**snipBỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**BÁO CÁO THỰC TẬP**

**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài*:**

**XÂY DỰNG ỨNG DỤNG HỌC TỪ VỰNG TIẾNG ANH TRÊN NỀN TẢNG MOBIE**

**Giảng Viên hướng dẫn : ThS. HUỲNH TRUNG TRỤ**

**Sinh viên thực hiện : NGUYỄN TRỌNG TÀI**

**Mã số sinh viên : N15DCCN055**

**Lớp : D15CQIS01-N**

**Khoá : 2015- 2020**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 08/2020**

# LỜI CẢM ƠN

Kính thưa quý thầy cô!

Khoảng thời gian học tập tại trường Học viện Công nghệ Bưu chính Viễn thông cơ sở tại Tp. Hồ Chí Minh đã giúp em có được những kiến thức quý giá , giúp em trưởng thành hơn trong cuộc sống.Và bây giờ khoảng thời gian cùng với những kỷ niệm đẹp đó đã sắp khép lại em xin được gửi lời cảm ơn đến tất cả thầy cô đã tận tâm giảng dạy truyền đạt những bài học, những kinh nghiệm quý báu.

Trong các thầy cô trực tiếp giảng dạy cho em, em xin được gửi lời cảm ơn đến thầy Ths. Lưu Nguyễn Kỳ Thư người đã truyền đạt cho em nhiều kinh nghiệm, nhiều bài học hay có thể ứng dụng vào công việc cũng như chuyên ngành hệ thống. Thầy đã giúp em có nhiều va chạm kiến thức tiếp cận gần với thực tiễn hơn trước khi em bước vào môi trường làm việc thực sự và giúp em đứng vững hơn với mọi vấn đề trong công việc.

Lời cảm ơn sau cùng em xin gửi đến thầy Ths. Huỳnh Trung Trụ người trực tiếp hướng dẫn em hoàn thành kỳ thực tập tốt nghiệp, hết lòng hỗ trợ mọi khó khăn trong quá trình thực tập của em. Cảm ơn với những bài học hữu ích mà thầy đã truyền đạt trong các môn học đã giúp em có nhiều tiền đề, phương hướng để học tập.

Tuy nhiên, vì thời gian và kiến thức còn hạn chế em đã cố gắng rất nhiều để hoàn thành đề tài này, nhưng trong quá trình làm bài em không tránh khỏi những thiếu sót, kính mong được lời nhận xét và góp ý của thầy cô.

Một lần nữa, em xin chân thành cảm ơn !

TP.HCM, Ngày tháng 08 năm 2020

Sinh viên thực hiện

Nguyễn Trọng Tài

# MỤC *LỤC*

[I. LỜI CẢM ƠN 1](#_Toc47364982)

[II. MỤC *LỤC* 2](#_Toc47364983)

[III. Chuong 1: GIỚI THIỆU ĐỀ TÀI 3](#_Toc47364984)

[A. Mục tiêu đề tài: 3](#_Toc47364985)

[B. Giới thiệu công nghệ: 3](#_Toc47364986)

[1. Tổng quan. 3](#_Toc47364987)

[2. Framework Asp.net core 3.1. 3](#_Toc47364988)

[3. Giới thiệu EntityFramwork Core. 5](#_Toc47364989)

[4. Microsoft SqlServer. 6](#_Toc47364990)

[5. Giới thiệu công nghệ Framework React native. 11](#_Toc47364991)

[6. Giới thiệu công nghệ Libarary React redux. 12](#_Toc47364992)

[C. Chức năng ứng dụng 15](#_Toc47364993)

[1. Tổng quan. 15](#_Toc47364994)

[2. Chức năng Server side. 15](#_Toc47364995)

[3. Chức năng Client side. 15](#_Toc47364996)

[IV. CHƯƠNG 2: SƠ LƯỢC VỀ CÔNG CỤ XÂY DỰNG HỆ THỐNG. 16](#_Toc47364997)

[A. Giới thiệu Visual Studio Code. 16](#_Toc47364998)

[B. Giới thiệu về Visual Studio. 18](#_Toc47364999)

[V. CHƯƠNG 3: THIẾT KẾ DATABASE. 26](#_Toc47365000)

[A. Xác định thực thể. 26](#_Toc47365001)

[B. Mô hình ERD. 26](#_Toc47365002)

[C. Mô Hình Dữ Liệu Quan Hệ. 27](#_Toc47365003)

[D. Từ điển dữ liệu. 27](#_Toc47365004)

[E. Diagram. 29](#_Toc47365005)

# Chuong 1: GIỚI THIỆU ĐỀ TÀI

## Mục tiêu đề tài:

Chúng ta đều biết tiếng Anh là một trong những ngôn ngữ được sử dụng rất rộng rãi và trở nên thông dụng nhất trên thế giới. Vì thế lợi của việc học tiếng Anh là không hề nhỏ :

* Lợi ích đầu tiên khi đi đến bất quốc gia nào đều có thể dùng tiếng anh để giao tiếp.
* Các trang web thông tin báo chí,web site bán hàng, công trình nghiên cứu, tài liệu học thuật có giá trị cao đa phần đều viết bằng tiếng anh, nếu chúng ta học được tiếng anh có thể nói là tiến rất gần với các nguồn tư liệu trên.
* Các công cụ dịch vụ, các ứng dụng phổ biến cũng chiếm phần lớn là tiếng anh và để có thể sử dụng dễ dàng hơn không gì khác ngoài việc học tiếng anh.

Từ đó em nhận thấy việc học tiếng anh trong hiện tại là việc rất quan trọng, và để học tiếng anh tốt hơn theo quan điểm cá nhân của em thì học từ vững là điều cốt yếu đầu tiên cần phải làm. Để có thể tăng sự tiện dụng hơn cho việc học từ vựng cũng như trau dồi kiến thức trong cộng đồng em đã nghĩ đến việc xây dựng ứng dụng hỗ trợ học từ vựng tiếng anh trên nên tảng điện thoại thông minh. Trong thời đại công nghệ số hiện tại hầu hết tất cả mọi người ai cũng có thể dễ dàng có được một chiếc điện thoại thông minh. và đó cũng là một điều kiện rất thuận tiện để sử dụng ứng dụng mọi lúc, mọi nơi cơ động về mặt thời gian .

## Giới thiệu công nghệ:

### Tổng quan.

* Giới thiệu công nghệ Framework Asp.net core 3.1.
* Giới thiệu EntityFramwork Core.
* Cơ chế xác thực trong Identity trong EntityFramework.
* Giới thiệu công nghệ Microsoft SqlServer.
* Giới thiệu công nghệ Framework React native.
* Giới thiệu công nghệ Libarary React redux.

### Framework Asp.net core 3.1.

#### Framework ASP.NET Core là gì?

Đầu tiền Asp core là một framework mã nguồn mở đa nền tảng (cross-platform) và có thể sử dụng đẻ xây dựng các ứng dụng điện toán đám mây, web api, web app, IoT,….

Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. Nó được thiết kế để cung cấp và tối ưu development framework cho những ứng dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise.

Nó bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy chúng ta giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bản thân người lập trình. Chúng ta có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.

Đồng thời nó đã trở thành một mã nguồn mở. Đây là một thay đổi rất lớn và theo em là quan trọng nhất của ASP.NET Core. Điều mà trước đây khó có một lập trình viên nào có thể nghĩ đến. Có lẽ đó cũng là một xu thế mà các ngôn ngữ lập trình hiện nay đang hướng tới.

#### Hoàn cảnh ra đời của ASP.NET Core

Bản phát hành đầu tiên của ASP.NET đã xuất hiện cách đây hơn 15 năm về trước, nó là một phần của .NET Framework. Từ đó, hàng triệu lập trình viên đã sử dụng nó để xây dựng những ứng dụng web tuyệt vời, và trên những năm đó Microsoft đã phát triển thêm nhiều tính năng mới.

ASP.NET Core không phải là phiên bản tiếp theo của ASP.NET, nó là một cái tên mới được xây dựng từ đầu, ASP.NET Core có một số thay đổi kiến trúc lớn, đó là kết quả của việc học hỏi rất nhiều từ các framework module hóa khác. ASP.NET Core không còn dựa trên System.Web.dll nữa. Nó được dựa trên một tập hợp các gói, các module hay cũng được gọi là các Nuget packages. Điều này cho phép chúng ta tối ưu ứng dụng của chúng ta để chỉ bao gồm những packages nào cần thiết. Lợi ích của nó là giúp cho ứng dụng nhỏ hơn, bảo mật chặt chẽ hơn, giảm sự phức tạp, tối ưu hiệu suất hoạt động và giảm chi phí, thời gian cho việc phát triển.

Với ASP.NET Core chúng đạt được những nền tảng cải tiến dưới đây:

* Hợp nhất việc xây dựng web UI và web APIs
* Tích hợp những client-side frameworks hiện đại và những luồng phát triển
* Hệ thống cấu hình dựa trên môi trường đám mây thật sự
* Dependency injection được xây dựng sẵn
* HTTP request được tối ưu nhẹ hơn
* Có thể host trên IIS hoặc self-host trong process của riêng bạn
* Được xây dựng trên .NET Core, hỗ trợ thực sự app versioning
* Chuyển các thực thể, thành phần, module như những NuGet packages
* Những công cụ mới để đơn giản hóa quá trình phát triển web hiện đại
* Xây dựng và chạy đa nền tảng(Windows, Mac và Linux)
* Mã nguồn mở và tập trung vào cộng đồng

### Giới thiệu EntityFramwork Core.

#### EntityFramwork Core là gì?.

Entity Framework (EF) Core là phiên bản nhẹ, có thể mở rộng, mã nguồn mở và đa nền tảng của công nghệ truy cập dữ liệu Entity Framework phổ biến. EF Core có thể đóng vai trò là trình ánh xạ quan hệ đối tượng (ORM), cho phép các nhà phát triển .NET làm việc với cơ sở dữ liệu bằng các đối tượng .NET và loại bỏ nhu cầu đối với hầu hết các mã truy cập dữ liệu mà họ thường cần viết. EF Core hỗ trợ nhiều công cụ cơ sở dữ liệuCơ chế xác thực trong Identity trong EntityFramework.

#### Đối tượng ánh xạ quan hệ (ORM).

* **Trước khi ORM ra đời:**
* Các lập trình viên phải viết mã để duy trì kết nối tới cơ sở dữ liệu.
* Họ phải thiết kế các lớp miền bên trong chương trình, theo cách thủ công theo lược đồ cơ sở dữ liệu.
* Lấy dữ liệu từ cơ sở dữ liệu và ánh xạ lại theo các đối tượng lớp.
* Lưu dữ liệu trở lại cơ sở dữ liệu từ ứng dụng cũng cần loại quy trình tương tự.
* **Lợi ích của (ORM):**

ORM Framework tạo ra các đối tượng (như trong Lập trình hướng đối tượng) và ánh xạ với các bảng trong cơ sở dữ liệu. Sau đó, lập trình viên sử dụng các đối tượng này để tương tác với cơ sở dữ liệu. Vì vậy, ý tưởng chính là giúp lập trình viên khỏi phải viết mã SQL và tối ưu hóa. Ví dụ: chúng ta đang viết một ứng dụng bằng C sharp và cơ sở dữ liệu của chúng ta bằng SQL. Như vậy C Sharp và Sql là những nền tảng khác nhau và có định dạng khác nhau. Làm thế nào chúng có thể giao tiếp với nhau? - Chúng ta cần phần mềm trung gian để kết nối hai ngôn ngữ trên lại với nhau. ORM tự động ánh xạ dữ liệu với các lớp có liên quan trong chương trình khi chúng ta truy xuất dữ liệu từ cơ sở dữ liệu hoặc chèn đối tượng mà chúng ta đã tạo vào cơ sở dữ liệu, nó sẽ tự động dịch sang mã mà SQL và thực thi đoạn mã đã dịch bên dưới hệ quản trị cơ sở dữ liệu để truy xuất hoặc thêm vào cơ sở dữ liệu.

