UNIVERSITY OF HERTFORDSHIRE

School of Computer Science

BSc Honors Computer Science (Online)

6WCM0009 - E-Learning Applications Design and Development

Final Report

SQL Tutor

Student Number: 15000887

Academic Supervisor: Dr. Mariana Lilley

September 2016

Abstract

The purpose of this report is to present the academic foundation and methodology used to develop an E-Learning solution. This solution was developed using ASP.NET/VB.NET and is intended to support individual learners in developing their practical skills in the use of Structured Query Language (SQL).

Studies have shown that learning occurs at various psychological levels and students can easily become overwhelmed with too much information. This information has formed the basis of the method in which the learning content is presented to the user in the form of micro lessons. Exercises are intended to be short, specific and focused. To facilitate learners with varying preferences, links to other types of media will be provided, in relation to the subject matter.

The project has also placed some emphasis on responsive web design, due to the increase of mobile device usage. The report presents all of the associated research material in greater detail.

Acknowledgements:

This project would not have been possible without the guidance of CTS lecturer Mr. Kerron Ramganesh and the clarification provided by our academic leader, Dr. Mariana Lilley.

Table of Contents

Chapter 1. Introduction

1. Introduction

This final report presents holistic view on all the activities involved in the development of an e-learning application. This web application is focused toward helping learners develop their skills in the use of SQL. The report also includes all the phases of the development process, challenges encountered and a comprehensive view on the final result.

1.2 The Subject

The concept of e-learning, quite simply, refers to the delivery of knowledge by the use of internet based content and mechanisms. The process incorporates the use of different technologies to create an environment, similar to that of the traditional classroom. In a traditional scenario, a lecturer is able to read the reaction of his audience and adapt the delivery of his content accordingly in the typical classroom. The use of eye contact and gestures demands attention and questions can be asked to students to stimulate thought and discussion, which aids in the learning process.

The challenge, however, with an e-learning system is to be able to facilitate the transfer of knowledge to the learner, in a way that is appealing to their particular needs and at the same time, being able to overcome the constraints caused in the absence of physical presence.

The e-learning system is intended to contain elements that is in alignment with learning theory, to facilitate the transfer of knowledge to its target audience, as well as functions and features which would encourage the use of the system.

1.3 Project Aims

The aim of this project is to develop an e-learning application. This e-learning application is intended to support individual learners in developing their practical skills in the use of Structured Query Language. Many academic studies have shown the processes involved in the learning process. Consequently, this project aims to use these studies to present the subject matter, in alignment with learning theory. The following points outlines the Core and Advanced project objectives.

1.4 Core Project Objectives

- To develop a prototype web application using ASP.NET, intended to support individual learners in developing their practical skills in the use of Structured Query Language (SQL).
- Conduct thorough research about e-learning (definition), e-learning theories and the activities associated with the learning process.
- To incorporate various aspects of e-learning technologies, as well as learning theories to support the transfer of knowledge to the user and enhance the learning process.
- Choose and adopt a formal approach to the development of the application, including a comprehensive Integrated Development Environment.
- Evaluate the prototype against both functional and non-functional requirements.

1.5 Advanced Project Objectives

- Implement a mechanism to provide instant feedback, based on user input.
- Track user progress and allow the user to resume a lesson at the point it was left.
- Learn and Implement web technologies such as CSS, JavaScript/JQuery to enhance user experience.
- Develop a responsive web application, adjusting the layout based on the device it is being viewed on.

1.6 Project Plan

A project plan was developed by identifying the critical activities required to complete the project and the associated deadlines.

Objective	**Deadline**
Requirements analysis	8th June, 2016
Progress report 1	15th June, 2016
Background research	22nd June, 2016
Database design	27th June, 2016
Develop prototype (Basic Functionality)	29th June, 2016
Progress report 2	7th July, 2016
Evaluate prototype	12th July 2016
Further requirements analysis	14th July 2016
ASP.NET research	19th July 2016
VB.NET research	20th July, 2016
CSS research	21st July, 2016
Responsive Design research	22nd July, 2016
Application design	28th July, 2016
Implementation	19th August, 2016
Testing	22nd August, 2016
Evaluation	25th August 2016

Chapter 2. Background Research

2.1 Introduction

The purpose of this chapter is to present the main findings associated with key subject areas, in relation to learning theory, similar e-learning websites and Structured Query Language. These findings will then support a list of functional and non-functional requirements and establish the foundation for the rest of the project and software development.

