# Glimpse Iteration 8

Iteration 8 has been fine but we have completed 4 out 6 user stories. We have completed a total of 60 user story points. The reason for this is that we are not sure on the requirements for these user stories. For the mobile application, we have moved some our filtering functionality to our web service instead of having it on application. We're currently working on the continuous integration for automated testing and will most likely use visual studio's continuous integration feature. However, we also created powershell script to maintain our versioning and will use it to maintain our nugget packages. The high fidelity prototype is near completion with a few minor elements to review and make our final decisions. The logging for our application is also in progress and will be done in our controllers which are used by our mobile and web application. Every call to the controllers will get logged in a file which will be sent to our Azure Blob Storage. We are also working on giving a new fresh look to our web application with a new template but this will be for Release 3. For Release 3, we are going over many parts of the application focusing on improving the quality of some features, fixing our infrastructure , unit tests and starting our UI.

### Team members

| Name and Student id | GitHub id | Number of story points that member was an **author** on. |
|---|---|---|
| **Sam Kazerouni** 26658415 | sammykaz | 0 |
| Omer Deljanin 27064438 | bosniak47 | 13 |
| Jonathan Phillips 26699596 | PhillipsJP | 0 |
| Eric Leon-Rodas 26648754 | RiceAndBeanz | 34 |
| Joseph Daaboul 27077890 | jdaaboul | 34 |
| Asma Laaribi 29326197 | asmalaaribi | 13 |

# Project summary

Glimpse is a location based application that aims to provide the most up to date information related to local promotions and sales to its buyers. Local businesses often have small scale promotions or one day deals that are communicated to clients in person as they enter the store. This app will give these businesses a platform to

advertize such sales to a wider audience. It will help with giving them exposure to potential clients who are passing by. The main objective of the application is to efficiently advertise promotions that might have otherwise gone unnoticed to likely clients. Buyers will be able to see the sales that are taking place in the stores or businesses close by. It is providing them with relevant information based on their current location. Glimpse will offer its users a map view of the available promotions as well as a mosaic view where these promotions will be displayed as tiles. One of the main features of the application is the follow option which allows users to be more selective and follow their favorite businesses and see their promotions on the map. The users will also be able to receive notifications from the vendors they are following as soon as there is a new promotion. Another great feature is the live chat option which provides a communication channel between the users and the business. In terms of tracking how much exposure a promotion has received, the number of users that have seen it will be displayed and visible for the business it originated from. We envision this mobile application to be an innovative marketing tool that is able to keep up with daily changes and challenges of a local commercial market.

# Velocity

| Iterations | Completed User Stories | USP | Velocity | Comments |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | Initial Configuration |
| 2 | 0 | 27 | 0 | Configuration Issues & No Unit Tests were Completed, Stories moved to next Iteration |
| 3 | 8 | 59 | 59 | |
| 4 | 5 | 40 | 40 | |
| 5 | 13 | 80 | 80 | Pushed 2 stories |
| 6 | 15 | 107 | 107 | Pushed 1, but exceeded estimated user story points with new priority stories |
| 7 | 7 | 65 | 65 | |
| 8 | 6 | 60 | 60 | Incomplete 2, |

| | | | | |
|---|---|---|---|---|
| | | | | requirements changing |

Average Velocity = 63 (Not taking into account Iteration 1)

# Overview Iteration 7,8,9

| ID | Name | Priority | USP | Completed? |
|---|---|---|---|---|
| **Iteration 7**<br>X Stories - 2 weeks ( USP) | | | | |
| 171 | Establish CardView mechanics using geolocation | High | 13 | Yes |
| 172 | Buyer : Add "Liked Promotions" List Section | Medium | 13 | Yes |
| 163 | Remove Sign Up and Login for Buyer | Medium | 5 | Yes |
| 161 | Web: Add Vendors section where the vendor can manage his promotion | High | 13 | Yes |
| 163 | Web : Adding extra pictures for the promotion | High | 8 | Yes |
| 160 | Web : Ability to edit vendor profile | Medium | 5 | Yes |
| 159 | Web : Ability to view promotion details from the map view | Low | 8 | Yes |

| | | | | |
|---|---|---|---|---|
| **Iteration 8**<br>X Stories - 2 weeks ( USP) | | | | Completed? |
| 176 | Enhance detailed view for each liked promotion | High | 21 | Yes |
| 173 | Enhance Map View with integrated liked promotion list | High | 21 | No |
| 175 | Remove Side Menu, Add Tabs with appropriate icons | Medium | 13 | Yes |

| | | | | |
|---|---|---|---|---|
| 179 | Web : Add Line chart representing promotion clicks depending on hour of day, day of week | Medium | 13 | Yes |
| 178 | Web : Show number of promotion views for every promotion | Medium | 13 | No |
| 180 | Web : Add Color Variation to filters (possibly using slidebar) | Low | 13 | Yes |