* **Các cách để sử dựng Entityframework core:**
* **Code First** - Cách tiếp cận Code First giúp bạn tạo các thực thể trong ứng dụng của mình bằng cách tập trung vào các yêu cầu miền..
* **Model First** - Trong cách tiếp cận Model First, bạn có thể tạo EDM trước, sau đó tạo cơ sở dữ liệu từ nó. Thông thường, bạn sẽ tạo một EDM trống bằng Trình hướng dẫn mô hình dữ liệu thực thể trong Visual Studio, xác định các thực thể và mối quan hệ của chúng trong Visual Studio, sau đó tạo cơ sở dữ liệu từ mô hình được xác định này. (Đã xóa trong lõi khung thực thể).
* **Database First**  - Bạn có thể sử dụng phương pháp tiếp cận cơ sở dữ liệu trước nếu cơ sở dữ liệu đã được thiết kế và sẵn sàng. Theo cách tiếp cận này, Mô hình dữ liệu thực thể (EDM) được tạo từ cơ sở dữ liệu cơ bản.

### Microsoft SqlServer.

#### Microsoft SqlServer là gì?.

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System (RDBMS) ) sử dụng câu lệnh SQL (Transact-SQL) để trao đổi dữ liệu giữa máy Client và máy cài SQL Server. Một RDBMS bao gồm databases, database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS.

SQL Server được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. SQL Server có thể kết hợp “ăn ý” với các server khác như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server….

Một vài ấn bản SQL Server:

* Enterprise : chứa tất cả cá đặc điểm nổi bật của SQL Server, bao gồm nhân bộ máy cơ sở dữ liệu và các dịch vụ đi kèm cùng với các công cụ cho tạo và quản lý phân cụm SQL Server. Nó có thể quản lý các CSDL lớn tới 524 petabytes và đánh địa chỉ 12 terabytes bộ nhớ và hỗ trợ tới 640 bộ vi xử lý(các core của cpu)
* Standard : Rất thích hợp cho các công ty vừa và nhỏ vì giá thành rẻ hơn nhiều so với Enterprise Edition, nhưng lại bị giới hạn một số chức năng cao cấp (advanced features) khác, edition này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.
* Developer : Có đầy đủ các tính năng của Enterprise Edition nhưng được chế tạo đặc biệt như giới hạn số lượng người kết nối vào Server cùng một lúc…. Ðây là phiên bản sử dụng cho phát triển và kiểm tra ứng dụng. Phiên bản này phù hợp cho các cá nhân, tổ chức xây dựng và kiểm tra ứng dụng
* Workgroup: ấn bản SQL Server Workgroup bao gồm chức năng lõi cơ sở dữ liệu nhưng không có các dịch vụ đi kèm. Chú ý phiên bản này không còn tồn tại ở SQL Server 2012.
* Express : SQL Server Express dễ sử dụng và quản trị cơ sở dữ liệu đơn giản. Được tích hợp với Microsoft Visual Studio, nên dễ dàng để phát triển các ứng dụng dữ liệu, an toàn trong lưu trữ, và nhanh chóng triển khai. SQL Server Express là phiên bản miễn phí, không giới hạn về số cơ ở dữ liệu hoặc người sử dụng, nhưng nó chỉ dùng cho 1 bộ vi xử lý với 1 GB bộ nhớ và 10 GB file cơ sở dữ liệu. SQL Server Express là lựa chọn tốt cho những người dùng chỉ cần một phiên bản SQL Server 2005 nhỏ gọn, dùng trên máy chủ có cấu hình thấp, những nhà phát triển ứng dụng không chuyên hay những người yêu thích xây dựng các ứng dụng nhỏ.

#### Lịch sử ra đời SQL Server và các phiên bản.

Phiên bản đầu tiên của Microsoft SQL Server ra đời đầu tiên vào năm 1989 cho các hệ điều hành chạy 16 bít với SQL Server phiên bản 1.0 và tiếp tục phát triển cho tới ngày nay.

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại một engine mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một  bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên version 8.0 (SQL Server 2000) thì những cải tiến chủ yếu là mở rộng các tính năng về web và làm cho SQL Server 2000 đáng tin cậy hơn.

Một điểm đặc biệt đáng lưu ý ở phiên bản 2000 là Multiple-Instance. Tức là chúng ta có thể cài dặt phiên bản 2000 chung với các phiên bản trước mà không cần phải gỡ chúng. Nghĩa là chúng ta có thể chạy song song version 6.5 hoặc 7.0 với phiên bản 2000 trên cùng một máy (điều này không thể xảy ra với các phiên bản trước đây). Khi đó phiên bản cũ trên máy tính của chúng ta là Default Instance còn phiên bản 2000 mới vừa cài sẽ là Named Instance.

Từ tháng 10 năm 2016, các phiên bản sau được Microsoft hỗ trợ:

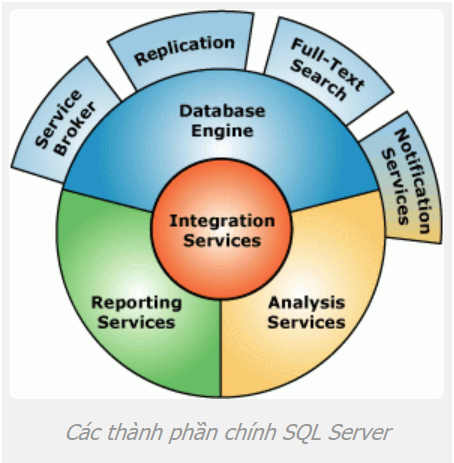
* SQL Server 2008 R2
* SQL Server 2012
* SQL Server 2014
* SQL Server 2016

Phiên bản hiện tại là Microsoft SQL Server 2016, xuất bản vào ngày 1/6/2016, SQL Server 2016 chỉ hỗ trợ cho các bộ vi xử lý 64 bít.

#### Các thành cơ bản trong SQL Server.

SQL Server được cấu tạo bởi nhiều thành phần như Database Engine, Reporting Services, Notification Services, Integration Services, Full Text Search Service…. Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

#### Database Engine.

* ** Lõi của SQL Server:**
  + Ðây là một engine có khả năng chứa data ở các quy mô khác nhau dưới dạng table và support tất cả các kiểu kết nối (data connection) thông dụng của Microsoft như
  + ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC).
  + Ngoài ra nó còn có khả năng tự điều chỉnh (tune up) ví dụ như sử dụng thêm các tài nguyên (resource) của máy khi cần và trả lại tài nguyên cho hệ điều hành khi một user log off.
* **Replication.**
  + Cơ chế tạo bản sao (Replica): Giả sử bạn có một database dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Một ngày đẹp trời bạn muốn có một cái database giống y hệt như thế trên một server khác để chạy báo cáo (report database) (cách làm này thường dùng để tránh ảnh hưởng đến performance của server chính). Vấn đề là report server của bạn cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. Bạn không thể dùng cơ chế back up and restore trong trường hợp này. Thế thì bạn phải làm sao? Lúc đó cơ chế replication của SQL Server sẽ được sử dụng để bảo đảm cho dữ liệu ở 2 database được đồng bộ (synchronized). Replication sẽ được bàn kỹ trong bài 12
* **Integration Services (DTS) :**
  + Integration Services là một tập hợp các công cụ đồ họa và các đối tượng lập trình cho việc di chuyển, sao chép và chuyển đổi dữ liệu.
  + Nếu bạn làm việc trong một công ty lớn trong đó data được chứa trong nhiều nơi khác nhau và ở các dạng khác nhau cụ thể như chứa trong Oracle, DB2 (của IBM), SQL Server, Microsoft Access….Bạn chắc chắn sẽ có nhu cầu di chuyển data giữa các server này (migrate hay transfer) và không chỉ di chuyển bạn còn muốn định dạng (format) nó trước khi lưu vào database khác, khi đó bạn sẽ thấy DTS giúp bạn giải quyết công việc trên dễ dàng.
* **Analysis Services:**
  + Một dịch vụ phân tích dữ liệu rất hay của Microsoft.
  + Dữ liệu (Data) chứa trong database sẽ chẳng có ý nghĩa gì nhiều nếu như bạn không thể lấy được những thông tin (Information) bổ ích từ đó. Do đó Microsoft cung cấp cho bạn một công cụ rất mạnh giúp cho việc phân tích dữ liệu trở nên dễ dàng và hiệu quả bằng cách dùng khái niệm hình khối nhiều chiều (multi-dimension cubes) và kỹ thuật “khai phá dữ liệu” (data mining).
* **Notification Services:**
  + Dịch vụ thông báo Notification Services là nền tảng cho sự phát triển và triển khai các ứng dụng tạo và gửi thông báo. Notification Services có thể gửi thông báo theo địch thời đến hàng ngàn người đăng ký sử dụng nhiều loại thiết bị khác nhau.
* **Reporting  Services:**
  + Reporting Services bao gồm các thành phần server và client cho việc tạo, quản lý và triển khai các báo cáo. Reporting Services cũng là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.
* **Full Text Search Service:**
  + Dịch vụ SQL Server Full Text Search là một dịch vụ đặc biệt cho đánh chỉ mục và truy vấn cho dữ liệu văn bản không cấu trúc được lưu trữ trong các CSDL SQL Server. Đánh chỉ mục với Full Text Search có thể dduwowcj tạo trên bất kỳ cột dựa trên dữ liệu văn bản. Nó sẽ rất hiệu quả cho việc tìm các sử dụng toán tử LIKE trong SQL với trường hợp tìm văn bản.
* **Service Broker :**
  + Được sử dụng bên trong mỗi Instance, là môi trường lập trình cho việc các ứng dụng nhảy qua các Instance. Service Broker giao tiếp qua giao thức TCP/IP và cho phép các component khác nhau có thể được đồng bộ cùng nhau theo hướng trao đổi các message. Service Broker chạy như một phần của bộ máy cơ sở dữ liệu, cung cấp một nền tảng truyền message tin cậy và theo hàng đợi cho các ứng dụng SQL Server.

### Giới thiệu công nghệ Framework React native.

#### React Native là gì?

React Native là một framework do công ty công nghệ nổi tiếng Facebook phát triển nhằm mục đích giải quyết bài toán hiệu năng của Hybrid và bài toán chi phí khi mà phải viết nhiều loại ngôn ngữ native cho từng nền tảng di động.

