

Governors State University

OPUS Open Portal to University Scholarship

All Capstone Projects

Student Capstone Projects

Fall 2020

Better Reviews

Mauro Patino

Follow this and additional works at: <https://opus.govst.edu/capstones>

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the [Governors State Computer Science Department](#)

This Capstone Project is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.

Better Reviews

By

Mauro Patino

B.S. Computer Science, Governors State University, 2000

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science



Governors State University
University Park, IL 60484

2020

ABSTRACT

Better Reviews is a real-time web resource that allows registered individuals to create, edit and submit testimonials about products, organizations, and services they have connected with in their local community.

Better Reviews provides individuals (reviewers) with a location to share their story about the experience they have had with a business, product(s) or service provider. These reviews allow other users and visitors to the site to make better informed decisions about upcoming purchases and service contracts.

Better Reviews allows registered users to provide text descriptions of their experience, in addition to providing a simple visual ranking system (for example one-to-five stars) which quickly indicates a proposed value for the product or service which is being described. Registered users may also upload pictures to accompany their review.

The audience for both text and visual reviews is intended to be other registered members of this community, who will also need to create a profile (i.e., register) before they are permitted to access or create reviews through this portal. Users will have a user interface which allows them to search for reviews by keyword, and sort reviews by category, type, or location.

Service providers, including business owners and other agents of a businesses or organization who are reviewed on Eight Reviews will be able to read and respond to reviews by becoming a registered user, and logging into the system. A 'Featured Review' (i.e., advertising placement) is able to be included with Better Reviews which will allow a business to enjoy additional visibility at the top of a category or business geographic search result.

An imbedded communications system is included in Better Reviews which provides registered users with the ability to subscribe to and review comments to business sites, and provides an interface that administrative users can use to reach out to users and businesses with information pertaining site rules or marketing opportunities.

Administrative users who maintain the site are also provided with privileges to update, upgrade, create and delete business profiles and user reviews which do not meet site quality or appropriateness guidelines. These admin users are the theoretical business owners of Better Reviews, and for the purpose of this project are the Team Members listed above as working on the project.

Table of Content

1	<i>Project Description</i>	1
1.1	Competitive Information	1
1.2	Relationship to Other Applications/Projects	1
1.3	Assumptions and Dependencies	2
1.4	Future Enhancements.....	2
1.5	Definitions and Acronyms	3
2	<i>Technical Description</i>	4
2.1	Project/Application Architecture	4
2.2	Project/Application Information flows	4
2.3	Interactions with other Projects (if Any)	5
2.4	Interactions with other Applications	5
2.5	Capabilities	5
2.6	Risk Assessment and Management.....	6
3	<i>Project Requirements</i>	7
3.1	Identification of Requirements	7
3.2	Operations, Administration, Maintenance and Provisioning (OAM&P)	8
3.3	Security and Fraud Prevention.....	9
3.4	Release and Transition Plan.....	9
4	<i>Project Design Description</i>	10
5	<i>Project Internal/external Interface Impacts and Specification</i>	39
6	<i>Project Design Units Impacts</i>	64
6.1	Functional Area/Design Unit A	64
6.1.1	<i>Functional Overview</i>	64
6.1.2	<i>Impacts</i>	64
6.1.3	<i>Requirements</i>	64
6.2	Functional Area/Design Unit B	67
6.2.1	<i>Functional Overview</i>	67
6.2.2	<i>Impacts</i>	67
6.2.3	<i>Requirements</i>	67
6.3	Functional Area/Design Unit B	69
6.3.1	<i>Functional Overview</i>	69
6.3.2	<i>Impacts</i>	69
6.3.3	<i>Requirements</i>	69
7	<i>Open Issues</i>	72
8	<i>Acknowledgements</i>	73
9	<i>References</i>	74
10	<i>Appendices</i>	76

1 Project Description

1.1 Competitive Information

A variety of public and consumer facing review sites already exist on the web. General sites which feature multiple categories of reviews and rating systems include:

Yelp	https://www.yelp.com/
Google Rating	https://www.google.com/intl/en_us/business/
Facebook	https://www.facebook.com/marketingAPAC/reviews/
Yellow Pages	https://www.yellowpages.com/
Angie's List	https://www.angieslist.com/
Trip Advisor	https://www.tripadvisor.com/

In addition, niche review sites also exist. For example, sites which provide reviews targeted toward restaurants include:

Zagat Restaurants	https://zagat.com/
Restaurant.com	https://www.restaurant.com/
Open Table	https://www.restaurant.com/

1.2 Relationship to Other Applications/Projects

Review sites both in niche and general categories typically make use of a template which provides a business/organization with the ability to create (or claim) a profile with basic facts about their business.

Basic profiles found on sites (like those listed above), as well as the stie that is being described by this document for Better Reviews, includes:

- Business Name (and dba names)
- Business Address (and map pinpoint location)
- Business Phone, Fax, Email, Website
- Business Hours
- Business Specialties
- Business Photos

Better Reviews has included these basic profile features into its design.

In addition to the standard business profile, customers and business clients are invited to share the experience they have had with the business by posting 'a review'. Much like the template that the business is invited to follow, review sites typically follow a template for the review which includes:

- Reviewer Name
- Reviewer Rating (Stars, points, or other graphical favorability scale)
- Reviewer Description (narrative, story, or text description)
- Photos/Videos or other Media

Better Reviews has included these reviewer features into its design.

1.3 Assumptions and Dependencies

The primary assumptions made for this project is that while we use a representative sample of data from businesses located in the greater Chicago area, none of the data which is included in the presentation of this application represents 'true' or accurate review and business data.

We present the data, businesses, photos, and descriptions (both imaginary and real-world locations) as representative models of how businesses could or should be represented in the application.

No effort, or agreements have been created or exist between any business which might be seen in the application and the real world. Our data is fictitious and imaginary, and users, viewers or readers of our documents and users of our apps should understand that business models are not real. No compensation, transactions or reviews which might be included as test data or created by people using this application should be considered to be 'valid, truthful or real'.

Dependencies:

To illustrate the real-world potential of this application we have used and created a test database based off data we have culled from Yellow Pages (yellowpages.com) which is made available publicly through the yp.com API. This data is used to test the generation of categories, to do geo-location targeting (via Google Maps API), and to otherwise test the creation, deletion and updating of business profiles as well as user reviews of those sites. We depend on the API's of these public companies, and the data that these API's reveal to produce some site features. However, these API's do not compromise the core of the applications, and no 'code', only data, from these sites are returned as 'content' on the site.

Other technology used for the coding are described in Section 2.1 Application Architecture.

1.4 Future Enhancements

Many of the review sites also enable communication between a business and reviewers of that business either through a message system or through interaction between user profiles. For platforms like Facebook, messages can go from a user directly to a business owner via Facebook messenger. Other services, like Yelp permit conversations to occur between users and business owners in a more public fashion (where the entire conversation is exposed to the public).

A number of review sites provide business users with additional tools which can include the ability to contest reviews and challenge comments left by users about their business or services. Google provides an 'instant challenge feature' where any user or business owner can flag a comment as inappropriate, obscene, or defamatory which will trigger a review by a content moderator. Some sites like Yelp also offer the ability for business owners to purchase services related to taking down reviews or burying poor reviews for a charge.

Better Reviews has identified this area as an area of future growth which can be implemented into future versions of its design.

Finally, as an economic revenue stream, many site reviews will offer a variety of lead generation services to businesses. From generating recommendations, to servicing phone call (click-through dialing), to spot advertising and special coupons, the number of services vary greatly from one service to another. Larger platforms like Google, Yelp and Facebook typically wrap these services into contracts which are included in advertising (display and click-through) agreements.

Better Reviews has identified the benefits of these economic models and anticipates that marketing and sales features should be included in upcoming and future releases.

1.5 Definitions and Acronyms

Admin: A user with administrative privileges to all manage (create, edit, update and delete) all business listings, as well as all users (create, edit, update and delete), and finally to manage the roles of registered users and to create new categories of businesses on the site are admin users.

Business: A business is an organization, group or other entity which provides good or services to customers, clients, or other businesses. In this application a business could be a restaurant serving food, or an accountant providing tax return services. In the illustrations provided as examples in this document a Montessori (a private school) is created as a business.

Business Owner: It is not necessary to formalize or restrict a 'business owner' to be an individual who is a sole, or majority shareholder of a Business (previously defined). For our purpose, a business owner could be a manager or a supervisor in a business, or might even be an employee (or an agent of the business) who is taking on the responsibility of managing a profile (create, edit, update and other wise maintain) the listing of a business identified in the application.

Category: This is a list of types of businesses or organizations. For illustration we provide Restaurant, Entertainment and Other as initial business categories, however admins for the site have the ability to add additional categories which might be identified with such keywords as "Automotive, Travel, Medical, Gyms, Grocery, Nail/Beauty, Salon, Dentist, Pharmacy, etc.).

Claimed Business: A business is claimed by a business owner and become 'verified' as being claimed. Unclaimed businesses display a 'claim this business' button which allows a registered user to send a message to the site admin requesting that they be allowed to manage the identified business profile.

Cropping: Images that are uploaded to the site are stored in the dimensions which they are received, however each image is 'cropped' to make it square (even on top and sides) for display in photo album-type business profile page displays.

DOT.NET: A programming framework (by Microsoft) use in the development of this application see section 5.

HTML: Hyper Text Markup Language the markup language used for front-end (web pages) for this application.

Image: A picture or a graphic which is uploaded through the graphical user interface. Both registered users providing a business review, as well as business owners are allowed to upload an image to a business profile.

MVC: Model View Controller, a programming paradigm used in the development of this app. See Section 5 for more info.

Message: A message is a text artifact that is exchanged between two users. We use the words Notification and Notification center to indicate a message and the graphical user display where Notifications are received, read, replied to and otherwise managed (i.e., deleted).

Rating: The application provides a graphical rating system (stars from one to five) where a registered user (see definition below) is able to indicate an increasing strength of the business by highlighting few to many stars. The rating system one to five stars indicates poor performance (one star) to high performance (five stars).

Registered user: Users who would like to either review a business (comment about a business page) are required to register with the site. Registered users must provide a first and last name, as well as an email address in order to become a registered user. An individual who would like to become a business owner, must first be a registered user.

Review: A registered user (see definition below) is provided with an opportunity to provide a critique (information provided as text, graphical star performance indication, and to upload images) which are related to any business on the site they have received goods or services from. Reviews, which are identified for a particular business (see definition above) are grouped together as a stream of reviews with the most recent reviews at the top of the stream. Business owners (see definition above) are automatically subscribed to see all reviews which are posted about their business.

Site Visitor: Individuals who visit the site are allowed to browse listings and view business profiles without having to provide any personal details. These individuals are site visitors. Site visitors can become registered users.

The Site: The application being proposed is a web application called Better Reviews. The entire implementation of the application is referred to throughout the document as either 'the site', 'the app', or to Better Reviews.

2 Project Technical Description

2.1 Application Architecture

Responsive Front End:

HTML-5, CSS-3, ECMA/JavaScript, Bootstrap

Application Layer:

Microsoft Dot.Net Framework, with Integrated Cookie/Session management

Federated Login (Twitter, Facebook, Google)

Database:

Microsoft SQL-Server with Entity Web Framework

Web Server:

IIS Windows Server,

Version Control:

Git-Hub (version control)

Architecture Design Abstractions:

Model View Controller, Entity Framework, Object-Oriented

2.2 Application Information flows

Better Reviews is a Dot.NET MVC Application which generated dynamic HTML/CSS/JavaScript displays on the front end and makes use of a MS-SQL Server Database layer for storage retrieval and updating of data submitted by users.

The MVC Model separates the concerns of Views (User Interface), from C# Classes (Controllers), and Data (abstracted into two concerns, Models (inside the MVC Application), and the actual Database).

In the most generic sense the data flow from a user to the application takes place via a web interface or the Internet:

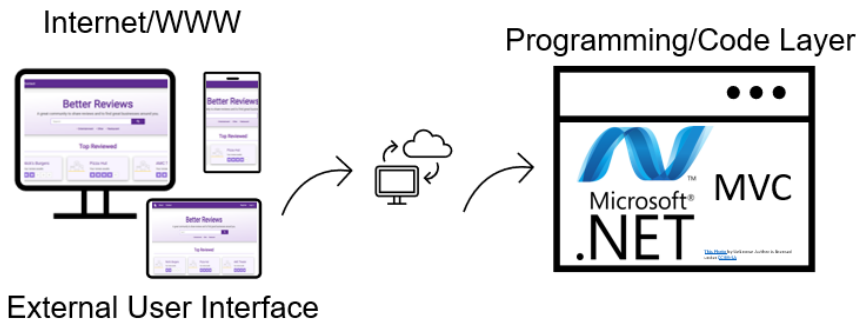


Fig 2.1 Information flow from User to App

Internally the separation of concerns between MVC are represented as individual application layers:

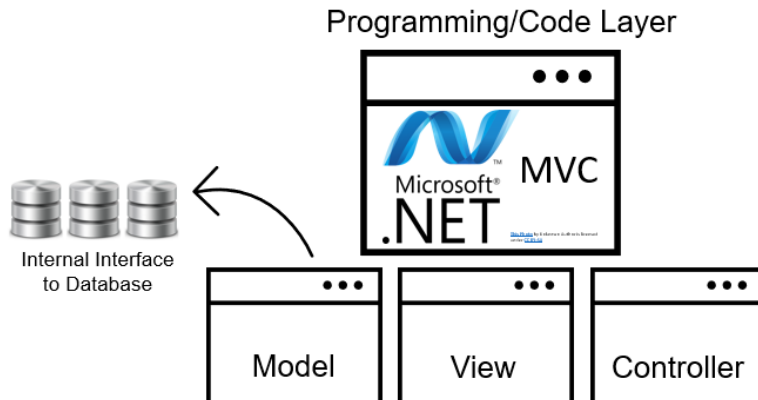


Fig 2.2 Internal Applications and Database Layer

Finally, within the Dot.NET Application the Functional Areas are represented as individual Applications:

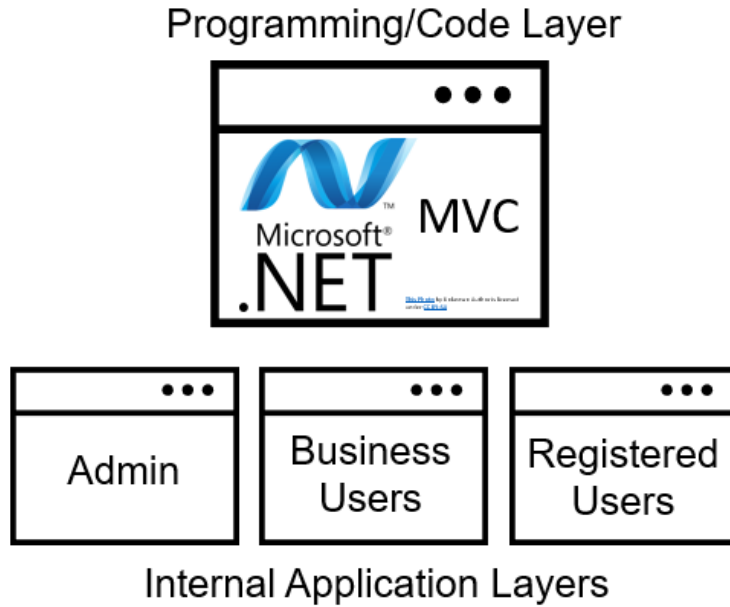


Fig 2.3 Programming and Code Layer

2.3 Interactions with other Projects (if Any)

Better Reviews does not have any interactions with any other GSU project or public site.

2.4 Interactions with other Applications

Google Maps API, Bootstrap, SQL-Server (Database)

2.5 Capabilities

Users:

- New User Registration
- New User Password Setup/Reset Function
- New User Profile Management

Business Users:

- New Business Creation
- Business Claim Feature
- Business Profile Update
- Business Photo Posting
- Subscribe to Reviews on Business
- Business Site Account Delete

Site Reviewer:

- Create New Review
- Delete/Update Existing Review
- Site Reviewer Profile Update
- Site Review Account Delete

Admin:

- Create/Edit/Update/Delete User(s)
- Create/Edit/Update/Delete Review(s)
- Create/Edit/Update/Delete Business
- Message Users (Reviewers and Businesses)

2.6 Risk Assessment and Management

There are two laws which directly pertain to management of user-generated content, posted to a site which describes other people, businesses, or public figures:

Section 512(c) of the Digital Millennium Copyright Act

<https://www.law.cornell.edu/uscode/text/17/512>

Section 230 of the Communications Decency Act

<https://www.law.cornell.edu/uscode/text/47/230>

In addition, even though this site is intended for users in the U.S., any data which would amount to Personally Identifiable Information (PII: [wikipedia.org/wiki/Personal_data](https://en.wikipedia.org/wiki/Personal_data)) is obligated to be protected as described in the EU General Data Protection Regulation

<https://eur-lex.europa.eu/eli/reg/2016/679/oj>

In the U.S. individual states each mandate that PII must be properly secured and managed. While our site does not intend to generate, solicit, or store information from users related to categories which directly qualify as PII, it is likely that related-PII (or linked PII) data might also be acquired. The most visible of state laws related to storage of PII is the California Data Privacy Act

https://wikipedia.org/wiki/California_Consumer_Privacy_Act

Illinois (for which this project is initialized and generally describes) has a statute related to PII as well. See the State of Illinois Compiled Statue known as the 'Personal Information Protection Act'

<https://www.ilga.gov/legislation/ilcs/ilcs3.asp?ActID=2702>

In addition to understanding the requirements meant for storage of data, and even though our data collection methods are not designed to acquire information, we must be diligent in safeguarding any information the site does receive, as 'not requesting PII data' does not excuse an organization from protecting such information even if it is inadvertently acquired. For example, acquiring a First Name, Middle Initial and Last name is under Illinois code PII. Therefore, it is mandatory that any data collected on the site be considered potentially covered by PII and properly safeguarded in a straightforward and auditable manner.

For this project, we have therefore elected to not publish the application or the database (with data) to a publicly facing IP address or application server.

3 Project Requirements

3.1 Identification of Requirements

Requirements Groups:

- 01 User Features**
- 02 Business Users**
- 03 Site Admin**

<GSU-GS_FL2020 User-Features-01>

Site Users Requirements

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01.01>

Console Access (GUI) for New User

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01.02>

Keyword-Based Search Box

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01.03>

Advanced search by category, rating, location, etc.

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01.04>

Featured businesses on the front page

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01.05>

Registration, Login, Profile Management for Site Reviewers

Implementation: Mandatory

<GSU-GS_FL2020 User-Feature-01>

Review a Business Feature for Registered Users

Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Features-02>

Business Users Requirements

Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Feature-02.01>

Create Business (GUI) for New Business

Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Feature-02.02>

Update Business (GUI) for Business Listing

Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Feature-02.03>
Add and Remove Photo(s) for Business Listing
Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Feature-02.04>
Subscribe to Reviews (posted by other users)
Implementation: Mandatory

<GSU-GS_FL2020 Business-User-Feature-02.05>
Send/Receive Notifications from Site Admin
Implementation: Mandatory

<GSU-GS_SP2020-1 Admin-Features-03>
Site Admin Requirements
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.01>
Admin Console (GUI) for Site Administrators
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.02>
Manage Users
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.03>
Manage Users Roles
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.04>
Manage Business Categories/Create Category
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.05>
Send/Respond to User Notifications/Messages
Implementation: Mandatory

<GSU-GS_FL2020 Site-Admin-Feature-03.06>
Update/Add, Edit Existing Business
Implementation: Mandatory

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

Technical Requirements

HTML5, CSS3, JavaScript, Bootstrap

Responsive design

User authentication and authorization (i.e., role-based access) > SQL Database

Programming using high level programming language

State management using cookies, states, etc.

Validations

User friendliness (e.g., smooth user flow, auto complete, single sign-on, ...)

Documentation:

The final project documentation should encompass detailed specification including project scope, technical description, project requirements and analysis, design description, and so on. Student should utilize various UML diagrams to describe process flow, data flows, file designs, entity relationship, I/O designs, and so on. The documentation should also include information gathering, and report activities and transition from requirements into system analysis and design, and final implementation.

3.3 Security and Fraud Prevention

Code management and control is handed through a private Git Hub Repository (the name of this repository, and its URL(s) are not being identified in this document). The repository is request-based with pull-push permission available to each member of the project team and commit administration by the repository owner. (Mauro Patino).

Because of the nature of risk, and laws governing user-supplied data to a review-user generated content application (see discussion in section 2.6 above) all URS's shown in the documentation images are <localhost>; meaning that they are run on a local/non-connected machine. In addition while the SLQ code provided in the appendix section (see section 10 below) can be used to re-create the entire database illustrated throughout this document, true user data and reviews are not included in the supplied master SQL database file.

Where live implementation of the database and application does exist, user roles and privileges are managed through the use of user roles including Admin (administration of site, categories, users, businesses, images), Business Roles (user-level administration of business profile information including business name, phone, email, URL, photos and location), and Registered Users (user-level control over individual contributions (i.e., reviews), posted by that user which can include text, images and ratings of businesses or organizations). Site admins have control over both Business Users as well as Site Reviewers, including the ability to remove and edit content and users.

3.4 Release and Transition Plan

No plans have been made to release this project in a commercial or non-for-profit fashion. At this time the site (application interface, code base, database components and related data) are limited for review through the Office of the Dean of the Mathematics and Engineering Department at Governors State University, University Park, Illinois USA.

No implied or explicit license, rights including transferability are provided to any individual not named as a direct contributor specifically listed on the first page of this document including (Franciskovich, Anthony; Garwood, Clinton; Miulli, Eric; Patel, Ashitaben Hemalkumar; Patino, Mauro). Release of the application (and its component parts) is only available through unanimous written consent from each contributing party, to any potential receiving party.

4 Project Design Description

As described in section 3.1 identification of requirements, evidence of mandatory features to be implemented in Better Reviews will be illustrated. The features to be shown include:

User-Features

- 1.1 Console Access (GUI) for New User
- 1.2 Keyword-Based Search Box
- 1.3 Advanced search by category, rating, location, etc.
- 1.4 Featured businesses on the front page
- 1.5 Registration, Login, Profile Management for Site Reviewers
- 1.6 Review a Business Feature for Registered Users

Business Users Requirements

- 2.1 Create Business (GUI) for New Business
- 2.2 Update Business (GUI) for Business Listing
- 2.3 Add and Remove Photo(s) for Business Listing
- 2.4 Subscribe to Reviews (posted by other users)
- 2.5 Send/Receive Notifications from Site Admin

Site Admin Requirements

- 3.1 Admin Console (GUI) for Site Administrators
- 3.2 Manage Users
- 3.3 Manage Users Roles
- 3.4 Manage Business Categories/Create Category
- 3.5 Send/Respond to User Notifications/Messages
- 3.6 Update/Add, Edit Existing Business

Figure 4.1: Better Reviews Home Page

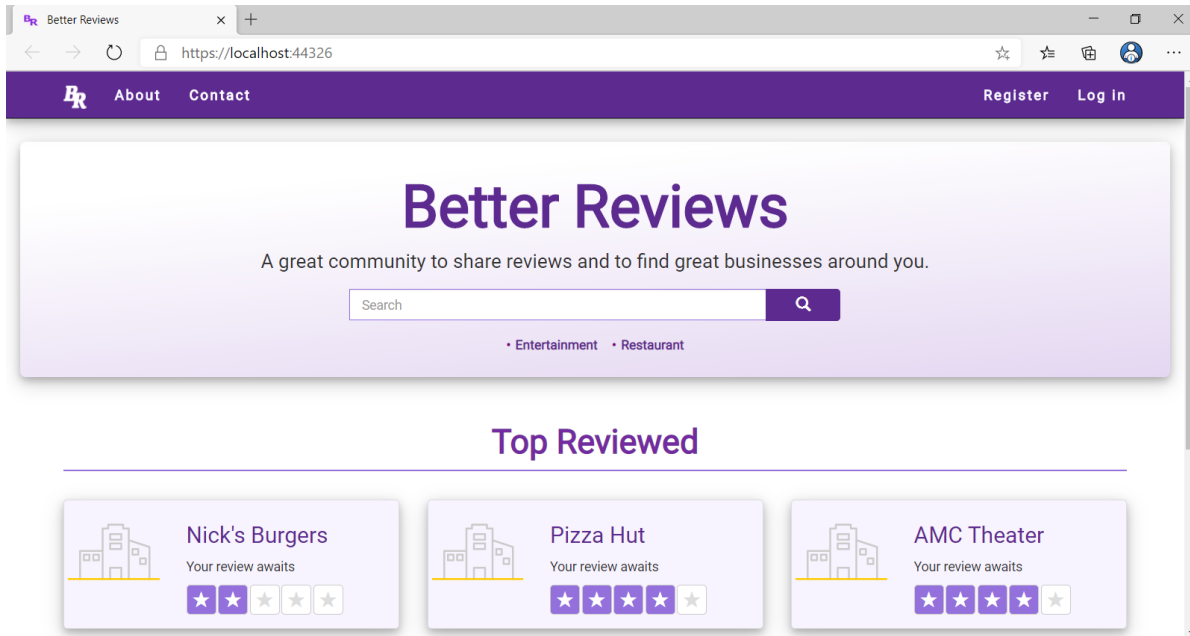


Figure 4.2: Users Click on “Register” at top left of screen and a new user Form is presented to them.