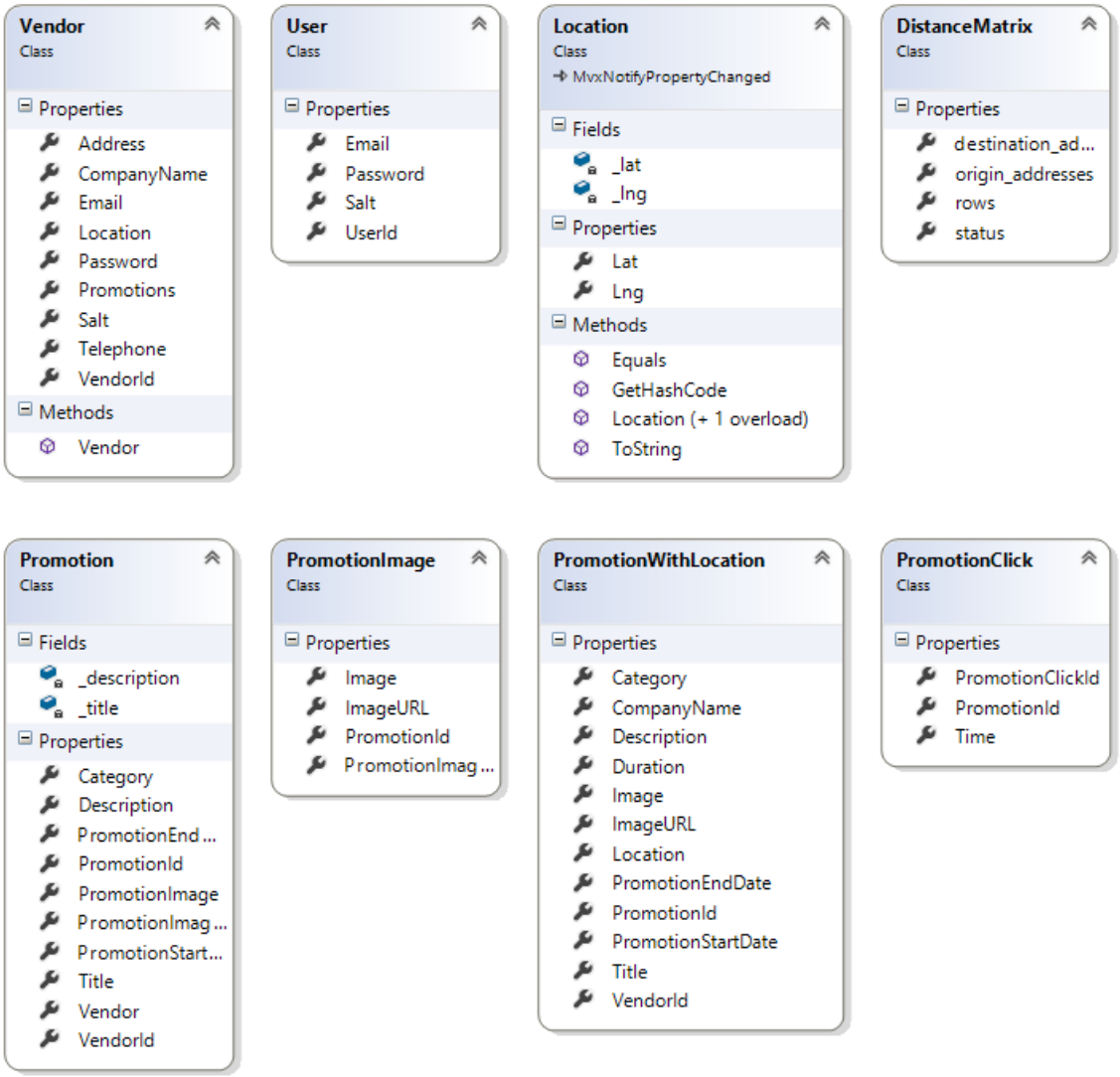
| Iteration 9 - Release 3<br>X Stories - 2 weeks ( USP) | | | | Completed? |
|---|---|---|---|---|
| 178 | Web : Show number of promotion views for every promotion | Medium | 8 | In Progress |
| 232 | Web : Display a vendor's own promotions | Medium | 13 | In Progress |
| 231 | Improve or Add Unit tests for Web application | Medium | 8 | In Progress |
| 230 | Add new Web UI template | Medium | 13 | In Progress |
| 212 | Web: View the extra pictures | Medium | 8 | In Progress |
| 211 | Web: Filter the promotions | Medium | 13 | In Progress |
| 223 | Improve or Add Unit Tests for mobile application | Medium | 0 | In Progress |
| 222 | Move hashed password from API to a service | Medium | 0 | In Progress |
| 224 | Develop High Fidelity Prototype for UI | Medium | 0 | In Progress |
| 225 | Proper Image Processing | Medium | 0 | In Progress |
| 226 | Send Emails from an Web Service | Medium | 0 | In Progress |

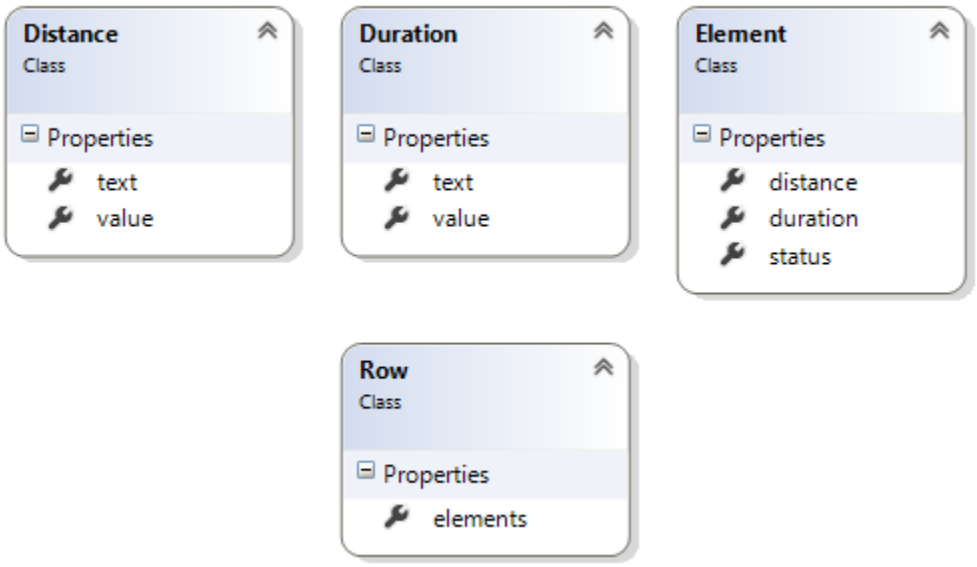
| | | | | |
|---|---|---|---|---|
| 227 | Complete the map page layout | Medium | 5 | In Progress |
| 221 | Detach Vendor functionalities From Main App and move to a new Merchant App | Medium | 0 | In Progress |

# Architecture and Class diagram

The Models package contains all the models required which are mapped to the ones used in the database. These are the current models for our application and will be updated in the future.