Chúng ta sẽ build được ứng dụng Native, và chúng ta cũng có thể build ứng dụng đó một cách đa nền tảng (multi-platform) chứ không phải là một “mobile web app”, không phải là “HTML5 app”, và cũng không phải là một “hybrid app” hay cũng không chỉ build trên iOS hay Android mà chúng ta build và chạy được cả hai hệ sinh thái.

Một điểm mà ract native mang lại là giảm chi phí recompile của Native bằng cách sử dụng Hot-Loading tức là chúng ta không cần phải build lại ứng dụng từ đầu nên việc chỉnh sửa diễn ra rất nhanh chóng. Giúp cho lập trình viên có thể thấy được những chỉnh sửa của họ một cách nhanh chóng trực quan, không còn phải bỏ quá nhiều thời gian trong việc build và chạy lại ứng dụng nữa.

Và điểm lợi hại kế tiếp của React Native đó chính là chúng ta chỉ cần sử dụng JS để phát triển được một ứng dụng di động hoàn chỉnh, đồng thời giải quyết được các vấn đề mà Native App gặp phải đã nêu ở trên. Và nó có thể kết hợp với code native như Swift, Java, v.v…

#### Native App là gì?

**Native App** là tên gọi dành cho những ứng dụng được xây dựng và phát triển bằng những công cụ do chính nhà phát triển cung cấp cho lập trình viên. Hiện nay, trên thế giới có hai nhà phát triển lớn tồn tại song song và cạnh tranh lẫn nhau đó là Android và iOs.

**Native App** được tạo ra bằng ngôn ngữ của hệ điều hành và các tính năng có sẵn trên hệ điều hành đó. Điều này cho phép các ứng dụng vận hành với tốc độ cao nhất mà không cần thông qua bất kì ứng dụng bên thứ ba hay engine nào.

Các ứng dụng Android có thể được viết bằng hai ngôn ngữ là Java và Kotlin. Trong khi đó, iOs app được viết bằng ngôn ngữ hướng đối tượng Swift, trước đây là Objective C. Công cụ hỗ trợ phát triển ứng dụng cho hệ điều hành Android và iOs có thể tham khảo là Android Studio và XCode.

#### Cách hoạt động của React Native

Bằng cách tích hợp 2 thread là Main Thread và JS Thread cho ứng dụng mobile. Với Main Thread sẽ đảm nhận vai trò cập nhật giao diện người dùng(UI). Sau đó sẽ xử lý tương tác người dùng. Trong khi đó, JS Thread sẽ thực thi và xử lý code Javascript. Hai luồng này hoạt động độc lập với nhau.

Để tương tác được với nhau hai Thread sẽ sử dụng một Bridge(cầu nối). Cho phép chúng giao tiếp mà không phụ thuộc lẫn nhau, chuyển đổi dữ liệu từ thread này sang thread khác. Dữ liệu từ hai Thread được vận hành khi tiếp nối dữ liệu cho nhau.

#### Ưu và nhược điểm của React Native là gì?

React Native có khá nhiều ưu điểm. Dù vậy nó cũng tồn tại một số nhược điểm. Hãy cùng Mắt Bão tìm hiểu chi tiết bên dưới đây!

* **Ưu điểm:**
* Tối ưu thời gian.
* Hiệu năng ổn định.
* Tiết kiệm chi phí.
* Đội ngũ phát triển ứng dụng không quá lớn.
* Ứng dụng tin cậy, ổn định.
* Xây dựng ứng dụng ít native code nhất cho nhiều hệ điều hành khác nhau.
* Trải nghiệm người dùng tốt hơn khi so sánh với ứng dụng Hybrid.
* **Nhược điểm:**
* Yêu cầu Native code.
* Hiệu năng kém hơn so với Native App.
* Bảo mật chưa thật sự tốt do dùng JS.
* Quản lý bộ nhớ.
* Tùy biến chưa thật sự tốt ở một số module.

### Giới thiệu công nghệ Libarary React redux.

#### ****Redux là gì?****

**Redux** là một predictable state management tool cho các ứng dụng Javascript. Nó giúp bạn viết các ứng dụng hoạt động một cách nhất quán, chạy trong các môi trường khác nhau (client, server, and native) và dễ dàng để test. [**Redux**](https://topdev.vn/blog/?s=redux) ra đời lấy cảm hứng từ tư tưởng của ngôn ngữ **Elm** và kiến trúc **Flux** của Facebook. Do vậy Redux thường dùng kết hợp với React.

* **Lý do ra đời Redux :**

Do yêu cầu cho các ứng dụng single-page sử dụng Javascript ngày càng trở lên phức tạp thì code của chúng ta phải quản lý nhiều state hơn, Với Redux state của ứng dụng được giữ trong một nơi gọi là store và mỗi componentđều có thể access bất kỳ state nào mà chúng muốn từ chúng store này.

#### ****Tại sao ta lại cần state management tool****

Hầu hết các thư viện như React, Angular, etc được xây dựng theo một cách sao cho các components đến việc quản lý nội bộ các state của chúng mà không cần bất kỳ một thư viện or tool nào từ bên ngoài.

Nó sẽ hoạt động tốt với các ứng dụng có ít components nhưng khi ứng dụng trở lên lớn hơn thì việc quản lý states được chia sẻ qua các components sẽ biến thành các công việc lặt nhặt.

Trong một app nơi data được chia sẻ thông qua các components, rất dễ nhầm lẫn để chúng ta có thể thực sự biết nơi mà một state đang live. Một sự lý tưởng là data trong một component nên live trong chỉ một component. Vì vậy việc share data thông qua các components anh em sẽ trở nên khó khăn hơn.

Ví dụ, trong react để share data thông qua các components anh em, một state phải hoạt động trong component cha. Một method để cập nhật chính state này sẽ được cung cấp bởi chính component cha này và truyền như props đến các components con.

* **Đây là một ví dụ:**



Giờ chúng ta hãy tưởng tượng rằng nếu một state phải được chia sẻ giữa các component cách khá xa nhau trong một cây components và state này phải được truyền từ một component đến một component khác cho đến khi nó đến được nơi mà nó được gọi.

Cơ bản là state mà chúng ta đang nói đến phải được nhấc lên một component cha gần nhất và tiếp nữa cho đến khi nó đến được cái component tổ tiên chứa tất cả các components nó cần cái state này sau đó truyền cái state này xuống. Điều này sẽ khiến state trở nên khó hơn trong việc maintain và less predictable.

Điều này khiến cho bộ phận quản lý state trong app trở lên bừa bộn cũng như app trở nên vô cùng phức tạp. Đó là lý do tại sao chúng ta cần một state management tool như Redux.

## Chức năng ứng dụng

### Tổng quan.

* Chức năng Server side.
* Chức năng Client side.

### Chức năng Server side.

#### Api service

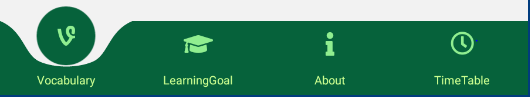
* Kết xuất dữ liệu từ database tạo ra các danh sách cần cho việc truy vấn từ các client:
  + Danh sách các chủ điểm từ vựng (VocabularyCategory).
  + Danh sách từ vựng theo một chủ điểm.
  + Danh sách từ vựng của tất cả các chủ điểm được sắp xếp theo các tiêu chí tăng dần, giảm dần….
* Phần quyền user:
  + Nhân viên nhập liệu : nhân viên nhập liệu thuộc đội học thuộc tiếng Anh, nhập liệu các chủ điểm từ vựng và thông tin từ vựng căn bản đầu tiên của app, tạo dữ liệu cho lần đầu khi người dùng sử dụng app.
  + Người dùng app:
    - có thể thêm các chủ điểm mới và thông tin từ vựng cho chủ điểm mới đó.
    - Có thể chia sẻ hoặc không chia sẻ các chủ điểm của mình tạo ra cho mọi người
* Chức năng cho phép người dùng tạo chủ điểm mới và thông tin từ vựng cho chủ điểm đó, chủ điểm mới tạo ra sẽ ở chế độ private – chỉ cho phép người tạo ra được phép truy xuất.
* Chức năng cho phép người dùng chia sẻ các chủ điểm của mình cho mọi người.
* Chức năng cho phép người dùng tạo các danh sách từ vựng từ các category đã tạo của mình và của người khác ở chế độ công khai (Learning Goal), danh sách này có ý nghĩa là mục tiêu học theo phân cụm người dùng tự tạo ra.
* Chức năng cho phép người dùng lập lịch báo thức ( TableTime ) và kích hoạt app vào các khung giờ do người dùng định nghĩa, Mỗi một TableTime sẽ gắn với một danh sách mục tiêu học ( LearningGoal ).

#### Giao dien admin

### Chức năng Client side.

#### Giao diện trang Home của app:

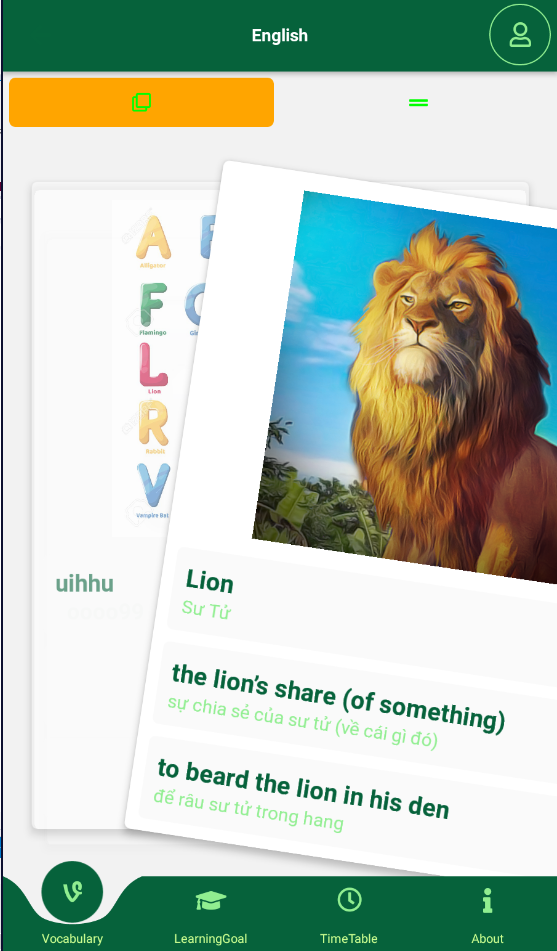
* Hiển thị các điều hướng đến các chúc năng (Danh mục từ vựng – Vocabulary Category, Mục tiêu học – LearningGoal, thông tin – About, Lặp lịch - TableTime).