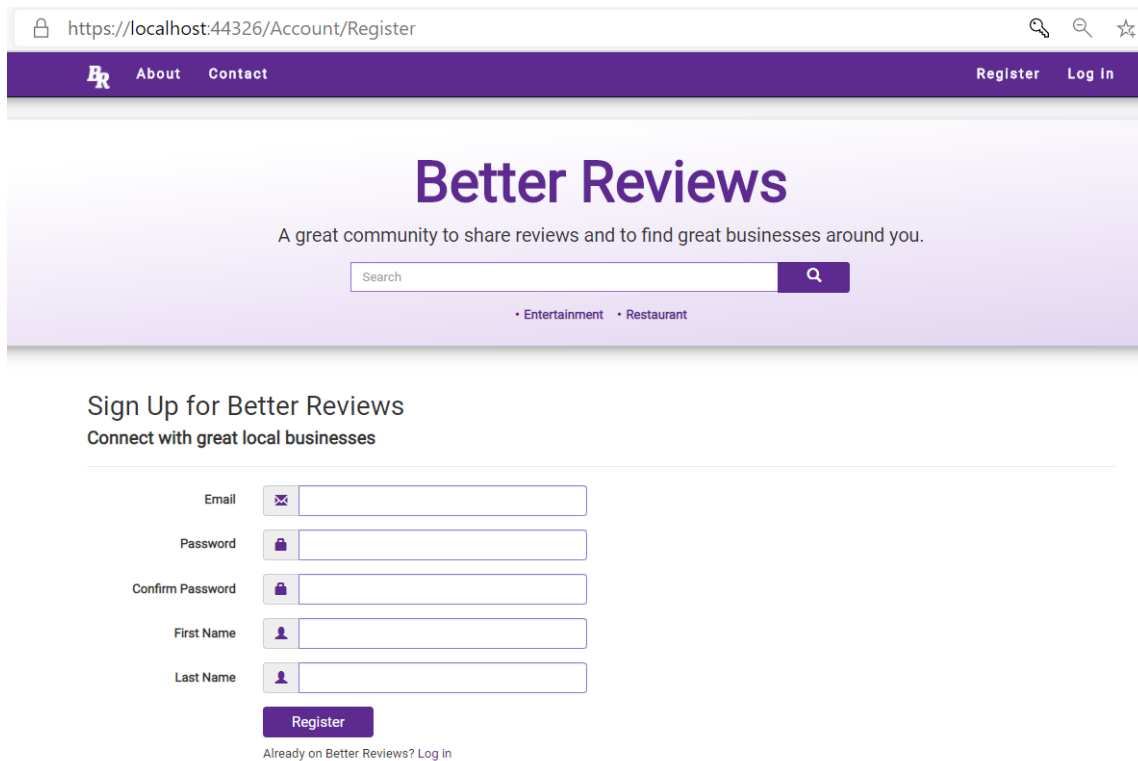


Figure 4.3: The webform is coded with validation and error checking to ensure proper security of entered data: Password Authentication (Rule #1)

Sign Up for Better Reviews

Connect with great local businesses

- The Password must be at least 6 characters long.

Email

Password

Confirm Password

First Name

Last Name

Figure 4.4: Password Authentication (Rule #2)

Sign Up for Better Reviews

Connect with great local businesses

- Passwords must have at least one digit ('0'-'9'). Passwords must have at least one uppercase ('A'-'Z').

Email

Password

Confirm Password

First Name

Last Name

Figure 4.5: Valid Registration Form (Rule #2)

Sign Up for Better Reviews

Connect with great local businesses

Email

Password

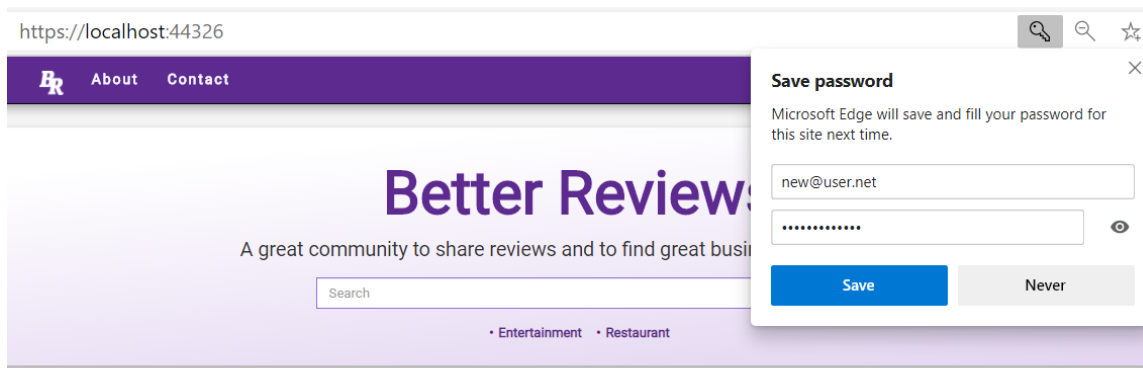
Confirm Password

First Name

Last Name

Already on Better Reviews? [Log in](#)

Figure 4.6: Once initial registration is complete browser sessions enable secure password storage:



END User-Feature: 1.1 Console Access (GUI) for New User

User-Feature: 1.2 Keyword-Based Search Box for Registered Users (page 1 of 3)

Figure 4.7: Any user to the site can browse listings even if they are not logged in. (Log In top right is available)

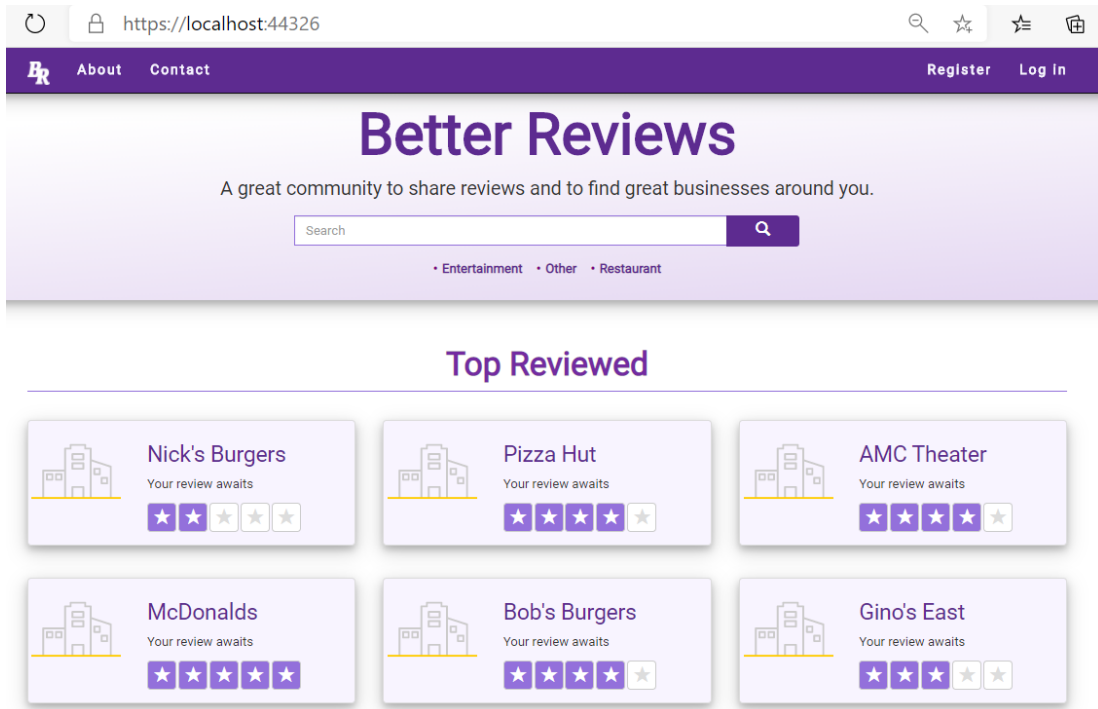


Figure 4.8: Shows active listing with anonymous user (not logged in)

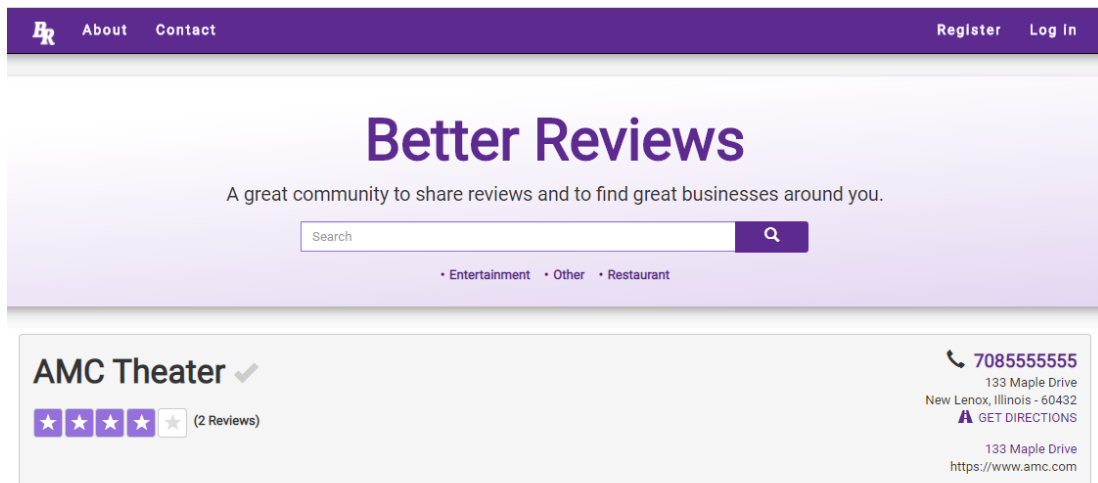


Figure 4.9: Users can also login to the site using credentials created from Feature 1.1

Log in

Use a local account to log in

Email

Password

Remember me?

[Log in](#)

[New to Better Reviews? Register](#)

Categories are created as users enter businesses or organizations (this will be shown in feature 2.1

Figure 4.10: Showing Category: Restaurant:

Better Reviews

A great community to share reviews and to find great businesses around you.

 [Q](#)

• Entertainment • Other • Restaurant

City
New Lenox, Illinois



1. Bob's Burgers
★★★★☆ (1 Reviews)

7085555555
133 Maple Drive
New Lenox, Illinois - 60432

Figure 4.11: Showing Category: Entertainment

Better Reviews

A great community to share reviews and to find great businesses around you.

 [Q](#)

• Entertainment • Other • Restaurant

City
New Lenox, Illinois



1. AMC Theater
★★★★☆ (2 Reviews)

7085555555
133 Maple Drive
New Lenox, Illinois - 60432

User-Feature: 1.2 Keyword-Based Search Box for Registered Users (page 3 of 3)

Figure 4.12: Showing Category Other:

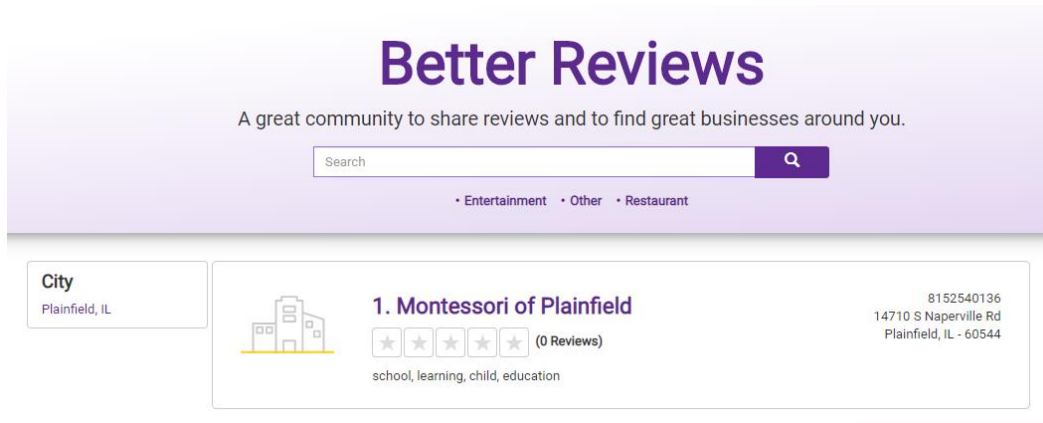
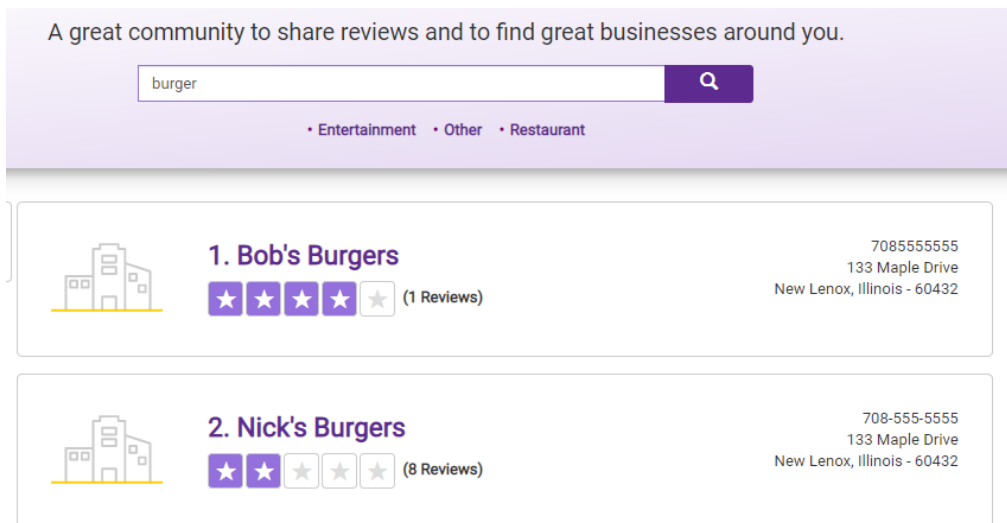


Figure 4.13: In addition to browsing by category, users can also search for businesses by name or keyword:



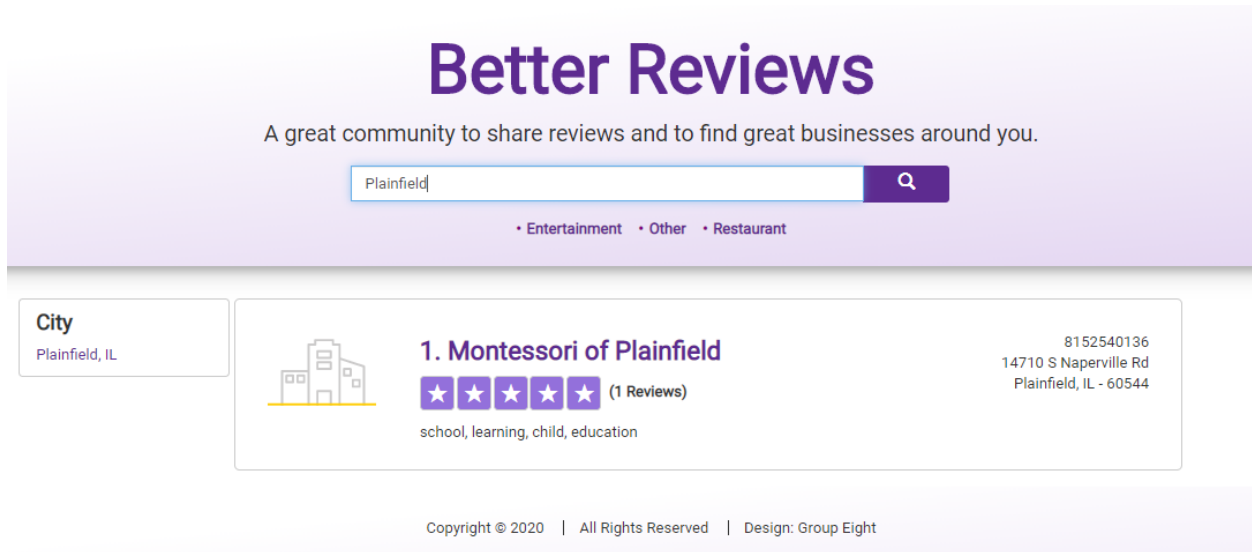
Figure 4.14: Keyword matches are suggested in the drop-down menu, and results are sorted based on keyword:



END User-Feature: 1.2 Keyword-Based Search Box for Registered Users

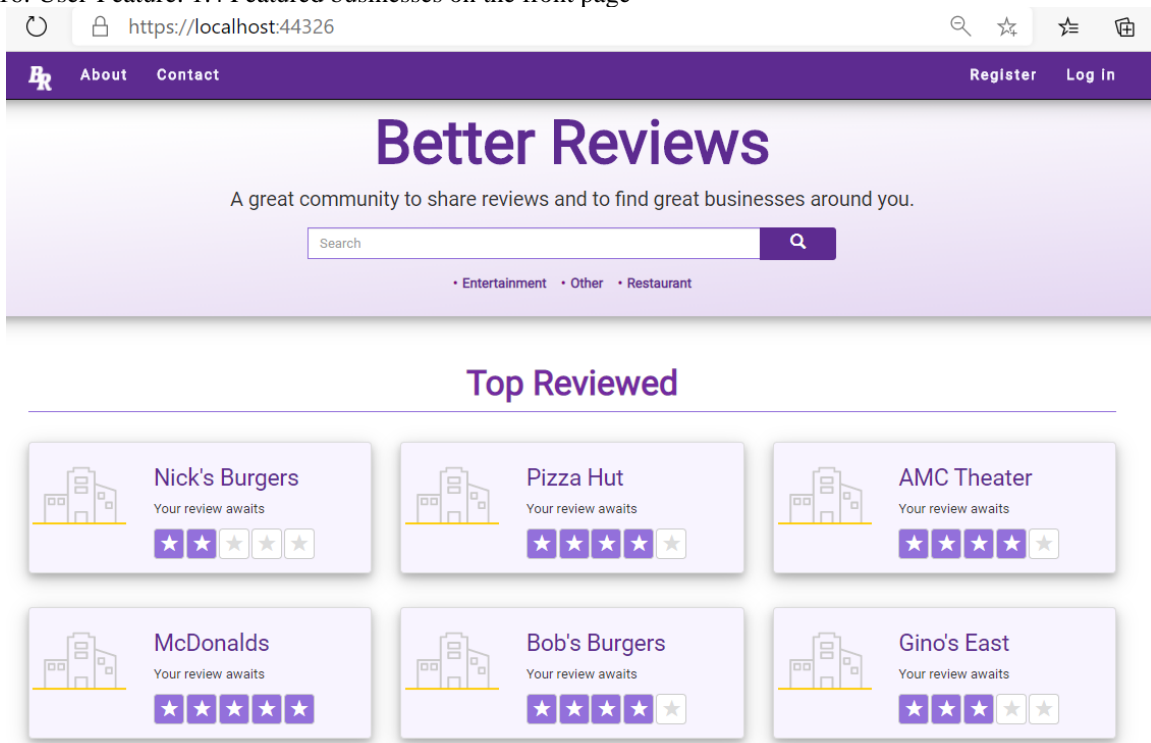
User-Feature: 1.3 Advanced search by category, rating, location, etc. (page 3 of 3)

Figure 4.15: Search by Keyword: City



END User-Feature: 1.3 Advanced search by category, rating, location

Figure 4.16: User-Feature: 1.4 Featured businesses on the front page



END User-Feature: 1.4 Featured businesses on the front page

Figure 4.17: Forgot Password Sub-Feature:

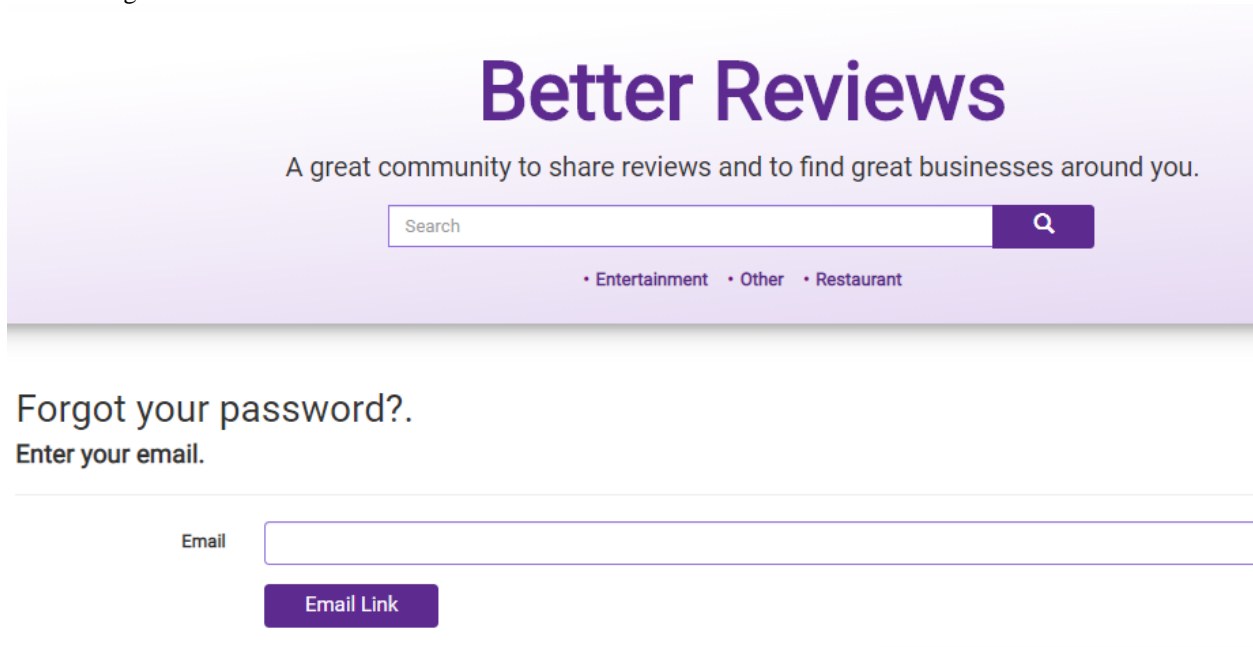


Figure 4.18: Password confirmation screen

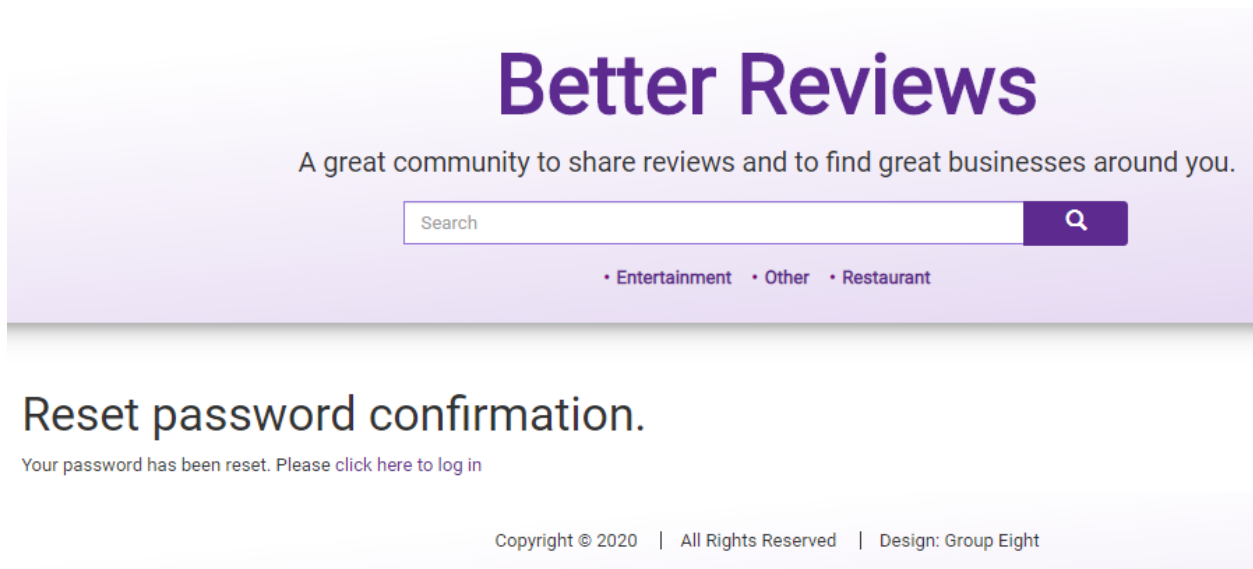


Figure 4.19: Profile Administration:

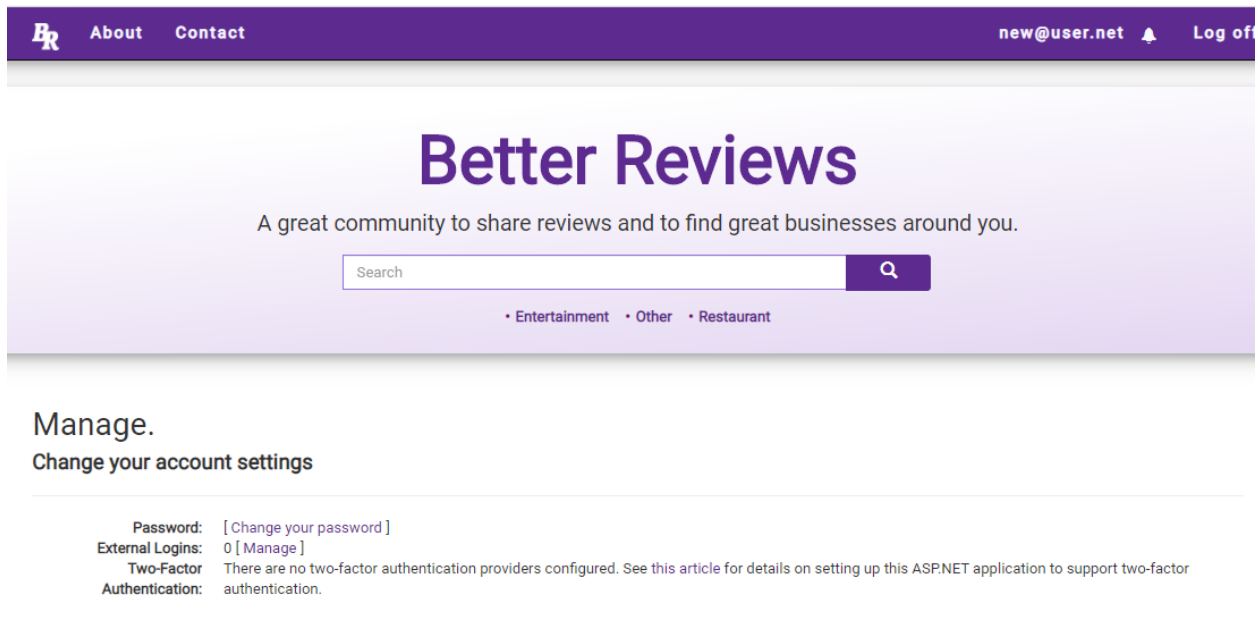


Figure 4.20: Password Administration:

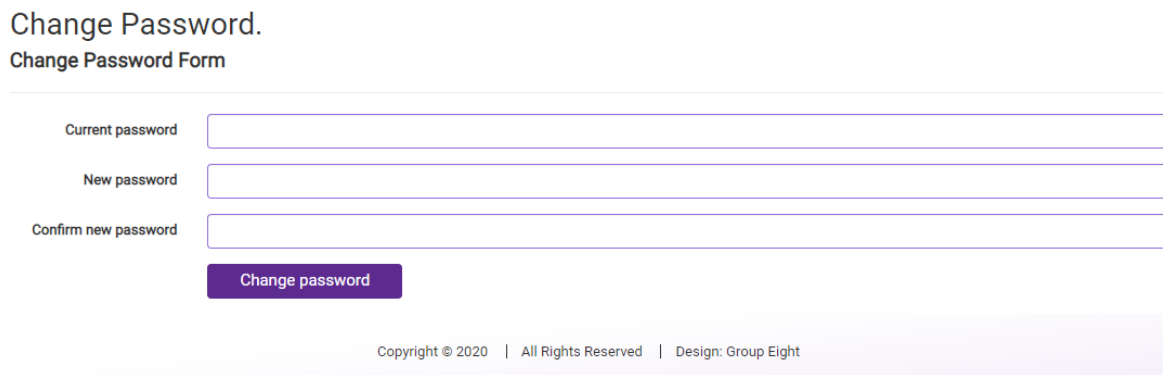


Figure 4.21: External Login Management:

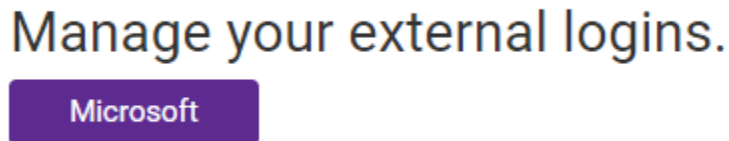
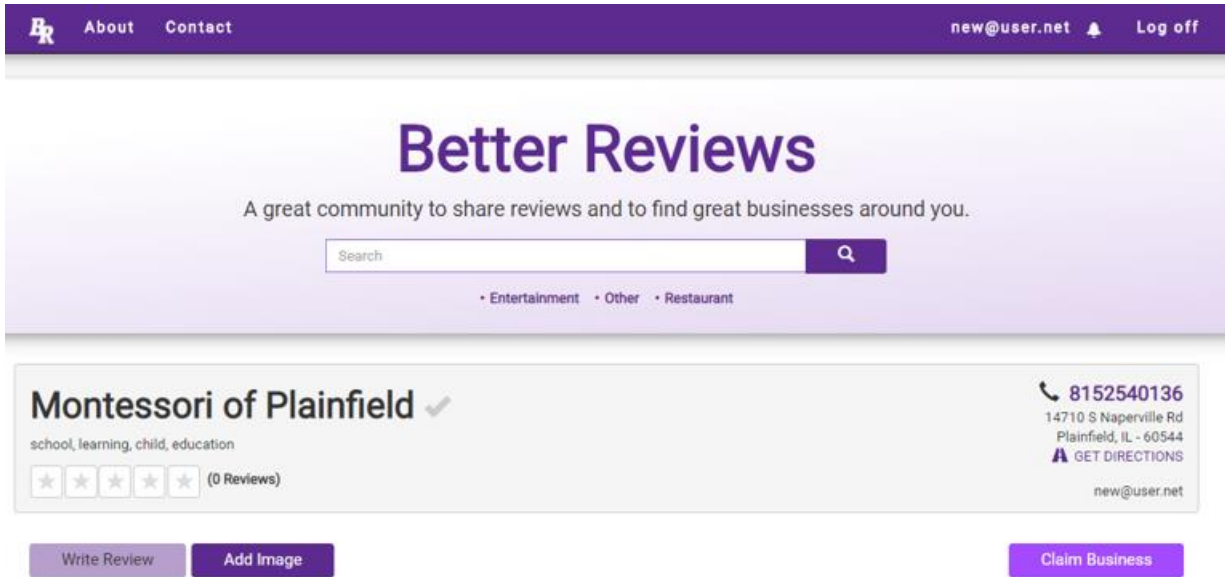


Figure 4.22: Find Business to Review:



Reviews

Figure 4.23: Click “Write Review”:

Write Review

Rating



Select your rating

Review

Write something here....

Upload File

Choose File No file chosen

Create

Figure 4.24: Add text to Review Box

Write Review

Rating

★ ★ ★ ★ ★ Select your rating

Review

Montessori Brings Out the Best Qualities in Your Child At Montessori of Plainfield and Montessori of Frankfort, we take a "whole child" approach that brings out the very best potential qualities in your child]

Upload File

Choose File No file chosen

Create

Figure 4.25: Star Ratings have Additional Semantic Indicators: Star Ratings are selected by clicking on Stars:

Rating

★ ★ ★ ★ ★ Select your rating

Review

Excellent! I highly recommend.

Rating

★ ★ ★ ★ ☆ Select your rating

Review

Yes! I'm a fan.

Rating

★ ★ ★ ☆ ☆ Select your rating

Review

Pretty Good!

Rating

★ ★ ☆ ☆ ☆ Select your rating

Review

Decent, but not the best.

Rating

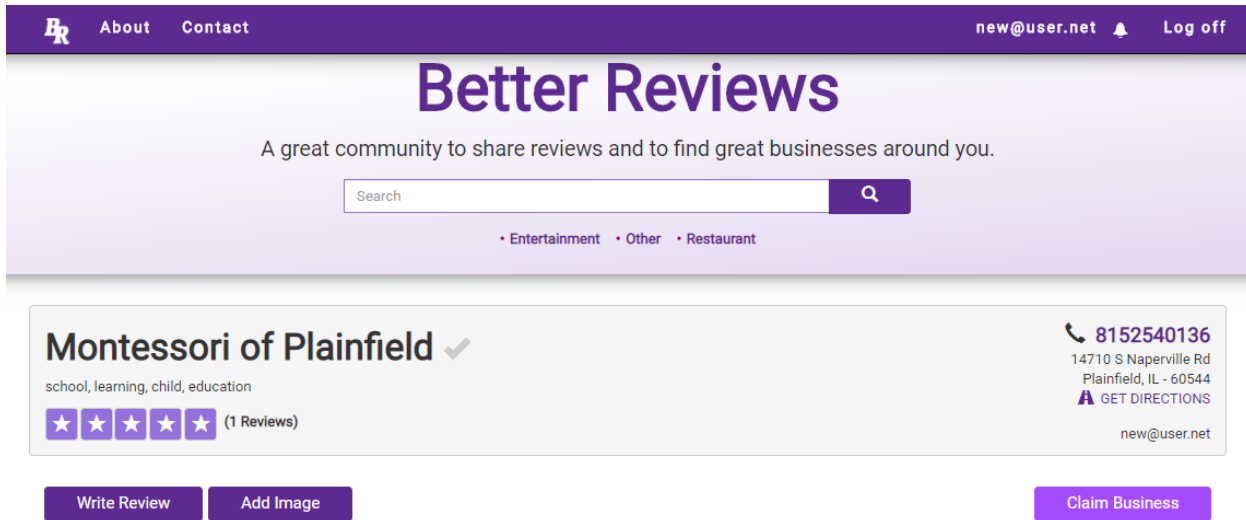
★ ☆ ☆ ☆ ☆ Select your rating

Review

I've had better!

User-Feature: 1.6 Review a Business Feature for Registered Users (page 3 of 3)

Figure 4.26: When Text is Complete User Clicks “Complete” and Review is posted to the site:



END User-Feature: 1.6 Review a Business Feature for Registered Users (page 3 of 3)

Business Users Requirements

Business User-Feature: 2.1 Create Business (GUI) for New Business (page 1 of 3)

Figure 4.27: A New User is offered an option to Add a Business:

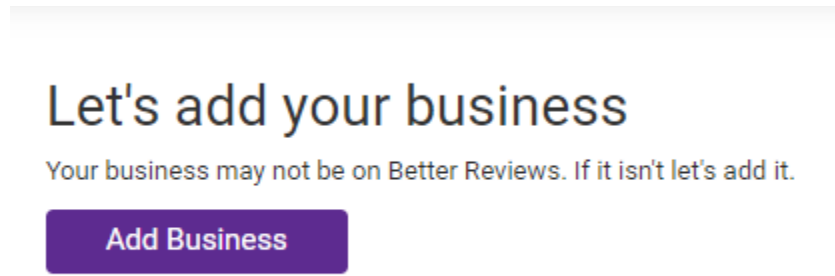


Figure 4.28: Clicking the “Add Business” Button reveals a new business registration form:

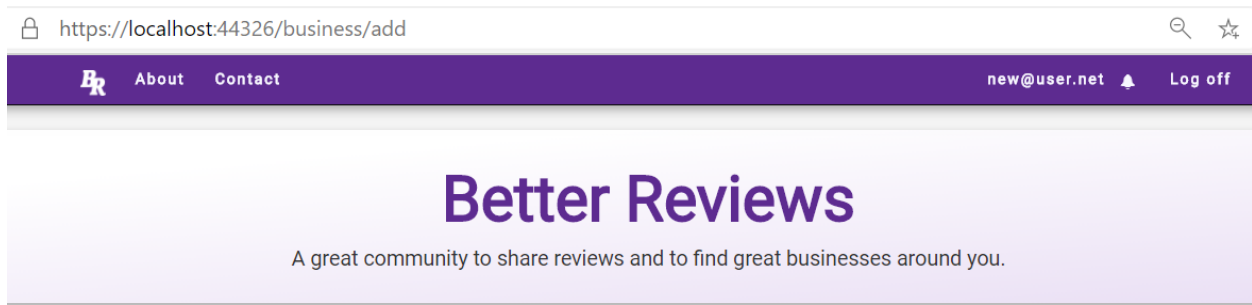


Figure 4.29: Form for Add Business:

Add a New Business


<p>Business Name <input type="text"/></p>	<p>Zip Code <input type="text"/></p>
<p>Business Category <input type="text"/></p>	<p>Country <input type="text"/></p>
<p>Keywords <input type="text"/></p>	<p>Phone Number <input type="text"/></p>
<p>Address <input type="text"/></p>	<p>Website Address <input type="text"/></p>
<p>City <input type="text"/></p>	<p>Email Address <input type="text"/></p>
<p>State <input type="text"/></p>	<p><input type="checkbox"/> I'm not a robot  reCAPTCHA Privacy - Terms</p>
	<p>Add Business</p>

Figure 4.30: Registration Form Supports Auto Fill from Previous Browser Registration Entries:

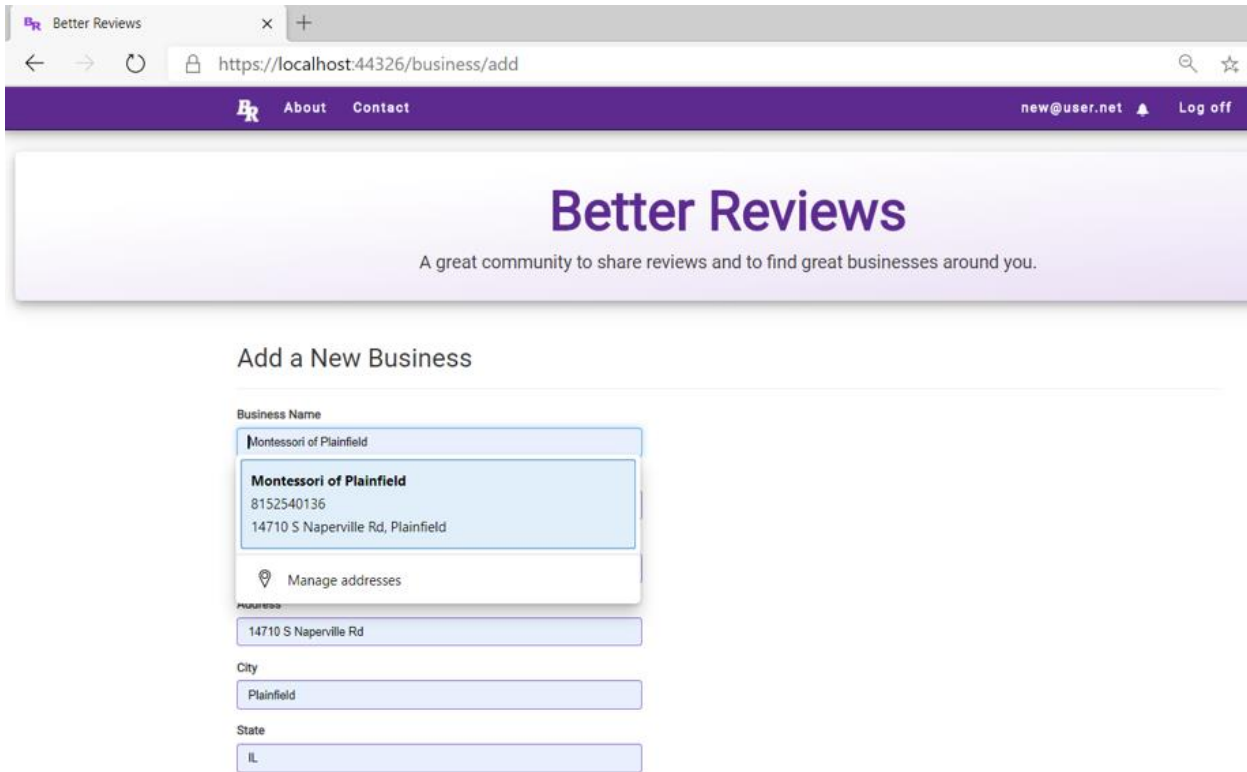


Figure 3.31: Select Category List is a Drop-Down Menu:

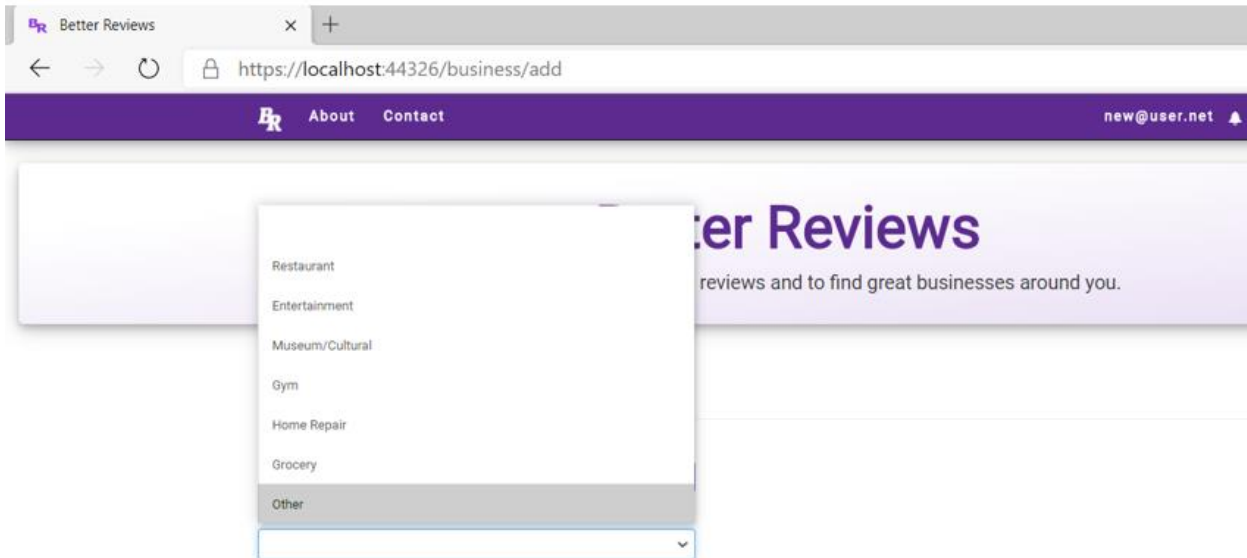


Figure 4.32: Once the user has valid information, their registration attempt is challenged with a Captcha:

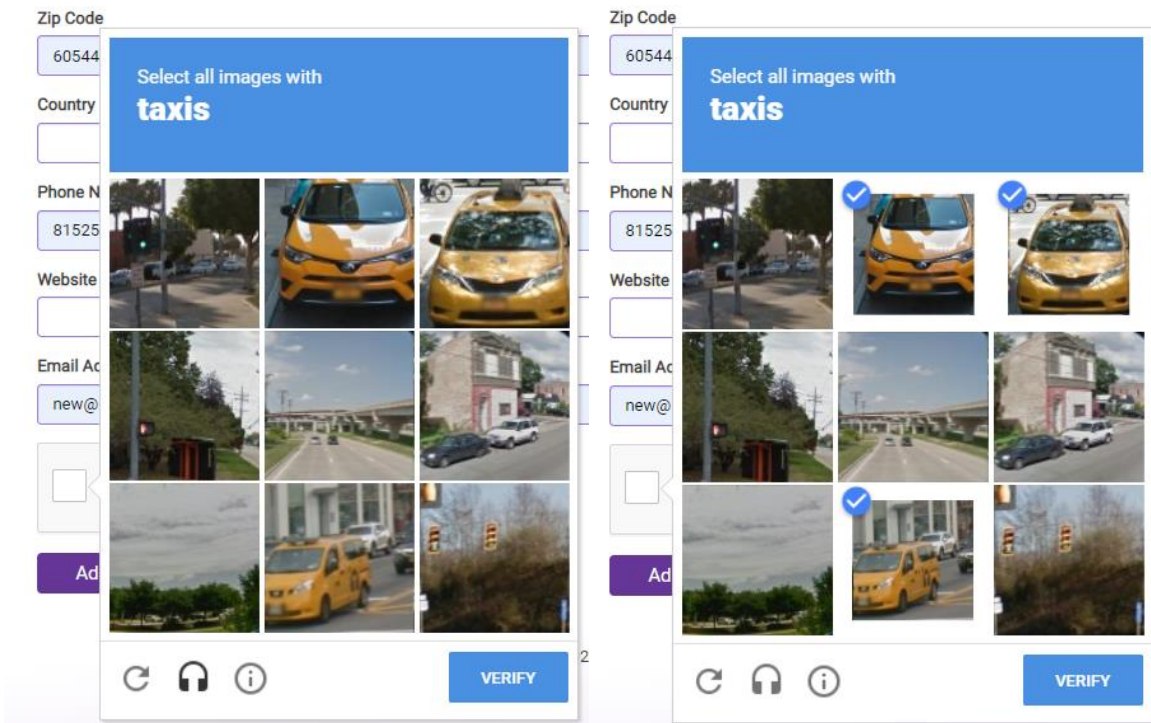
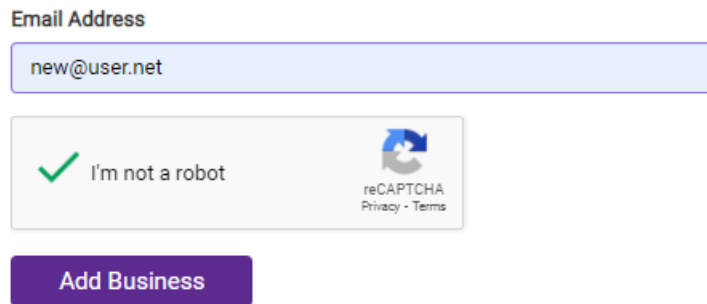


Figure 4.33: The Captcha is validated by API Call:



The User Clicks “Add Business” button to complete the registration.

END Business User-Feature: 2.2 Update Business (GUI) for Business Listing

In order for a user to 'Edit' a business they have to complete a 'Claim' on the business:
Figure 4.34: See Button on lower right of Business Page:

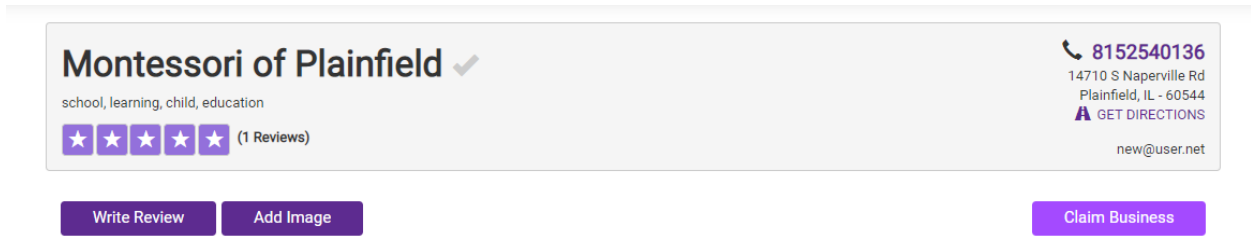


Figure 4.35: Clicking on the Button Reveals a Message Bos, where a user can compose a message to the Site Admin

Claim Business: Montessori of Plainfield

Request Message

This is my business

Submit Claim

Figure 4.36: The user enters text into the Claim Business Form and Clicks Submit Claim. A confirmation is displayed.



Figure 4.37 Admin Side, the message is received by the Site Administrator who see the message appear in their mailbox.

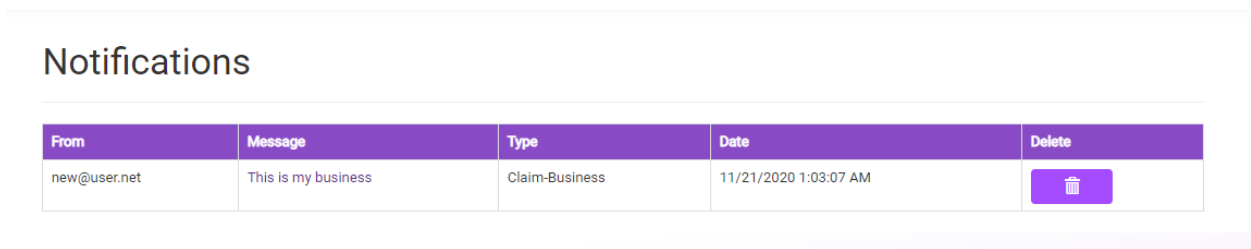


Figure 4.38: The site Admin can Approve or reject the Claim by using the Claim Approval Form:

Claim Approval

Requestor: new@user.net

Business Name: Montessori of Plainfield

Requestor Message: This is my business

Approve

Reject

Figure 4.39: Once the site Admin clicks Approve a confirmation message is displayed to the admin user:

Claim Approved

The request has been approved

Figure 4.40: After the Admin has approved the claim (or if it is denied), a Message is sent to the user. The user receives an alert (tag) in the status bar (top right of the main login screen).



Figure 4.41: The User can view the acceptance message in their Notifications message center.

Notifications

From	Message	Type	Date	Delete
new@user.net	You request has been approved	Claim-Approved	11/21/2020 1:05:43 AM	
new@user.net	This is my business	Claim-Business	11/21/2020 1:03:07 AM	

Figure 4.42: The next time the user (now business manager) visits the Business Listing, an Edit Button appears:

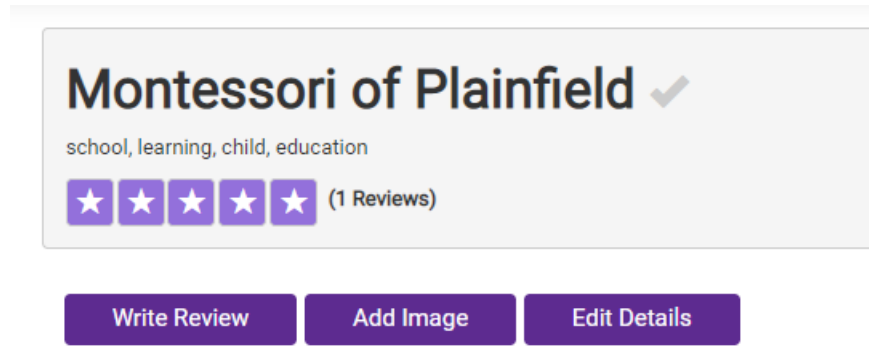


Figure 4.43: Clicking the “Edit Details” Button reveals a web form for editing the details of the business:

Edit Business

Business Name	<input type="text" value="Montessori of Plainfield"/>
BusinessCategoryId	<input style="border-bottom: 1px solid #ccc;" type="text" value="Other"/>
Keywords	<input type="text" value="school, learning, child, education"/>
Website Address	<input type="text"/>
Email Address	<input type="text" value="new@user.net"/>
Phone Number	<input type="text" value="8152540136"/>
Address	<input type="text" value="14710 S Naperville Rd"/>
City	<input type="text" value="Plainfield"/>
State	<input type="text" value="IL"/>
Zip Code	<input type="text" value="60544"/>
Country	<input type="text" value="Will"/>

END Business User-Feature: 2.2 Update Business (GUI) for Business Listing

Figure 4.44: To upload (or delete) an images to a business a user must be registered (as a user) and have submitted a valid claim request. The logged in user then when visiting the business listing will see an “Add Image” button on the main Business Listing page:

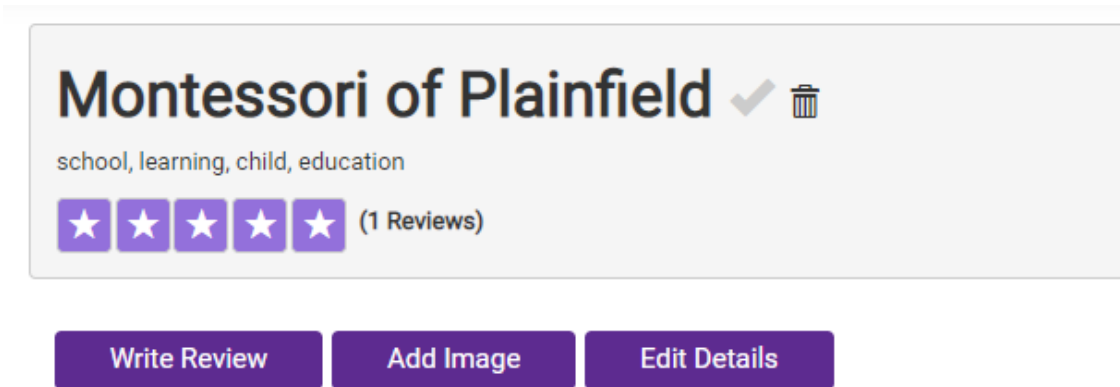


Figure 4.45: A File Picker Web Form is Presented

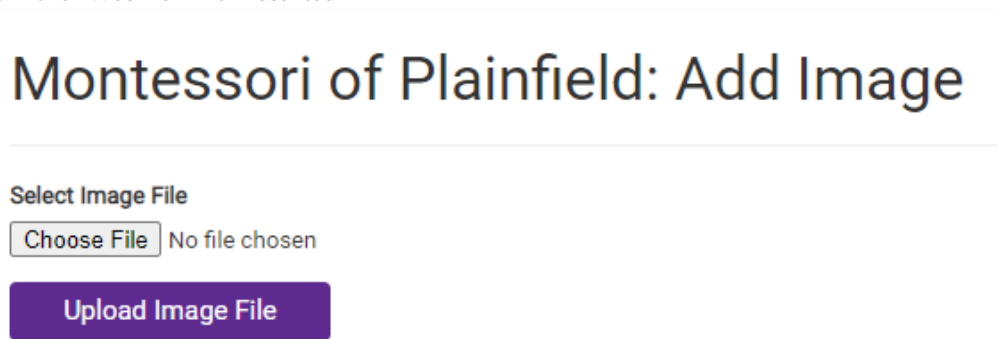


Figure 4.46: The user navigates to the image they wish to upload and select if from their file system:

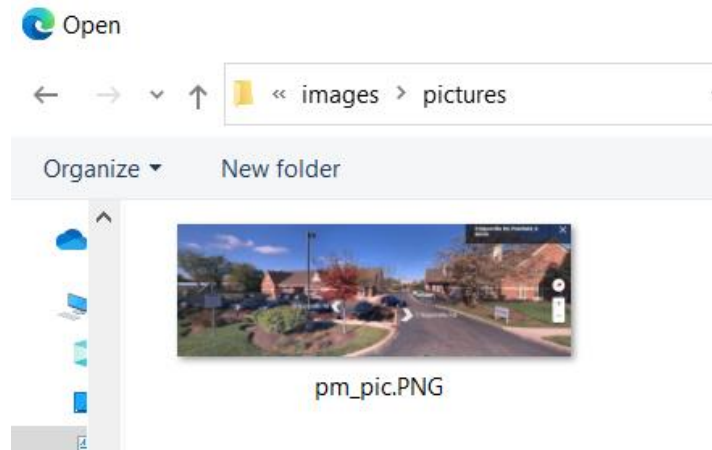


Figure 4.47: A confirmation (the file name is displayed) in the form, and the user clicks the Upload Image File to complete the process.