2.2 Learning Theory

According to *Deans for Impact, The Science of Learning 2015*, “learning occurs at various psychological levels”. The process involves the transfer of information in the brain from working memory to long term memory. This process also suggests that it is possible for the learner to become overwhelmed if presented with too much information at once. In reviewing this article, it was understood that the use of worked examples reduces cognitive overload, as solutions are first demonstrated systematically to the learner. Following this demonstration, the user is then asked to solve the problem independently.

Given the aforementioned, this theory can be applied to an E-Learning application. Learning content could be presented to the learner, focusing on one specific learning outcome at once. Each lesson is first demonstrated to the learner, using examples that link the lesson to a real life scenario. The learner is then asked to demonstrate their understanding of the content delivered.

Further studies on learning theory revealed that the topic of behaviorism has been a major contributor to one's understanding of the learning process. The theory of behaviorism involves the psychological study of stimulus, stimulus response and the relationship between the two (Watson, J.B 2013). According to this theory, learning can be stimulated through positive reinforcement and a strong reward system.

Although some studies have shown that positive reinforcement can have aversive effects (The Behavior Analyst, 2003), studies that are more recent have shown that it did in fact achieve greater results (Nauert PhD, R. 2015). Based on this information, it can be a good design decision to exclude any form of punishment within the E-Learning application. For example, the learner can simply be asked to "try again", rather than being told "You are incorrect".

In keeping with Watson's philosophy, the learner's stimulus can be in the form of a point system, upon successful completion of a lesson. The use of badges and ranks can encourage the learner to learn and achieve more. This introduces an element of positive reinforcement to enhance the learning process (Skinner B.F., 1969; Maag, 2001). In the next section, the subject of gamification will be analyzed in relation to Watson's theory of behaviorism.

2.3 Gamification

It has been proven by some researchers that gamification has become a popular tactic to encourage specific behaviors, increase motivation and engagement. Studies have also shown that stages and milestones are powerful tools used to motivate a learner. These tools also enable an instructor to sequence knowledge and quantify what the student needs to learn at the end of each stage or milestone (Hsin-Yuan Huang W.; Soman d. 2013).

By applying this concept, learning content is designed into stages, increasing in complexity as the user progresses. To ensure that the learning content is delivered in a sequential manner, higher levels are inaccessible, preventing the user from progressing unless the previous objective is passed.

This mechanism enables each level to focus on a pre-determined learning outcome, maintaining the objective of reducing cognitive overload.

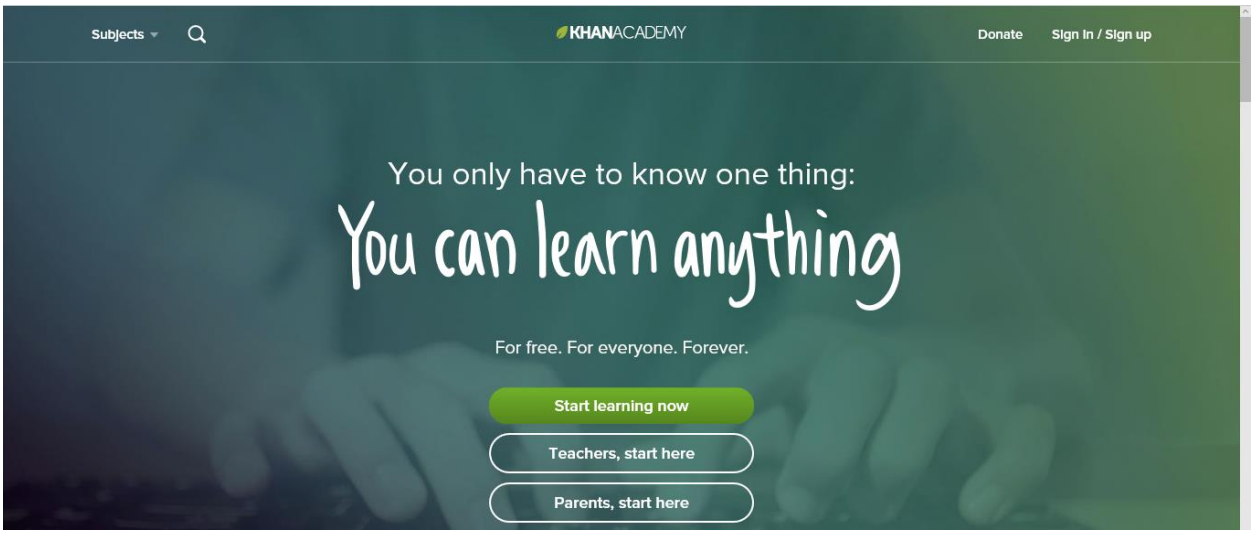
Studies have also shown that diverse learning environments and rewards aim at increasing motivation and higher levels of engagement in the learning process (Kapp M. K., 2012). This directly corresponds with the learning theory of behaviorism, stimulating the desired behavior through rewards and badges. Further research has also shown that adding gamification elements increases information retention. These elements assist people with attention disorders, capturing the attention of those who have trouble focusing in a normal learning setting (Meredith A. L). Several other professionals endorsed the use of Gamification in an article titled "How Gamification Reshapes learning".