Hình 1: Menu điều hướng

#### Giao diện Từ Vựng – Vocabulary Category:

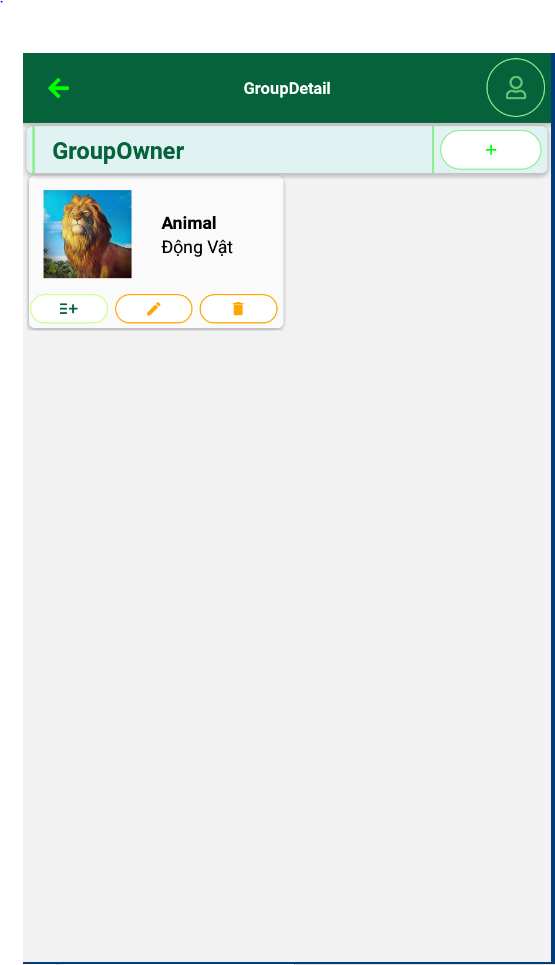
* các danh mục sẽ được chia theo ba Group (GroupOwner - Bản thân người dùng, GroupAdmin – Do hệ thống nhập liệu, GroupOther - của những người khác).



Hình 2: Giao diện vocabulary và vocabulary Category

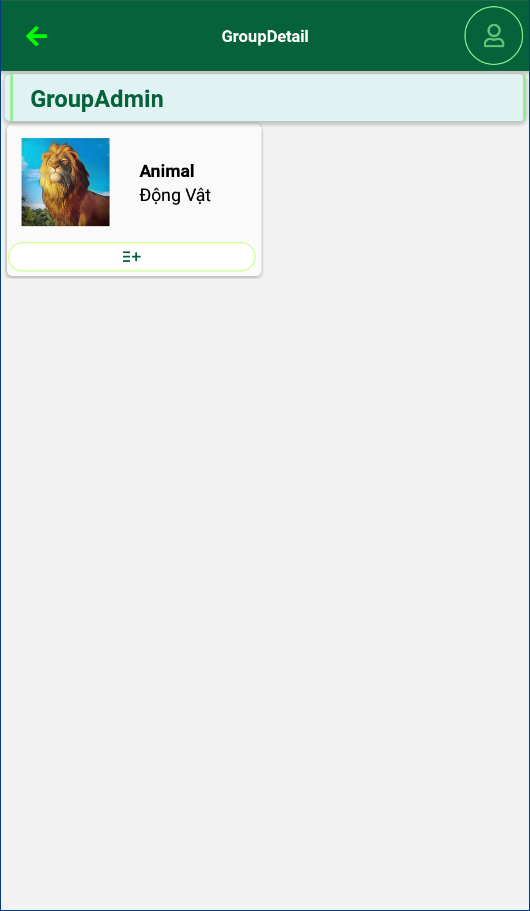
#### Giao diện hiển thị tất cả Category của một Group

* Group là của Owner: giao diện sẽ cho phép thao tác thêm, xóa, sửa đổi.



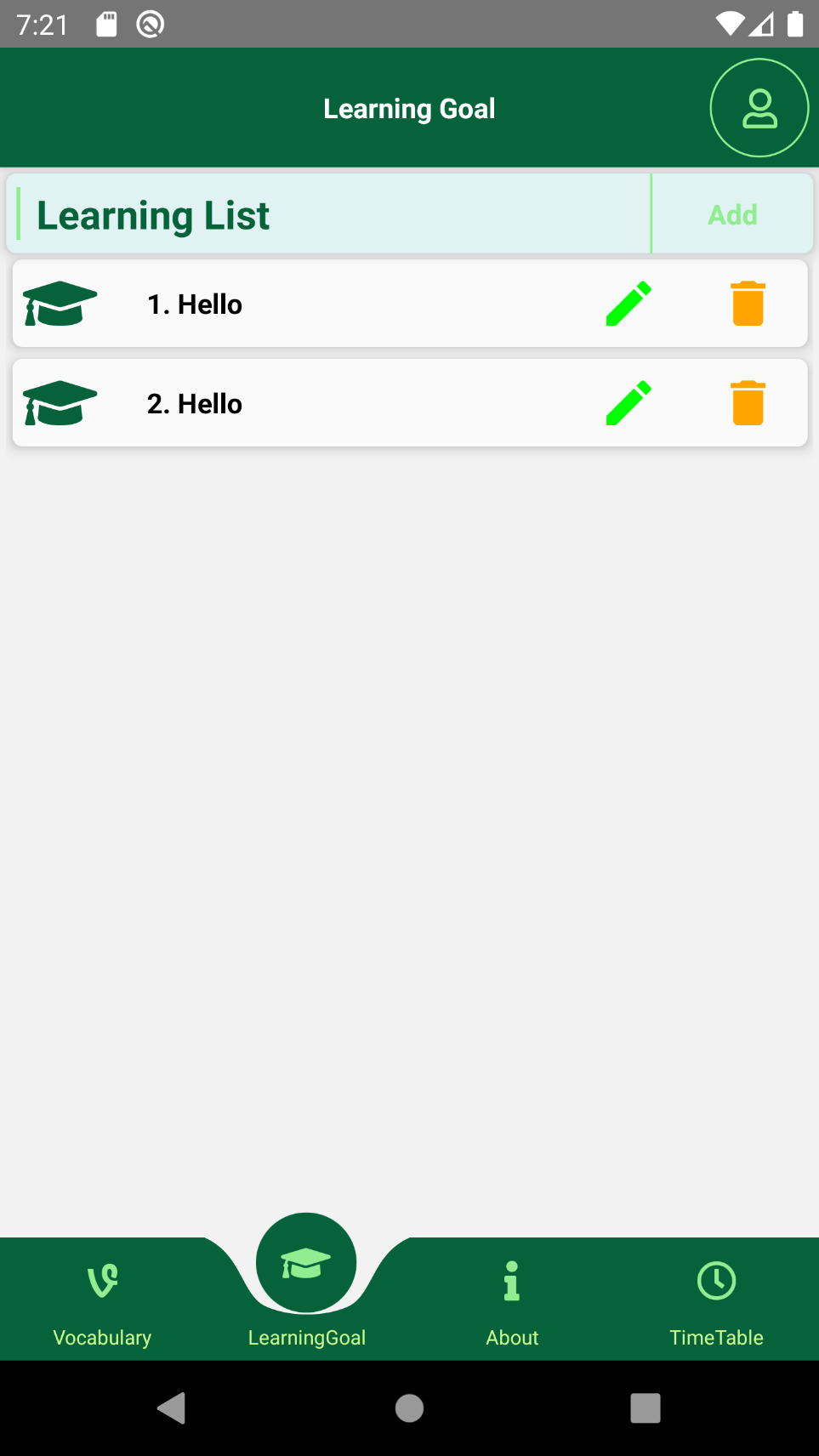
Hình 3: Giao diện chi tiết một GroupOwner.

* GroupAdmin và GroupOther: Chương trình chỉ cho người dùng được xem.



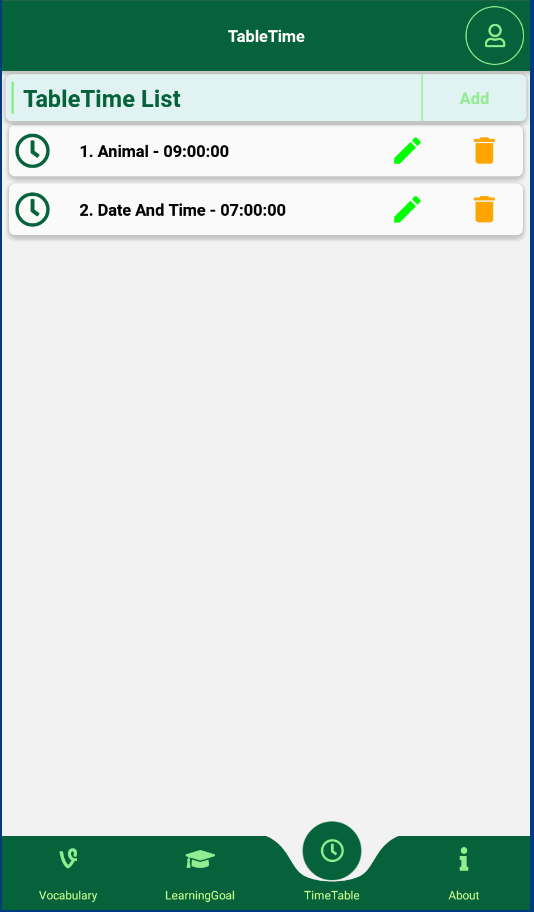
Hình 4: Chi tiết GroupAdmin.

#### Giao diện mục tiêu học – Learning Goal



Hình 5: Giao diện LearningGoal.