Montessori of Plainfield: Add Image

Select Image File

Choose File pm_pic.PNG

Upload Image File

Figure 4.48: The Image is added to the business listing. In the initial display the image is slightly cropped (into a rectangle).

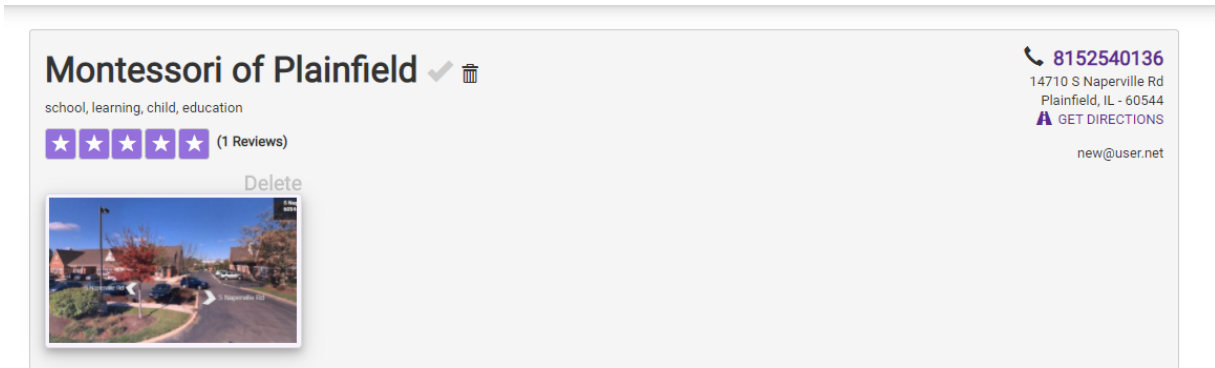
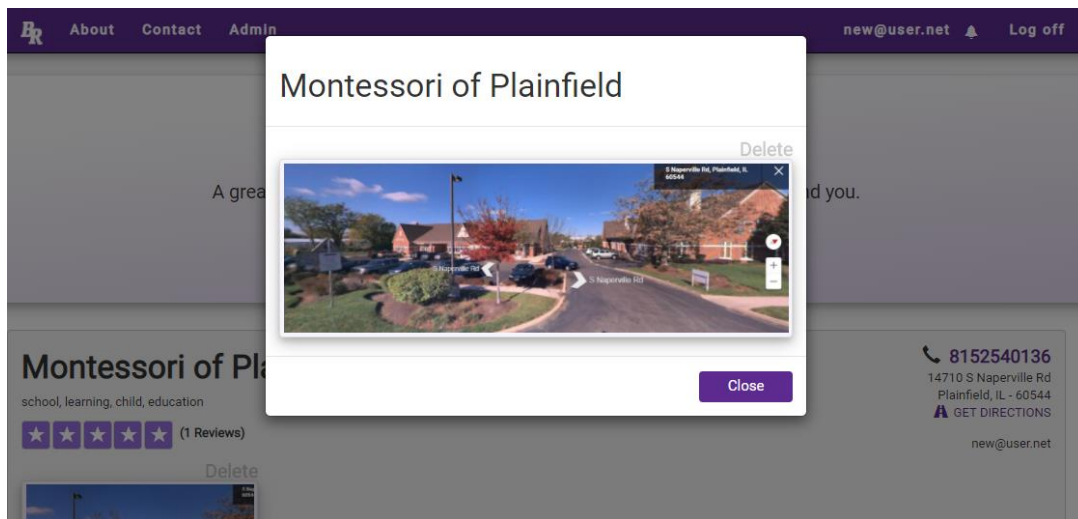


Figure 4.49: The full picture can be viewed by clicking on the thumbnail version of it from the Business Listing page. The picture is revealed in a modal box:

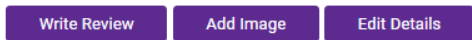
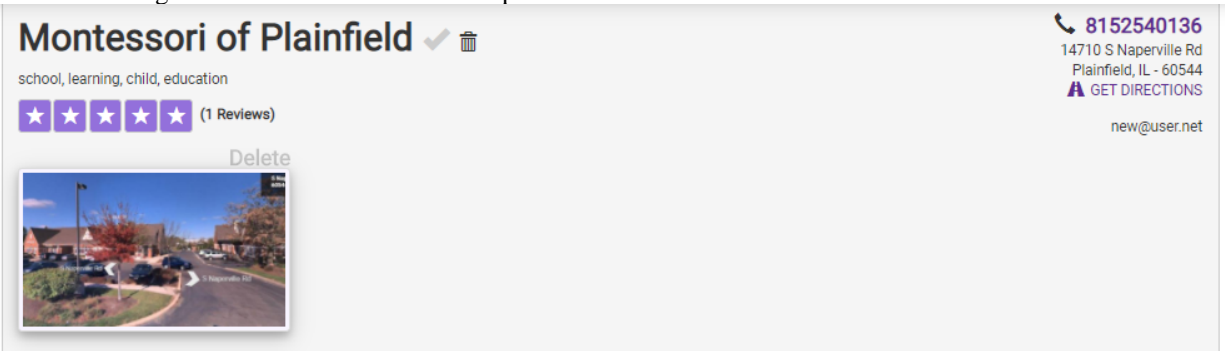


END Business User-Feature: 2.3 Add and Remove Photo(s) for Business Listing

Business User-Feature: 2.4 Subscribe to Reviews (posted by other users)

Once a business user is confirmed as the business owners, they have an automatic subscription to the reviews posted to their business site. A business user who is logged into the system can find all the reviews (including new ones) posted below their business profile.

Figure 4.50: The image below shows the user review posted in User Feature 1.6.



Reviews



Figure 4.51: The Business manager has the option to 'respond' to a user using the user Message feature. See the "Respond" button at the lower right.



END Business User-Feature: 2.4 Subscribe to Reviews (posted by other users)

Figure 4.52: Registered Users have the option to send a message directly to the site admin using the “Contact” button at the top of the page.



Figure 4.53: Clicking on “Contact” opens a web form to send a message:

A contact form titled "Get in touch with Better Reviews". It features three input fields on the left: "Name", "Email", and "Subject". To the right is a larger "Message" text area. Below the fields is a purple "Send Message" button. At the bottom, there is a footer: "Copyright © 2020 | All Rights Reserved | Design: Group Eight".

Figure 4.54: Like other features in the App, Validation takes place on this form.

The contact form from Figure 4.53, but with a validation error. The "Email" field is highlighted with a red border, and a tooltip with an exclamation mark icon says "Please fill out this field.".

Figure 4.55: The web form also provides auto fill for previously provided information:

The contact form with an auto-fill dropdown menu open over the "Name" field. The dropdown shows a "New User" entry with the email "new@user.net" and a "Manage" option with a location pin icon. The "Send Message" button is visible at the bottom.

Figure 4.56: A properly filled out Message Request:

Get in touch with Better Reviews

New User

new@user.net

Send Message to Admin

This is a message being sent to the Site Admin. Thanks for a great Site!!

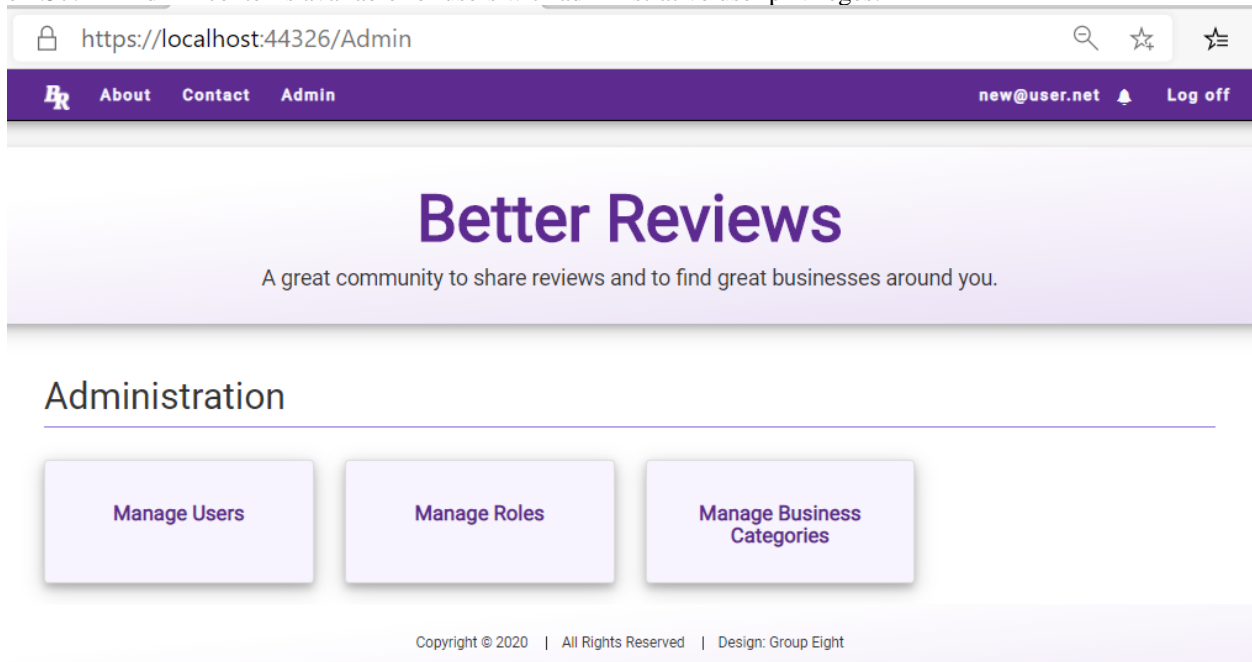
Send Message

The message shows up in the Admin Message Notification Center and correspondence between the users can occur through the notification center as seen before in Business User Feature 2.2

END Business User-Feature: 2.2 Update Business (GUI) for Business Listing

Admin User-Feature: 3.1 Admin User-Feature: Admin Console (GUI) for Site Administrators

Figure 4.57: An Admin center is available for users with administrative user privileges:



END 3.1 Admin User-Feature: Admin Console (GUI) for Site Administrators

3.2 Admin User-Feature: Manage Users (page 1 of 2)

Figure 4.58: Clicking on the Manage Users Button above reveals the User Management Center

Manage Users

Username	Last Name	Firt Name	Roles
mauro.patino@gmail.com			
test@gmail.com			
test2@gmail.com			
test3@gmail.com			
mauro.patino@commscope.com	Patino	Mauro	
mpatino2@student.govst.edu	Patino	Mauro(School)	
new@user.net	User	New	Admin
test4@gmail.com	User	Test4	

3.2 Admin User-Feature: Manage Users (page 2 of 2)

Figure 4.59: Clicking on a username (a registered user) opens an interface to edit settings for the selected user:

Edit User

User Info

UserName test3@gmail.com

First Name

Last Name

Roles

Admin

Save

END 3.2 Admin User-Feature: Manage Users

3.3 Admin User-Feature: Manage Users Roles

To change the status of a user, for example from a registered user to an admin the interface an existing admin user will log in, access the Manage Users Screen, Select the user, and enable the check-box at the bottom of the screen.

Figure 4.60: In this illustration, the user test@gmail.com will be converted from a simple registered user to an admin:

test@gmail.com			
----------------	--	--	--

Figure 4.61: Shows Toggle switch option for Roles

Roles

Admin

Save

Roles

Admin

Save

Figure 4.62: Shows Updated user with new Admin role

test@gmail.com		Admin
----------------	--	-------

END 3.3 Admin User-Feature: Manage Users Roles

3.4 Admin User-Feature: Manage Business Categories/Create Category (page 1 of 2)

Multiple types of roles can be created for the application which will allow fine-grain control over the features made available to individual and groups of users who interact with the app.

Figure 4.63: Admin users have the ability to create new roles by clicking the “Manage Roles” Button:

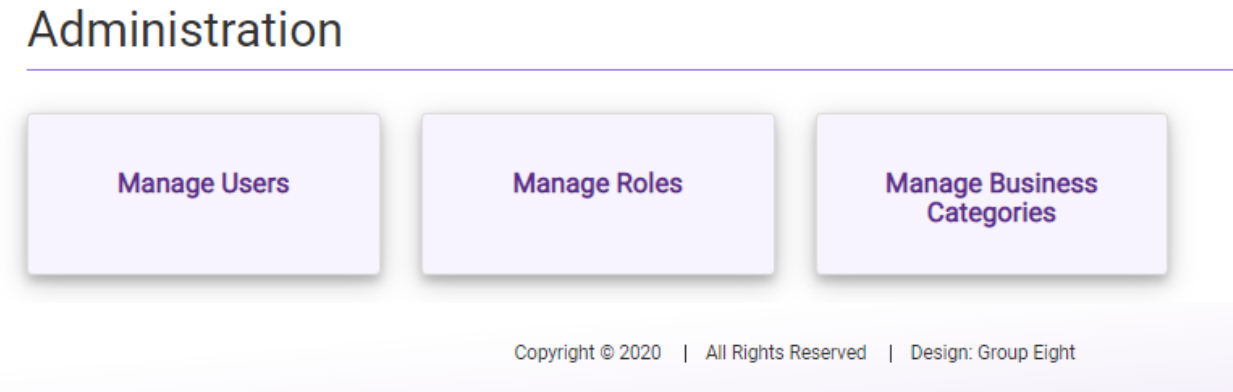


Figure 4.64: Clicking the Button displays an interface which allows an admin to create or remove a user role:

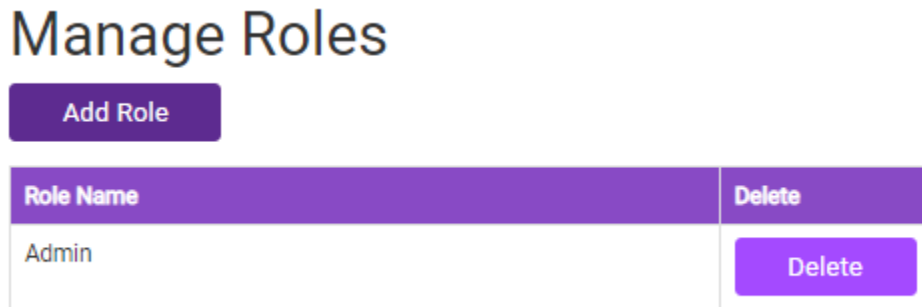
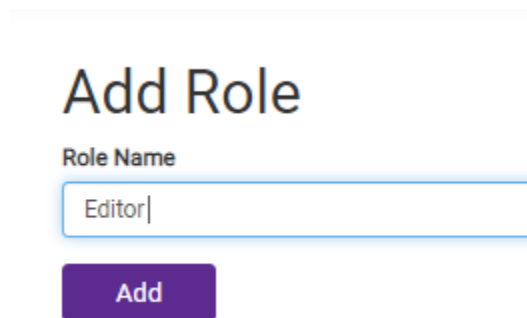
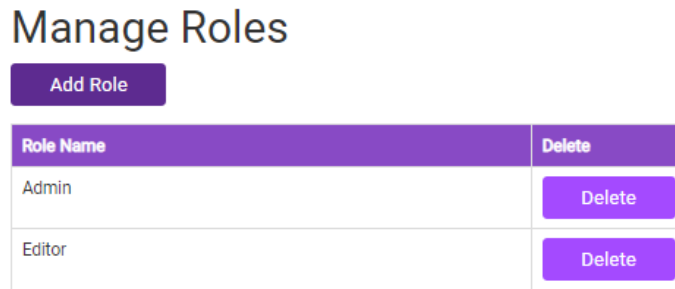


Figure 4.65: To add a new role(for example an “Editor”) the admin user can click Add Role and type in the name for the new user category:



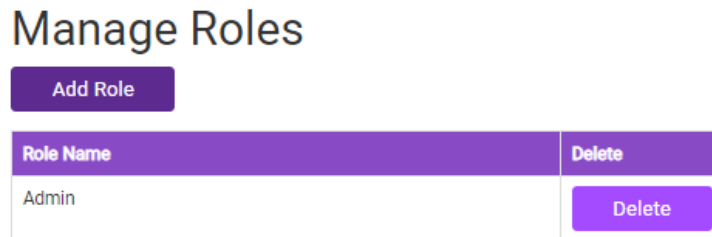
3.4 Admin User-Feature: Manage Business Categories/Create Category (page 2 of 2)

Figure 4.66: The new category is added to the list of User Roles:



If a user category is no longer needed, the logged in Admin user can click the Delete button for the role which is no longer required/useful.

Figure 4.67: The role is then removed, and the list is re-displayed without the category present:

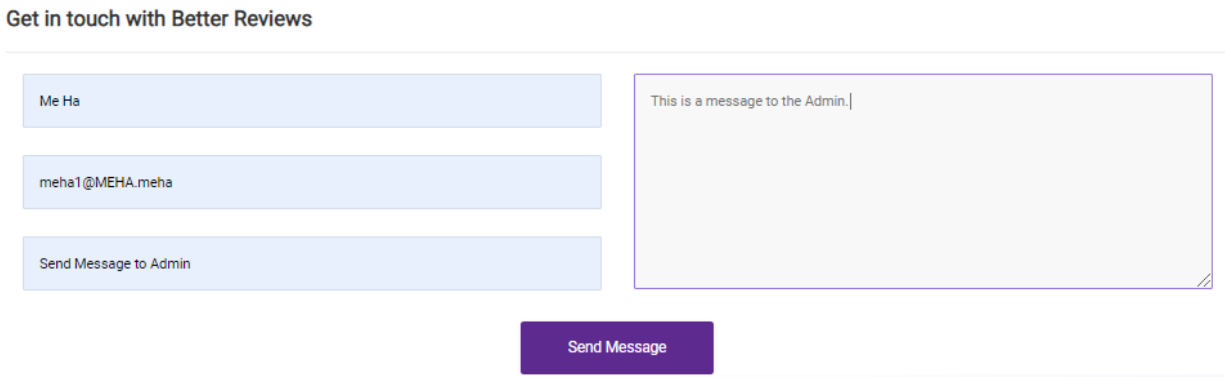


END 3.4 Admin User-Feature: Manage Business Categories/Create Category

3.5 Admin User-Feature: Send/Respond to User Notifications/Messages (Page 1 of 2)

Any users has the ability to send messages to Admin(s) using the contact button at the top of the site. Admins receive messages to their Notification center, and correspondence between users take place through this interface.

Figure 4.68 To illustrate a user (Me Ha) is logged in and access the Send Message Interface:



3.5 Admin User-Feature: Send/Respond to User Notifications/Messages (Page 2 of 2)

Figure 4.69: When the admin user logs in, they will receive an alert in the notification message bar:

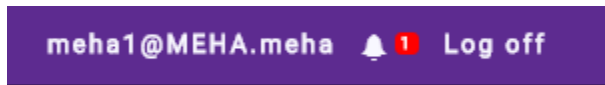


Figure 4.70: Clicking on the Bell Icon, the admin user can then go directly to the message center where the new message is listed:

Notifications

From	Message
meha1@MEHA.meha	This is a message to the Admin

Figure 4.71: The admin user can respond to the message which triggers an alert in the message center bar (see below). The users can then correspond to one another through the Notification center which has been illustrated in Business User Feature(s) 2.5:



END 3.5 Admin User-Feature: Send/Respond to User Notifications/Messages

3.6 Admin User-Feature: Update/Add, Edit Existing Business (page 1 of 3)

By default, All Admin users are provided with access to create, update/edit or remove businesses. To illustrate this the new user Me Ha will be upgraded to become an Admin user, and then once logged in will visit an existing business. What will be revealed is that the “Edit” business button will be active for each existing business.

Figure 4.72: Me Ha is initially not an admin user:

meha1@MEHA.meha	Ha	Me	
-----------------	----	----	--

Figure 4.73: Me Ha is elevated to admin user level:

meha1@MEHA.meha	Ha	Me	Admin
-----------------	----	----	-------

3.6 Admin User-Feature: Update/Add, Edit Existing Business (page 2 of 2)

Figure 4.74: The user Me Ha logs in and will visit the three businesses listed on the Featured Business Page:

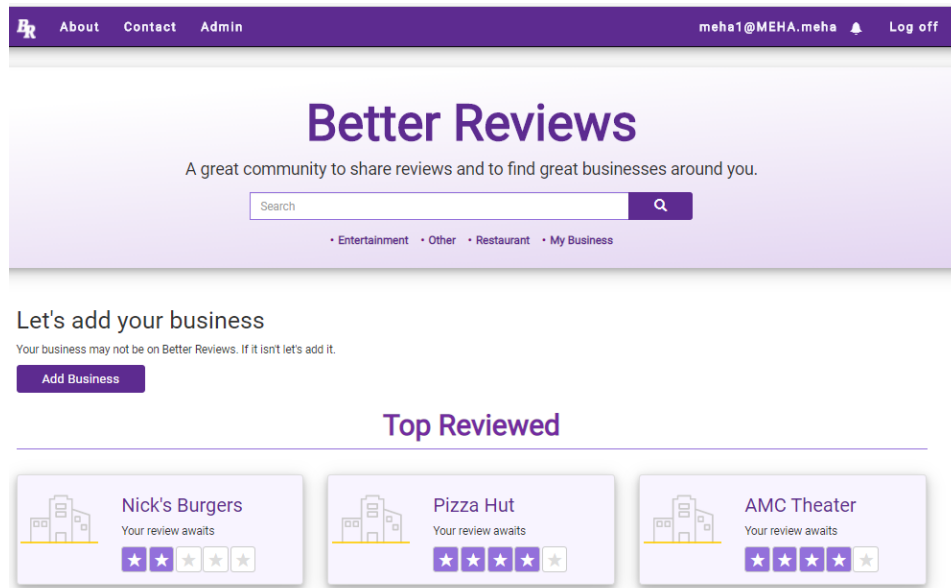


Figure 4.75: Admin User (Me HA) has the ability to Delete (trash can icon) update (Write Review, Add Image and Edit Details) for Nick's Burgers:

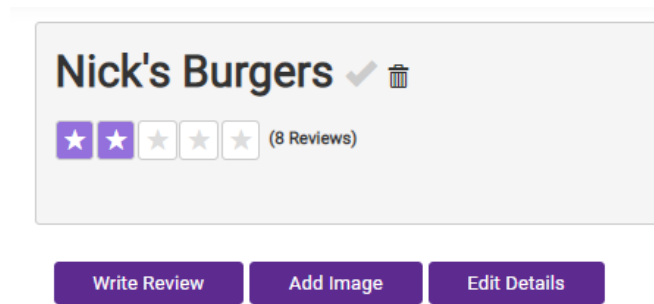
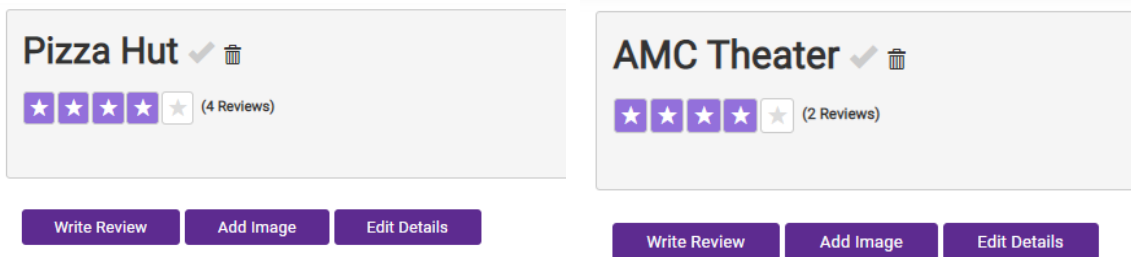


Figure 4.76: Similarly Me Ha also has admin privileges for Pizza Hut and AMC Theaters:



END 3.6 Admin User-Feature: Update/Add, Edit Existing Business

5 Internal/external Interface Impacts and Specification

The programming paradigm selected for this application follows the MVC or Model, View, Controller software design pattern (see <https://en.wikipedia.org/wiki/Model-view-controller>).

