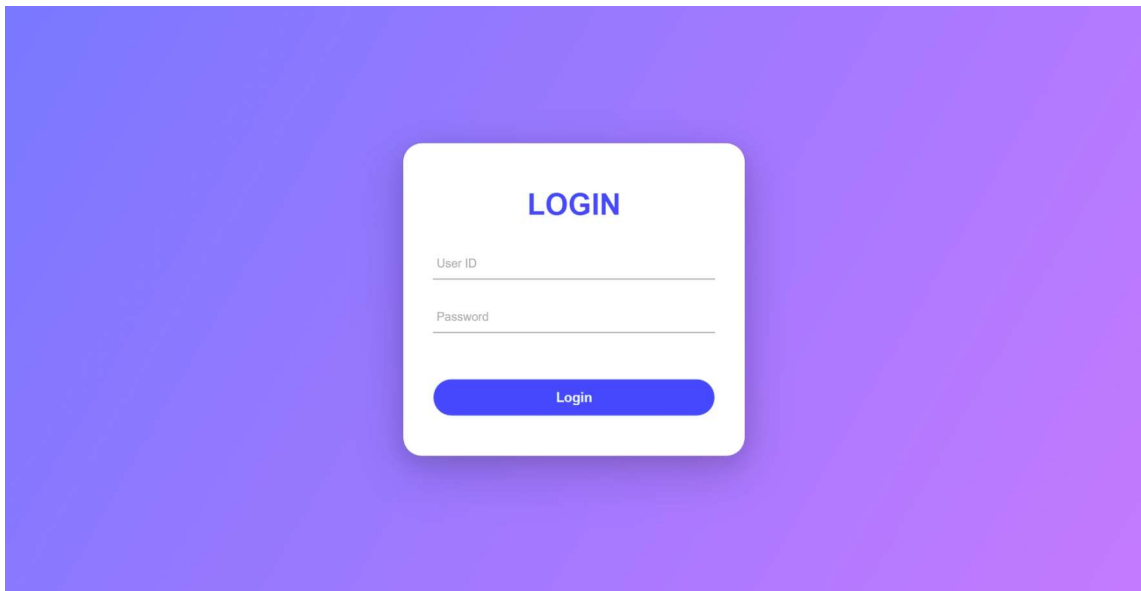
Danh sách sinh viên:

STT	Mã số sinh viên	Họ và tên	Nữ	Lớp	Số điện thoại
1	102210028	Nguyễn Viết Hoài Bảo		21TCLC_NHAT1	0786428244
2	102210029	Hoàng Lê Anh Bình	x	21TCLC_NHAT1	0383505178
3	102210030	Nguyễn Đức Chung		21TCLC_NHAT1	0378427964
4	102210031	Lê Minh Duy		21TCLC_NHAT1	0777512925
5	102210032	Nguyễn Thị Thu Hà	x	21TCLC_NHAT1	0988274803
6	102210033	Nguyễn Hữu Hiệp		21TCLC_NHAT1	0349839628
7	102210034	Phạm Trung Hiếu		21TCLC_NHAT1	0978177429
8	102210035	Trần Kim Hiếu		21TCLC_NHAT1	0768527926
9	102210036	Châu Diễm Hoàng	x	21TCLC_NHAT1	0905560643
10	102210037	Đoàn Thị Phước Huyền	x	21TCLC_NHAT1	0769989432
11	102210038	Trương Công Hoàng Long		21TCLC_NHAT1	0762668222
12	102210039	Đặng Nhật Minh		21TCLC_NHAT1	0796250097
13	102210040	Nguyễn Trương Anh Minh		21TCLC_NHAT1	0769639972
14	102210041	Trần Lê Minh		21TCLC_NHAT1	0967201345
15	102210042	Nguyễn Minh Ngọc		21TCLC_NHAT1	0868208370
16	102210043	Nguyễn Ngọc Bảo Nhân		21TCLC_NHAT1	0935149368
17	102210044	Nguyễn Hồ Minh Quân		21TCLC_NHAT1	0704410611
18	102210045	Trần Tấn Thành		21TCLC_NHAT1	0369740905
19	102210046	Huỳnh Thị Hoàng Thư	x	21TCLC_NHAT1	0702484055
20	102210047	Ngô Mậu Trường		21TCLC_NHAT1	0921233432
21	102210048	Phạm Thành Vinh		21TCLC_NHAT1	0348219257
22	102210049	Nguyễn Phạm Nhật Vỹ	x	21TCLC_NHAT1	0905873860
23	102210338	Bùi Tuấn Anh		21TCLC_NHAT1	0342973670
24	102210339	Nguyễn Văn Bách		21TCLC_NHAT1	0837385323
25	102210340	Nguyễn Gia Bảo		21TCLC_NHAT1	0373650403
26	102210341	Huỳnh Hải Đăng		21TCLC_NHAT1	0787614533
27	102210342	Nguyễn Đức Dũng		21TCLC_NHAT1	0869805823
28	102210343	Thái Khắc Dược		21TCLC_NHAT1	0987125147
29	102210344	Phạm Đình Hà		21TCLC_NHAT1	0362651324
30	102210345	Hoàng Đức Khánh		21TCLC_NHAT1	0975658549
31	102210346	Nguyễn Thanh Liêm		21TCLC_NHAT1	0869108754
32	102210347	Đỗ Đình Mạnh		21TCLC_NHAT1	0332842309
33	102210348	Nguyễn Bình Minh		21TCLC_NHAT1	0981527513
34	102210349	Đặng Ngọc Nam		21TCLC_NHAT1	0939884875
35	102210350	Phan Thị Thanh Nhân	x	21TCLC_NHAT1	0388291839
36	102210351	Phạm Quang Nhật		21TCLC_NHAT1	0932417536