#### Giao diện Lặp Lịch – TableTime

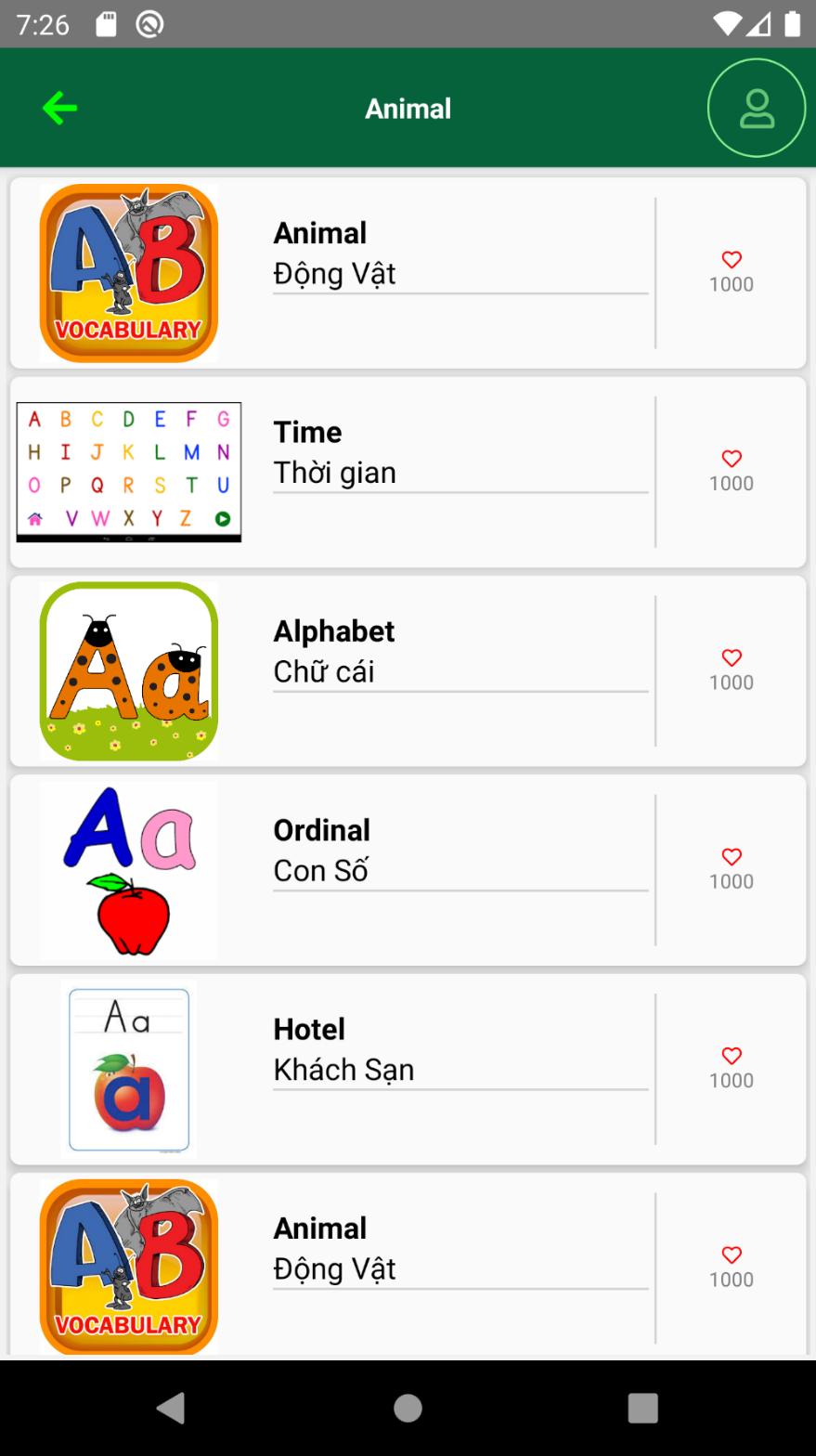


Hình 6: Giao diện TableTime

#### Giao diện thông tin – About

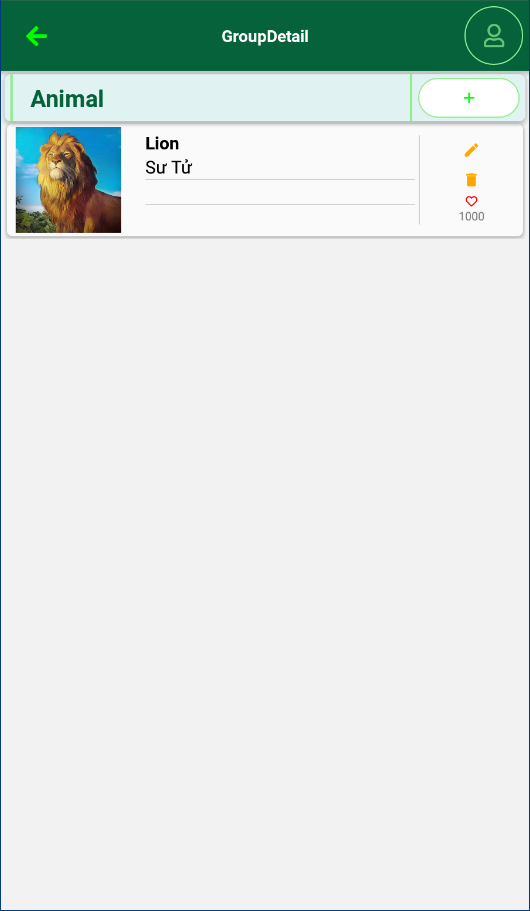
#### Giao diện chi tiết của một chủ điểm - Vocabulary category

* + Các chủ điểm của hệ thống và người dùng khác



Hình 7: Giao diện chi tiết Chủ điểm Của Hệ Thóng và người dùng khác.

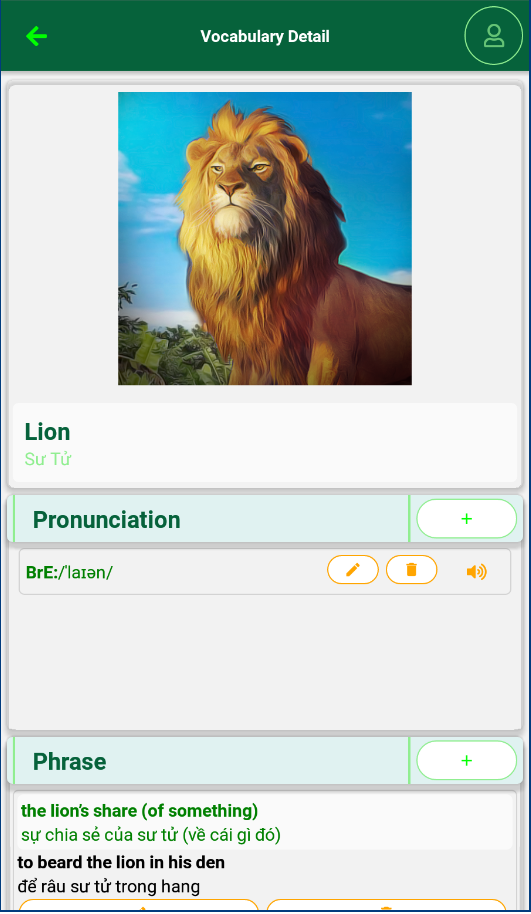
* + Chủ điểm do người dùng tạo ra.



Hình 8: Giao diện chi tiết chủ điểm người dùng.

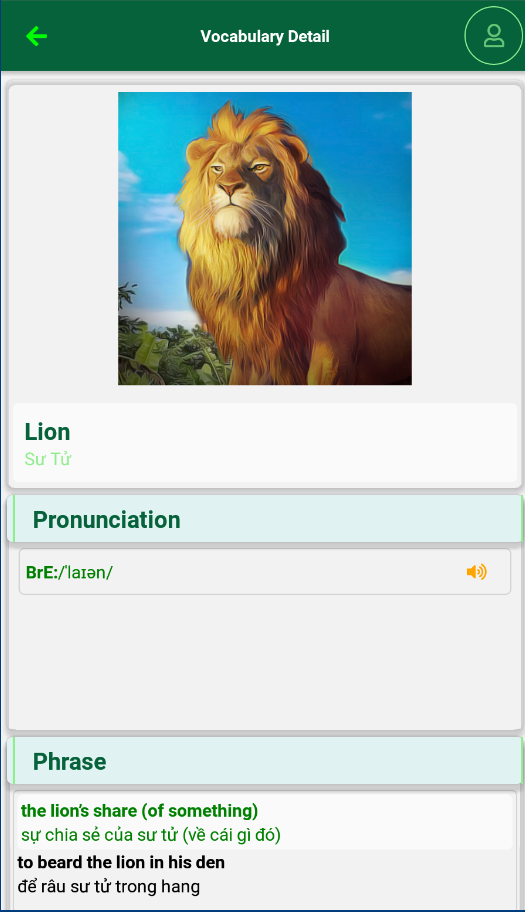
#### Giao chi tiết từ vựng.

* Người dùng.



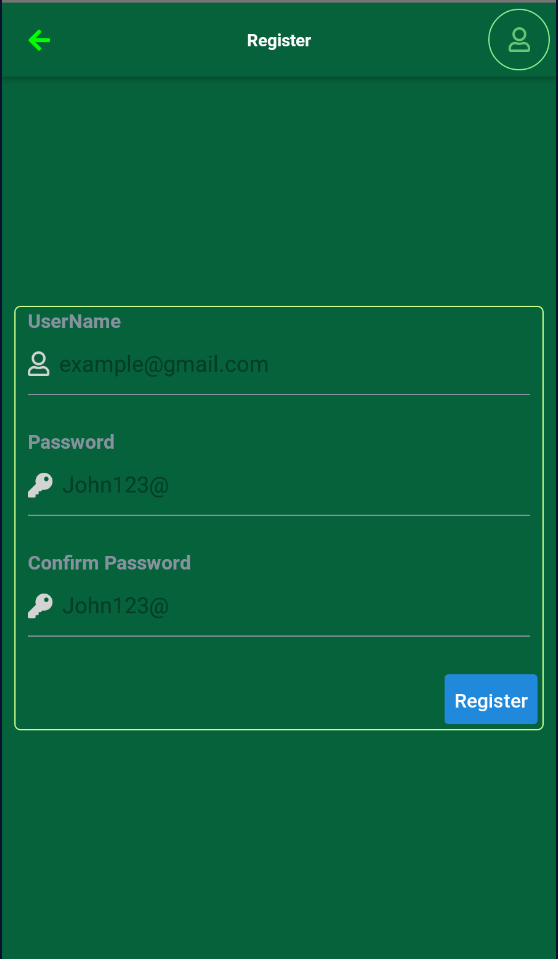
Hình 9:Chi tiết từ vựng của người dùng.

* Hệ thống và người dùng khác.



Hình 10: Chi tiết từ vựng của hệ thống và người dùng khác

* Giao diện đăng nhập (login), đăng ký (register).



Hình : Giao diện đăng nhập, đăng ký

# CHƯƠNG 2: SƠ LƯỢC VỀ CÔNG CỤ XÂY DỰNG HỆ THỐNG.

## Giới thiệu Visual Studio Code.

#### Visual studio Code là gì?

**Visual Studio Code** (VS Code hay VSC) là một trong những trình soạn thảo mã nguồn phổ biến nhất được sử dụng bởi các lập trình viên. Nhanh, nhẹ, hỗ trợ đa nền tảng, nhiều tính năng và là mã nguồn mở chính là những ưu điểm vượt trội khiến VS Code ngày càng được ứng dụng rộng rãi. Visual Studio Code được phát triển bởi Microsoft. Nó được xem là một sự kết hợp hoàn hảo giữa IDE và Code Editor.

Visual Studio Code hỗ trợ chức năng debug, đi kèm với Git, có syntax highlighting, tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho phép người dùng thay đổi theme, phím tắt, và các tùy chọn khác.

#### Một số tính năng của Visual Studio.

* **Hỗ trợ nhiều ngôn ngữ lập trình**

Visual Studio Code hỗ trợ nhiều ngôn ngữ lập trình như C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript, … Vì vậy, nó dễ dàng phát hiện và đưa ra thông báo nếu chương chương trình có lỗi.

* **Hỗ trợ đa nền tảng**

Các trình viết code thông thường chỉ được sử dụng hoặc cho Windows hoặc Linux hoặc Mac Systems. Nhưng Visual Studio Code có thể hoạt động tốt trên cả ba nền tảng trên.

* **Cung cấp kho tiện ích mở rộng**

Trong trường hợp lập trình viên muốn sử dụng một ngôn ngữ lập trình không nằm trong số các ngôn ngữ Visual Studio hỗ trợ, họ có thể tải xuống tiện ích mở rộng. Điều này vẫn sẽ không làm giảm hiệu năng của phần mềm, bởi vì phần mở rộng này hoạt động như một chương trình độc lập.

* **Kho lưu trữ an toàn.**

Đi kèm với sự phát triển của lập trình là nhu cầu về lưu trữ an toàn. Với Visual Studio Code, người dùng có thể hoàn toàn yên tâm vì nó dễ dàng kết nối với Git hoặc bất kỳ kho lưu trữ hiện có nào.

* **Hỗ trợ web.**

Visual Studio Code hỗ trợ nhiều ứng dụng web. Ngoài ra, nó cũng có một trình soạn thảo và thiết kế website.

* **Lưu trữ dữ liệu dạng phân cấp.**

Phần lớn tệp lưu trữ đoạn mã đều được đặt trong các thư mục tương tự nhau. Ngoài ra, Visual Studio Code còn cung cấp các thư mục cho một số tệp đặc biệt quan trọng.