This modern programming methodology separates the concerns and code of an application into three separate layers, with the Model as the database layer, the View as the front end or the user interface, and the Controller as the classes which act as the middleware or the programming layer.

By adopting this separation of concerns within the application individual components (for example the front end or web pages) can be independently updated or even completely swapped out with a new design without impacting the programming (Controller) or Database layers which it is connected to.

Likewise, if an eventual application is run on the cloud, and a cloud-based database management system is selected other than SQL-Server (the current DBMS that the application uses) is used, the other components of the application do not need to each be changed to accommodate the change to the DBMS.

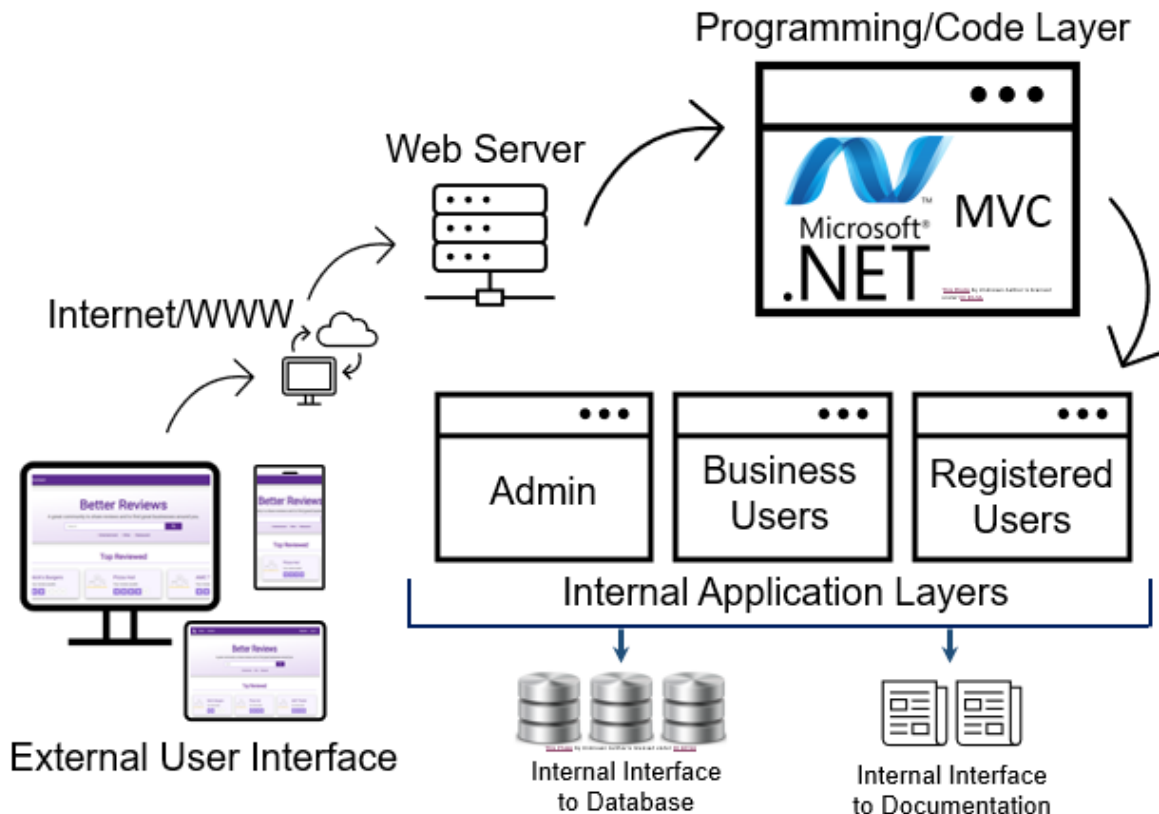
In the application being disclosed here, the MVC layers are depicted in the chart below.

View (External User Interface)

Controller (The Programming/Code Layer, including the three separate internal apps (Admin, Business and User))

Model (The connected Database).

Figure 5.1: Better Reviews System Design Overview:



In addition to brief explanations of each of the components provided in the illustration above, this section will include a fine-grain depiction of the components which are included in each of the MVC layers.

The Main Solution or File Explorer Window is shown first to provide an orientation as to the layout of the files, folders, and MVC components which are present (see graphic to the left side below).

A discussion of each of the individual MVC component will follow this summary.

Figure 5.2: Solutions Explorer View

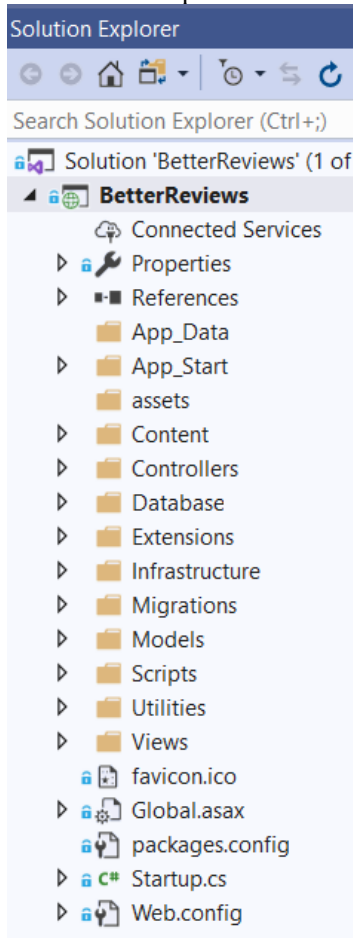
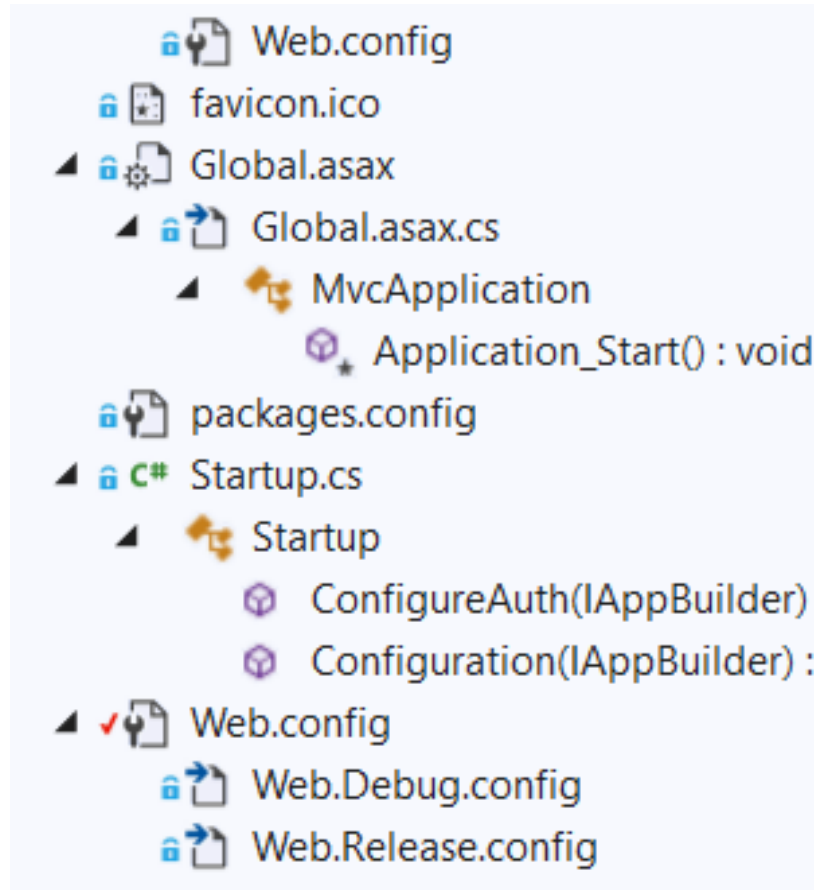


Figure 5.3: Expanded View of Solutions Explorer System files



Web Configuration and Miscellaneous System Files (Shown above in graphic to the Right)

It should be noted that the Web Server/Internet WWW layer pictured in the graphic at the start of this section is not a traditional component of the MVC paradigm. Although it is theoretically present in any web application that has ever been published, it is considered a generic interface, which could be handled by any web server or cloud service provider configuration.

For this project, an Internet Information Services (IIS (see wikipedia.org/wiki/Internet_Information_Services) Web server is used for development and testing, however we have also included the Razor Markup Language library (see new get package listing at the end of this section, or nuget.org/packages/Microsoft.AspNet.Core.Razor/ for more details), which allows the application to run using the more modern Razor web services implementation.

Application Views

External User Interface (displays on desktop, laptop, tablet and mobile devices)

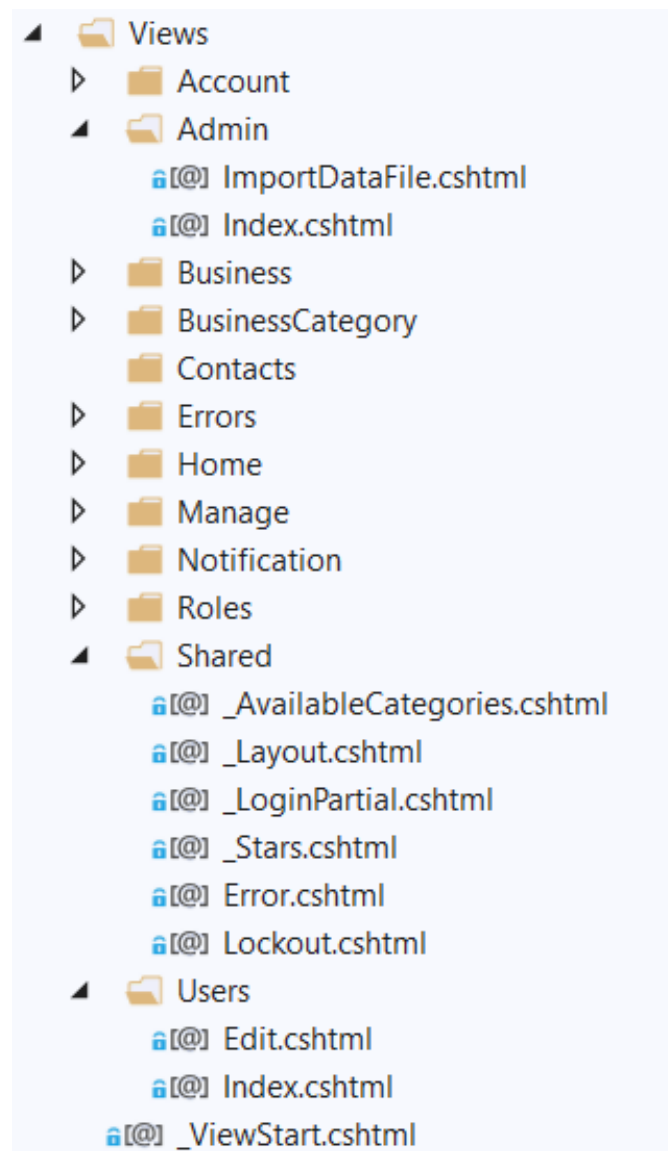
The application is displayed using HTML/CSS, JavaScript, and Bootstrap, and is created as a responsive website. This means that the display updated dynamically to the device which a user accesses the app on. For example, a full-size desktop or laptop screen will display the application in a traditional wide-screen mode which is historically in landscape orientation.

Similarly where a user displays the application on a tablet (i.e., iPad or Samsung Tablet), the display will still be in horizontal mode, but will be more compressed, adjusting for the limited screen size. Finally on displays which are portrait (or taller than they are wide) the display again adjusts, and many images are either eliminated entirely, or are dramatically scaled due the dramatic reduction in available screen real estate.

Interfaces Affected:

User Interface (Desktop, Laptop, Tablet, Mobile Device Graphical User Interface (GUI))

Figure 5.4: Solutions Explorer Showing Views: (Main List)



Additional View(s) Details

Figure 5.5: Account Admin and Business Views

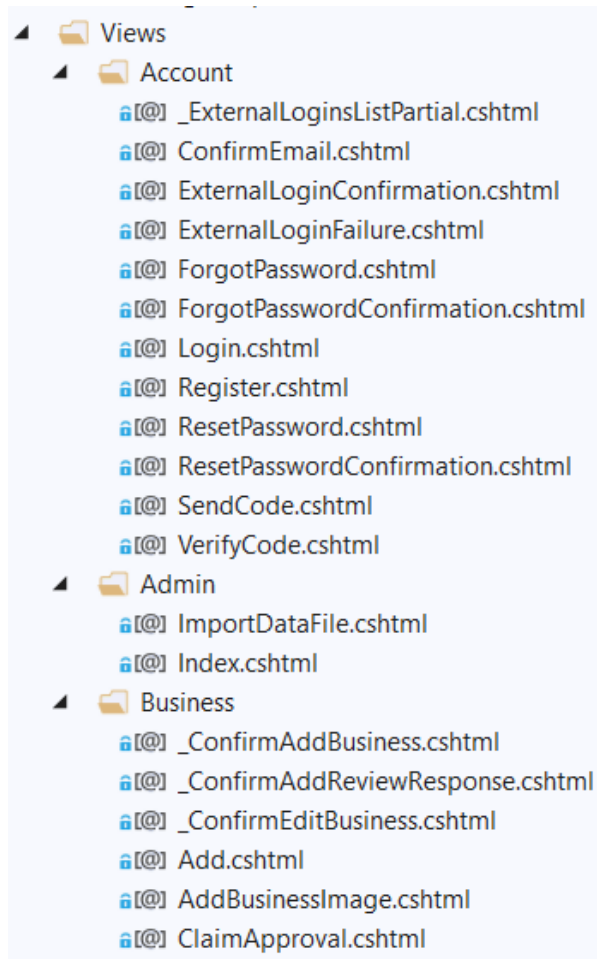
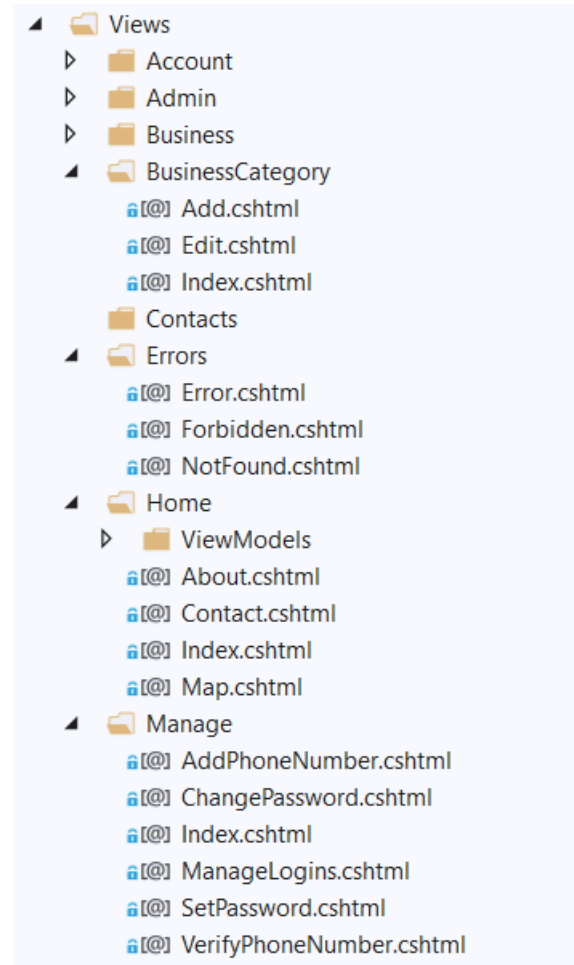


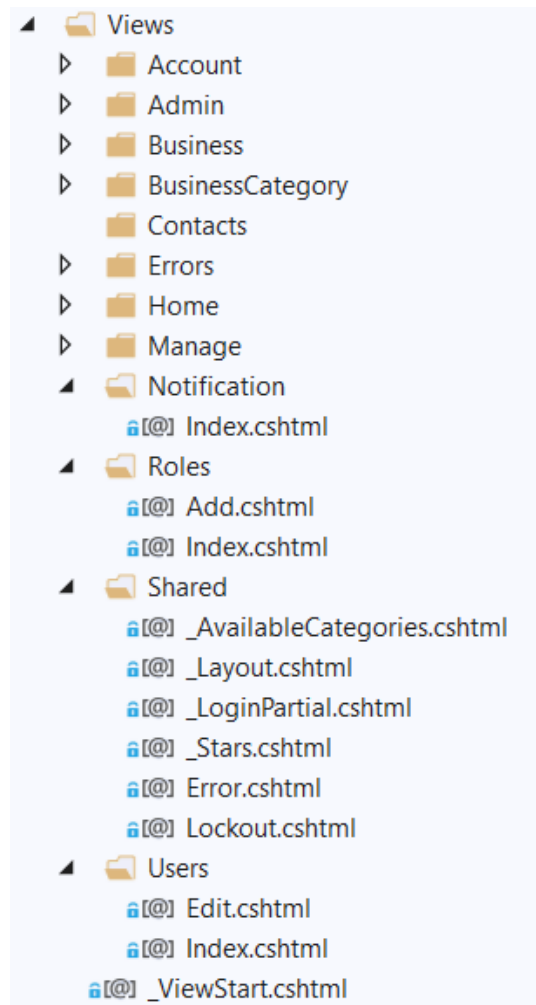
Figure 5.6 (Figure 5.4 Continued)



Additional View(s) Details

Of particular interest in the views folder shown below is the Shared Folder. The majority of the views used in the external GUI interface make use of the Shared View (one or more) in their implementation. For example the (Notification > Index.cshtml) view inherits the (Shared > Layout.cshtml) view, and then extends this template with the features which are relevant to its individual page. This inheritance scheme (code reuse) is applied throughout the application as a way to reduce the total amount of repeated code that would otherwise be required.

Figure 5.7: Account Admin and Business Views:



Database Layer (Represents Internal Interface to Database Abstraction)

The discussion of “Models’ in the Application Model Layer of MVC, for this application will be illustrated both via the ERD and Entities found exploring the database with Visual Explorer in SQL Microsoft Management Studio (MMS) as also as found in the Solutions and File Explorer for Microsoft Visual Studio.

Figure 5.8: Database viewed through Visual Studio

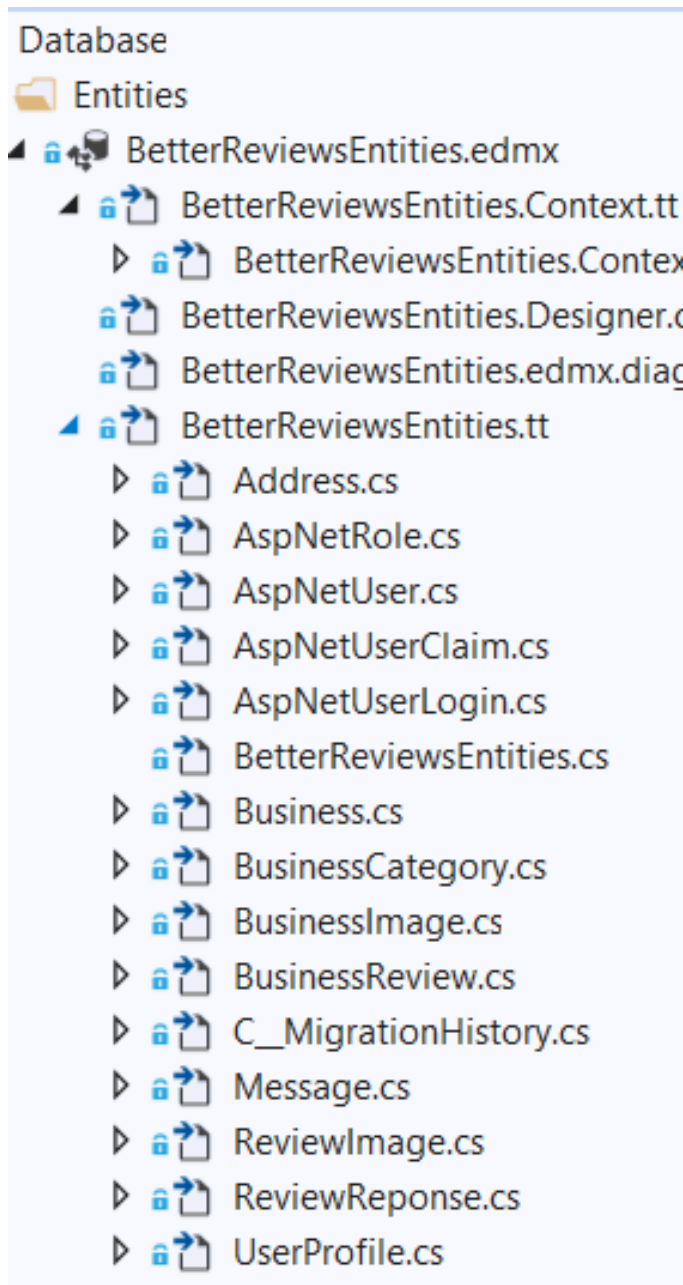
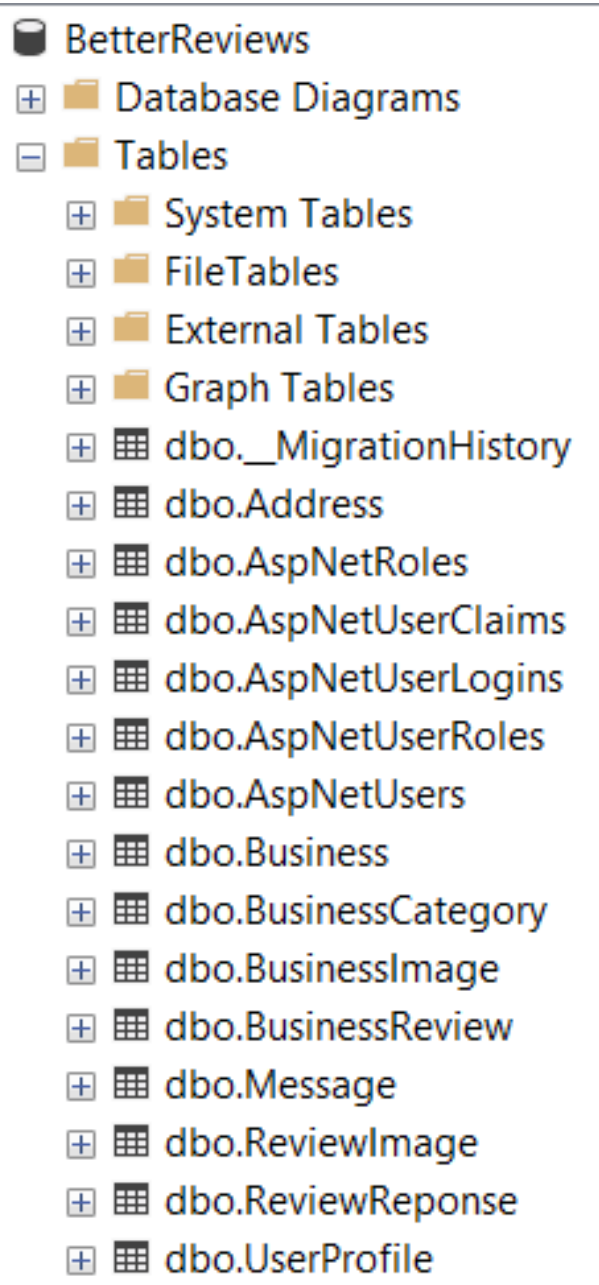


Figure 5.9: Database viewed through MMS



Database Layer (SQL Server; Entity Framework)

As can be seen in the illustration below, taken from the Better Reviews Database using Microsoft Management Studio (MMS) the tables that are revealed in the Visual Studio version are the same as if the Database were directly accessed through a separate interface tool.

Although the attached document (see appendix Structured Query Language (SQL) Master Code File: betterreviews_sql_script.sql) can be used to recreate the database being illustrated with this application, the database was developed over time. In addition for users wishing to recreate the database using MS Visual Studio, the database can be created by accessing the backpack file (included in Files) view and then running the identified master SQL file.