2.4 Similar websites teaching SQL

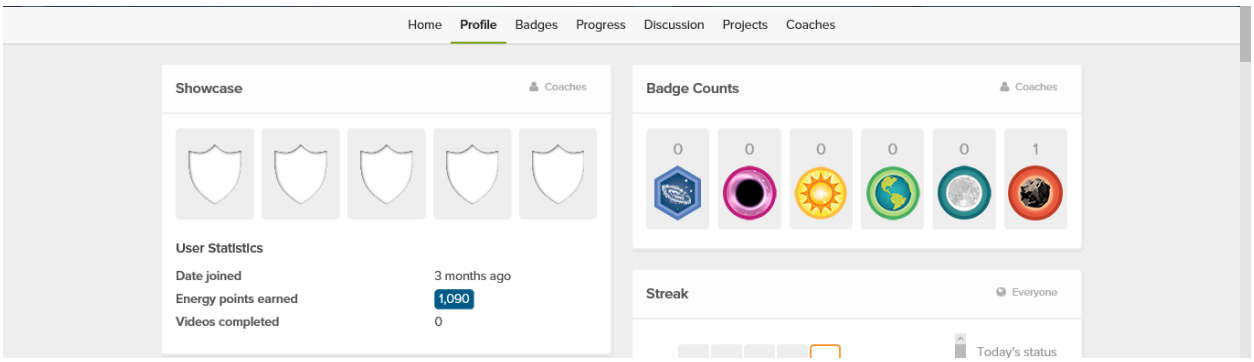
Using Google as an online search tool, it was determined that among the top eLearning websites offering SQL training were Khan Academy, w3Schools and CodeCademy. The main features and elements of these websites were analyzed, in relation to learning theory.

2.4.1 Khan Academy

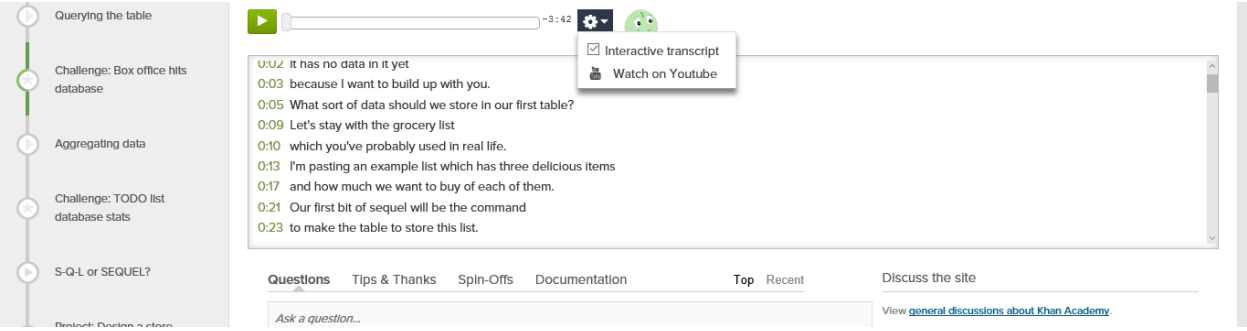
Navigation into the website began with a landing page containing an inspirational message. The page also contained an interesting color scheme of green, which stimulates positive emotional responses (Kaya, Naz; Epps, Helen H, 2004). The color green also represents balance, harmony and equilibrium, being within the center of the color spectrum (Wright A. 2016).