Hình 10. Trang chủ

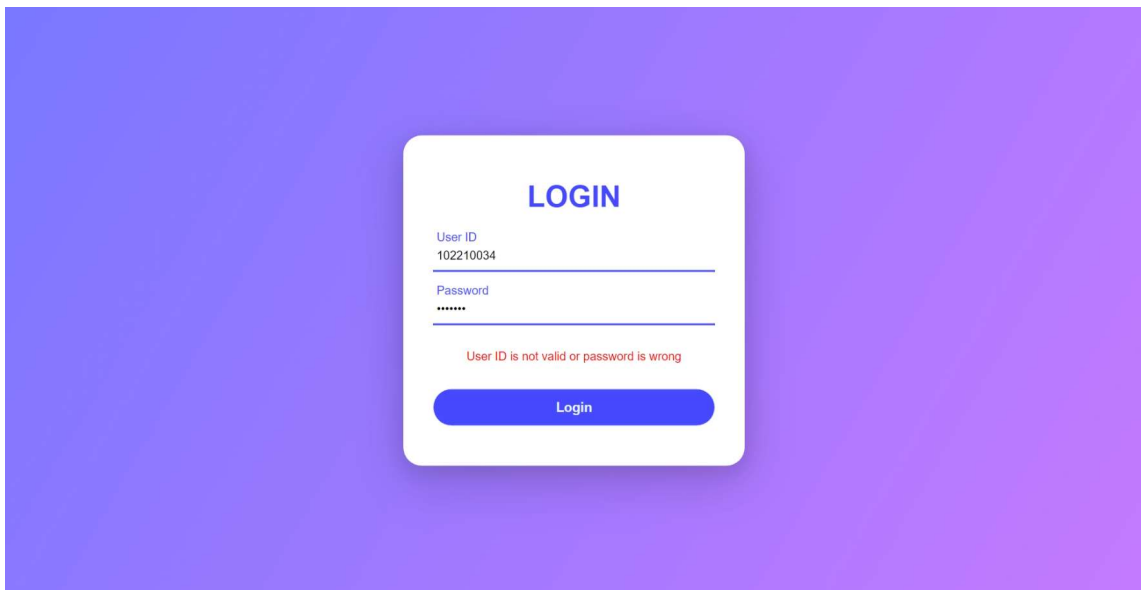
Hiện tại trang chủ của hệ thống chỉ hiển thị danh sách sinh viên trong hệ thống và đường dẫn đến trang đăng nhập. Danh sách sinh viên hiện tại chỉ bao gồm các sinh viên của lớp sinh hoạt 21TCLC_NHAT1 để làm mẫu.

Từ trang chủ, nhấn vào nút *Đăng nhập* để thực hiện đăng nhập vào hệ thống.



Hình 11. Trang đăng nhập

Khi nhập tài khoản và mật khẩu không khớp với dữ liệu của hệ thống, trang sẽ có dòng “*User ID is not valid or password is wrong*”. Ở phần này, hệ thống sẽ không phân biệt cụ thể là sai tài khoản hay mật khẩu.



Hình 12. Trang đăng nhập lỗi

Ngược lại, khi đăng nhập vào hệ thống thành công, trang sẽ tự động chuyển tới trang chủ và hiển thị thêm các chức năng để tương tác với tài khoản sinh viên đó.

Hệ thống sẽ hiển thị ảnh đại diện ở bên góc trên bên phải của tài khoản sinh viên đăng nhập vào. Ngoài ra, phần tiêu đề trang khi được đăng nhập cũng sẽ biến mất, thay vào đó là hai chức năng của tài khoản sinh viên là đăng ký học phần mới và xem thời

PBL2: DỰ ÁN CƠ SỞ LẬP TRÌNH

khoá biểu theo học phần đã đăng ký hiện tại. Tương tự như trang chủ, trang chủ khi được đăng nhập cũng chỉ có chức năng xem danh sách sinh viên trong hệ thống hiện tại.



Đăng ký học

Thời khoá biểu



Phạm Trung Hiếu

Danh sách sinh viên:

STT	Mã số sinh viên	Họ và tên	Nữ	Lớp	Số điện thoại
1	102210028	Nguyễn Viết Hoài Bảo		21TCLC_NHAT1	0786428244
2	102210029	Hoàng Lê Anh Bình	x	21TCLC_NHAT1	0383505178
3	102210030	Nguyễn Đức Chung		21TCLC_NHAT1	0378427964
4	102210031	Lê Minh Duy		21TCLC_NHAT1	0777512925
5	102210032	Nguyễn Thị Thu Hà	x	21TCLC_NHAT1	0988274803
6	102210033	Nguyễn Hữu Hiệp		21TCLC_NHAT1	0349839628
7	102210034	Phạm Trung Hiếu		21TCLC_NHAT1	0978177429
8	102210035	Trần Kim Hiếu		21TCLC_NHAT1	0768527926
9	102210036	Châu Diễm Hoàng	x	21TCLC_NHAT1	0905560643
10	102210037	Đoàn Thị Phước Huyền	x	21TCLC_NHAT1	0769989432
11	102210038	Trương Công Hoàng Long		21TCLC_NHAT1	0762668222
12	102210039	Đặng Nhật Minh		21TCLC_NHAT1	0796250097
13	102210040	Nguyễn Trương Anh Minh		21TCLC_NHAT1	0769639972
14	102210041	Trần Lê Minh		21TCLC_NHAT1	0967201345
15	102210042	Nguyễn Minh Ngọc		21TCLC_NHAT1	0868208370
16	102210043	Nguyễn Ngọc Bảo Nhân		21TCLC_NHAT1	0935149368
17	102210044	Nguyễn Hồ Minh Quân		21TCLC_NHAT1	0704410611
18	102210045	Trần Tấn Thành		21TCLC_NHAT1	0369740905
19	102210046	Huỳnh Thị Hoàng Thư	x	21TCLC_NHAT1	0702484055
20	102210047	Ngô Mậu Trường		21TCLC_NHAT1	0921233432
21	102210048	Phạm Thành Vinh		21TCLC_NHAT1	0348219257
22	102210049	Nguyễn Phạm Nhật Vỹ	x	21TCLC_NHAT1	0905873860
23	102210338	Bùi Tuấn Anh		21TCLC_NHAT1	0342973670
24	102210339	Nguyễn Văn Bách		21TCLC_NHAT1	0837385323
25	102210340	Nguyễn Gia Bảo		21TCLC_NHAT1	0373650403
26	102210341	Huỳnh Hải Đăng		21TCLC_NHAT1	0787614533
27	102210342	Nguyễn Đức Dũng		21TCLC_NHAT1	0869805823
28	102210343	Thái Khắc Dược		21TCLC_NHAT1	0987125147
29	102210344	Phạm Đình Hà		21TCLC_NHAT1	0362651324
30	102210345	Hoàng Đức Khánh		21TCLC_NHAT1	0975658549
31	102210346	Nguyễn Thanh Liêm		21TCLC_NHAT1	0869108754
32	102210347	Đỗ Đình Mạnh		21TCLC_NHAT1	0332842309
33	102210348	Nguyễn Bình Minh		21TCLC_NHAT1	0981527513
34	102210349	Đặng Ngọc Nam		21TCLC_NHAT1	0939884875
35	102210350	Phan Thị Thanh Nhân	x	21TCLC_NHAT1	0388291839
36	102210351	Phạm Quang Nhựt		21TCLC_NHAT1	0932417536

Hình 13. Trang chủ sau khi đăng nhập



Khi di chuyển chuột vào phần ảnh đại diện và tên sinh viên, sẽ có một bảng chọn gồm hai chức năng: vào trang cá nhân để xem thông tin, đăng xuất khỏi tài khoản sinh viên hiện tại.



Hình 14. Thanh điều hướng sau khi đăng nhập

4.2.2. Trang đăng ký học

Đối với các sinh viên chưa thực hiện đăng ký bất cứ môn học nào, hệ thống sẽ hiển thị như hình dưới.

<div>  <div> Đăng ký học Thời khoá biểu </div> <div>  <div>Phạm Trung Hiếu</div> <div> Trang cá nhân Đăng xuất </div> </div> </div>					
Lớp đã đăng ký					
STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
Chưa đăng ký lớp học phần nào					
Lớp có thể đăng ký					
STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
1	0000000000000000001	Tiếng Nhật 3 (CNTT)	Công ty Sun	8/20	Đăng ký
2	0000000000000000002	Kinh tế chính trị Mác - Lênin	Nguyễn Lê Thu Hiền	6/20	Đăng ký
3	0000000000000000013	Phân tích & thiết kế giải thuật	Phan Thanh Tao	5/20	Đăng ký
4	0000000000000000003	TN Vật lý (Điện-Từ-Quang)	Phan Liễn	5/20	Đăng ký
5	0000000000000000007	Lý thuyết thông tin	Ninh Khánh Duy	5/20	Đăng ký
6	0000000000000000004	Xác suất thống kê	Tôn Thất Tú	5/20	Đăng ký
7	0000000000000000005	Lập trình hướng đối tượng	Đặng Hoài Phương	5/20	Đăng ký
8	0000000000000000010	TH Lập trình hướng đối tượng	Đặng Hoài Phương	4/20	Đăng ký
9	0000000000000000006	Cơ sở dữ liệu	Trương Ngọc Châu	5/20	Đăng ký
10	0000000000000000008	Nguyên lý hệ điều hành	Trần Hồ Thuý Tiên	5/20	Đăng ký
11	0000000000000000009	PBL 2: Dự án cơ sở lập trình	Đặng Thiên Bình	5/20	Đăng ký

Hình 15. Trang đăng ký tin chỉ khi chưa đăng ký

Trang *Đăng ký học* được chia thành hai phần: *Lớp đã đăng ký*, *Lớp có thể đăng ký*. Ở phần *Lớp đã đăng ký*, trang sẽ hiển thị những thông tin học phần đã được sinh viên đăng ký. Nếu như chưa có học phần nào được đăng ký, phần bảng thông tin học phần sẽ hiển thị “*Chưa đăng ký lớp học phần nào*”. Ngược lại, khi đã có học phần đăng ký, trang sẽ hiển thị như hình dưới.



Lớp đã đăng ký

STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
1	00000000000000000001	Tiếng Nhật 3 (CNTT)	Công ty Sun	9/20	Hủy
2	00000000000000000002	Kinh tế chính trị Mác - Lênin	Nguyễn Lê Thu Hiền	7/20	Hủy
3	00000000000000000003	TN Vật lý (Điện-Từ-Quang)	Phan Liễn	6/20	Hủy
4	00000000000000000013	Phân tích & thiết kế giải thuật	Phan Thanh Tao	6/20	Hủy

Lớp có thể đăng ký

STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
1	00000000000000000007	Lý thuyết thông tin	Ninh Khánh Duy	5/20	Đăng ký
2	00000000000000000004	Xác suất thống kê	Tôn Thất Tú	5/20	Đăng ký
3	00000000000000000005	Lập trình hướng đối tượng	Đặng Hoài Phương	5/20	Đăng ký
4	00000000000000000010	TH Lập trình hướng đối tượng	Đặng Hoài Phương	4/20	Đăng ký
5	00000000000000000006	Cơ sở dữ liệu	Trương Ngọc Châu	5/20	Đăng ký
6	00000000000000000008	Nguyên lý hệ điều hành	Trần Hồ Thuý Tiên	5/20	Đăng ký
7	00000000000000000009	PBL 2: Dự án cơ sở lập trình	Đặng Thiễn Bình	5/20	Đăng ký

Hình 16. Trang đăng ký tin chỉ khi đã đăng ký

Ở phần *Lớp có thể đăng ký*, các thông tin các lớp có thể đăng ký sẽ xuất hiện ở đây. Các lớp có thể đăng ký bao gồm các lớp còn tồn tại chỗ trống, không bị cản thời khoá biểu với thời khoá biểu hiện tại của các học phần mà sinh viên đã đăng ký. Nếu như không còn lớp học phần phù hợp để đăng ký, phần bảng thông tin sẽ hiển thị “*Không có lớp học phần có thể đăng ký*”.