Figure 5.10: The Better Reviews ERD is shown in its entirety below, and subsequent pages will break out individual sections, and show Table details with greater detail:

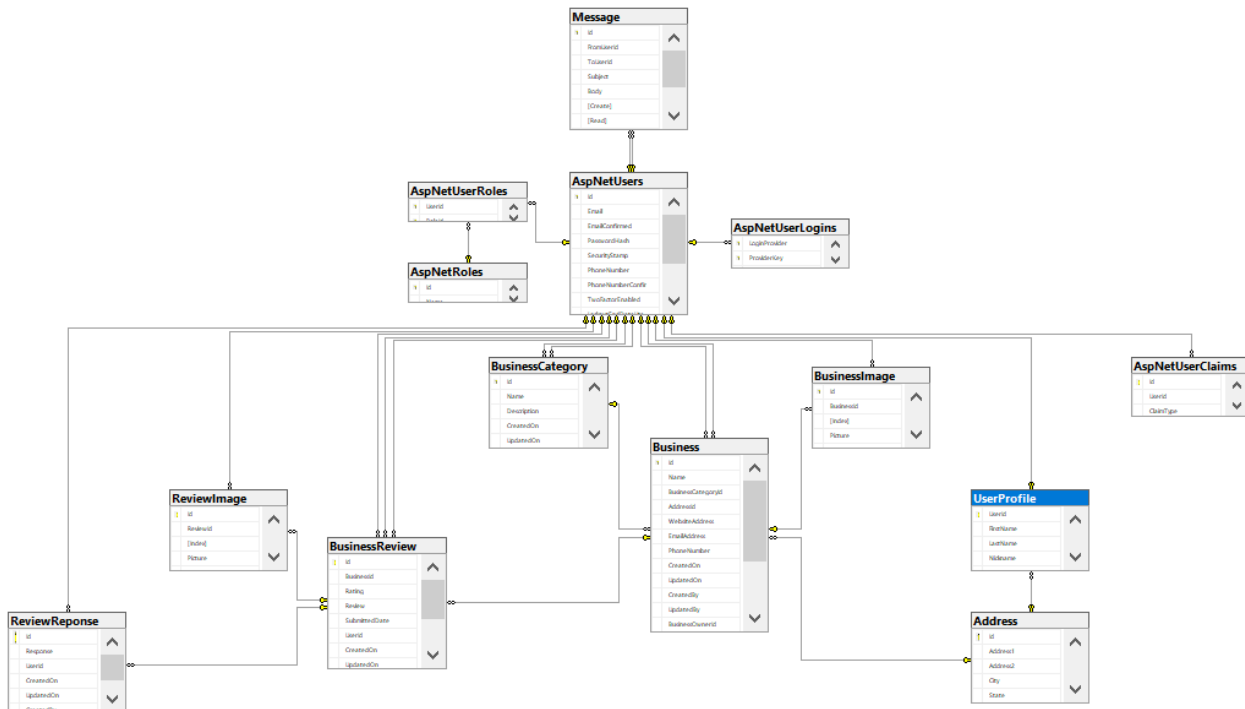


Figure 5.11: Database Entities and Column Data:

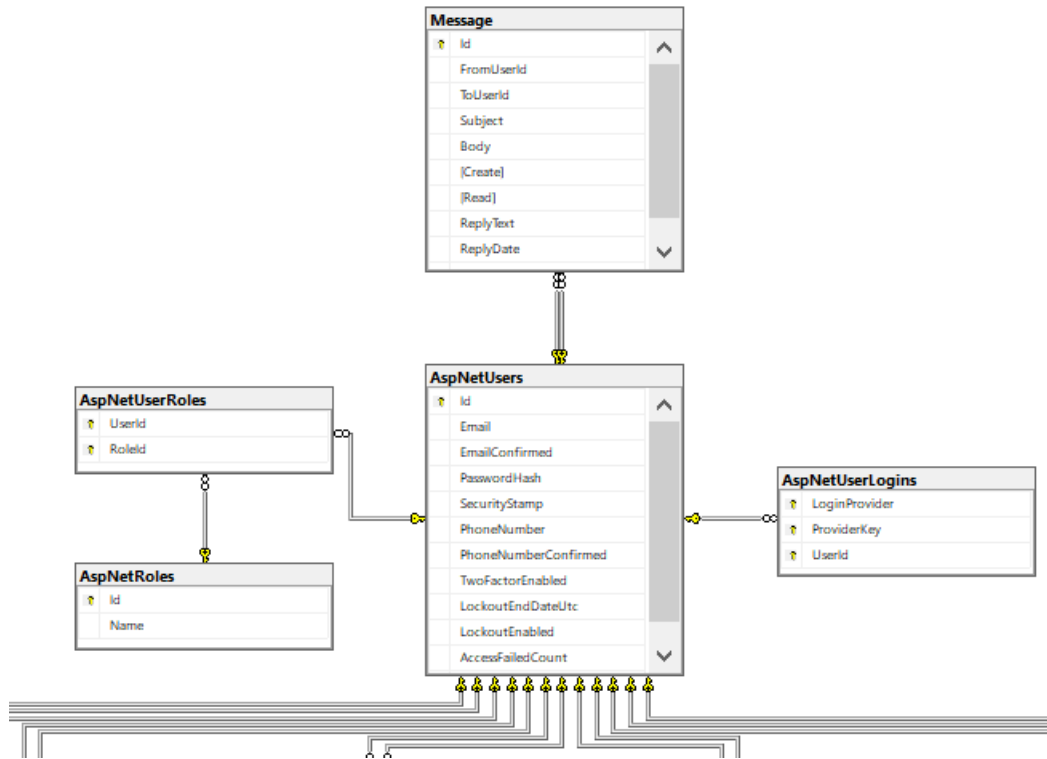


Figure 5.12: Message Table:
Message:

```

    dbo.Message
    Columns
    Id (PK, int, not null)
    FromUserId (FK, nvarchar(128), not null)
    ToUserId (FK, nvarchar(128), not null)
    Subject (nvarchar(250), null)
    Body (nvarchar(max), null)
    Create (timestamp, null)
    Read (bit, not null)
    ReplyText (nvarchar(max), null)
    ReplyDate (datetime, null)
    Archive (bit, null)
    
```

Figure 5.13: AspNetTable
AspNetUsers

```

    dbo.AspNetUsers
    Columns
    Id (PK, nvarchar(128), not null)
    Email (nvarchar(256), null)
    EmailConfirmed (bit, not null)
    PasswordHash (nvarchar(max), null)
    SecurityStamp (nvarchar(max), null)
    PhoneNumber (nvarchar(max), null)
    PhoneNumberConfirmed (bit, not null)
    TwoFactorEnabled (bit, not null)
    LockoutEndDateUtc (datetime, null)
    LockoutEnabled (bit, not null)
    AccessFailedCount (int, not null)
    UserName (nvarchar(256), not null)
    
```

Figure 5.14: AspNetUserRoles Table:
AspNetUserRoles

```

    dbo.AspNetUserRoles
    Columns
    LoginProvider (PK, nvarchar(128), not null)
    ProviderKey (PK, nvarchar(128), not null)
    UserId (PK, FK, nvarchar(128), not null)
    
```

Figure 5.15: AspNetUserLogins Table
AspNetUserLogins

```

    dbo.AspNetUserLogins
    Columns
    LoginProvider (PK, nvarchar(128), not null)
    ProviderKey (PK, nvarchar(128), not null)
    UserId (PK, FK, nvarchar(128), not null)
    
```

Figure 5.16: AspNetRoles Table

```

    dbo.AspNetRoles
    Columns
    Id (PK, nvarchar(128), not null)
    Name (nvarchar(256), not null)
    
```

Figure 5.17: AspNetUserClaims Table

```

    dbo.AspNetUserClaims
    Columns
    Id (PK, int, not null)
    UserId (FK, nvarchar(128), not null)
    ClaimType (nvarchar(max), null)
    ClaimValue (nvarchar(max), null)
    
```

Figure 5.18: Additional Database Entities and Columns (Shown in Bottom Left of Original ERD)

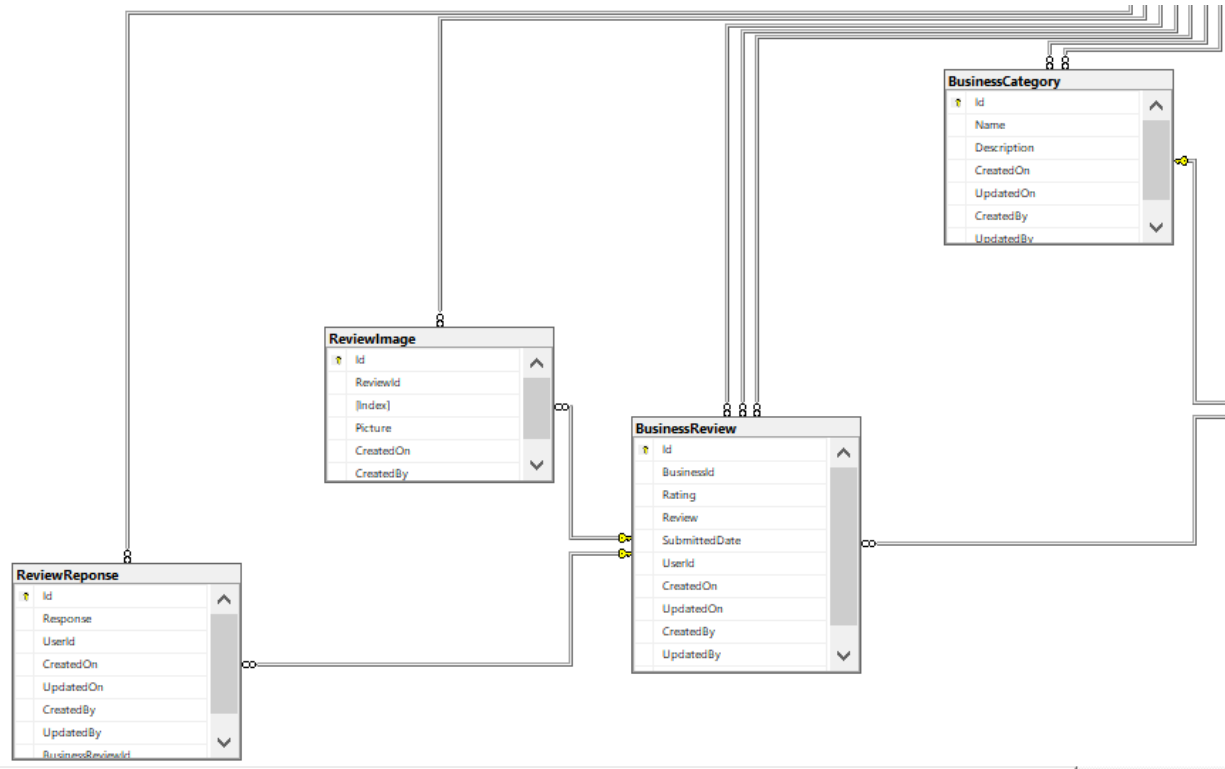


Figure 5.19: Review Response

```

dbo.ReviewResponse
├── Columns
│   ├── Id (PK, int, not null)
│   ├── Response (nvarchar(max), null)
│   ├── UserId (FK, nvarchar(128), null)
│   ├── CreatedOn (datetime, null)
│   ├── UpdatedOn (datetime, null)
│   ├── CreatedBy (nvarchar(128), null)
│   ├── UpdatedBy (nvarchar(128), null)
│   └── BusinessReviewId (FK, int, not null)

```

Figure 5.20: Business Review

```

dbo.BusinessReview
├── Columns
│   ├── Id (PK, int, not null)
│   ├── BusinessId (FK, int, not null)
│   ├── Rating (int, not null)
│   ├── Review (nvarchar(max), not null)
│   ├── SubmittedDate (datetime, not null)
│   ├── UserId (FK, nvarchar(128), not null)
│   ├── CreatedOn (datetime, null)
│   ├── UpdatedOn (datetime, null)
│   ├── CreatedBy (FK, nvarchar(128), null)
│   ├── UpdatedBy (FK, nvarchar(128), null)
│   └── ReviewResponseId (int, null)

```

Figure 5.21: Review Image

```

dbo.ReviewImage
├── Columns
│   ├── Id (PK, int, not null)
│   ├── ReviewId (FK, int, not null)
│   ├── Index (int, null)
│   ├── Picture (image, not null)
│   ├── CreatedOn (datetime, null)
│   └── CreatedBy (FK, nvarchar(128), null)

```

Figure 5.22: Business Category

```

dbo.BusinessCategory
├── Columns
│   ├── Id (PK, int, not null)
│   ├── Name (nvarchar(80), not null)
│   ├── Description (nvarchar(128), null)
│   ├── CreatedOn (datetime, null)
│   ├── UpdatedOn (datetime, null)
│   ├── CreatedBy (FK, nvarchar(128), null)
│   └── UpdatedBy (FK, nvarchar(128), null)

```

Figure 5.23: Additional Database Entities and Columns (Shown in Bottom Right of Original ERD)

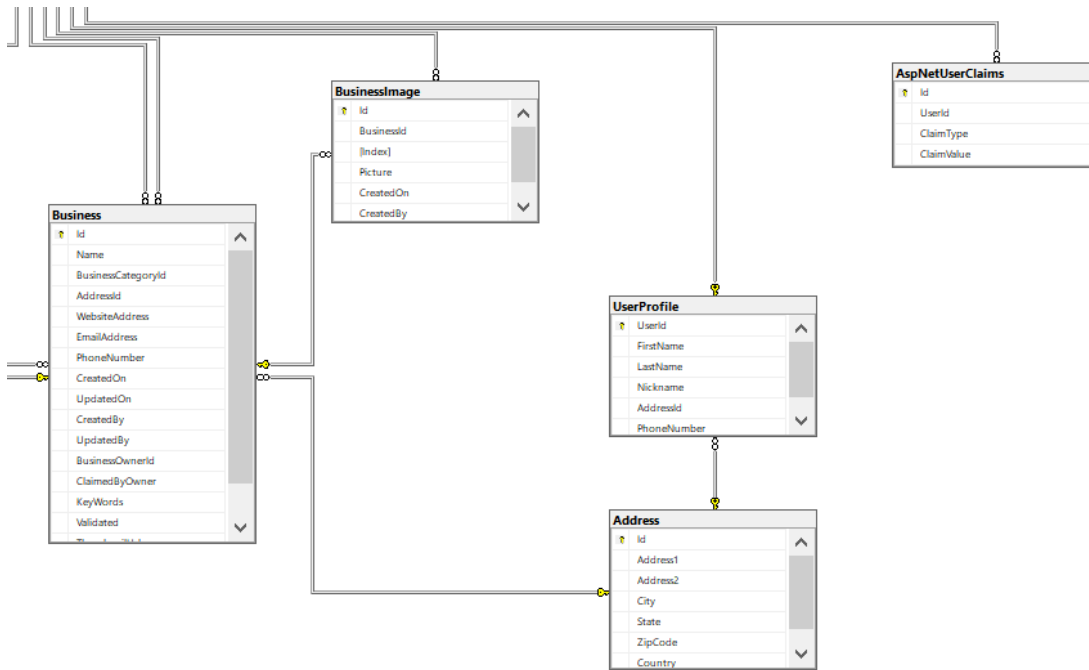


Figure 5.24: Business Image

```

dbo.BusinessImage
├── Columns
│   ├── Id (PK, int, not null)
│   ├── BusinessId (FK, int, not null)
│   ├── Index (int, null)
│   ├── Picture (image, not null)
│   ├── CreatedOn (datetime, null)
│   └── CreatedBy (FK, nvarchar(128), null)
    
```

Figure 5.25: User Profile

```

dbo.UserProfile
├── Columns
│   ├── UserId (PK, FK, nvarchar(128), not null)
│   ├── FirstName (nvarchar(50), null)
│   ├── LastName (nvarchar(50), null)
│   ├── Nickname (nvarchar(50), null)
│   ├── AddressId (FK, int, null)
│   └── PhoneNumber (nvarchar(50), null)
    
```

Figure 5.26: Business

```

dbo.Business
├── Columns
│   ├── Id (PK, int, not null)
│   ├── Name (nvarchar(128), not null)
│   ├── BusinessCategoryId (FK, int, null)
│   ├── AddressId (FK, int, null)
│   ├── WebsiteAddress (nvarchar(max), null)
│   ├── EmailAddress (nvarchar(128), null)
│   ├── PhoneNumber (nvarchar(50), null)
│   ├── CreatedOn (datetime, null)
│   ├── UpdatedOn (datetime, null)
│   ├── CreatedBy (FK, nvarchar(128), null)
│   ├── UpdatedBy (FK, nvarchar(128), null)
│   ├── BusinessOwnerId (nvarchar(128), null)
│   ├── ClaimedByOwner (bit, null)
│   ├── KeyWords (nvarchar(max), null)
│   ├── Validated (bit, null)
│   ├── ThumbnailUrl (nvarchar(max), null)
│   └── Blurb (nvarchar(max), null)
    
```

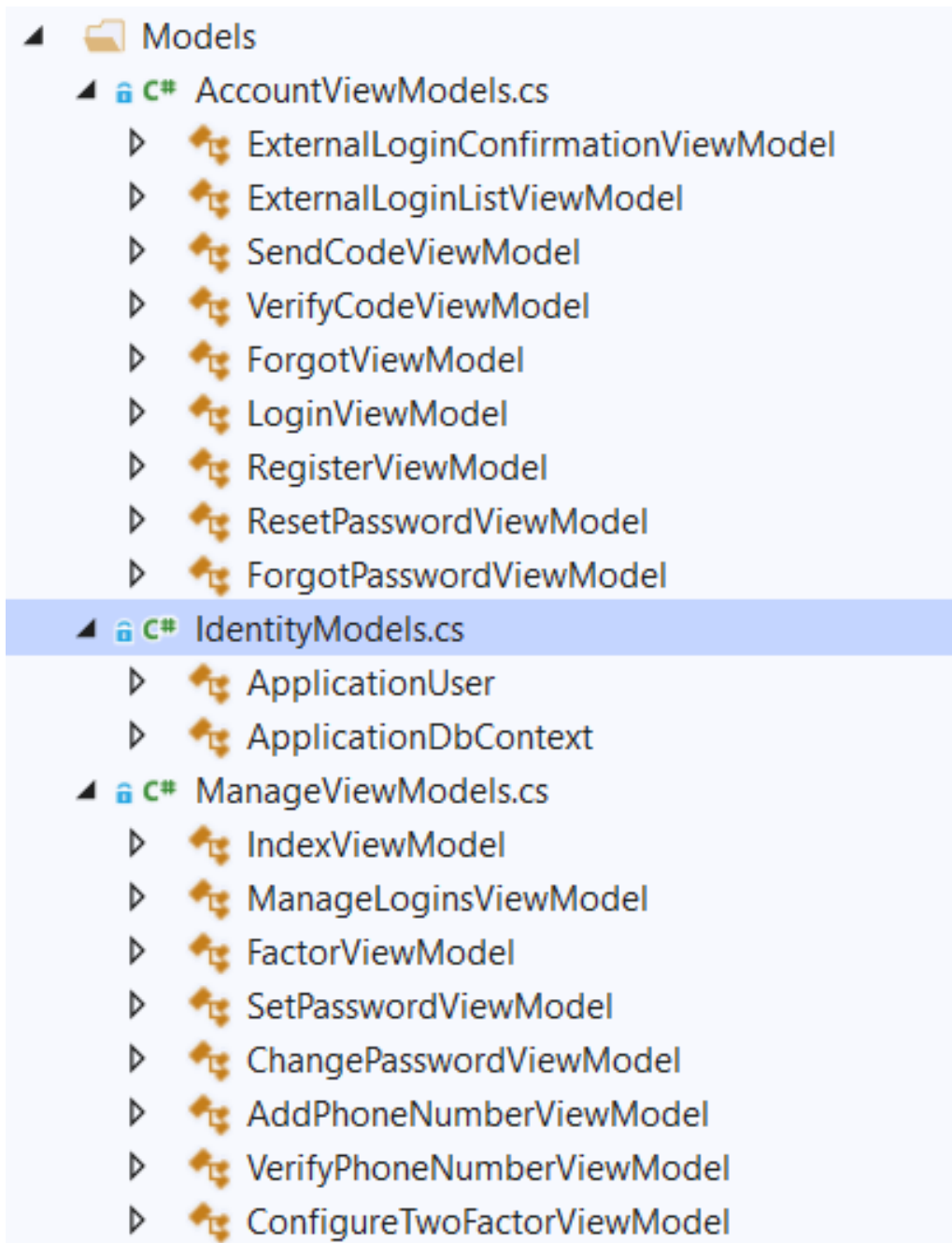
Figure 5.27: Address

```

dbo.Address
├── Columns
│   ├── Id (PK, int, not null)
│   ├── Address1 (nvarchar(128), not null)
│   ├── Address2 (nvarchar(50), null)
│   ├── City (nvarchar(50), not null)
│   ├── State (nvarchar(50), not null)
│   ├── ZipCode (nvarchar(50), not null)
│   └── Country (nvarchar(50), not null)
    
```

The Models included in the Visual Studio Application are aligned to the information that is retrieved and accessed in the database, and while the SQL structures are identified above, the Models as seen below act as the interface between the Database and application.

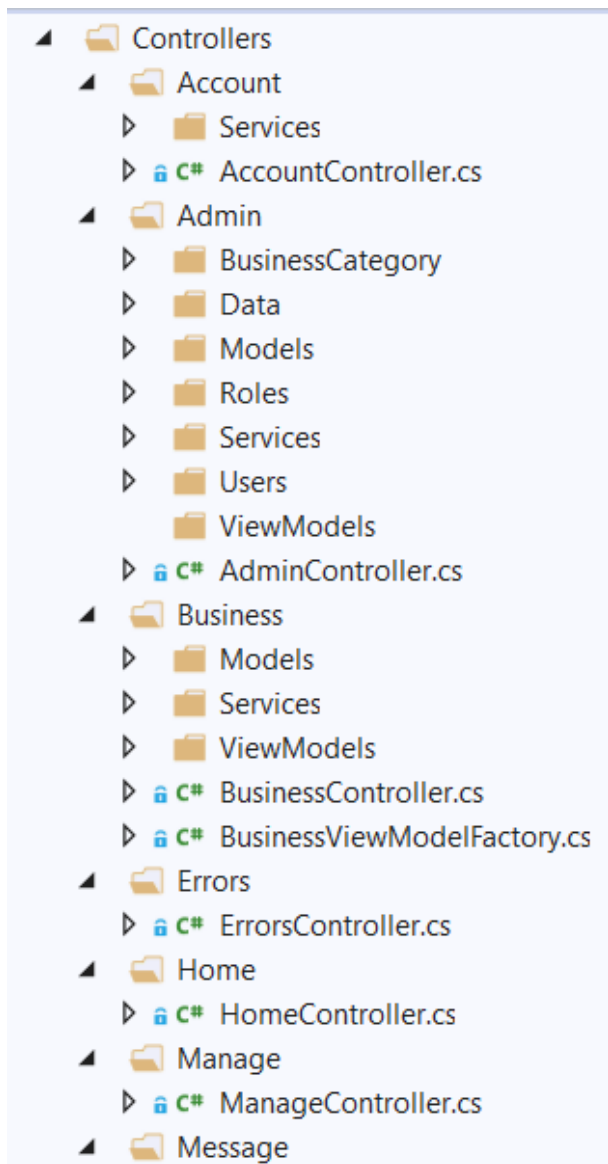
Figure 5.28: Models (Main List)



The Logical abstraction for the Better Reviews application is handled in the Controllers layer of the MVC paradigm. This layer represents the Web Application (including the internal Admin, Business and User applications).

A discussion related to the selection and organization of the controllers (as seen below) is provided in section six (Design Unit Impacts). For this section the controllers that are included in the application will be simply be graphically identified as images (see below).

Figure 5.29: Application Solution Explorer: Main View of Controllers



Additional Controller Details:

Figure 5.30: Notification Search Controllers

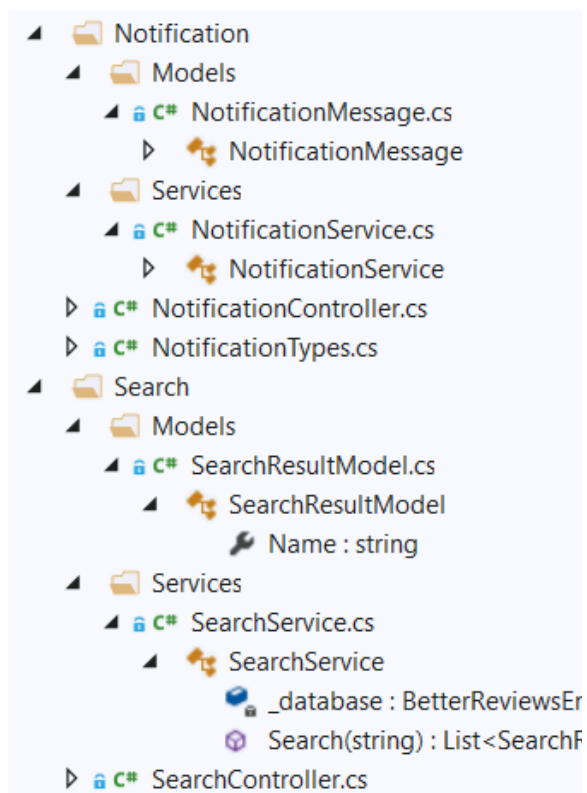


Figure 5.31: Business Import Controllers

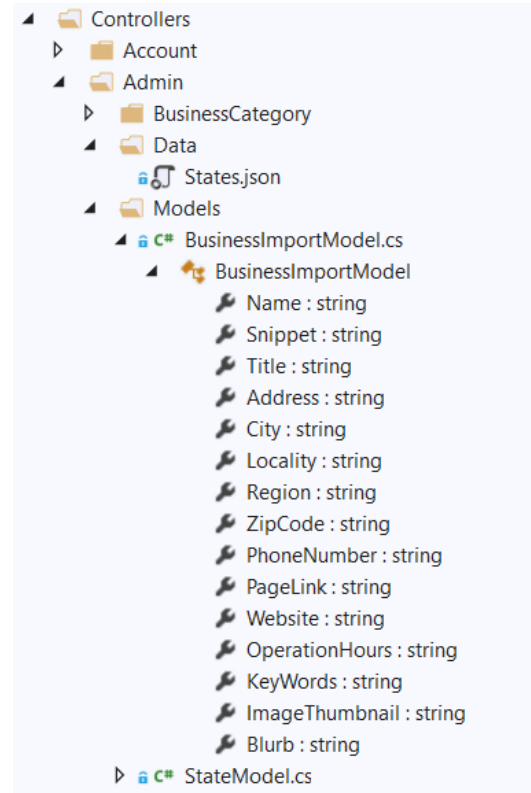


Figure 5.32: Role Model Controllers

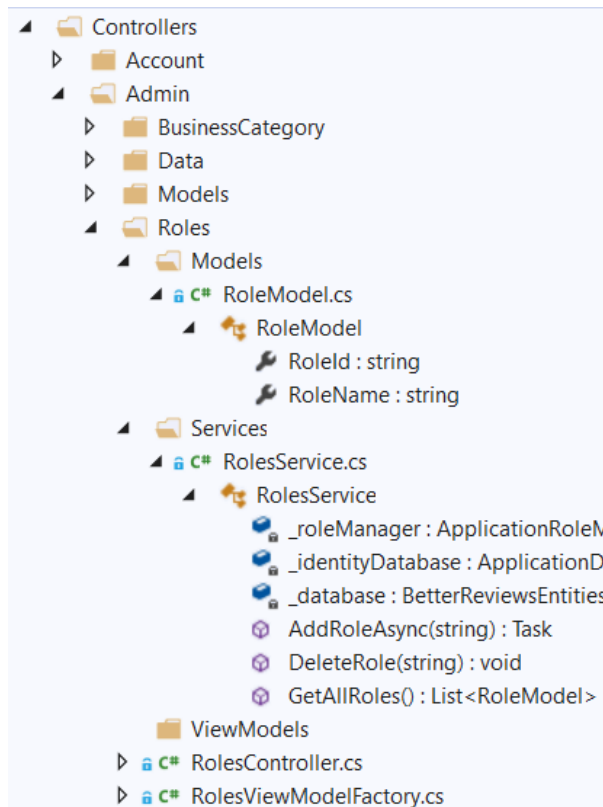
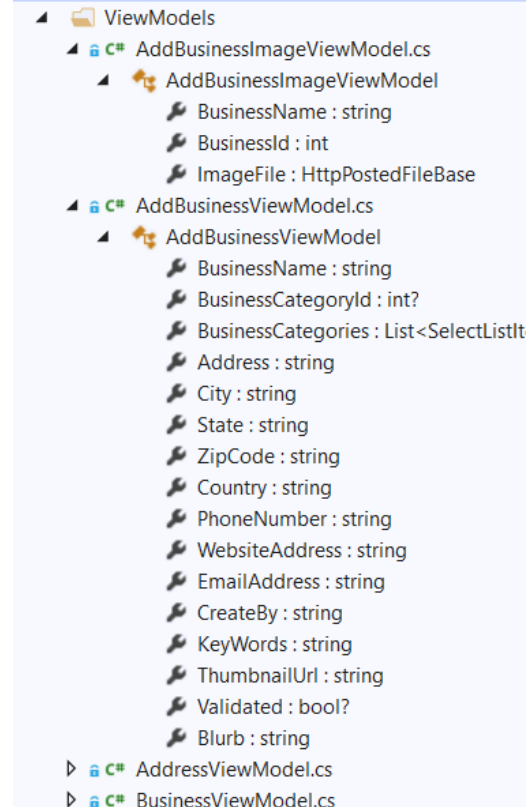