* **Hỗ trợ viết Code.**

Một số đoạn code có thể thay đổi chút ít để thuận tiện cho người dùng. Visual Studio Code sẽ đề xuất cho lập trình viên các tùy chọn thay thế nếu có.

* **Hỗ trợ thiết bị đầu cuối.**

Visual Studio Code có tích hợp thiết bị đầu cuối, giúp người dùng khỏi phải chuyển đổi giữa hai màn hình hoặc trở về thư mục gốc khi thực hiện các thao tác.

* **Màn hình đa nhiệm.**

Người dùng Visual Studio Code có thể mở cùng lúc nhiều tệp tin và thư mục – mặc dù chúng không hề liên quan với nhau.

* **Intellisense.**

Hầu hết các trình viết mã đều có tính năng nhắc mã Intellisense, nhưng ít chương trình nào chuyên nghiệp bằng Visual Studio Code. Nó có thể phát hiện nếu bất kỳ đoạn mã nào không đầy đủ. Thậm chí, khi lập trình viên quên không khai báo biến, Intellisense sẽ tự động giúp họ bổ sung các cú pháp còn thiếu.

* **Hỗ trợ Git.**

Visual Studio Code hỗ trợ kéo hoặc sao chép mã trực tiếp từ GitHub. Mã này sau đó có thể được thay đổi và lưu lại trên phần mềm.

* **Bình luận.**

Việc để lại nhận xét giúp người dùng dễ dàng nhớ công việc cần hoàn thành.

## Giới thiệu về Visual Studio.

#### Microsoft Visual Studio.

**Microsoft Visual Studio** là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C,[4] C++ và C++/CLI (thông qua Visual C++), VB.NET (thông qua Visual Basic.NET), C# (thông qua Visual C#) và F# (như của Visual Studio 2010). Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng rẽ. Nó cũng hỗ trợ XML/XSLT, HTML/XHTML, JavaScript và CSS.

Microsoft cung cấp phiên bản "Express" (đối với phiên bản Visual Studio 2013 trở về trước) và "Community" (đối với bản Visual Studio 2015 trở về sau) là phiên bản miễn phí của Visual Studio.

#### Cấu trúc.

Visual Studio không hỗ trợ cho bất kỳ ngôn ngữ lập trình nào về giải pháp hoặc công cụ thực chất, thay vào đó nó cho phép cắm chức năng được mã hóa như là một VSPackage. Khi cài đặt, các chức năng có sẵn như là một dịch vụ. IDE cung cấp ba dịch vụ: SVsSolution cung cấp khả năng liệt kê các dự án và các giải pháp; SVsUIShell cung cấp cửa sổ và giao diện người dùng và SVsShell. Ngoài ra, IDE cũng có trách nhiệm điều phối và cho phép truyền thông giữa các dịch vụ. Tất cả các biên tập viên, nhà thiết kế, các loại dự án và các công cụ khác được thực hiện theo VSPackages. Visual Studio sử dụng COM để truy cập VSPackages. Visual Studio SDK cũng bao gồm Managed Package Framework (MPF) là một tập hợp quản lý bao bọc quanh các COM-interfaces cho phép các gói được viết bằng bất kỳ ngôn ngữ nào.Tuy nhiên, MPF không cung cấp tất cả các chức năng bộc lộ trong Visual Studio COM-interfaces.Các dịch vụ có thể được tiêu thụ để tạo ra các gói khác, để thêm chức năng cho Visual Studio IDE.

Hỗ trợ cho các ngôn ngữ lập trình được thêm vào bằng cách sử dụng một VSPackage đặc biệt được gọi là một dịch vụ ngôn ngữ. Một dịch vụ ngôn ngữ định nghĩa giao tiếp khác nhau mà việc thực hiện VSPackage có thể thực hiện để hỗ trợ thêm cho các chức năng khác nhau. Các chức năng có thể được thêm vào theo cách này bao gồm cú pháp màu, hoàn thành báo cáo kết quả, kết hợp đôi, công cụ chú giải tham số thông tin, danh sách thành viên và đánh dấu lỗi trên nền biên dịch. Nếu giao diện được thực hiện, các tính năng sẽ có sẵn ngôn ngữ. Dịch vụ ngôn ngữ sẽ được thực hiện trên cơ sở mỗi ngôn ngữ. Việc triển khai có thể tái sử dụng mã từ phân tích cú pháp hoặc trình biên dịch cho ngôn ngữ.Dịch vụ ngôn ngữ có thể được triển khai hoặc trong mã nguồn gốc hoặc mã số quản lý. Đối với mã nguồn gốc, thì cả COM-interfaces gốc hoặc Babel Framework (một phần của Visual Studio SDK) đều có thể được sử dụng.Đối với mã số quản lý thì các MPF sẽ bao hàm các dịch vu quản lý văn bản.

Visual Studio không bao gồm bất kỳ Hệ thống quản lý phiên bản hỗ trợ kiểm soát mã nguồn nhưng nó xác định hai cách thay thế cho các hệ thống kiểm soát mã nguồn để tích hợp với IDE. Một VSPackage kiểm soát mã nguồn có thể cung cấp giao diện người dùng tùy chỉnh của riêng mình. Ngược lại, một plugin kiểm soát mã nguồn bằng cách sử dụng MSSCCI (Microsoft Source Code Control Interface) cung cấp một tập các chức năng được sử dụng để thực hiện chức năng kiểm soát mã nguồn khác nhau, với một giao diện người dùng Visual Studio tiêu chuẩn.MSSCCI lần đầu tiên được sử dụng để tích hợp Visual SourceSafe với Visual Studio 6.0 nhưng sau đó được mở ra thông qua Visual Studio SDK. Visual Studio.NET 2002 dùng MSSCCI 1.1, và Visual Studio.NET 2003 dùng MSSCCI 1.2. Visual Studio 2005, 2008 và 2010 dùng MSSCCI 1.3.

Visual Studio hỗ trợ chạy nhiều cá thể của môi trường (tất cả đều có VSPackages riêng của mình). Những trường hợp sử dụng các registry hives khác nhau để lưu trữ trạng thái cấu hình và được phân biệt bởi AppID (Application ID). Các trường hợp được đưa ra bởi một AppId-specific.exe cụ thể mà lựa chọn AppID, thiết lập các hive gốc và khởi chạy IDE. VSPackages đăng ký một AppID được tích hợp với VSPackages khác cho AppID đó. Các phiên bản sản phẩm khác nhau của Visual Studio được tạo ra bằng cách sử dụng AppIds khác nhau. Các sản phẩm phiên bản Visual Studio Express được cài đặt với AppIds riêng nhưng với các sản phẩm Standard, Professional và Team Suite chia sẻ cùng AppID. Do đó, người ta có thể cài đặt các phiên bản Express song song với các phiên bản khác, không giống như các phiên bản khác cập nhật các cài đặt tương tự. Phiên bản Professional bao gồm các VSPackages khổng lồ trong phiên bản Standard và Team. Hệ thống AppID được thừa hưởng bởi Visual Studio Shell trong Visual Studio 2008.

#### Tính năng.

* **Biên tập mã**

Giống như bất kỳ IDE khác, nó bao gồm một trình soạn thảo mã hỗ trợ tô sáng cú pháp và hoàn thiện mã bằng cách sử dụng IntelliSense không chỉ cho các biến, hàm và các phương pháp mà còn các cấu trúc ngôn ngữ như vòng điều khiển hoặc truy vấn. IntelliSense được hỗ trợ kèm theo cho các ngôn ngữ như XML, Cascading Style Sheets và JavaScript khi phát triển các trang web và các ứng dụng web. Các đề xuất tự động hoàn chỉnh được xuất hiện trong một hộp danh sách phủ lên trên đỉnh của trình biên tập mã. Trong Visual Studio 2008 trở đi, nó có thể được tạm thời bán trong suốt để xem mã che khuất bởi nó. Các trình biên tập mã được sử dụng cho tất cả các ngôn ngữ được hỗ trợ.

Các trình biên tập mã Visual Studio cũng hỗ trợ cài đặt dấu trang trong mã để điều hướng nhanh chóng. Hỗ trợ điều hướng khác bao gồm thu hẹp các khối mã lệnh và tìm kiếm gia tăng, ngoài việc tìm kiếm văn bản thông thường và tìm kiếm Biểu thức chính quy Các trình biên tập mã cũng bao gồm một bìa kẹp đa mục và một danh sách công việc. Các trình biên tập mã hỗ trợ lưu lại các đoạn mã được lặp đi lặp lại nhằm để chèn vào mã nguồn sử dụng về sau. Một công cụ quản lý cho đoạn mã được xây dựng là tốt. Những công cụ này nổi lên như các cửa sổ trôi nổi có thể được thiết lập để tự động ẩn khi không sử dụng hoặc neo đậu đến các cạnh của màn hình. Các trình biên tập mã Visual Studio cũng hỗ trợ cải tiến mã nguồn bao gồm tham số sắp xếp lại, biến và phương pháp đổi tên, khai thác và đóng gói giao diện các lớp thành viên bên trong những trạng thái giữa những thứ khác.

Visual Studio có tính năng biên dịch nền (còn gọi là biên dịch gia tăng). Như mã đang được viết, Visual Studio biên dịch nó trong nền để cung cấp thông tin phản hồi về cú pháp và biên dịch lỗi, được đánh dấu bằng một gạch dưới gợn sóng màu đỏ. Biên dịch nền không tạo ra mã thực thi, vì nó đòi hỏi một trình biên dịch khác hơn là để sử dụng tạo ra mã thực thi. Biên dịch nền ban đầu được giới thiệu với Microsoft Visual Basic nhưng bây giờ đã được mở rộng cho tất cả các ngôn ngữ.

* **Trình gỡ lỗi.**

Visual Studio có một trình gỡ lỗi hoạt động vừa là một trình gỡ lỗi cấp mã nguồn và là một trình gỡ lỗi cấp máy. Nó hoạt động với cả hai mã quản lý cũng như ngôn ngữ máy và có thể được sử dụng để gỡ lỗi các ứng dụng được viết bằng các ngôn ngữ được hỗ trợ bởi Visual Studio. Ngoài ra, nó cũng có thể đính kèm theo quy trình hoạt động và theo dõi và gỡ lỗi những quy trình. Nếu mã nguồn cho quá trình hoạt động có sẵn, nó sẽ hiển thị các mã như nó đang được chạy. Nếu mã nguồn không có sẵn, nó có thể hiển thị các tháo gỡ. Các Visual Studio debugger cũng có thể tạo bãi bộ nhớ cũng như tải chúng sau để gỡ lỗi. Các chương trình đa luồng cao cấp cũng được hỗ trợ. Trình gỡ lỗi có thể được cấu hình sẽ được đưa ra khi một ứng dụng đang chạy ngoài Visual Studio bị treo môi trường.