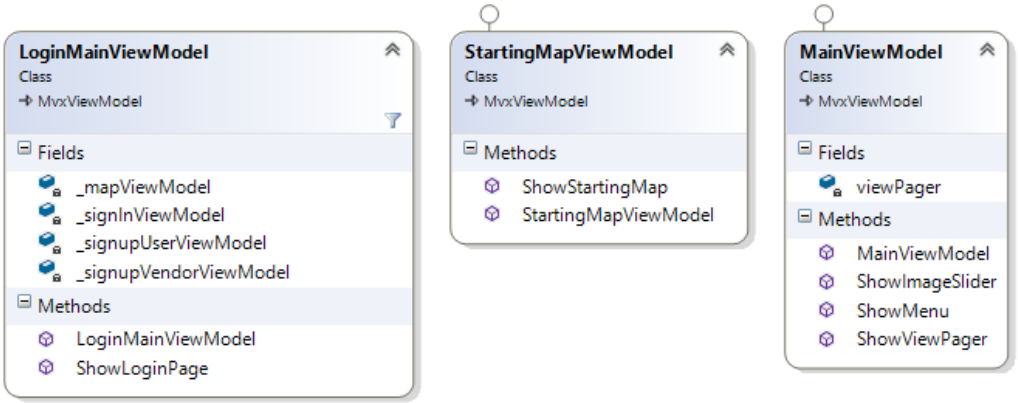
**Models Package:**


Vendor
Class
Properties
Address
CompanyName
Email
Location
Password
Promotions
Salt
Telephone
VendorId
Methods
Vendor
User
Class
Properties
Email
Password
Salt
UserId
Location
Class
MvxNotifyPropertyChanged
Fields
_lat
_lng
Properties
Lat
Lng
Methods
Equals
GetHashCode
Location (+ 1 overload)
ToString
DistanceMatrix
Class
Properties
destination_ad...
origin_addresses
rows
status
Promotion
Class
Fields
_description
_title
Properties
Category
Description
PromotionEnd...
PromotionId
PromotionImage
PromotionImag...
PromotionStart...
Title
Vendor
VendorId
PromotionImage
Class
Properties
Image
ImageURL
PromotionId
PromotionImag...
PromotionWithLocation
Class
Properties
Category
CompanyName
Description
Duration
Image
ImageURL
Location
PromotionEndDate
PromotionId
PromotionStartDate
Title
VendorId
PromotionClick
Class
Properties
PromotionClickId
PromotionId
Time