Figure 5.33: View Models (Figure 3.1)



Additional Controller Details:

Figure 5.34: Business Controllers (initial)

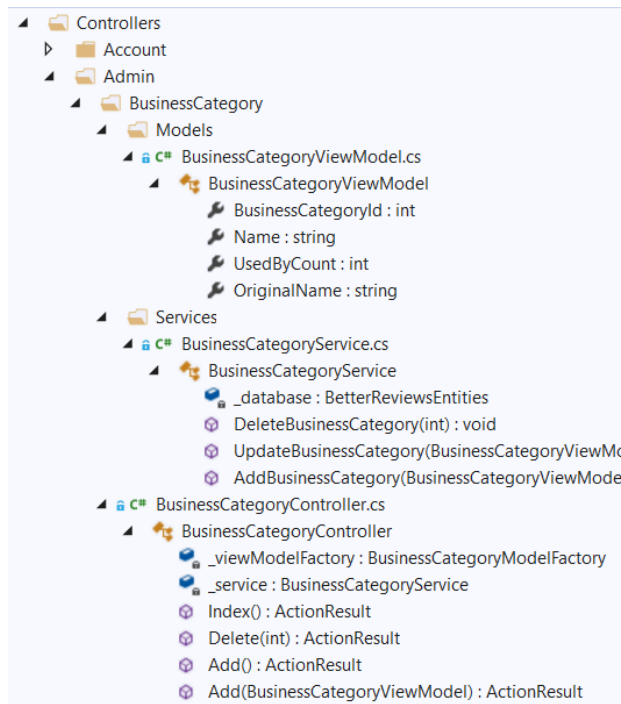


Figure 5.35: Business Controllers (advanced)

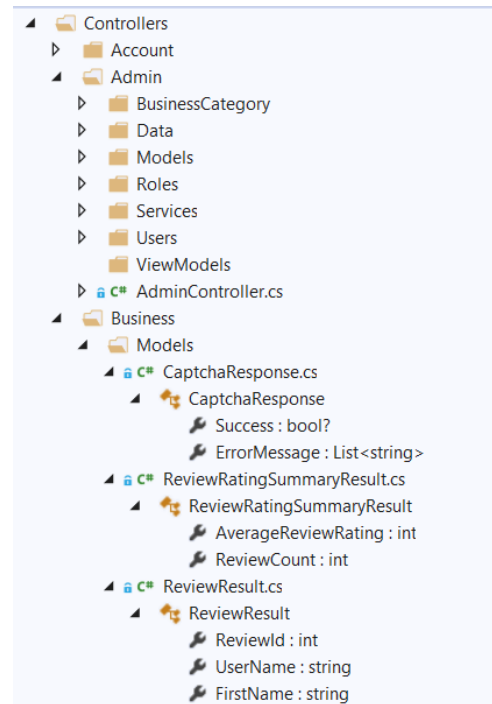


Figure 5.36: Business View Model Controllers

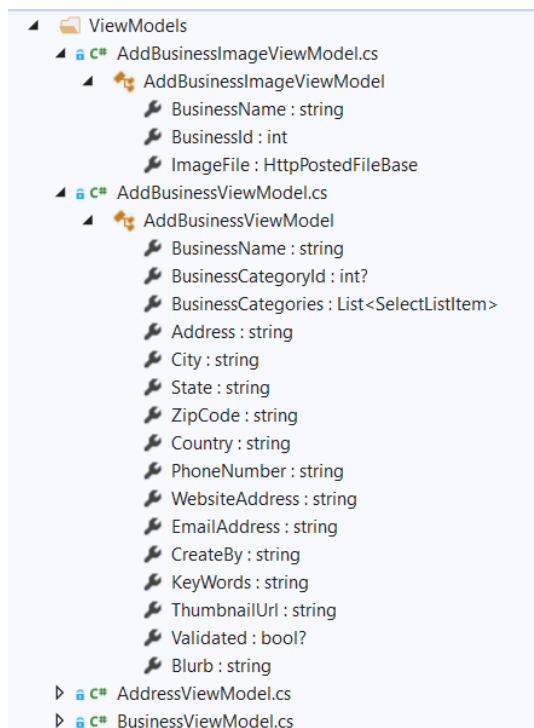
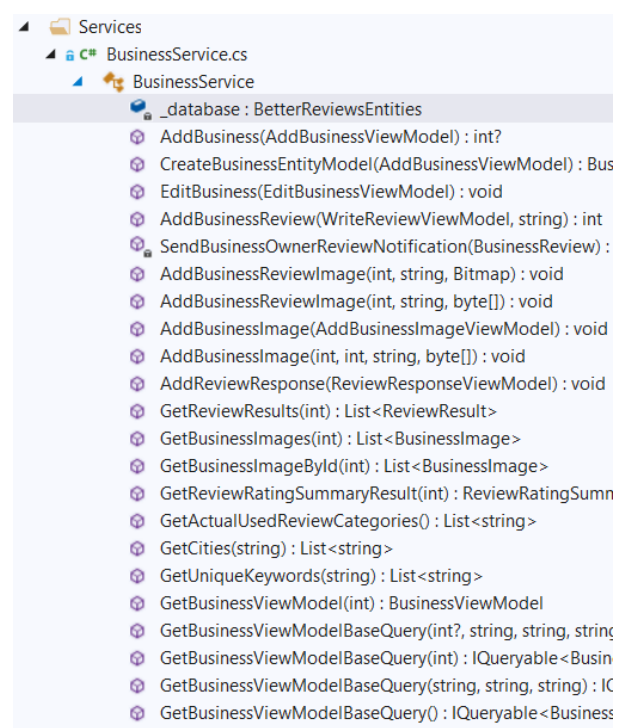


Figure 5.37: Business Services Controllers



Documentation

The last component identified in the original system-wide graphical image on the first page of section 5 is related to internal interface for documentation. Our documentation (including this document) is stored in a shared repository provided through GSU student login to OneDrive. In addition our code repository is hosted and shared on a private GitHub interface, which allows us to interact and share code (and push updates) as a team.

First, we provide a representation of the files which are shared on One Drive (external to the application), and then we will identify (graphically and as full text) the entire set of software dependencies which are embedded into the Better Reviews App.

File System Documents (External Interface via MS OneDrive)

Figure 5.38: Group File Storage

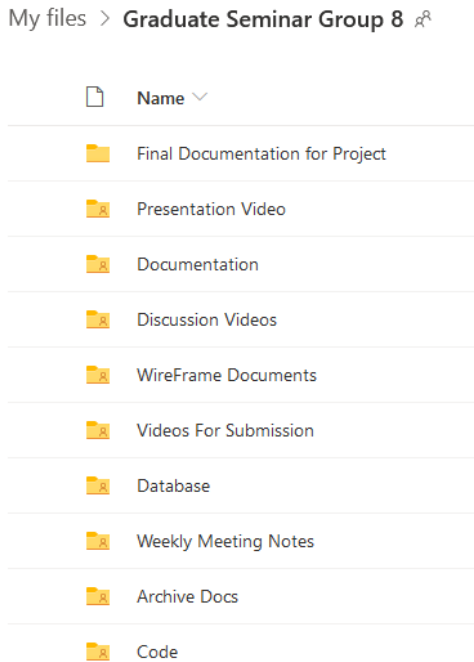


Figure 5.39: Additional Files/Folder on OneDrive Shared Storage

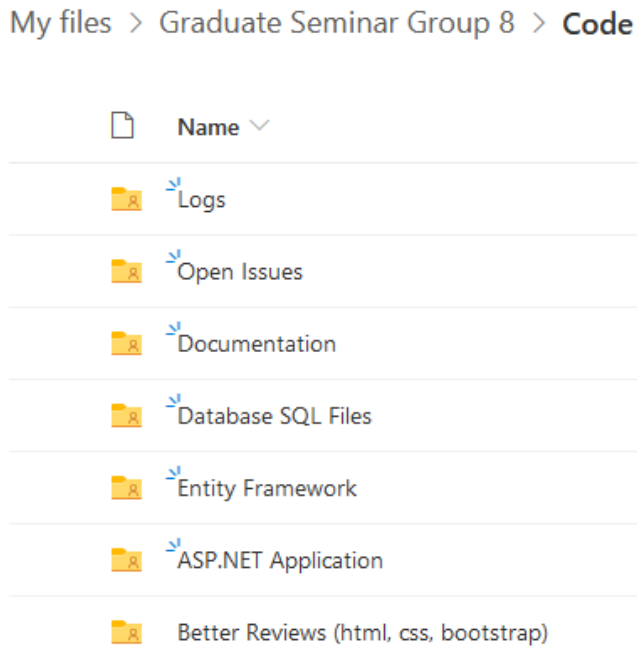


Figure 5.40: Documentation Files

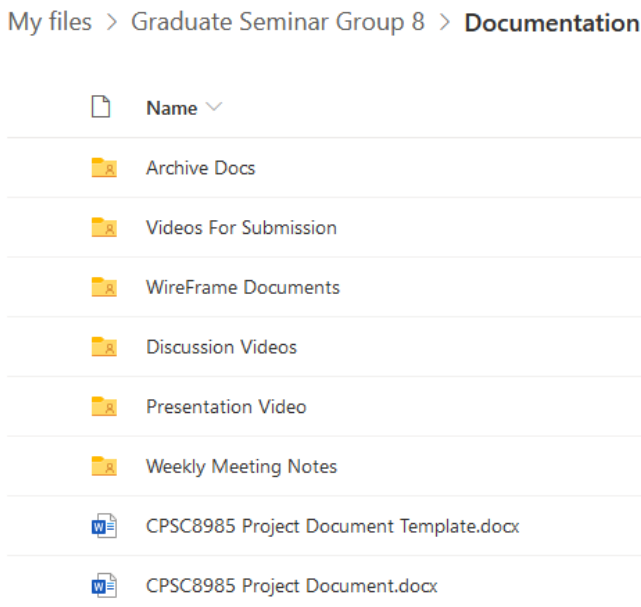














































































































Figure 5.41: ASP.NET and Entity Framework Library Dependencies (Internal Interface)














	Antlr by Sam Harwell, Terence Parr ANother Tool for Language Recognition, is a language tool that provides a framework for constructing recognizers, interpreters, compi...	v3.5.0.2
	bootstrap by Twitter, Inc. Bootstrap framework in CSS. Includes fonts and JavaScript	v3.4.1 v4.5.3
	EntityFramework by Microsoft Entity Framework is Microsoft's recommended data access technology for new applications.	v6.2.0 v6.4.4
	jQuery by jQuery Foundation, Inc. jQuery is a new kind of JavaScript Library. jQuery is a fast and concise JavaScript Library that simplifies HTML...	v3.4.1 v3.5.1
	jQuery.Validation by Jörn Zaefferer This jQuery plugin makes simple clientside form validation trivial, while offering lots of option for customization. That makes a good choice if...	v1.17.0 v1.19.2
	Microsoft.AspNet.Identity.Core by Microsoft Core interfaces for ASP.NET Identity.	v2.2.3
	Microsoft.AspNet.Identity.EntityFramework by Microsoft ASP.NET Identity providers that use Entity Framework.	v2.2.3
	Microsoft.AspNet.Identity.Owin by Microsoft Owin implementation for ASP.NET Identity.	v2.2.3
	Microsoft.AspNet.Mvc by Microsoft This package contains the runtime assemblies for ASP.NET MVC.	v5.2.7
	Microsoft.AspNet.Razor by Microsoft This package contains the runtime assemblies for ASP.NET Web Pages.	v3.2.7
	Microsoft.AspNet.Web.Optimization by Microsoft ASP.NET Optimization introduces a way to bundle and optimize CSS and JavaScript files.	v1.1.3
	Microsoft.AspNet.WebApi by Microsoft This package contains everything you need to host ASP.NET Web API on IIS.	v5.2.7
	Microsoft.AspNet.WebApi.Client by Microsoft This package adds support for formatting and content negotiation to System.Net.Http.	v5.2.7



















	Microsoft.AspNet.WebApi.Core by Microsoft	v5.2.7
	This package contains the core runtime assemblies for ASP.NET Web API.	
	Microsoft.AspNet.WebApi.WebHost by Microsoft	v5.2.7
	This package contains everything you need to host ASP.NET Web API on IIS.	
	Microsoft.AspNet.WebPages by Microsoft	v3.2.7
	This package contains core runtime assemblies shared between ASP.NET MVC and ASP.NET Web Pages.	
	Microsoft.AspNetCore.Http by Microsoft	v2.2.2
	ASP.NET Core default HTTP feature implementations.	
	Microsoft.AspNetCore.Http.Abstractions by Microsoft	v2.2.0
	ASP.NET Core HTTP object model for HTTP requests and responses and also common extension methods for registering middleware in an IAppl...	
	Microsoft.AspNetCore.Http.Features by Microsoft	v2.2.0
	ASP.NET Core HTTP feature interface definitions.	
	Microsoft.AspNetCore.Mvc.Abstractions by Microsoft	v2.2.0
	ASP.NET Core MVC abstractions and interfaces for action invocation and dispatching, authorization, action filters, formatters, model binding, rout...	
	Microsoft.AspNetCore.Routing.Abstractions by Microsoft	v2.2.0
	ASP.NET Core abstractions for routing requests to application logic and for generating links.	
	Microsoft.AspNetCore.WebUtilities by Microsoft	v2.2.0
	ASP.NET Core utilities, such as for working with forms, multipart messages, and query strings.	
	Microsoft.Bcl.AsyncInterfaces by Microsoft	v1.1.1
	Provides the IAsyncEnumerable<T> and IAsyncDisposable interfaces and helper types for .NET Standard 2.0. This package is not required starting...	
	Microsoft.Bcl.HashCode by Microsoft	v1.1.0
	Provides the HashCode type for .NET Standard 2.0. This package is not required starting with .NET Standard 2.1 and .NET Core 3.0.	
	Microsoft.CodeDom.Providers.DotNetCompilerPlatform by	v2.0.1
	Replacement CodeDOM providers that use the new .NET Compiler Platform ("Roslyn") compiler as a service APIs.	
	Microsoft.CSharp by Microsoft	v4.7.0
	Provides support for compilation and code generation, including dynamic, using the C# language.	
	Microsoft.Data.SqlClient by Microsoft	v1.1.3
	Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, incl...	

	Microsoft.Bcl.HashCode by Microsoft	v1.1.0
	Provides the HashCode type for .NET Standard 2.0. This package is not required starting with .NET Standard 2.1 and .NET Core 3.0.	
	Microsoft.CodeDom.Providers.DotNetCompilerPlatform by	v2.0.1
	Replacement CodeDOM providers that use the new .NET Compiler Platform ("Roslyn") compiler as a service APIs.	v3.6.0
	Microsoft.CSharp by Microsoft	v4.7.0
	Provides support for compilation and code generation, including dynamic, using the C# language.	
	Microsoft.Data.SqlClient by Microsoft	v1.1.3
	Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, incl...	v2.0.1
	Microsoft.Data.SqlClient.SNI by Microsoft	v1.1.0
	Internal implementation package not meant for direct consumption. Please do not reference directly.	v2.1.1
	Microsoft.EntityFrameworkCore by Microsoft	v3.1.9
	Entity Framework Core is a lightweight and extensible version of the popular Entity Framework data access technology.	v5.0.0
	Microsoft.EntityFrameworkCore.Abstractions by Microsoft	v3.1.9
	Provides abstractions and attributes that are used to configure Entity Framework Core	v5.0.0
	Microsoft.EntityFrameworkCore.Analyzers by Microsoft	v3.1.9
	CSharp Analyzers for Entity Framework Core.	v5.0.0
	Microsoft.EntityFrameworkCore.Design by Microsoft	v3.1.9
	Shared design-time components for Entity Framework Core tools.	v5.0.0
	Microsoft.EntityFrameworkCore.Relational by Microsoft	v3.1.9
	Shared Entity Framework Core components for relational database providers.	v5.0.0
	Microsoft.EntityFrameworkCore.SqlServer by Microsoft	v3.1.9
	Microsoft SQL Server database provider for Entity Framework Core.	v5.0.0
	Microsoft.EntityFrameworkCore.Tools by Microsoft	v3.1.9
	Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	v5.0.0
	Microsoft.Extensions.Caching.Abstractions by Microsoft	v3.1.9
	Caching abstractions for in-memory cache and distributed cache. Commonly used types:	v5.0.0

	Microsoft.Extensions.Caching.Memory by Microsoft	v3.1.9
	In-memory cache implementation of Microsoft.Extensions.Caching.Memory.IMemoryCache.	v5.0.0
	Microsoft.Extensions.Configuration by Microsoft	v3.1.9
	Implementation of key-value pair based configuration for Microsoft.Extensions.Configuration. Includes the memory configuration...	v5.0.0
	Microsoft.Extensions.Configuration.Abstractions by Microsoft	v3.1.9
	Abstractions of key-value pair based configuration. Commonly used types:	v5.0.0
	Microsoft.Extensions.Configuration.Binder by Microsoft	v3.1.9
	Functionality to bind an object to data in configuration providers for Microsoft.Extensions.Configuration.	v5.0.0
	Microsoft.Extensions.DependencyInjection by Microsoft	v3.1.9
	Default implementation of dependency injection for Microsoft.Extensions.DependencyInjection.	v5.0.0
	Microsoft.Extensions.DependencyInjection.Abstractions by Microsoft	v3.1.9
	Abstractions for dependency injection. Commonly used types:	v5.0.0
	Microsoft.Extensions.Logging by Microsoft	v3.1.9
	Logging infrastructure default implementation for Microsoft.Extensions.Logging.	v5.0.0
	Microsoft.Extensions.Logging.Abstractions by Microsoft	v3.1.9
	Logging abstractions for Microsoft.Extensions.Logging. Commonly used types:	v5.0.0
	Microsoft.Extensions.ObjectPool by Microsoft	v2.2.0
	A simple object pool implementation.	v5.0.0
	Microsoft.Extensions.Options by Microsoft	v3.1.9
	Provides a strongly typed way of specifying and accessing settings using dependency injection.	v5.0.0
	Microsoft.Extensions.Primitives by Microsoft	v3.1.9
	Primitives shared by framework extensions. Commonly used types include: Microsoft.Extensions.Primitives.IChangeToken	v5.0.0
	Microsoft.Identity.Client by Microsoft	v3.0.8
	This package contains the binaries of the Microsoft Authentication Library for .NET (MSAL.NET).	v4.22.0
	Microsoft.IdentityModel.JsonWebTokens by Microsoft	v5.5.0
	Includes types that provide support for creating, serializing and validating JSON Web Tokens.	v6.8.0
	Microsoft.IdentityModel.Logging by Microsoft	v5.5.0
	Includes Event Source based logging support.	v6.8.0

	Microsoft.IdentityModel.Protocols by Microsoft	v5.5.0
	Provides base protocol support for OpenIdConnect and WsFederation.	v6.8.0
	Microsoft.IdentityModel.Protocols.OpenIdConnect by Microsoft	v5.5.0
	Includes types that provide support for OpenIdConnect protocol.	v6.8.0
	Microsoft.IdentityModel.Tokens by Microsoft	v5.5.0
	Includes types that provide support for SecurityTokens, Cryptographic operations: Signing, Verifying Signatures, Encryption.	v6.8.0
	Microsoft.jQuery.Unobtrusive.Validation by Microsoft	v3.2.11
	The jQuery Unobtrusive Validation library complements jQuery Validation by adding support for specifying validation options as HTML...	
	Microsoft.Net.Http.Headers by Microsoft	v2.2.0
	HTTP header parser implementations.	v2.2.8
	Microsoft.Owin by Microsoft	v4.0.1
	Provides a set of helper types and abstractions for simplifying the creation of OWIN components.	v4.1.1
	Microsoft.Owin.Host.SystemWeb by Microsoft	v4.0.1
	OWIN server that enables OWIN-based applications to run on IIS using the ASP.NET request pipeline.	v4.1.1
	Microsoft.Owin.Security by Microsoft	v4.0.1
	Common types which are shared by the various authentication middleware components.	v4.1.1
	Microsoft.Owin.Security.Cookies by Microsoft	v4.0.1
	Middleware that enables an application to use cookie based authentication, similar to ASP.NET's forms authentication.	v4.1.1
	Microsoft.Owin.Security.Facebook by Microsoft	v4.0.1
	Middleware that enables an application to support Facebook's OAuth 2.0 authentication workflow.	v4.1.1
	Microsoft.Owin.Security.Google by Microsoft	v4.0.1
	Contains middlewares to support Google's OAuth 2.0 authentication workflow.	v4.1.1
	Microsoft.Owin.Security.MicrosoftAccount by Microsoft	v4.0.1
	Middleware that enables an application to support the Microsoft Account authentication workflow.	v4.1.1
	Microsoft.Owin.Security.OAuth by Microsoft	v4.0.1
	Middleware that enables an application to support any standard OAuth 2.0 authentication workflow.	v4.1.1

	Microsoft.Owin.Security.Twitter by Microsoft	v4.0.1 v4.1.1
	Middleware that enables an application to support Twitter's OAuth 2.0 authentication workflow.	
	Microsoft.Web.Infrastructure by Microsoft	v1.0.0
	This package contains the Microsoft.Web.Infrastructure assembly that lets you dynamically register HTTP modules at run time.	
	Modernizr by Faruk Ateş, Paul Irish, Alex Sexton	v2.8.3
	Modernizr is a small and simple JavaScript library that helps you take advantage of emerging web technologies (CSS3, HTML 5) while still mai...	
	Newtonsoft.Json by James Newton-King	v12.0.2 v12.0.3
	Json.NET is a popular high-performance JSON framework for .NET	
	Owin by OWIN startup components contributors	v1.0.0
	OWIN IApplicationBuilder startup interface	
	PresentationFramework by dongz	v4.6.0
	PresentationFramework	
	System.Buffers by Microsoft	v4.5.1
	Provides resource pooling of any type for performance-critical applications that allocate and deallocate objects frequently.	
	System.Collections.Immutable by Microsoft	v1.7.1 v5.0.0
	This package provides collections that are thread safe and guaranteed to never change their contents, also known as immutable collections. Like s...	
	System.ComponentModel.Annotations by Microsoft	v4.7.0 v5.0.0
	Provides attributes that are used to define metadata for objects used as data sources.	
	System.Data.Common by Microsoft	v4.3.0
	Provides the base abstract classes, including System.Data.DbConnection and System.Data.DbCommand, for all data providers.	
	System.Diagnostics.DiagnosticSource by Microsoft	v4.7.1 v5.0.0
	Provides Classes that allow you to decouple code logging rich (unserializable) diagnostics/telemetry (e.g. framework) from code that c...	
	System.IdentityModel.Tokens.Jwt by Microsoft	v5.5.0 v6.8.0
	Includes types that provide support for creating, serializing and validating JSON Web Tokens.	
	System.Memory by Microsoft	v4.5.4
	Provides types for efficient representation and pooling of managed, stack, and native memory segments and sequences of such segments, al...	