Lớp đã đăng ký

STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
1	00000000000000000001	Tiếng Nhật 3 (CNTT)	Công ty Sun	9/20	Hủy
2	00000000000000000002	Kinh tế chính trị Mác - Lênin	Nguyễn Lê Thu Hiền	7/20	Hủy
3	00000000000000000003	TN Vật lý (Điện-Từ-Quang)	Phan Liên	6/20	Hủy
4	00000000000000000004	Xác suất thống kê	Tôn Thất Tú	6/20	Hủy
5	00000000000000000005	Lập trình hướng đối tượng	Đặng Hoài Phương	6/20	Hủy
6	00000000000000000006	Cơ sở dữ liệu	Trương Ngọc Châu	6/20	Hủy
7	00000000000000000007	Lý thuyết thông tin	Ninh Khánh Duy	6/20	Hủy
8	00000000000000000008	Nguyên lý hệ điều hành	Trần Hồ Thuỷ Tiên	6/20	Hủy
9	00000000000000000009	PBL 2: Dự án cơ sở lập trình	Đặng Thiên Bình	6/20	Hủy
10	00000000000000000010	TH Lập trình hướng đối tượng	Đặng Hoài Phương	5/20	Hủy
11	00000000000000000013	Phân tích & thiết kế giải thuật	Phan Thanh Tao	6/20	Hủy

Lớp có thể đăng ký

STT	Mã lớp học phần	Tên lớp học phần	Giảng viên	Đã đăng ký	
Không có lớp học phần có thể đăng ký					

Hình 17. Trang đăng ký khi không tồn tại lớp đăng ký

4.2.3. Trang thời khoá biểu



Thời khoá biểu theo học phần đã đăng ký

Tiết	2	3	4	5	6	7	8
1			7:00 - 9:50 Nguyên lý hệ điều hành			7:00 - 8:50 Kinh tế chính trị Mác - Lênin	
2		8:00 - 9:50 Tiếng Nhật 3 (CNTT)		7:00 - 10:50 TH Lập trình hướng đối tượng			
3							
4							
5							
6		12:00 - 13:50 Cơ sở dữ liệu		12:00 - 13:50 Lý thuyết thông tin			
7	12:00 - 14:50 Xác suất thống kê						
8		14:00 - 16:50 Lập trình hướng đối tượng		14:00 - 15:50 PBL 2: Dự án cơ sở lập trình			
9	15:00 - 16:50 Phân tích & thiết kế giải thuật						
10							
11							
12	17:00 - 20:50 TN Vật lý (Điện-Từ-Quang)						
13							
14							
15							

Hình 18. Trang thời khoá biểu

Phần thời khoá biểu sẽ hiển thị theo các học phần mà sinh viên đã đăng ký ở trang *Đăng ký học*. Bảng thời khoá biểu sẽ hiển thị phân theo ba buổi và các môn học bao gồm nhiều tiết sẽ được nhóm thành một cột để hiển thị thời gian và tên môn học. Với thời khoá biểu này, sinh viên có thể dễ dàng biết được số môn học trong ngày, số tiết học và thời gian học.

Ngoài ra, bảng thời khoá biểu sẽ có một cột bao gồm các học phần được tô màu hồng. Đây là các học phần mà sinh viên sẽ học hôm nay, các cột màu hồng sẽ được tự động thay đổi theo ngày xem của sinh viên.

4.2.4. Trang cá nhân

Trang cá nhân hiển thị thông tin trong hệ thống của mỗi tài khoản sinh viên bao gồm ảnh đại diện, họ và tên, mã số sinh viên, lớp, số điện thoại và ngày tháng năm sinh. Tuy nhiên, hệ thống hiện tại chỉ có khả năng hiển thị, chưa hỗ trợ chỉnh sửa thông tin cá nhân của sinh viên trên hệ thống.



[Đăng ký học](#)

[Thời khoá biểu](#)



Phạm Trung Hiếu



Trang 19. Trang cá nhân

4.3. Nhận xét

Ưu điểm	Nhược điểm
- Giao diện hiện tại dễ sử dụng, màu sắc thân thiện, có thể sử dụng với mọi loại màn hình máy tính.	- Ở phần đăng ký học phần, chưa hiển thị đầy đủ thông tin của các lớp học phần hiện tại do cơ sở dữ liệu chưa có đủ lớn. - Chưa tối ưu được số lượng thay đổi dữ liệu lên server do mỗi lần đăng ký, huỷ

- | | |
|---|---|
| <ul style="list-style-type: none">- Dễ theo dõi các thông tin đã hiển thị trên trang, các thông tin cần chú ý đã được đổi màu nổi bật hơn.- Qua quá trình thử nghiệm với sinh viên lớp 21TCLC_NHAT1, trang đã đáp ứng được các yêu cầu đăng ký học phần cơ bản của các bạn sinh viên, chưa có trường hợp thử nghiệm báo lỗi về server. | <ul style="list-style-type: none">đăng ký phải gửi request lên server một lần.- Chỉ mới có các chức năng cơ bản của trang đăng ký tín chỉ và chỉ sử dụng được trên máy tính. |
|---|---|

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

a. Kết luận

- Hệ thống đáp ứng được yêu cầu của bài toán đặt ra ở phần mở đầu.
- Áp dụng được các kiến thức đã học vào các phần hướng đối tượng và cơ sở dữ liệu của dự án.
- Quá trình thử nghiệm ban đầu chưa có lỗi của hệ thống.

b. Hướng phát triển

Hoàn thành giao diện đầy đủ của trang và thêm các chức năng đầy đủ tương tự như trang đăng ký tín chỉ của trường đại học Bách Khoa – đại học Đà Nẵng.