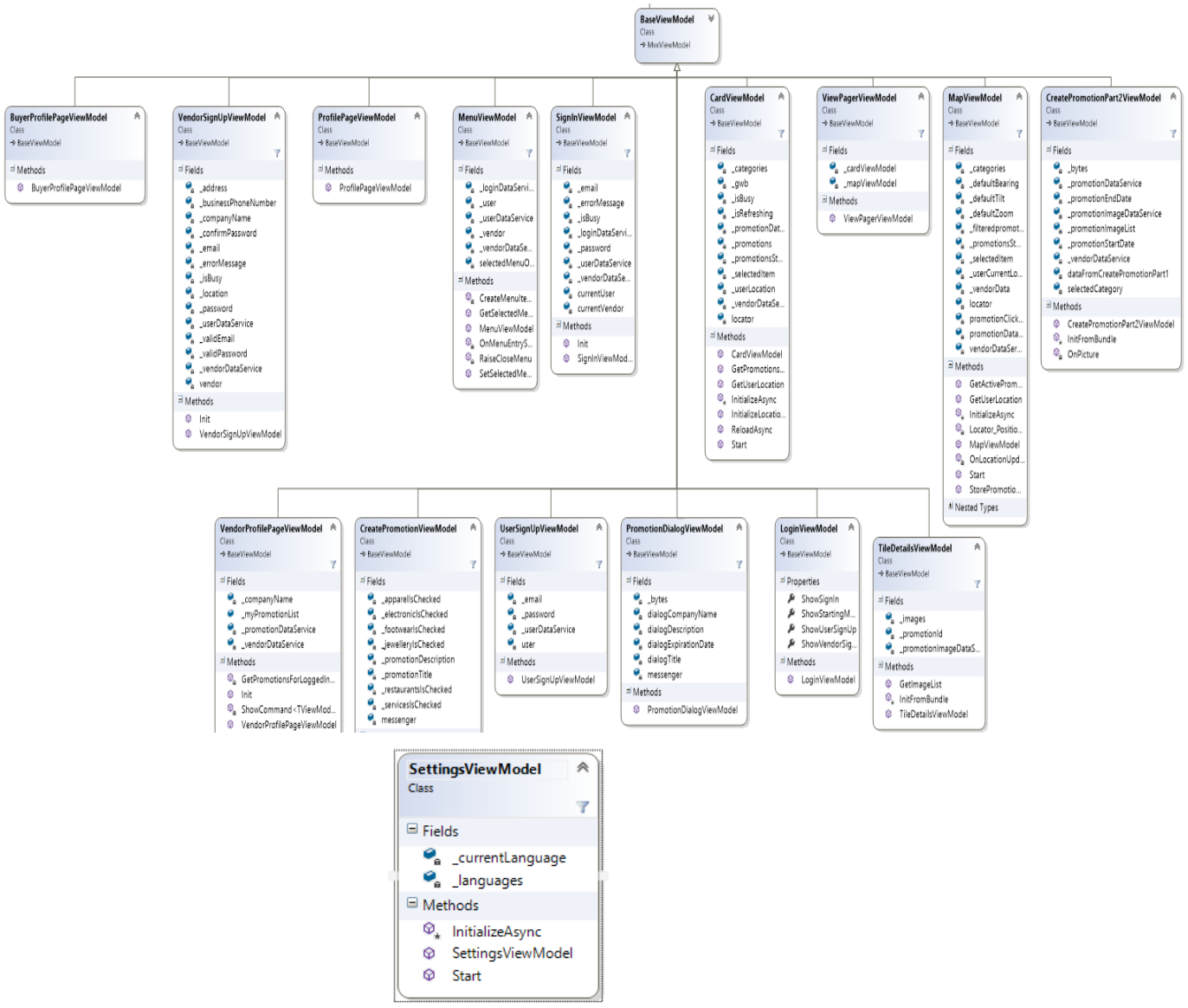

This package contains the view models for each activity in our application, currently we have LoginActivity, MainActivity and StartingMapActivity. These view models are used to update the views of these activities and contain logic related to user interface.

**Activities View Models Package:**



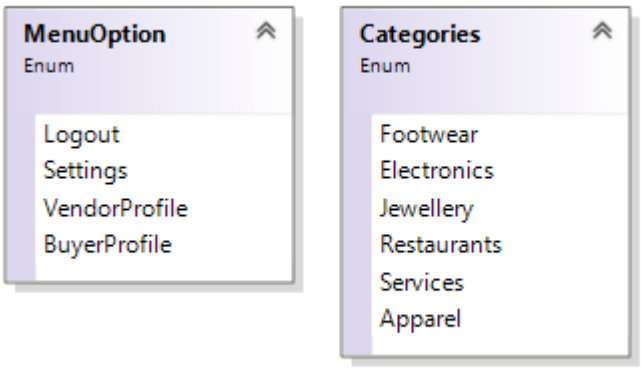
This is another View Models package, it updates the views, contains some business logic related to views, however these each fragment in our application has an associated view model which takes care of the fragments view.

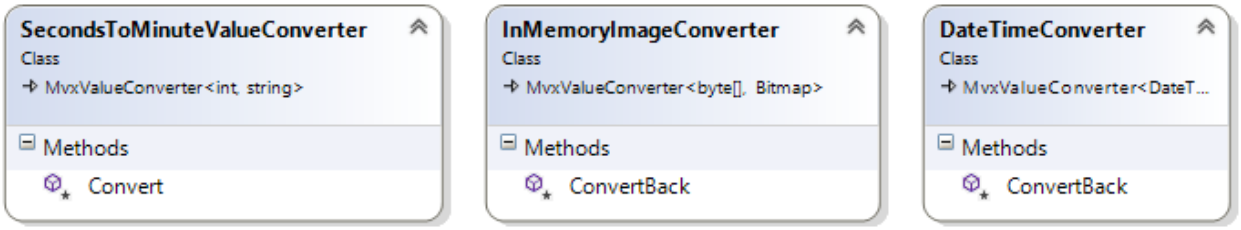
**Fragments View Models Package:**



The Utility package contains all utility classes and enums of the application. These utility classes are converters, tools, etc…