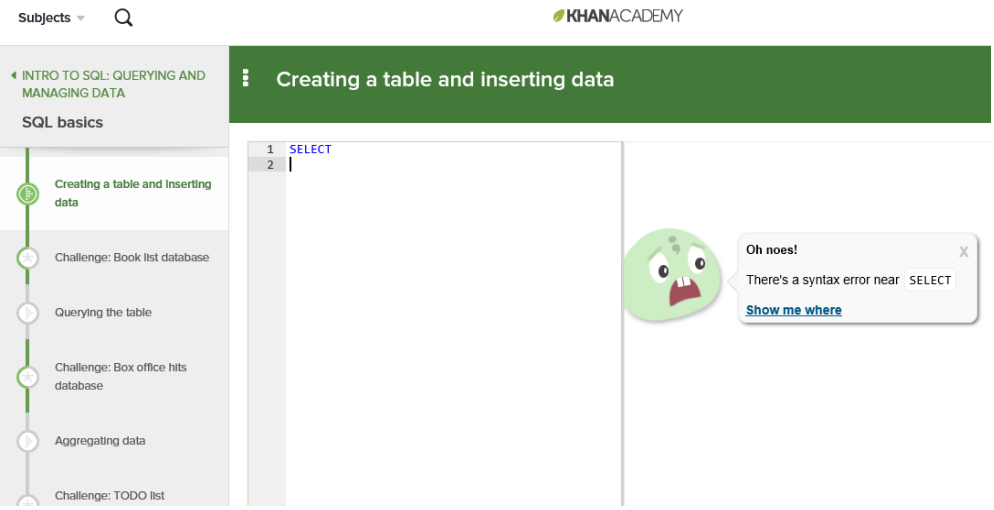
Khan Academy also contained a profile page, which contained various forms of user statistics and achievements. This corresponded with the philosophies of both Skinner and Watson by adding an element of positive reinforcement (Khanacademy.org, 2016).



Another interesting feature involves the user having the ability to choose the form in which the learning content is delivered. To achieve this, the learner is able to view a text transcript or watch a video\audio presentation. Learning theory of Kolb corresponds with this feature, as varying forms of learning content covers the four dimensions of learning.

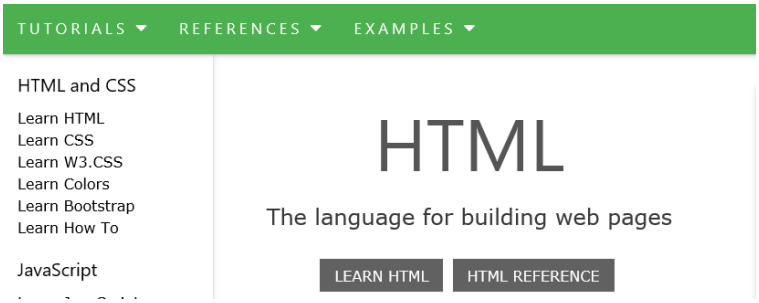


Other interesting features included direct interaction with the content, allowing the user to type SQL commands, with instant feedback displayed.

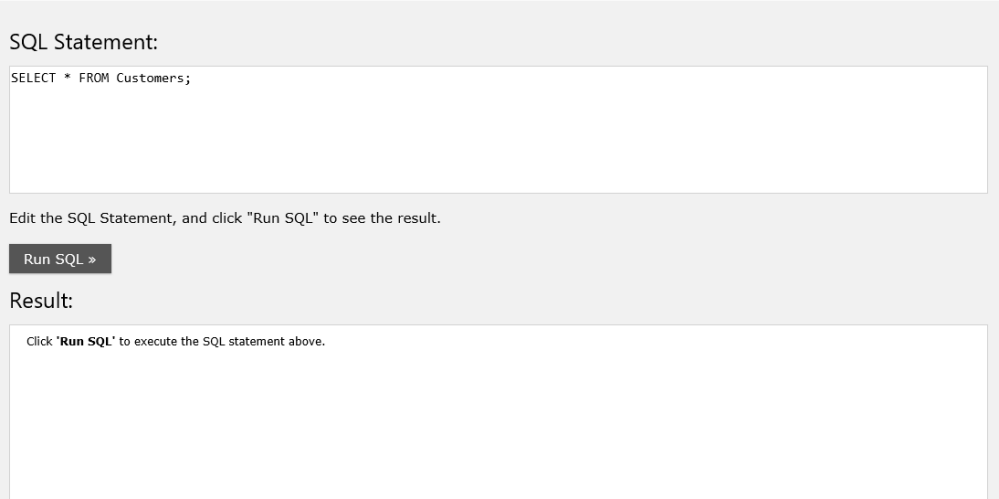


2.4.2 W3schools

W3schools was also listed as one of the top eLearning websites in existence today. In analyzing this website, some interesting elements were also identified, which was in alignment with eLearning theory. The main theme of the website is green, which, according to color psychology, represents balance and harmony (Wright A. 2016).