TÀI LIỆU THAM KHẢO

- [1] Mô hình MVC: <https://monamedia.co/mvc-la-gi-ung-dung-cua-mo-hinh-mvc-trong-lap-trinh/>
- [2] Django: <http://legiacong.blogspot.com/2021/04/django3-mo-hinh-mvt.html>
- [3] Angular: <https://angular.io/docs>

CODE

```
class Sex(IntEnum):
    MALE = 0
    FEMALE = 1
    OTHER = 2

class User:
    def __init__(self, userid='', name='', sex=Sex.OTHER,
phone_number='', birthday=''):
        self.__userid = userid
        self.__name = name
        self.__sex = sex
        self.__phone_number = phone_number
        self.__birthday = birthday

    def GetUserID(self) -> str:
        return self.__userid

    def GetUserName(self) -> str:
        return self.__name

    def GetUserSex(self) -> Sex:
        return self.__sex

    def GetUserPhoneNumber(self) -> str:
        return self.__phone_number

    def GetUserBirthday(self) -> str:
        return self.__birthday

    def SetUserName(self, name: str) -> None:
        self.__name = name

    def SetUserSex(self, sex: Sex) -> None:
        self.__sex = sex

    def SetUserPhoneNumber(self, phone_number : str) -> None:
        self.__phone_number = phone_number

    def SetUserBirthday(self, birthday : str) -> None:
        self.__birthday = birthday

    @staticmethod
```

```
def Authenticate(id, password) -> str:
    from Database.TokenDatabase import TokenDatabase
    from Database.StudentDatabase import StudentDatabase

    if not StudentDatabase.Login(id, password):
        return ''

    # if authentication pass
    dummy = hashlib.md5((id + password +
datetime.now().strftime('%m/%d/%Y-%H:%M:%S')).encode('utf-
8')).hexdigest()
    dummy1 = hashlib.sha256(dummy.encode('utf-8')).hexdigest()
    token = hashlib.sha512(dummy1.encode('utf-8')).hexdigest()
    TokenDatabase.Insert({'token': token, 'userid': id})
    return token

@staticmethod
def TokenAuthenticate(token):
    from Database.TokenDatabase import TokenDatabase

    UserID = TokenDatabase.Get(token)
    if UserID != None:
        return Student.GetByID(UserID, AsDict=True) # type: ignore
    return None

@staticmethod
def TokenLogout(token):
    from Database.TokenDatabase import TokenDatabase

    TokenDatabase.Delete(token)

class Student(User, IObject):

    def __init__(self, userid : str, name : str, sex : Sex,
phone_number : str, birthday : str, classname : str,
attendedClassSections = []) -> None: # type: ignore
        super().__init__(userid, name, sex, phone_number, birthday)
        self.__classname = classname
        self.__attendedClassSections = attendedClassSections

    def __str__(self):
        str = f'{self.GetUserID()} {self.GetUserName()}
{self.GetUserSex()} {self.GetClassName()}'
        return str
```

```
def AsDict(self):
    return {
        'studentId': self.GetUserID(),
        'name': self.GetUserName(),
        'sex': self.GetUserSex(),
        'class': self.GetClassName(),
        'phoneNumber': self.GetUserPhoneNumber(),
        'birthday': self.GetUserBirthday()
    }

def GetPrimaryKey(self) -> str:
    return self.__userid

def GetClassName(self) -> str:
    return self.__classname

def SetClassName(self, classname : str) -> None:
    self.__classname = classname

def GetAttendedClasses(self, AsDict = False):
    from Database.StudentDatabase import StudentDatabase

    if not StudentDatabase.IsClassesLoaded(self.GetUserID()):
        StudentDatabase.FetchFromDatabase(self.GetUserID())
    if not AsDict:
        return self.__attendedClassSections
    retList = []
    for classSection in self.__attendedClassSections:
        retList.append(classSection.AsDict())
    return retList

def _SetAttendedClasses(self, classes: list):
    self.__attendedClassSections = classes

def _AddClassSection(self, classSection):
    from Database.StudentDatabase import StudentDatabase
    if StudentDatabase.IsClassesLoaded(self.GetUserID()):
        StudentDatabase.FetchFromDatabase(self.GetUserID())
    self.__attendedClassSections.append(classSection)

def _RemoveClassSection(self, classSection):
    from Database.StudentDatabase import StudentDatabase
    if StudentDatabase.IsClassesLoaded(self.GetUserID()):
```

```
        StudentDatabase.FetchFromDatabase(self.GetUserID())

    if classSection in self.__attendedClassSections:
        self.__attendedClassSections.remove(classSection)

    @staticmethod
    def FromRecord(record) -> 'Student':
        if record == None:
            raise RecordException(f"Record is empty")
        for i in range(len(record)):
            ele = record[i]
            if ele == None:
                raise RecordException(f"Element {i} of Student record is
None or Empty")

        from Database.ClassSectionDatabase import ClassSectionDatabase
        classList = []
        try:
            for pk in record[6]:
                ClassSectionDatabase.FetchFromDatabase(pk[0], True)
                classList.append(ClassSectionDatabase.Get(pk[0]))
        except IndexError:
            pass
        student = Student(record[0], record[1], record[2], record[4],
record[5], record[3], classList)

        return student

    @staticmethod
    def GetAll(AsDict = False):
        from Database.StudentDatabase import StudentDatabase
        students = StudentDatabase.GetAll()
        if AsDict:
            return list([student.AsDict() for student in students])
        return students

    @staticmethod
    def GetByID(id, AsDict = False):
        from Database.StudentDatabase import StudentDatabase
        student = StudentDatabase.Get(id)
        if AsDict:
            return student.AsDict()
        return student
```

```
class Teacher(User, IObject):

    __teachingClassSections = []

    def __init__(self, userid : str, name : str, sex : Sex,