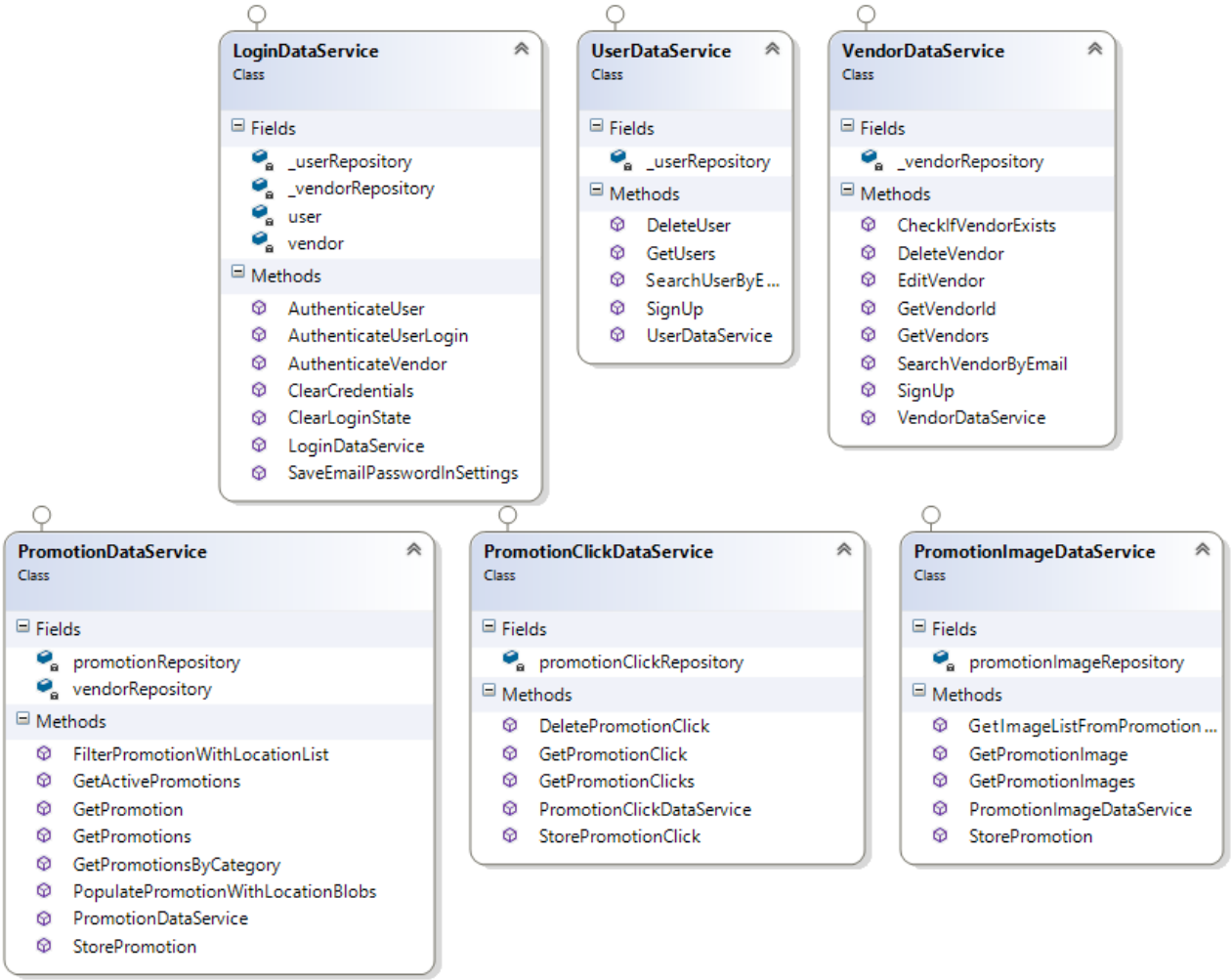
**Utility Package:**



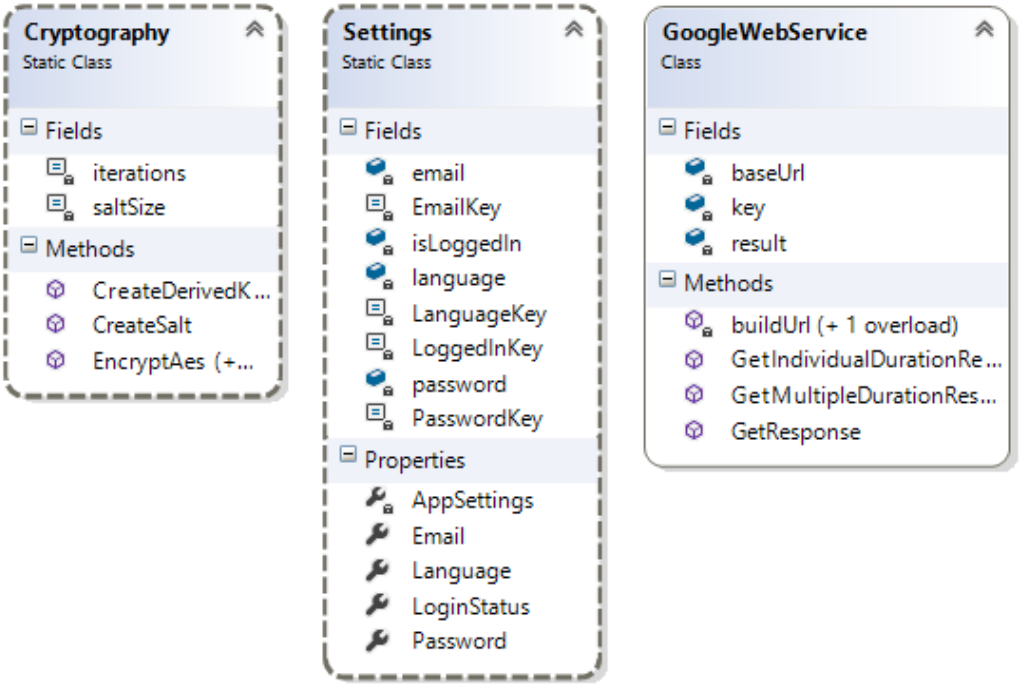
The Service packages are the most important packages in our application because they contain most of business logic related to the core of our application. They have IRepository types in their constructor using dependency injection, which retrieves a repository type and can interface to the database easily. There is no instantiation of these types and the singleton pattern is used for each repository type. This is all maintained in the framework. The Data services contain business logic related to data.

**Data Service Package:**



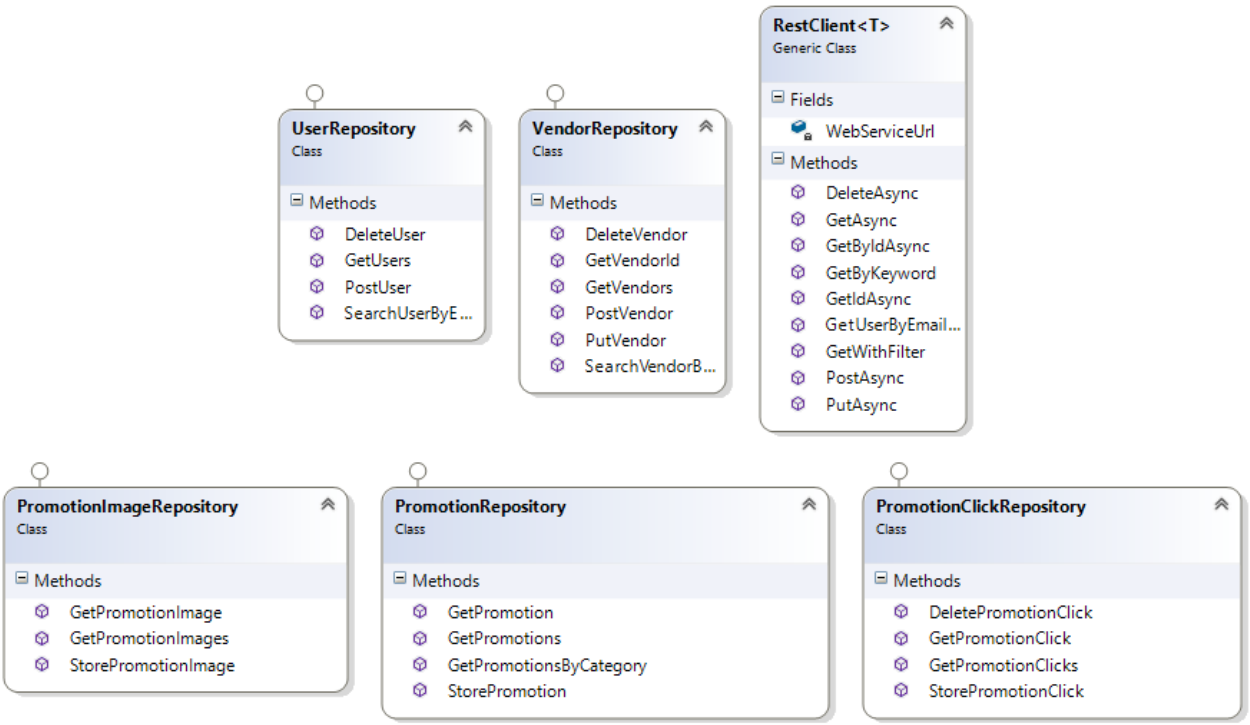
This is another service package which is very important. The General Service package contains core business logic and doesn't contain business logic related to data.