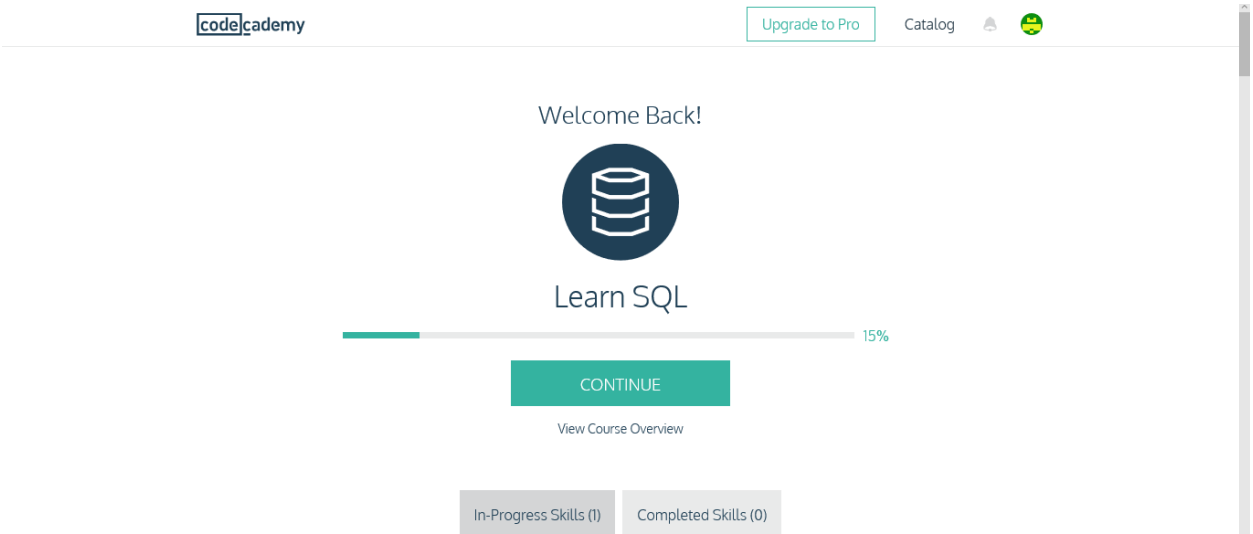
Tutorials were structured very specific, delivering one learning outcome at a time which corresponded with theories of cognitivism.



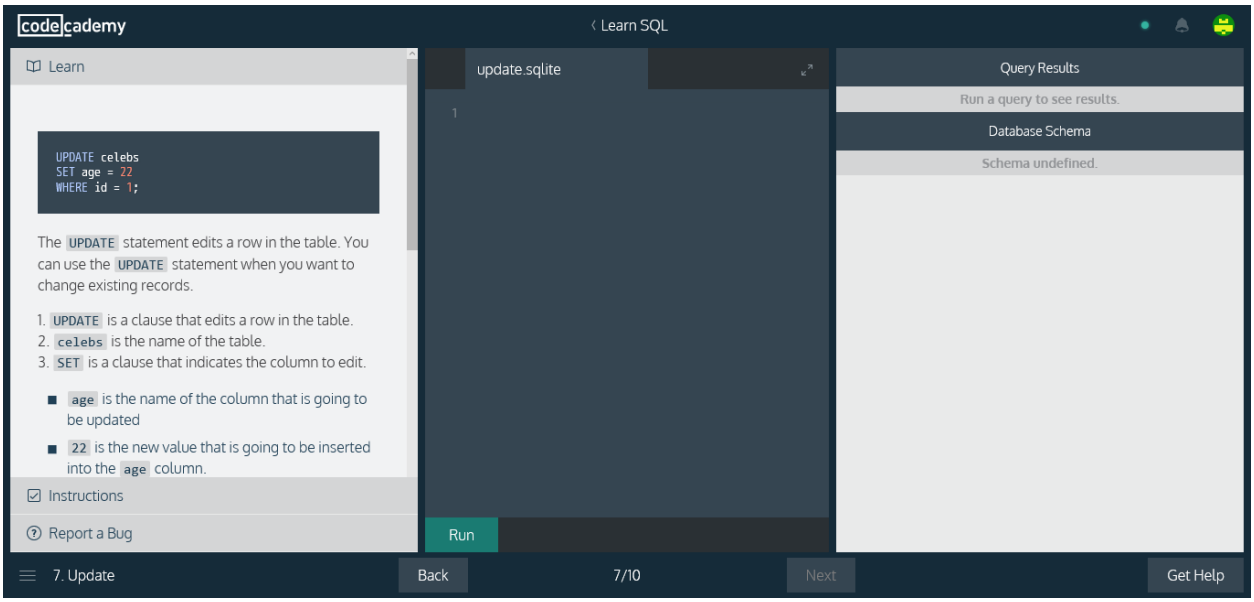
(W3schools.com, 2016).