phone_number : str, birthday : str, teachingClassSections = []): #
type: ignore
        super().__init__(userid, name, sex, phone_number, birthday)
        self.__teachingClassSections = teachingClassSections

    def AsDict(self):
        return {
            'teacherId': self.GetUserID(),
            'name': self.GetUserName(),
            'sex': self.GetUserSex(),
            'phoneNumber': self.GetUserPhoneNumber(),
            'birthday': self.GetUserBirthday()
        }

    def GetPrimaryKey(self) -> str:
        return self.__userid

    def GetTeachingClassSections(self, AsDict = False):
        if not AsDict:
            return self.__teachingClassSections
        retList = []
        for classSection in self.__teachingClassSections:
            retList.append(classSection.AsDict())
        return retList

    def _SetTeachingClassSections(self, classes: list):
        self.__teachingClassSections = classes

    @staticmethod
    def FromRecord(record) -> 'Teacher':
        if record == None:
            raise RecordException(f"Record is empty")
        for i in range(len(record)):
            ele = record[i]
            if ele == None:
                raise RecordException(f"Element {i} of Student record is
None or Empty")
```

```
        teacher = Teacher(record[0], record[1], record[2], record[3],
record[4])
        return teacher
```

```
@staticmethod
def GetByID(id, AsDict = False):
    from Database.TeacherDatabase import TeacherDatabase

    teacher = TeacherDatabase.Get(id)
    if AsDict:
        return teacher.AsDict()
    return teacher
```

```
@staticmethod
def GetAll(AsDict = False):
    from Database.TeacherDatabase import TeacherDatabase

    teachers = TeacherDatabase.GetAll()
    if AsDict:
        return list([teacher.AsDict() for teacher in teachers])
    return teachers
```

```
class IObject(ABC):
```

```
    @abstractmethod
    def AsDict(self) -> dict:
        pass
```

```
    @abstractmethod
    def GetPrimaryKey(self) -> str:
        pass
```

```
    @abstractmethod
    def FromRecord(record):
        pass
```

```
    @abstractmethod
    def GetAll(AsDict=False):
        pass
```

```
    @abstractmethod
    def GetByID(id, AsDict=False):
        pass
```

```
class ClassSection(IObject):
```



```
__sectionID: str
__subject: Subject
__teacher: Teacher
__start_time: int
__end_time: int
__capacity: int
__attendingStudents = []

def __init__(self, sectionID, subject, teacher, start_time,
end_time, capacity, attendingStudents = []):
    self.__sectionID = sectionID
    self.__subject = subject
    self.__teacher = teacher
    self.__start_time = start_time
    self.__end_time = end_time
    self.__capacity = capacity
    self.__attendingStudents = attendingStudents

def AsDict(self):
    return {
        "sectionID": self.GetClassSectionID(),
        "subjectName": self.GetSubject().GetSubjectName(),
        "teacherName": self.GetTeacher().GetUserName(),
        "startTime": self.GetPeriodTime()[0],
        "endTime": self.GetPeriodTime()[1],
        "current": self.GetCurrentNumberOfStudents(),
        "capacity": self.GetClassCapacity()
    }

def GetPrimaryKey(self) -> str:
    return self.__sectionID

def GetClassSectionID(self):
    return self.__sectionID

def GetSubject(self) -> Subject:
    return self.__subject

def GetTeacher(self) -> Teacher:
    return self.__teacher

def GetCurrentNumberOfStudents(self):
    from Database.Student_ClassSectionDatabase import
Student_ClassSectionDatabase
```

```
        return
Student_ClassSectionDatabase.CountBySectionID(self.GetClassSectionID())

    def GetClassCapacity(self):
        return self.__capacity

    def GetPeriodTime(self):
        return (self.__start_time, self.__end_time)

    def GetAttendingStudents(self, AsDict = False):
        from Database.ClassSectionDatabase import ClassSectionDatabase
        if not
ClassSectionDatabase.IsStudentsLoaded(self.GetClassSectionID()):
            ClassSectionDatabase.FetchFromDatabase(self.GetClassSectionI
D())

        if not AsDict:
            return self.__attendingStudents

        retList = []
        for student in self.__attendingStudents:
            retList.append(student.AsDict())
        return retList

    def _SetAttendingStudents(self, students : list):
        self.__attendingStudents = students

    @staticmethod
    def FromRecord(record):
        if record == None:
            raise RecordException(f"Record is empty")
        for i in range(len(record)):
            ele = record[i]
            if ele == None:
                raise RecordException(f"Element {i} of Student record is
None or Empty")
        sectionID = record[0]
        from Database.TeacherDatabase import TeacherDatabase
        from Database.SubjectDatabase import SubjectDatabase
        SubjectDatabase.FetchFromDatabase(record[1], True)
        TeacherDatabase.FetchFromDatabase(record[2], True)
        subject = SubjectDatabase.Get(record[1])
        teacher = TeacherDatabase.Get(record[2])
```

```
        if record[3] == None:
            startTime = 0
            endTime = 0
        else:
            timeData = record[3].split('-')
            startTime = int(timeData[0])
            endTime = int(timeData[1])
        capacity = record[4]

    from Database.StudentDatabase import StudentDatabase

    studentList = []
    try:
        for pk in record[5]:
            studentList.append(StudentDatabase.Get(pk[0]))
    except IndexError:
        pass

    classSection = ClassSection(sectionID, subject, teacher,
                                startTime, endTime, capacity, studentList)
    return classSection

    def AddStudent(self, studentID):