**General Service Package:**




**Repository Package:**

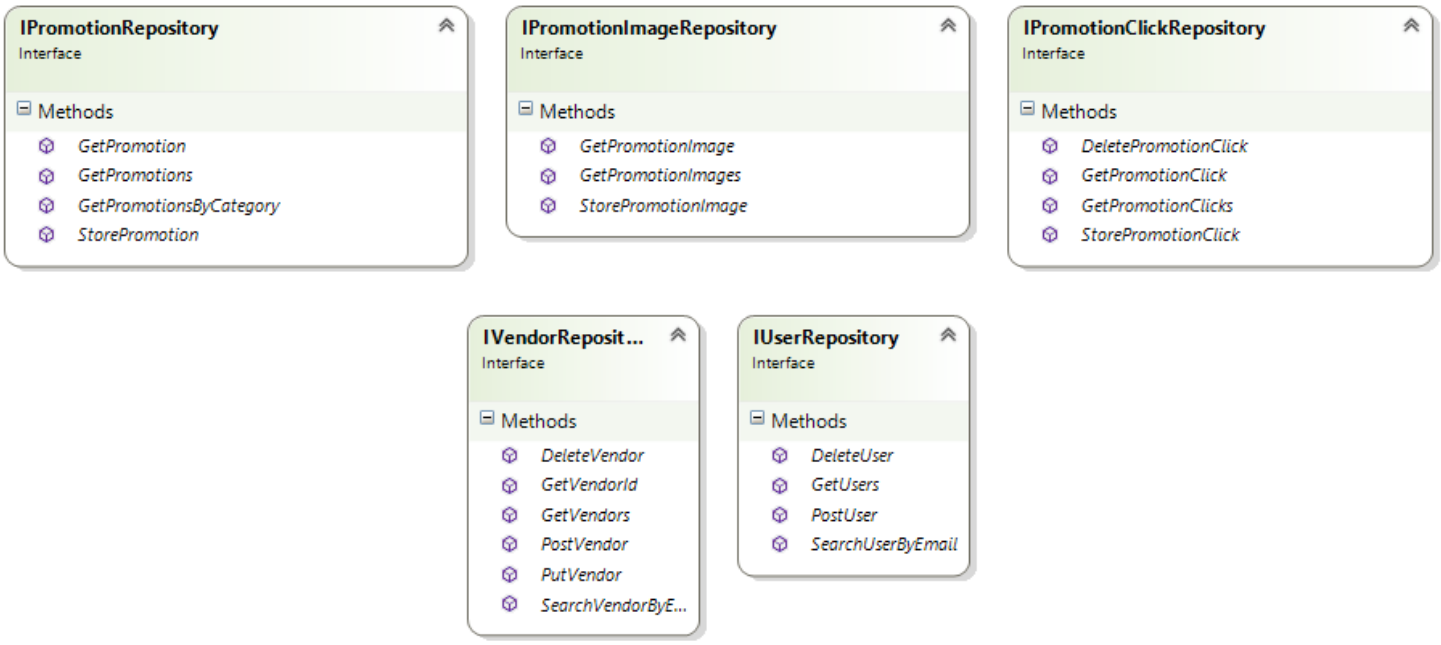
The repository classes are the commands to make calls to the database and use the RestClient class shown below. These are classes are the gateway to the database, they use the RestClient to send HTTP requests to the webservice which then sends a POST or GET to database. The response from webservice is then returned to the Restclient and passed onto the Repository.


UserRepository
Class
Methods
DeleteUser
GetUsers
PostUser
SearchUserByE...
VendorRepository
Class
Methods
DeleteVendor
GetVendorId
GetVendors
PostVendor
PutVendor
SearchVendorB...
RestClient<T>
Generic Class
Fields
WebServiceUrl
Methods
DeleteAsync
GetAsync
GetByIdAsync
GetByKeyword
GetIdAsync
GetUserByEmail...
GetWithFilter
PostAsync
PutAsync
PromotionImageRepository
Class
Methods
GetPromotionImage
GetPromotionImages
StorePromotionImage
PromotionRepository
Class
Methods
GetPromotion
GetPromotions
GetPromotionsByCategory
StorePromotion
PromotionClickRepository
Class
Methods
DeletePromotionClick
GetPromotionClick
GetPromotionClicks
StorePromotionClick