	System.Numerics.Vectors by Microsoft	v4.5.0
	Provides hardware-accelerated numeric types, suitable for high-performance processing and graphics applications.	
	System.Runtime.CompilerServices.Unsafe by Microsoft	v4.7.1
	Provides the System.Runtime.CompilerServices.Unsafe class, which provides generic, low-level functionality for manipulating pointers.	v5.0.0
	System.Text.Encodings.Web by Microsoft	v4.5.0
	Provides types for encoding and escaping strings for use in JavaScript, HyperText Markup Language (HTML), and uniform resource locators (UR...	v5.0.0
	System.Threading.Tasks.Extensions by Microsoft	v4.5.4
	Provides additional types that simplify the work of writing concurrent and asynchronous code.	
	WebGrease by webgrease@microsoft.com	v1.6.0
	Web Grease is a suite of tools for optimizing javascript, css files and images.	
	WindowsBase by Microsoft Corp.	v4.6.1055
	WindowsBase.dll NuGet Package	
	X.PagedList by Copyright Troy Goode, Ernado © 2018	v7.9.0
	Library for easily paging through any IEnumerable/IQueryable in .NET and .NET Core	v8.0.7
	X.PagedList.Mvc by Copyright Troy Goode, Ernado © 2019	v7.9.1
	Library for easily paging through any IEnumerable/IQueryable in ASP.NET MVC	v8.0.7
	X.PagedList.Web.Common by Copyright Troy Goode, Ernado © 20	v7.9.1
	Package Description	v8.0.7

Finally, for full-text referencing and easy keyword lookup, the list of NuGet Package Dependencies represented above as images are listed below alphabetically:

Antlr
Bootstrap
EntityFramework
JQuery
jQuery.Validation
Microsoft.AspNetCore.Identity.Core
Microsoft.AspNetCore.Identity.EntityFramework
Microsoft.AspNetCore.Identity.Owin
Microsoft.AspNetCore.Mvc
Microsoft.AspNetCore.Razor
Microsoft.AspNetCore.Web.Optimization
Microsoft.AspNetCore.WebApi
Microsoft.AspNetCore.WebApi.Client
Microsoft.AspNetCore.WebApi.Core
Microsoft.AspNetCore.WebApi.WebHost
Microsoft.AspNetCore.WebPages
Microsoft.AspNetCore.Http
Microsoft.AspNetCore.Http.Abstractions
Microsoft.AspNetCore.Http.Features
Microsoft.AspNetCore.Mvc.Abstractions
Microsoft.AspNetCore.Routing.Abstractions
Microsoft.AspNetCore.WebUtilities
Microsoft.Bcl.AsyncInterfaces
Microsoft.Bcl.HashCode
Microsoft.CodeDom.Providers.DotNetCompilerPlatform
Microsoft.CSharp
Microsoft.Data.SqlClient
Microsoft.Data.SqlClient.SNI
Microsoft.ExtityFrameworkCore.Abstractions
Microsoft.ExtityFrameworkCore.Analyzers
Microsoft.ExtityFrameworkCore.Design
Microsoft.ExtityFrameworkCore.Relational
Microsoft.ExtityFrameworkCore.SqlServer
Microsoft.ExtityFrameworkCore.Tools
Microsoft.Extensions.Caching.Abstractions
Microsoft.Extensions.Caching.Memory
Microsoft.Extensions.Configuration
Microsoft.Extensions.Configuration.Abstractions
Microsoft.Extensions.Configuration.Binder
Microsoft.Extensions.DependencyInjection
Microsoft.Extensions.DependencyInjection.Abstractions
Microsoft.Extensions.Logging
Microsoft.Extensions.Logging.Abstractions
Microsoft.Extensions.ObjectPool
Microsoft.Extensions.Options
Microsoft.IdentityModel.Primitives
Microsoft.IdentityModel.Client
Microsoft.IdentityModel.JsonWebTokens
Microsoft.IdentityModel.Logging
Microsoft.IdentityModel.Protocols
Microsoft.IdentityModel.Protocols.OpenId.Connect
Microsoft.IdentityModel.Tokens
Microsoft.jQuery.Unobtrusive.Validation
Microsoft.Net.Http.Headers

Full-text reference of NuGet Package Dependencies (continued):

Microsoft.Owin
Microsoft.Owin.Host.SystemWeb
Microsoft.Owin.Security
Microsoft.Owin.Security.Cookies
Microsoft.Owin.Security.Facebook
Microsoft.Owin.Security.Google
Microsoft.Owin.Security.MicrosoftAccount
Microsoft.Owin.Security.OAuth
Microsoft.Owin.Security.Twitter
Microsoft.Web.Infrstruture
Modernizr
Newtonsoft.Json
Owin
PresentationFramework
System.Buffers
System.Collections.Immutable
System.ComponentModel.Annotations
System.Data.Common
System.Diagnostics.DiagnosticSource
System.IdentityModel.Tokens.Iwt
System.Memory
System.Numerics.Vectors
System.Runtime.CompilerServices.Unsafe
System.Text.Encodings.Web
System.Threading.Tasks.Extensions
WebGrease
WindowsBase
X.PagedList
X.PagedList.Web.Mvc
X.PagedList.Web.Common

6 Design Units Impacts

Three Functional Areas are described in this section, specifically FDA-1: Registered User Functions, FDA-2 Business Functions and FDA-3 Administrative Functions. These sections correspond to Section 3 (Identification of Requirements) and will primarily focus on the Controllers used to manage the functionality in each of these three corresponding sections. A list of all controllers (beyond those which are briefly illustrated below, or otherwise described partially as in text as a list of methods), is provided in Section 5 (Controllers Section).

6.1 Functional Area – Registered Users /Design Unit A

6.1.1 Functional Overview

Better Reviews separates a registered users from a site visitor with through the act of registration. A Registered User is then the foundation for the development of features found through the rest of the site. The act of registering a user and the set of features which are provided to all users (including those who do not register) are critical to describe as its own Functional Area. Therefore this section will focus on the initial registration components. The extended functions which grow from this functional area will be described in the next two sections.

6.1.2 Impacts

Both Business as well as Admin (each described below as separate Functional Areas) extend the initial registration of a site visitor to provide them with special privileges. These special privileges are used in both the Business and Admin function areas (see below).The ability for a Business user to take control of a Business profile, and to correspond with admin users are functions described in the Business Section (see next). The ability to register users, to edit their privileges and also to delete registered users are a function of Admin users (see last).

6.1.3 Requirements

The requirements list for this section are initially listed in Section 3.1, and are repeated here for clarity:

User-Features-01: Site Users Requirements

- User-Feature-01.01; Console Access (GUI) for New User S
- User-Feature-01.02; Keyword-Based Search Box
- User-Feature-01.03; Advanced search by category, rating, location, etc.
- User-Feature-01.04; Featured businesses on the front page
- User-Feature-01.05; Registration, Login, Profile Management for Site Reviewers
- User-Feature-01.06; Review a Business Feature for Registered Users

The aspects required for registering a user (list of functional requirements above) are handled by the Account Controller. Briefly the Account Controller includes the following methods.

Class AccountController : Controller

- AccountController
- ApplicationSignInManager
- ApplicationUserManager
- Login
- VerifyCode
- Register
- ConfirmEmail
- ForgotPassword
- ForgotPasswordConfirmation
- ResetPassword
- ExternalLogin
- SendCode
- ExternalLoginCallback
- ExternalLoginConfirmation
- LogOff
- Dispose

Written in C# the account methods are designed to be Self-Describing, and for example where a method listed above is named “LogOff” the user, when calling the method through the User Interface would expect to see their account profile status be reset to not logged in.

Of particular note in the list above, the Methods = External indicate external services such as Google, Twitter and Facebook login libraries, and Forgot Password and Code methods are designed to help users who have forgotten their login credentials retrieve them in a seamless manner.

While the entire Code Base is available for inspection as the attached application (see appendix):

Figure 6.1: a representative sample of code (the first 95 lines of code in the Account Controller) is provided for inspection.

```
AccountController.cs
BetterReviews
BetterReviews.Controllers.AccountCor
_signInManager

1 using System;
2 using System.Globalization;
3 using System.Linq;
4 using System.Security.Claims;
5 using System.Threading.Tasks;
6 using System.Web;
7 using System.Web.Mvc;
8 using Microsoft.AspNet.Identity;
9 using Microsoft.AspNet.Identity.Owin;
10 using Microsoft.Owin.Security;
11 using BetterReviews.Models;
12 using BetterReviews.Controllers.Account.Services;
13
14 namespace BetterReviews.Controllers
15 {
16     [Authorize]
17     public class AccountController : Controller
18     {
19         private ApplicationSignInManager _signInManager;
20         private ApplicationUserManager _userManager;
21         private AccountService _accountService = new AccountService();
22
23         public AccountController()
24         {
25         }
26
27         public AccountController(ApplicationUserManager userManager,
28             ApplicationSignInManager signInManager )
29         {
30             UserManager = userManager;
31             SignInManager = signInManager;
32         }
33
34         public ApplicationSignInManager SignInManager
35         {
36             get
37             {
38                 return _signInManager ?? HttpContext.GetOwinContext
39                     ().Get<ApplicationSignInManager>();
40             }
41             private set
42             {
43                 _signInManager = value;
44             }
45         }
46
47         public ApplicationUserManager UserManager
48         {
49             get
50             {
51                 return _userManager ?? HttpContext.GetOwinContext
52                     ().GetUserManager<ApplicationUserManager>();
53             }
54         }
55     }
56 }
```

Figure 6.2: Continued sample of code (lines 50 – 94 lines of code in the Account Controller).

```
AccountController.cs - X
BetterReviews BetterReviews.Controllers.AccountCor _signInManager
50 }
51 private set
52 {
53     _userManager = value;
54 }
55 }
56
57 //
58 // GET: /Account/Login
59 [AllowAnonymous]
60 public ActionResult Login(string returnUrl)
61 {
62     ViewBag.ReturnUrl = returnUrl;
63     return View();
64 }
65
66 //
67 // POST: /Account/Login
68 [HttpPost]
69 [AllowAnonymous]
70 [ValidateAntiForgeryToken]
71 public async Task<ActionResult> Login(LoginViewModel model, string
    returnUrl)
72 {
73     if (!ModelState.IsValid)
74     {
75         return View(model);
76     }
77
78     // This doesn't count login failures towards account lockout
79     // To enable password failures to trigger account lockout,
80     change to shouldLockout: true
81     var result = await SignInManager.PasswordSignInAsync
82     (model.Email, model.Password, model.RememberMe,
83     shouldLockout: false);
84     switch (result)
85     {
86     case SignInStatus.Success:
87         return RedirectToLocal(returnUrl);
88     case SignInStatus.LockedOut:
89         return View("Lockout");
90     case SignInStatus.RequiresVerification:
91         return RedirectToAction("SendCode", new { ReturnUrl =
92         returnUrl, RememberMe = model.RememberMe });
93     case SignInStatus.Failure:
94     default:
95         ModelState.AddModelError("", "Invalid login attempt.");
96         return View(model);
97     }
98 }
99 }
```

6.2 Functional Area Business Users / Design Unit B

6.2.1 Functional Overview

Building off the Functional Design Unit A (The registration of users) this Design Unit (B) provides registered users with a significant increase site tools. In particular the ability to Create a New Business, and also to Review a Business are both included in this functional Business Area.

6.2.2 Impacts

Business Users, who are also Registered Users, have the ability to interact with Site Admins, a special set of user who are described as their own Functional Business area in the section that follows. The interaction between Registered Business Users and Admin users are primarily connected to the Claim of a Business, and the Elevation of that user to a new set of privileges. The proper creation, administration, and establishment of features for business users therefore impacts the Admin functionality as will be seen in the next section.

6.2.3 Requirements

Business-User-Features-02:

- Business Users Requirements
- Business-User-Feature-02.01; Create Business (GUI) for New Business
- Business-User-Feature-02.02; Update Business (GUI) for Business Listing
- Business-User-Feature-02.03; Add and Remove Photo(s) for Business Listing
- Business-User-Feature-02.04; Subscribe to Reviews (posted by other users)
- Business-User-Feature-02.05; Send/Receive Notifications from Site Admin

Class BusinessController : Controller

- Search
- Index
- ShowNonValidatedOnly
- NonValidatedIndex
- Add
- Detail
- EditDetails
- AddBusinessImage
- WriteReview
- ReviewResponse
- AvailableCategories
- GetBusinessThumbnailImage
- GetBusinessImage
- DeleteBusinessImage
- ClaimBusiness
- ClaimBusinessRequestConfirmation
- ClaimApproval
- ClaimApprovedRequest

As was described in the previous section, The Class seen above Business Controller has self-describing methods (the list that follows the controller header), which generally describe the features which are implemented via the Controller. In this case instead of simply providing the setup and headers for the code, the code for three methods (Write Review, Edit Details and Claim Business) will provided for inspection.

Figure 6.3: Write Review Method of the BusinessController C# Class:

```
BusinessController.cs
BetterReviews
BetterReviews.Features.Business.BusinessController
WriteReview(int id)

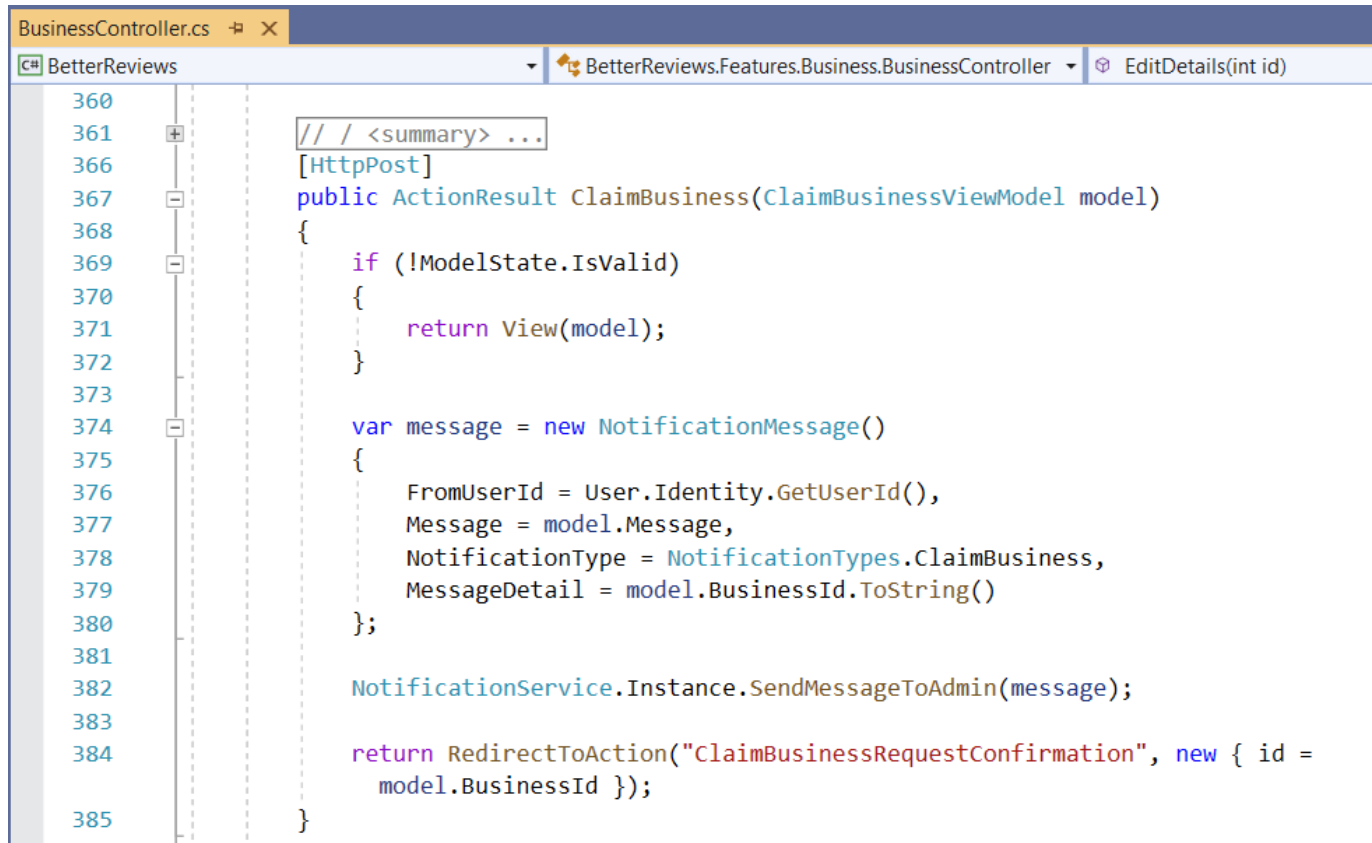
230     /// <summary>
231     /// Posts a populated review object to the database
232     /// </summary>
233     /// <param name="model"></param>
234     /// <returns></returns>
235     [HttpPost]
236     public ActionResult WriteReview(WriteReviewViewModel model)
237     {
238         if (ModelState.IsValid)
239         {
240             var userId = HttpContext.User.Identity.GetUserId();
241             int reviewId = _businessService.AddBusinessReview(model, userId);
242
243             // Add user supplied image here
244             if (model.ImageFile != null)
245             {
246                 byte[] imageBytes = Utilities.ImageHelper.ConvertStreamToBytes
247                     (model.ImageFile.InputStream);
248                 _businessService.AddBusinessReviewImage(reviewId, userId, imageBytes);
249             }
250
251             return RedirectToAction("detail", new { id = model.BusinessId });
252         }
253
254         return View(model);
255     }
256 }
```

Figure 6.4: Add (a Business) Method of the Business Controller C# Class:

```
BusinessController.cs
BetterReviews
BetterReviews.Features.Business.BusinessController
EditDetails(int id)

102     /// <summary>
103     /// Posts business details model to the database
104     /// </summary>
105     /// <param name="model"></param>
106     /// <returns></returns>
107     [HttpPost]
108     public ActionResult Add(AddBusinessViewModel model)
109     {
110         var recaptchaResponse = Request.Form["g-recaptcha-response"];
111         var isRecaptchaValid = true || _businessService.ValidateReCaptcha(recaptchaResponse);
112
113         if(isRecaptchaValid && ModelState.IsValid)
114         {
115             model.CreateBy = HttpContext.User.Identity.GetUserId();
116             var businessId = _businessService.AddBusiness(model);
117             return RedirectToAction("Detail", new { id = businessId.Value });
118         }
119
120         if (!isRecaptchaValid)
121         {
122             ViewBag.RecaptchaMessage = "Please validate \"I'm not a robot\"";
123         }
124
125         model.BusinessCategories = _viewModelFactory.CreateAddViewModel().BusinessCategories;
126         return View(model);
127     }
128 }
```


Figure 6.5: Claim Business Method of the Business Controller C# Class:



```
BusinessController.cs -P X
BetterReviews BetterReviews.Features.Business.BusinessController EditDetails(int id)
360
361 // / <summary> ...
366 [HttpPost]
367 public ActionResult ClaimBusiness(ClaimBusinessViewModel model)
368 {
369     if (!ModelState.IsValid)
370     {
371         return View(model);
372     }
373
374     var message = new NotificationMessage()
375     {
376         FromUserId = User.Identity.GetUserId(),
377         Message = model.Message,
378         NotificationType = NotificationTypes.ClaimBusiness,
379         MessageDetail = model.BusinessId.ToString()
380     };
381
382     NotificationService.Instance.SendMessageToAdmin(message);
383
384     return RedirectToAction("ClaimBusinessRequestConfirmation", new { id =
385         model.BusinessId });
385 }
```

6.3 Functional Area Administrative Users / Design Unit C

6.3.1 Functional Overview

Finally the functionality of the Admin user will be described as its own Design unit. The Admin user is the super user for the site, and while they have the privileges to add and remove other users, businesses, reviews and to elevate privileges of other user, they too first must be registered with the site (see discussion in Design Unit A above).

6.3.2 Impacts

The admin user has the greatest impact over the other two types of users Business and Registered User, however this set of users will by far constitute the smallest number of actual users working with or interacting with the site. While this limited audience has tremendous power, the influence of the Design Area is mostly restricted to limiting, editing or removing content and information posted by the two other major groups of users.

6.3.3 Requirements

Section 3.1 Admin-Features-03: Site Admin Requirements

Site-Admin-Feature-03.01; Admin Console (GUI) for Site Administrators

Site-Admin-Feature-03.02; Manage Users

Site-Admin-Feature-03.03; Manage Users Roles

Site-Admin-Feature-03.04; Manage Business Categories/Create Category

Site-Admin-Feature-03.05; Send/Respond to User Notifications/Messages

Site-Admin-Feature-03.06; Update/Add, Edit Existing Business

Class Admin.BusinessCategory Controller
CreateIndexListModel
Delete
Add
Edit

Class Admin.Roles Controller
CreateRolesListModel
Add
Delete

Class Controllers.Notification
GetUserMessages
DeleteMessage

Illustrating the code for the Admin section:

Figure 6.6: Admin Business Category Controller (Beginning Section)

```
BusinessCategoryController.cs
BetterReviews
BetterReviews.Controllers.Admin.Business
_viewModelFactory
1 using BetterReviews.Controllers.Admin.BusinessCategory.Models;
2 using BetterReviews.Controllers.Admin.BusinessCategory.Services;
3 using System.Web.Mvc;
4
5 namespace BetterReviews.Controllers.Admin.BusinessCategory
6 {
7     [RoutePrefix("admin/businesscategory")]
8     [Route("{action=index}")]
9     public class BusinessCategoryController : Controller
10    {
11        private readonly BusinessCategoryModelFactory _viewModelFactory = new
12            BusinessCategoryModelFactory();
13        private readonly BusinessCategoryService _service = new
14            BusinessCategoryService();
15
16        public ActionResult Index()
17        {
18            var model = _viewModelFactory.CreateIndexListModel();
19            return View(model);
20        }
21
22        [HttpPost]
23        public ActionResult Delete(int id)
24        {
25            _service.DeleteBusinessCategory(id);
26            return RedirectToAction("Index");
27        }
28    }
29 }
```

Figure 6.7: Admin Roles Controller (Beginning Section)

```
RolesController.cs  [X]
BetterReviews
BetterReviews.Controllers.Admin.Roles.R
_viewModelFactory

1  using BetterReviews.Controllers.Admin.Roles.Models;
2  using BetterReviews.Controllers.Admin.Roles.Services;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Web;
7  using System.Web.Mvc;
8
9  namespace BetterReviews.Controllers.Admin.Roles
10 {
11     [RoutePrefix("admin/roles")]
12     [Route("{action=index}")]
13     public class RolesController : Controller
14     {
15         private readonly RolesViewModelFactory _viewModelFactory = new
16             RolesViewModelFactory();
17         private readonly RolesService _rolesService = new RolesService();
18
19         // GET: Roles
20         public ActionResult Index()
21         {
22             var model = _viewModelFactory.CreateRolesListModel();
23             return View(model);
24         }
25
26         [HttpGet]
27         public ActionResult Add()
```

Figure 6.8: Notification Controller (Site Messages)

```
NotificationController.cs  [X]
BetterReviews
BetterReviews.Controllers.Notificati
Index()

1  using BetterReviews.Controllers.Notification.Services;
2  using Microsoft.AspNet.Identity;
3  using System.Web.Mvc;
4
5  namespace BetterReviews.Controllers.Notification
6  {
7      [Authorize]
8      public class NotificationController : Controller
9      {
10         [HttpGet]
11         public ActionResult Index()
12         {
13             var model = NotificationService.Instance.GetUserMessages
14                 (User.Identity.GetUserId());
15             return View(model);
16         }
17
18         [HttpPost]
19         public ActionResult Delete(int id)
20         {
21             NotificationService.Instance.DeleteMessage(id);
22             return RedirectToAction("Index");
23         }
24     }
25 }
```

7 *Open Issues*

There are no open issues related to this project.

8 *Acknowledgements*

This project was created as a team effort by the following Students at Governor's State University

Franciskovich, Anthony
Garwood, Clinton
Miulli, Eric
Patel, Ashitaben Hemalkumar
Patino, Mauro

We would like to thank the Governor's State University Division of Science Mathematics and Technology including:

Dr. (Xin) Jasmine Chen, Assistant Professor, xchen3@govst.edu

Bryce Johnsen, Academic Advisor, bjohnsen2@govst.edu

Nancy Rios, Office Support Specialist, nrios@govst.edu

9 *References*

Referenced Businesses and Public Product Review Sites:

Yelp	https://www.yelp.com/
Google Businesses	https://www.google.com/intl/en_us/business/
Facebook Businesses	https://www.facebook.com/marketingAPAC/reviews/
Yellow Pages	https://www.yellowpages.com/
Angie's List	https://www.angieslist.com/
Trip Advisor	https://www.tripadvisor.com/
Zagat (Restaurants Reviews)	https://zagat.com/
Restaurant.com	https://www.restaurant.com/
Open Table	https://www.restaurant.com/

Laws Regulations and Legal Conventions:

Section 512(c) of the Digital Millennium Copyright Act

<https://www.law.cornell.edu/uscode/text/17/512>

https://en.wikipedia.org/wiki/Digital_Millennium_Copyright_Act

Section 230 of the Communications Decency Act

<https://www.law.cornell.edu/uscode/text/47/230>

https://wikipedia.org/wiki/Section_230_of_the_Communications_Decency_Act

Personally Identifiable Information Overview

https://wikipedia.org/wiki/Personal_data

European Union General Data Protection Regulation

<https://eur-lex.europa.eu/eli/reg/2016/679/oj>

https://wikipedia.org/wiki/General_Data_Protection_Regulation

California Data Privacy Act

<https://www.oag.ca.gov/privacy/ccpa>

https://wikipedia.org/wiki/California_Consumer_Privacy_Act

State of Illinois: Personal Information Protection Act

<https://www.ilga.gov/legislation/ilcs/ilcs3.asp?ActID=2702>

Additional References of Software, Topics and Themes:

Internet Information Services Web Server (IIS)

http://www.wikipedia.org/wiki/Internet_Information_Services

Razor Markup Language

<http://www.nuget.org/packages/Microsoft.AspNet.Core.Razor>

MVC or Model, View, Controller Software Design Pattern

<https://en.wikipedia.org/wiki/Model-view-controller>

Microsoft.com Documentation MVC Views

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/>

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/>

<https://docs.microsoft.com/en-us/aspnet/core/mvc/models/>

ASP.NET MVC

<https://en.wikipedia.org/wiki/ASP.NET>

Creative Commons Images and Fair Use License

Microsoft ASP.NET Logo

<https://devblogs.microsoft.com/visualstudio/wp-content/uploads/sites/4/2019/01/visualstudio-1.png>

GitHub Logo: Where the world builds software

<https://github.com>

Application Layer Image

<https://stackoverflow.com/questions/35834262/having-trouble-with-deployment-diagram>

<https://creativecommons.org/licenses/by-sa/3.0/>

Database Image

Photo by Unknown Author is licensed under CC BY-SA-NC

<https://nixfaq.org/2013/08/top-features-of-mysql.html>

<https://creativecommons.org/licenses/by-nc-sa/3.0/>

10 Appendices

Structured Query Language (SQL) Master Code File: betterreviews_sql_script.sql

Description: This SQL File will allow the database, including the tables, constraints and data to be reconstructed.

The recommended SQL engine (i.e., exported from) is SQL Server using SQL Server Management Studio.

Initial Business Data File (JSON) Sample Business Data: restaurants_json

Description: This JSON file (structured objects) is the referenced test data, collected from the Yellow Pages API.

Application File (Better Reviews) Compressed (Zip) File

ASP.NET MVC Application with SQL Server (MSSQL Server) Database. Recommended Visual Studio

Recommended Development Environment: Microsoft Visual Studio Community 2019 Version 16.7.7