        from Database.Student_ClassSectionDatabase import
Student_ClassSectionDatabase
        from Database.StudentDatabase import StudentDatabase
        from Database.ClassSectionDatabase import ClassSectionDatabase
        from UserManager.User import Student

        if not
ClassSectionDatabase.IsStudentsLoaded(self.GetClassSectionID()):
            ClassSectionDatabase.FetchFromDatabase(self.GetClassSectionI
D())

        student: Student = StudentDatabase.Get(studentID) #type: ignore
        self.__attendingStudents.append(student)
        student._AddClassSection(self)
        Student_ClassSectionDatabase.Insert((studentID,
self.GetClassSectionID()))

    def RemoveStudent(self, studentID):
        from Database.Student_ClassSectionDatabase import
Student_ClassSectionDatabase
        from Database.StudentDatabase import StudentDatabase
```

```
        from UserManager.User import Student

        from Database.ClassSectionDatabase import ClassSectionDatabase
        if not
ClassSectionDatabase.IsStudentsLoaded(self.GetClassSectionID()):
            ClassSectionDatabase.FetchFromDatabase(self.GetClassSectionI
D())

        student: Student = StudentDatabase.Get(studentID) # type: ignore
        try:
            self.__attendingStudents.remove(student)
            student._RemoveClassSection(self)
        except:
            print('Delete Error')
            Student_ClassSectionDatabase.Delete((studentID,
self.GetClassSectionID()))

    @staticmethod
    def GetByID(id, AsDict = False):
        from Database.ClassSectionDatabase import ClassSectionDatabase

        classSection = ClassSectionDatabase.Get(id)
        if AsDict:
            return classSection.AsDict()
        return classSection

    @staticmethod
    def GetAll(AsDict = False):
        from Database.ClassSectionDatabase import ClassSectionDatabase

        classSections = ClassSectionDatabase.GetAll()
        if AsDict:
            return list([classSection.AsDict() for classSection in
classSections])
        return classSections

class Subject(IObject):
    __subjectID: str
    __subjectName: str

    def __init__(self, subjectID, subjectName):
        self.__subjectID = subjectID
        self.__subjectName = subjectName

    def AsDict(self):
```

```
        return {
            "subjectID": self.GetSubjectID(),
            "subjectName": self.GetSubjectName()
        }

    def GetPrimaryKey(self) -> str:
        return self.__subjectID

    def GetSubjectID(self):
        return self.__subjectID

    def GetSubjectName(self):
        return self.__subjectName

    def SetSubjectName(self, subjectName : str):
        self.__subjectName = subjectName

    @staticmethod
    def FromRecord(record):
        if record == None:
            raise RecordException(f"Record is empty")
        for i in range(len(record)):
            ele = record[i]
            if ele == None:
                raise RecordException(f"Element {i} of Student record is
None or Empty")
            subjectID = record[0]
            subjectName = record[1]
            subject = Subject(subjectID, subjectName)
        return subject

    @staticmethod
    def GetByID(id, AsDict=False):
        from Database.SubjectDatabase import SubjectDatabase

        subject = SubjectDatabase.Get(id)
        if AsDict:
            return subject.AsDict()
        return subject

    @staticmethod
    def GetAll(AsDict=False):
        from Database.SubjectDatabase import SubjectDatabase
```

```
        subjects = SubjectDatabase.GetAll()
        if AsDict:
            return list([subject.AsDict() for subject in subjects])
        return subjects
    }
}

export class ClassectionService {
    constructor(private http: HttpClient) {}

    private handleError<T>(operation = 'operation', result?: T) {
        return (error: any): Observable<T> => {
            console.error(error);
            return of(result as T);
        };
    }

    private baseUrl = BASE_URL + `/api/classection`;

    getClasses(): Observable<Class[]> {
        return this.http
            .get<Class[]>(this.baseUrl)
            .pipe(catchError(this.handleError<Class[]>('getClasses', [])));
    }

    getClass(id: string): Observable<Class> {
        const url = this.baseUrl + `/${id}`;

        return this.http
            .get<Class>(url)
            .pipe(catchError(this.handleError<Class>('getClasses')));
    }

    getClassesAttendedByStudent(studentID: string): Observable<Class[]> {
        const url = this.baseUrl + `?sid=${studentID}`;

        return this.http
            .get<Class[]>(url)
            .pipe(
                catchError(this.handleError<Class[]>('getClassesAttendedByStudent', []))
            );
    }

    addClass(_class: Class, student: Student): void {
        const options = {
            headers: new HttpHeaders({
```

```
        'Content-Type': 'application/json',
    }),
    body: {
        sid: student.studentId,
    },
};
const url = this.baseURL + `/${_class.sectionID}`;

this.http.post(url, options).subscribe();
}

deleteClass(classID: string, studentID: Number): void {
    const options = {
        headers: new HttpHeaders({
            'Content-Type': 'application/json',
        }),
        body: {
            sid: studentID,
        },
    };
    const url = this.baseURL + `/${classID}`;

    this.http
        .delete(url, options)
        .pipe(catchError(this.handleError<Class[]>('deleteClass', [])))
        .subscribe();
}

getNewClasses(studentID: string): Observable<Class[]> {
    const url = this.baseURL + `?sid=${studentID}&mode=register`;

    return this.http
        .get<Class[]>(url)
        .pipe(catchError(this.handleError<Class[]>('getNewClasses', [])));
}
}