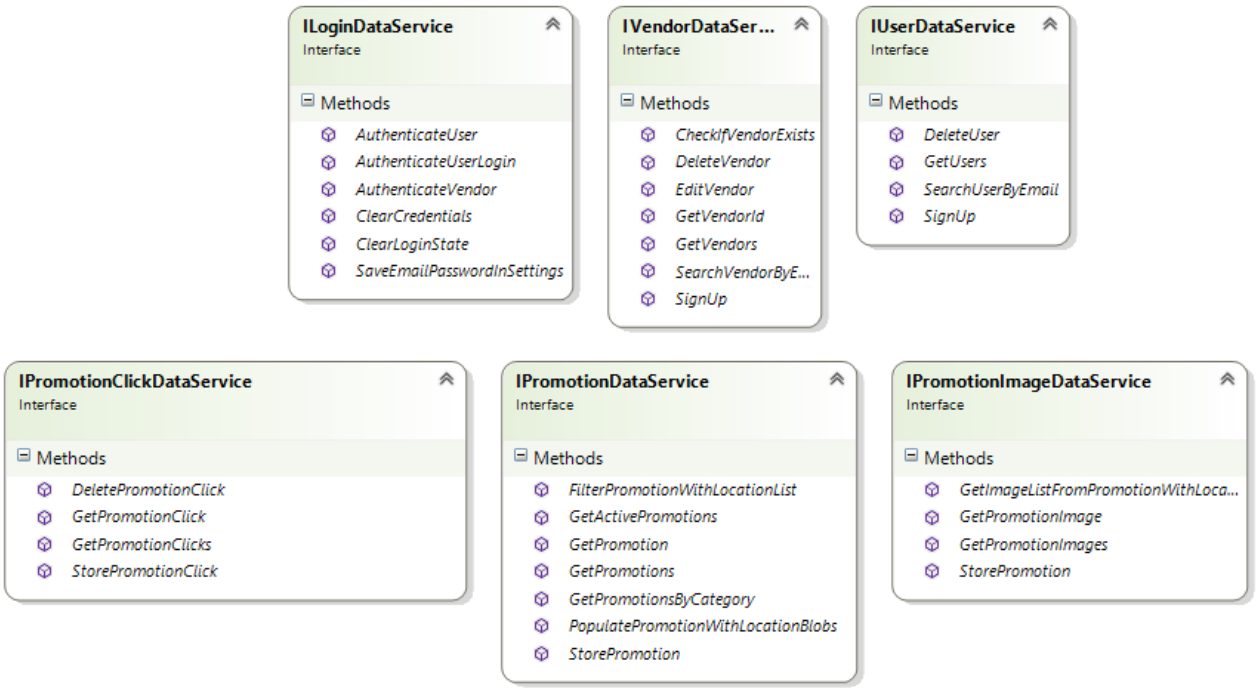
## Contracts:

These are the contracts packages. The repository and service contracts contain several methods which are used in our repositories and services respectively. We define them here before we use them in our application.

**Repository Contracts Package:**



## Service Contracts Package:



# Infrastructure

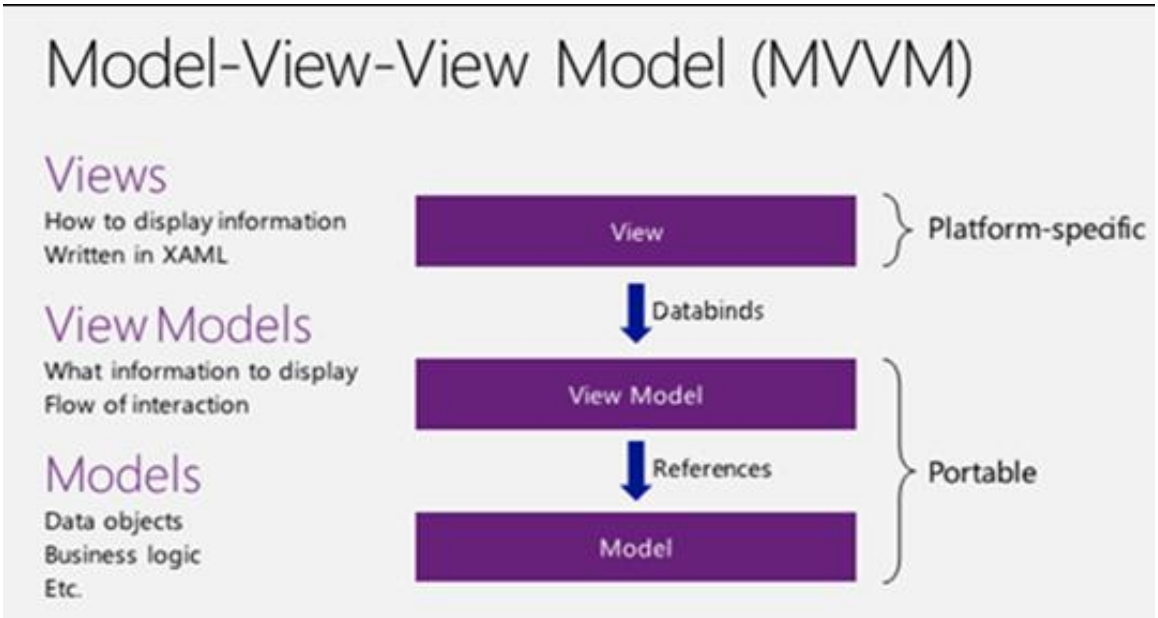
**MvvmCross Framework (Application framework)**
https://mvvmcross.com/

MvvmCross is a cross-platform mvvm framework that allows developers to create cross platform apps. Model-View-ViewModel is an software architectural pattern. MVVM is a variation of Martin Fowler's Presentation Model design pattern. Like Fowler's Presentation Model, MVVM abstracts a view's state and behavior. However, whereas the Presentation Model abstracts a view (i.e., creates a view model) in a manner not dependent on a specific user-interface platform.Several reasons why we are using

MVVM and Presentation Model both derive from the model–view–controller pattern (MVC). MVVM facilitates a separation of development of the graphical user interface (either as markup language or GUI code) from development of the business logic or back-end logic (the data model). The view model of MVVM is a value converter meaning the ViewModel is responsible for exposing (converting) the data objects from the model in such a way objects are easily managed and consumed. In this respect, the view model is more model than view, and handles most if not all of the view's display logic The view model may implement a mediator pattern, organizing access to the back-end logic around the set of use cases supported by the view.