Trình gỡ lỗi cho phép thiết lập các breakpoint (mà cho phép thực thi được tạm thời dừng lại tại một vị trí nhất định) và watch (trong đó giám sát các giá trị của biến là việc thực hiện tiến bộ). Breakpoint có thể có điều kiện, nghĩa là chúng được kích hoạt khi điều kiện được đáp ứng. Mã có thể được biểu diễn, tức là chạy một dòng (của mã nguồn) tại một thời điểm. Nó có hoặc là bước sang các chức năng để gỡ lỗi bên trong nó, hoặc là nhảy qua nó, tức là, việc thực hiện các chức năng không có sẵn để kiểm tra thủ công. Trình gỡ lỗi hỗ trợ Edit and Continue, nghĩa là, nó cho phép mã được chỉnh sửa khi nó đang được sửa lỗi (chỉ có 32 bit, không được hỗ trợ trong 64 bit). Khi gỡ lỗi, nếu con trỏ chuột di chuyển lên bất kỳ biến, giá trị hiện tại của nó được hiển thị trong phần chú giải ("chú thích dữ liệu"), nơi mà nó cũng có thể được thay đổi nếu muốn. Trong quá trình viết mã, các trình gỡ lỗi của Visual Studio cho phép một số chức năng được gọi ra bằng tay từ cửa sổ công cụ Immediate. Các thông số cho phương thức được cung cấp tại các cửa sổ Immediate.

* **Thiết kế**

Windows Forms Designer: được sử dụng để xây dựng GUI sử dụng Windows Forms; bố trí có thể được xây dựng bằng các nút điều khiển bên trong hoặc khóa chúng vào bên cạnh mẫu. Điều khiển trình bày dữ liệu (như hộp văn bản, hộp danh sách, vv) có thể được liên kết với các nguồn dữ liệu như cơ sở dữ liệu hoặc truy vấn. Các điều khiển dữ liệu ràng buộc có thể được tạo ra bằng cách rê các mục từ cửa sổ nguồn dữ liệu lên bề mặt thiết kế. Các giao diện người dùng được liên kết với mã sử dụng một mô hình lập trình hướng sự kiện. Nhà thiết kế tạo ra bằng C thăng hay VB.NET cho ứng dụng.

WPF Designer: có tên mã là Cider, được giới thiệu trong Visual Studio 2008. Giống như Windows Forms Designer, hỗ trợ kéo và thả ẩn dụ. Sử dụng tương tác người-máy nhắm mục tiêu theo Windows Presentation Foundation. Nó hỗ trợ các chức năng WPF bao gồm kết nối dữ liệu và tự động hóa bố trí quản lý. Nó tạo ra mã XAML cho giao diện người dùng. Các tập tin XAML được tạo ra là tương thích với Microsoft Expression Design, sản phẩm thiết kế theo định hướng. Các mã XAML được liên kết với mã đang sử dụng một mô hình code-behind.

Web designer/development: Visual Studio cũng bao gồm một trình soạn thảo và thiết kế trang web cho phép các trang web được thiết kế bằng cách kéo và thả các đối tượng. Nó được sử dụng để phát triển các ứng dụng ASP.NET và hỗ trợ HTML, CSS và JavaScript. Nó sử dụng mô hình code-behind để liên kết với mã ASP.NET. Từ Visual Studio 2008 trở đi, công cụ bố trí được sử dụng bởi các nhà thiết kế web được chia sẻ với Microsoft Expression Web. Ngoài ra ASP.NET MVC Framework hỗ trợ cho công nghệ MVC là tải xuống riêng biệt và dự án ASP.NET Dynamic Data có sẵn từ Microsoft.

Class designer: Các lớp thiết kế được dùng để biên soạn và chỉnh sửa các lớp (bao gồm cả các thành viên và truy cập của chúng) sử dụng mô hình UML. Các lớp thiết kế có thể tạo ra mã phác thảo C thăng và VB.NET cho các lớp và cá phương thức. Nó cũng có thể tạo ra sơ đồ lớp từ các lớp viết tay.

Data designer: Thiết kế dữ liệu có thể được sử dụng để chỉnh sửa đồ họa giản đồ cơ sở dữ liệu bao gồm các bảng, khóa chính, khóa ngoại và các rằng buộc. Nó cũng có thể được sử dụng để thiết kế các truy vấn từ các giao diện đồ họa.

Mapping designer: Từ Visual Studio 2008 trở đi, thiết kế ánh xạ được dùng bởi Language Integrated Query để thiết kế các ánh xạ giữa các giản đồ cơ sở dữ liệu và các lớp để đóng gói dữ liệu. Các giải pháp mới từ cách tiếp cận ORM, ADO.NET Entity Framework sẽ thay thế và cải thiện các công nghệ cũ.

* **Các công cụ khác**

Open Tabs Browser: được sử dụng để liệt kê tất cả thẻ đang mở và chuyển đổi giữa chúng. Được viện dẫn bằng cách sử dụng CTRL+TAB.

Properties Editor: được sử dụng để chỉnh sửa các thuộc tính trong một cửa sổ giao diện bên trong Visual Studio. Nó liệt kê tất cả các thuộc tính có sẵn (gồm chỉ đọc và những thuộc tính có thể được thiết lập) cho tất cả các đối tượng bao gồm các lớp, biểu mẫu, trang web và các hạng mục khác.

Object Browser: là một không gian tên và trình duyệt lớp thư viện cho Microsoft NET. Nó có thể được sử dụng để duyệt các không gian tên (được sắp xếp theo thứ bậc) trong Assembly (CLI). Các hệ thống phân cấp có thể hoặc không có thể phản ánh các tổ chức trong hệ thống tập tin.

Solution Explorer: theo cách nói trong Visual Studio, một giải pháp là một tập hợp các tập tin mã và các nguồn khác được sử dụng để xây dựng một ứng dụng. Các tập tin trong một giải pháp được sắp xếp theo thứ bậc, mà có thể có hoặc không thể phản ánh các tổ chức trong hệ thống tập tin. Solution Explorer được sử dụng để quản lý và duyệt các tập tin trong một giải pháp.

Team Explorer: được sử dụng để tích hợp các khả năng của Team Foundation Server, Revision Control System và là cơ sở cho môi trường CodePlex đối với dự án mã nguồn mở. Ngoài việc kiểm soát nguồn nó cung cấp khả năng xem và quản lý các công việc riêng lẻ (bao gồm cả lỗi, nhiệm vụ và các tài liệu khác) và để duyệt thống kê TFS. Nó được bao gồm như là một phần của một cài đặt TFS và cũng có sẵn để tải xuống cho Visual Studio. Team Explorer cũng có sẵn như là một môi trường độc lập duy nhất để truy cập các dịch vụ TFS.

Data Explorer: được sử dụng để quản lý cơ sở dữ liệu trên Microsoft SQL Server. Nó cho phép tạo ra và sửa đổi các bảng cơ sở dữ liệu (hoặc bằng cách ban hành các lệnh T-SQL hoặc bằng cách sử dụng các thiết kế dữ liệu). Nó cũng có thể được sử dụng để tạo các truy vấn và các thủ tục lưu trữ trong T-SQL hoặc trong Managed code thông qua SQL CLR. Có sẵn gỡ lỗi và hỗ trợ IntelliSense.

Server Explorer: công cụ được sử dụng để quản lý các kết nối cơ sở dữ liệu trên một máy tính truy cập được. Nó cũng được sử dụng để duyệt chạy Windows Services, quầy thực hiện, Windows Event Log và hàng đợi tin nhắn và sử dụng chúng như một nguồn dữ liệu.

Dotfuscator Software Services Community Edition: Visual Studio bao gồm một phiên bản light của sản phẩm PreEmptive Solutions' Dotfuscator cho mã gây rối và giảm kích thước ứng dụng. Khởi đầu với Visual Studio 2010, phiên bản này của Dotfuscator sẽ bao gồm khả năng Runtime Intelligence cho phép tác giả thu thập cách sử dụng của người dùng cuối, hiệu suất, tính ổn định và các thông tin từ các ứng dụng của họ chạy trong sản xuất.

Text Generation Framework: Visual Studio bao gồm một khung tạo văn bản đầy đủ được gọi là Text Template Transformation Toolkit T4 cho phép Visual Studio tạo ra tập tin văn bản từ các mẫu hoặc trong IDE hoặc thông qua mã.

ASP.NET Web Site Administration Tool: công cụ quản trị trang web ASP.NET cho phép cấu hình các trang web ASP.NET.

Visual Studio Tools for Office: Công cụ Visual Studio cho Ofice là một SDK và một add-in cho Visual Studio bao gồm các công cụ để phát triển cho các bộ Microsoft Office. Trước đây (với Visual Studio.NET 2003 và Visual Studio 2005) đó là một SKU riêng biệt mà chỉ hỗ trợ Visual C# Visual Basic.NET hoặc đã được đưa vào Team Suite. Với Visual Studio 2008, nó không còn là một SKU riêng biệt nhưng lại kèm trong các phiên bản chuyên nghiệp và cao hơn. Một thời gian chạy riêng biệt được yêu cầu khi triển khai các giải pháp VSTO.

* **Khả năng mở rộng.**

Visual Studio cho phép các nhà phát triển viết các phần mở rộng cho Visual Studio để mở rộng tính năng của nó. Những phần mở rộng "cắm vào" Visual Studio và mở rộng tính năng của nó. Các phần mở rộng đến ở dạng macro, add-in và các gói. Các macro đại diện cho các nhiệm vụ lặp đi lặp lại và hành động mà các nhà phát triển có thể ghi lại theo chương trình để tiết kiệm, p231 các cài đặt. Các chế độ biệt lập của vỏ tạo ra một AppID mới, nơi các gói được cài đặt. Những thứ này được bắt đầu với một thực thi khác nhau. Nó nhằm mục đích cho sự phát triển của môi trường phát triển tùy chỉnh, hoặc cho một ngôn ngữ cụ thể hoặc một kịch bản cụ thể. Các chế độ tích hợp cài đặt các gói vào AppID của các phiên bản Professional / Standard / Team System, do đó các công cụ tích hợp vào các phiên bản. Visual Studio Shell là miễn phí tải về.

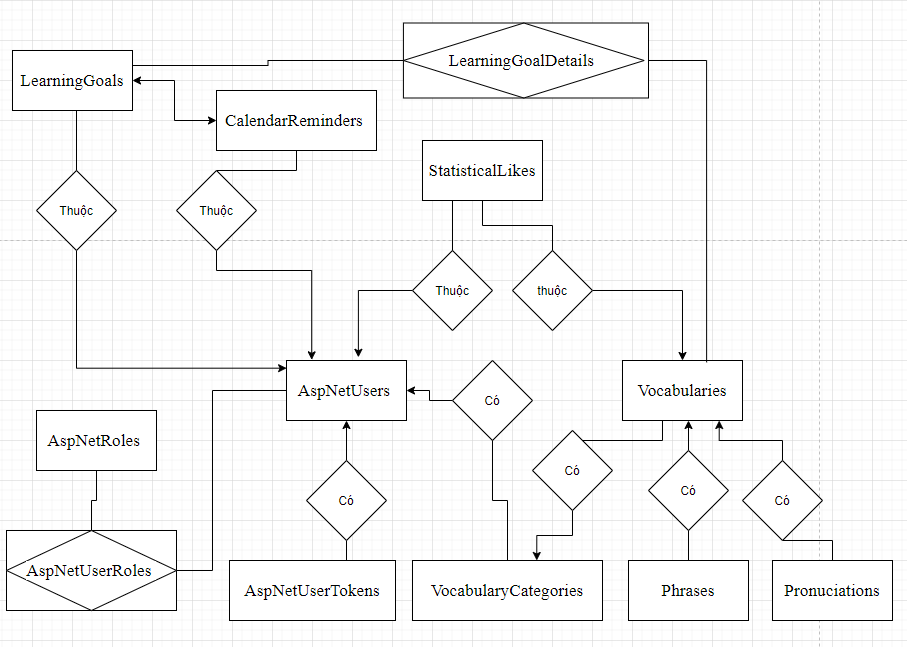
Sau khi phát hành Visual Studio 2008, Microsoft đã tạo ra Visual Studio Gallery. Nó phục vụ như vị trí trung tâm cho đăng tải thông tin về phần mở rộng cho Visual Studio. Phát triển cộng đồng cũng như phát triển thương mại có thể tải lên thông tin về các phần mở rộng của họ đến Visual Studio.NET 2002 thông qua Visual Studio 2010. Người sử dụng trang web có thể đánh giá và xem lại các phần mở rộng để giúp đánh giá chất lượng các phần mở rộng được đăng. RSS feed thông báo cho người dùng trên bản cập nhật tới trang web và các tính năng gắn thẻ cũng được lên kế hoạch.