export class LoginService {
    constructor(private http: HttpClient) {}

    private handleError<T>(operation = 'operation', result?: T) {
        return (error: any): Observable<T> => {
            console.error(error);
            return of(result as T);
        };
    }
}
```

```
}

private baseUrl = BASE_URL + `/api/user`;

getToken(id: string, password: string): Observable<any> {
  const url = `${this.baseUrl}/login?id=${id}&password=${password}`;
  return
this.http.get<any>(url).pipe(catchError(this.handleError<any>()));
}

loginByToken(): Observable<any> {
  const url =
`${this.baseUrl}/login?token=${localStorage.getItem('token')}`;
  return this.http
    .get<Student>(url)
    .pipe(catchError(this.handleError<Student>()));
}

Logout() {
  const url =
`${this.baseUrl}/logout/${localStorage.getItem('token')}`;
  return
this.http.delete<any>(url).pipe(catchError(this.handleError<any>()));
}
}

export class StudentService {
  constructor(private http: HttpClient) {}

  private handleError<T>(operation = 'operation', result?: T) {
    return (error: any): Observable<T> => {
      console.error(error);
      return of(result as T);
    };
  }

  private baseUrl = BASE_URL + `/api/user/student/`;

  getStudents(): Observable<Student[]> {
    return this.http
      .get<Student[]>(this.baseUrl)
      .pipe(catchError(this.handleError<Student[]>('getStudents', [])));
  }

  getStudent(id: String): Observable<Student> {
```

```
const url = `${this.baseURL}/${id}`;
return this.http
  .get<Student>(url)
  .pipe(catchError(this.handleError<Student>()));
}
}

export class ClassSignComponent implements OnInit {
  loginUser: any;

  constructor(
    private loginService: LoginService,
    private classSectionService: ClassSectionService
  ) {}

  classes: Class[] = [];
  newClasses: Class[] = [];

  ngOnInit(): void {
    this.loginService.loginByToken().subscribe((student) => {
      this.loginUser = student;
      if (this.loginUser.hasOwnProperty('error')) {
        location.replace('/main');
      }

      this.classSectionService
        .getClassesAttendedByStudent(this.loginUser['studentId'])
        .subscribe((data) => {
          (this.classes = data).sort((n1, n2) => {
            if (n1.sectionID > n2.sectionID) {
              return 1;
            }
            if (n1.sectionID < n2.sectionID) {
              return -1;
            }
            return 0;
          })
        });

      this.classSectionService
        .getNewClasses(this.loginUser['studentId'])
        .subscribe((data) => {
          this.newClasses = data;
        });
    });
  }
};
```

```
}

addClassSection(section: Class, student: Student): void {
  this.classSectionService.addClass(section, student);
  location.replace('/classSign');
}

deleteClassSection(section: Class, student: Student): void {
  this.classSectionService.deleteClass(section.sectionID,
student.studentId);
  location.replace('/classSign');
}
}

export class LoginBoxComponent implements OnInit {
  loginForm: any;
  error: String = '';
  constructor(
    private formBuilder: FormBuilder,
    private loginService: LoginService
  ) {}

  ngOnInit(): void {
    this.loginForm = this.formBuilder.group({
      id: ['', Validators.required],
      password: ['', Validators.required],
    });
  }

  Login() {
    if (this.loginForm.dirty && this.loginForm.valid) {
      this.loginService
        .getToken(this.loginForm.value.id,
this.loginForm.value.password)
        .subscribe((data) => {
          if (data.hasOwnProperty('token')) {
            localStorage.setItem('token', data['token']);
            window.location.href = '/';
          } else {
            this.error = data['error'];
          }
        });
    }
  }
}
```

```
export class NavbarComponent implements OnInit {
  @Input() loginName?: String;

  constructor(private loginService: LoginService) {}

  ngOnInit(): void {}

  Logout() {
    this.loginService.Logout().subscribe();
    location.replace('/main');
  }
}

export class ProfileComponent implements OnInit {
  loginUser: any = null;
  constructor(private loginService: LoginService) {}

  ngOnInit(): void {
    this.loginService.loginByToken().subscribe((student) => {
      this.loginUser = student;
      if (this.loginUser.hasOwnProperty('error')) {
        location.replace('/main');
      }
    });
  }
}

export class StudentListComponent implements OnInit {
  students: Student[] = [];
  loginUser: any;

  constructor(
    private studentService: StudentService,
    private loginService: LoginService
  ) {}

  getStudents(): void {
    this.studentService.getStudents().subscribe((students) =>
      (this.students = students).sort((n1, n2) => {
        if (n1.studentId > n2.studentId) {
          return 1;
        }
        if (n1.studentId < n2.studentId) {
          return -1;
        }
      })
    );
  }
}
```



```
        location.replace('/main');
    }

    this.classSectionService
        .getClassesAttendedByStudent(this.loginUser['studentId'])
        .subscribe((data) => {
            this.classes = data;
            for (let i = 0; i < this.classes.length; i++) {
                const classSection = this.classes[i];
                var dayOfWeek: number = Math.floor(classSection.startTime /
15);
                var start = classSection.startTime % 15;
                var duration = classSection.endTime - classSection.startTime
+ 1;

                console.log(classSection, dayOfWeek, start, duration);
                this.table[start][dayOfWeek] = [classSection.subjectName,
duration];
                for (let j = 1; j < duration; j++)
                    this.table[start + j][dayOfWeek] = ['#', 0];
            }
        });
    });
}
}
```