We took advantage of all the benefits that are provided by MVVMCross, such as bindings, converters, IoC, etc. For the Portable Class Libraries, the other option to sharing code is file-linking. PCL is neat, tidy, and requires no overhead when your common code library changes. File-linking is a manual process and can begin to get cumbersome after constant project changes. Furthermore, extensibility of the application can be easily done using the concept of plugins. Plugins can be added to the framework which can make our application grow much faster.

The disadvantage of using MvvmCross framework is that there a learning curve.

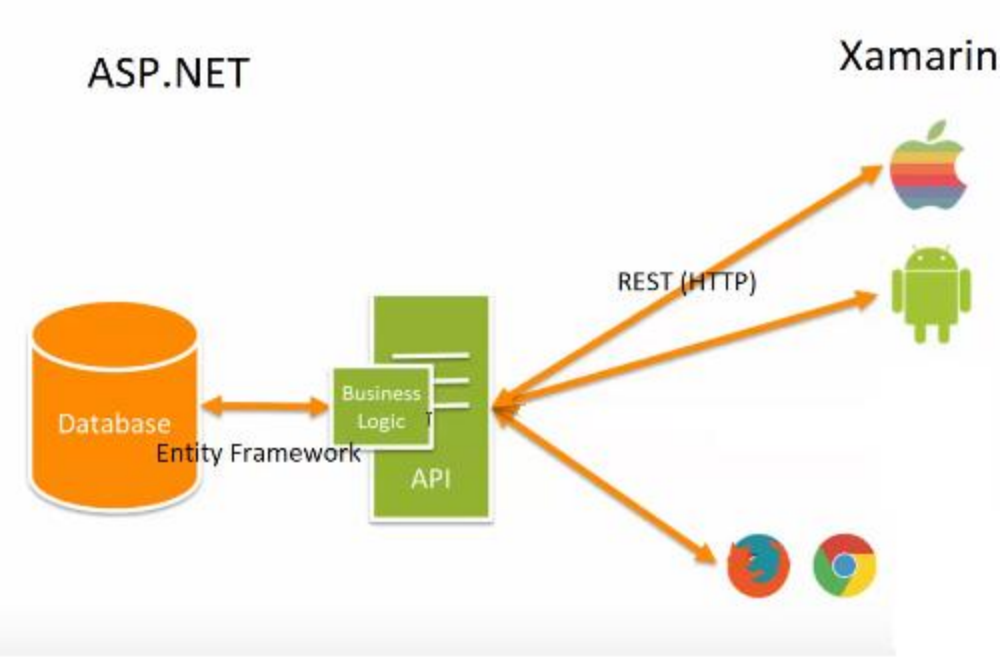



**Microsoft Azure Database**
https://azure.microsoft.com/en-us/services/sql-database/

We have selected Microsoft Azure Database because our project is based on .NET and we're using Visual Studio, therefore it makes sense using Microsoft Azure. The Azure plugin allows us to easily connect to our remote database using the Server Explorer in Visual Studio which allows us to quickly view data. Our remote server is located around Central Canada region which is relatively close to our location and runs our SQL database.

When comparing Amazon and Microsoft's database services, cost does come into play. Microsoft gives students a free 1 year subscription to use their services through the Microsoft Imagine program. Amazon also gives students a 150$ credit which was used. However, initially when we used Amazon Web Services, the configuration was slightly more complex and we realised that there are several hidden charges when using amazon servers and databases. These charges increased quickly and cost per usage time. This was terrible because there had to be further research on understanding what these charges ment and how to prevent them. Therefore, we decided to transition to Microsoft Azure and we have had no issues related to configuration, cost and service.

**Webservices**
In order to make our database transactions, we have made a WebServices project which communicates to our remote SQL database. It is the only gateway to our database and our application must use our WebServices in order to make requests. Our WebServices project uses Entity Framework for its Models and maps to the tables in the database. Our Core application which contains our shared business logic, uses a layer called "Repositories" where our RestClient is used. The RestClient sends CRUD operations using HTTP to our WebServices which then sends a request to our database. Moreover, a response is returned from our database to our WebService and back to the RestClient.




## Name Conventions

Standard C# Coding Convention: https://msdn.microsoft.com/en-us/library/ff926074.aspx

## Code

| File path with clickable GitHub link | Purpose (1 line description) |
|---|---|
| https://github.com/sammykaz/glimpse/blob/master/Glimpse.Core/RestClient/RestClient.cs | Generic class to be used by repositories that implements CRUD operations. Our RestClient for our web services. |
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core/Services/Data/PromotionClickDataService.cs | Our data service containing analytics logic which are used by merchants. |
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core/BlobClient/BlobClient.cs | This connects to our image service. |
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core/Services/General/GoogleWebService.cs | Our google web services for our distance calculations for the map. |
| https://github.com/sammykaz/glimpse/blob/master/Glimpse.Core/App.cs | Registers files ending with “Repository” and “Service” to IOC container, allows them to be used through interface and will only use a single instance. |

# Testing and Continuous Integration

List the **5** most important test with links below.

| Test File path with clickable GitHub link | What is it testing (1 line description) |
|---|---|
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core.UnitTests/Tests/Services/PromotionDataServiceTests.cs | All tests related to data access of promotions from merchants. |
| https://github.com/sammykaz/glimpse/blob/master/Glimpse.Core.UnitTests/Tests/Repository/VendorRepositoryTests.cs | When a vendor signs up, their credentials are stored on the database. |
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core.UnitTests/Tests/Services/GoogleWebServiceTests.cs | Google Web Service tests, test for individual responses and multiple responses for the distance. |
| https://github.com/sammykaz/glimpse/blob/develop/Glimpse.Core.UnitTests/Tests/Services/PromotionClicksDataServiceTests.cs | Tests related to merchant analytics |
| Other tests include System Tests for UI which are shown in Github. | |