# CHƯƠNG 3: THIẾT KẾ DATABASE.

## Xác định thực thể.

* AspNetUsers (**Id**,UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount).
* AspNetRoles (**Id**, Name, NormalizedName, ConcurrencyStamp).
* AspNetUserTokens( **UserId**, **LoginProvider**, **Name**, Value).
* VocabularyCategories( **Id**, Name, NameVN, IsShare, IsPublish).
* Vocabularies( **Id**, Word, WordVN, Image, IsShare, IsPublish).
* Pronuciations( **Id**, Transcription, LinkFile, Type).
* Phrases( **Id**, Sentence, SentenceVN, Content, ContentVN).
* LearningGoal ( **Id**,DateCreated,Name,AmountVocabulary)
* CalendarReminders ( Name, Description, TableTime, LongTime, IsLoop, **Id**)
* StatisticalLikes( **UserId**, **IdVocabulary** )

## Mô hình ERD.



## Mô Hình Dữ Liệu Quan Hệ.

* AspNetUsers (**Id**,UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount).
* AspNetRoles (**Id**, Name, NormalizedName, ConcurrencyStamp).
* AspNetUserRoles ( **UserId, RoleId**)
* AspNetUserTokens( **UserId**, **LoginProvider**, **Name**, Value).
* VocabularyCategories( **Id**, Name, NameVN, IsShare, **UserId**).
* Vocabularies( **Id**, Word, WordVN, Image, IsShare, **VocabularyCategoryId**).
* Pronuciations( **Id**, Transcription, LinkFile, Type, **VocabularyId**).
* Phrases( **Id**, Sentence, SentenceVN, Content, ContentVN, **VocabularyId**).
* LearningGoals ( **Id**, DateCreated, Name, AmountVocabulary, **UserId**)
* LearningGoalDetails (**VocabularyId, LearningGoalId**)
* CalendarReminders ( Name, Description, TableTime, LongTime, IsLoop, **Id, UserId,** IdLearningGoal)
* StatisticalLikes( **UserId**, **IdVocabulary** )

## Từ điển dữ liệu.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | nvarchar(450) | X | X |  |  |
| 2 | UserName | nvarchar(256) |  |  |  |  |
| 3 | NormalizedUserName | nvarchar(256) |  |  |  |  |
| 4 | Email | nvarchar(256) |  |  |  |  |
| 5 | NormalizedEmail | nvarchar(256) |  |  |  |  |
| 6 | EmailConfirmed | bit | X |  |  |  |
| 7 | PasswordHash | nvarchar(MAX) |  |  |  |  |
| 8 | SecurityStamp | nvarchar(MAX) |  |  |  |  |
| 9 | ConcurrencyStamp | nvarchar(MAX) |  |  |  |  |
| 10 | PhoneNumber | nvarchar(MAX) |  |  |  |  |
| 11 | PhoneNumberConfirmed | bit | X |  |  |  |
| 12 | TwoFactorEnabled | bit | X |  |  |  |
| 13 | LockoutEnd | datetimeoffset(7) |  |  |  |  |
| 14 | LockoutEnabled | bit | X |  |  |  |
| 15 | AccessFailedCount | int | X |  |  |  |

* AspNetUsers (**Id**,UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount).
* AspNetRoles (**Id**, Name, NormalizedName, ConcurrencyStamp).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | nvarchar(450) | X | X |  |  |
| 2 | Name | nvarchar(256) |  |  |  |  |
| 3 | NormalizedName | nvarchar(256) |  |  |  |  |
| 4 | ConcurrencyStamp | nvarchar(MAX) |  |  |  |  |

* AspNetUserRoles ( **UserId, RoleId**)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | UserId | nvarchar(450) | X | X | X |  |
| 2 | RoleId | nvarchar(450) | X | X | X |  |

* AspNetUserTokens( **UserId**, **LoginProvider**, **Name**, Value).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | UserId | nvarchar(450) | X | X | X |  |
| 2 | LoginProvider | nvarchar(450) | X | X |  |  |
| 3 | Name | nvarchar(450) | X | X |  |  |
| 4 | Value | nvarchar(MAX) |  |  |  |  |

* VocabularyCategories( Id, Name, NameVN, IsShare, IsPublish, UserId).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier | X | X |  |  |
| 2 | Name | nvarchar(MAX) |  |  |  |  |
| 3 | NameVN | nvarchar(MAX) |  |  |  |  |
| 4 | IsShare | bit | X |  |  |  |
| 5 | IsPublish | bit | X |  |  |  |
| 6 | UserId | nvarchar(450) |  |  | X |  |

* Vocabularies( **Id**, Word, WordVN, Image, IsShare, IsPublish **, VocabularyCategoryId**).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier | X | X |  |  |
| 2 | Word | nvarchar(MAX) |  |  |  |  |
| 3 | WordVN | nvarchar(MAX) |  |  |  |  |
| 4 | Image | nvarchar(MAX) |  |  |  |  |
| 5 | VocabularyCategoryId | uniqueidentifier | X |  | X |  |
| 6 | IsShare | bit | X |  |  |  |
| 7 | IsPublish | bit | X |  |  |  |

* Pronuciations( **Id**, Transcription, LinkFile, Type, **VocabularyId**).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier | X | X |  |  |
| 2 | Transcription | nvarchar(MAX) |  |  |  |  |
| 3 | LinkFile | nvarchar(MAX) |  |  |  |  |
| 4 | Type | nvarchar(MAX) |  |  |  |  |
| 5 | VocabularyId | uniqueidentifier | X |  | X |  |

* Phrases( **Id**, Sentence, SentenceVN, Content, ContentVN, **VocabularyId**).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier | X | X |  |  |
| 2 | Sentence | nvarchar(MAX) |  |  |  |  |
| 3 | SentenceVN | nvarchar(MAX) |  |  |  |  |
| 4 | [Content] | nvarchar(MAX) |  |  |  |  |
| 5 | ContentVN | nvarchar(MAX) |  |  |  |  |
| 6 | VocabularyId | uniqueidentifier | X |  | X |  |

* LearningGoals ( **Id**, DateCreated, Name, AmountVocabulary, **UserId**)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier | X | X |  |  |
| 2 | UserId | nvarchar(450) | X |  | X |  |
| 3 | DateCreated | datetime2(7) |  |  |  |  |
| 4 | Name | nvarchar(MAX) | X |  |  |  |
| 5 | AmountVocabulary | int |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | VocabularyId | uniqueidentifier | X | X | X |  |
| 2 | LearningGoalId | nvarchar(450) | X | X | X |  |

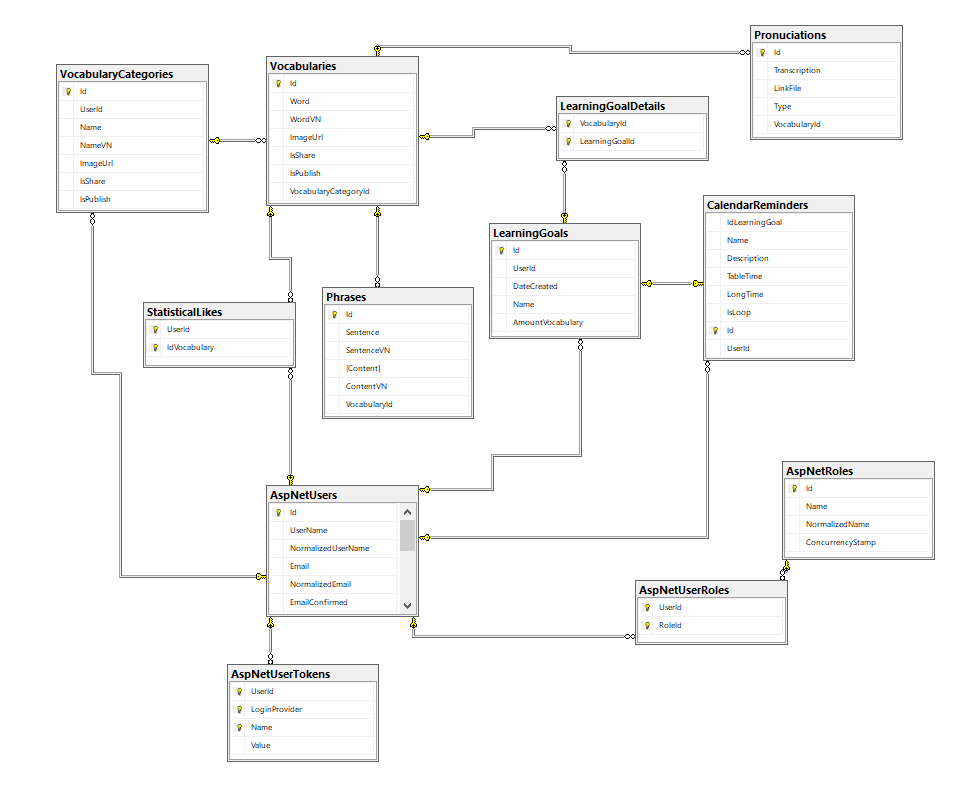
* LearningGoalDetails (**VocabularyId, LearningGoalId**)
* CalendarReminders ( Name, Description, TableTime, LongTime, IsLoop, **Id, UserId,** IdLearningGoal)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | Id | uniqueidentifier |  | X |  |  |
| 2 | IdLearningGoal | uniqueidentifier |  |  | X | X |
| 3 | Name | nvarchar(MAX) | X |  |  |  |
| 4 | Description | nvarchar(MAX) | X |  |  |  |
| 5 | TableTime | tinyint |  |  |  |  |
| 6 | LongTime | int |  |  |  |  |
| 7 | IsLoop | bit |  |  |  |  |
| 8 | UserId | nvarchar(450) | X |  | X |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stt | Tên trường | Kiểu | Not Null | Khóa chính | Khóa ngoại | Unique |
| 1 | UserId | uniqueidentifier | X | X | X |  |
| 2 | IdVocabulary | nvarchar(450) | X | X | X |  |

* StatisticalLikes( **UserId**, **IdVocabulary** )

## Diagram.



Hình 12: Diagram.