2.4.3 CodeCademy

Upon first login into CodeCademy, a user is taken to a start page, where they can continue a previously attempted lesson. This proved to be a very practical feature, as a returning user is able to jump in right where they left off. A progress bar representing the percentage completed also formed part of this welcome page (Google Books, 2016).



The learning workspace contained another appealing feature, with the instructions on the left panel, the input area in the middle, and output on the left.



However, it was observed that the delivery of the content was not quite clear, as it was difficult to determine where to start and what behavior was expected. To resolve this, visual elements in the eLearning application, such as showing and hiding web form controls could be used. In addition, highlighting borders can attract the user's attention and reduce cognitive load. This will assist the user in identifying the desired behavior (Codecademy, 2016).

All the above mentioned features were taken into consideration and formed the basis for the requirements analysis.

2.5 Responsive web design

According to the US Mobile app report, “The days of desktop dominance are over”. This whitepaper shows an increase in mobile usage, with smartphones and tablets having 60 percent of activities in the U.S alone. This is a clear indication that the average person spends more time using a smart device, when compared to a desktop. Therefore, web designers are now required to ensure that web projects are able to adapt to the device on which it is viewed. This adaptation is essential, as the alternative would be to send visitors with various devices to different sites (Bryant and Jones, 2012.).

Considering these facts, in order to understand the responsive web technologies in existence today, further research was conducted. Most of this research explained the use of relative widths and heights in the form of percentages, rather than absolute measurements (pixels). Media queries are also used to adapt the display to varying screen sizes, as well as adjusting the layouts as required (99designs blog, 2012). In conjunction with media queries, the CSS Flexbox has proven to facilitate responsive web design (YouTube, 2016)

Chapter 3. Requirements Analysis

3.1 Introduction

In this chapter, the information gathered from research conducted will be used as the foundation to formulate a list of functional and non-functional requirements. The methodology used for the requirements gathering involved the use of personas representing the targeted users. In addition, Use Case diagrams were used to illustrate different scenarios in order to determine the major web pages required.

3.2 Personas

A group of two fictitious users were formed to determine the intended users for the e-learning solution.

Persona 1

Name: Crystal Argos

Age: 22

Occupation: Student

Crystal is an aspiring programmer, beginning her academic study with Java and vb.net programming languages. She has gained sufficient knowledge in front end development and seeks to expand her development skills by learning SQL which was recommended by her lecturer. She is self-motivated and does not work well with groups. Her major weakness though is that she has a short attention span and can sometimes become easily distracted.

Persona 2

Name: Michael Yang

Age: 26

Occupation: Web Developer

Michael has been a web developer for the past 5 years. He is most experienced in JQuery and layout designing and has recently started a course in SQL at a local college. However, he is bombarded with notes from the lecturer and finds it very difficult to follow the lessons, without any practical examples. He is seeking to find an e-learning application to supplement the notes received in class and practice the SQL syntax.

3.3 Functional Requirements

Based on research conducted and the targeted uses, the following functional requirements were identified.

Number	Requirement	Rationale
F1	Login Page\Registration	As the user accesses the web application, he\she will be asked to login or register. This is to ensure that all users are authenticated before accessing the website so that user progress can be tracked.
F2	Micro lessons	Learning content should be structured with short, specific learning outcomes to cater for learners who are easily overloaded.
F3	Interactive	The delivery of the learning content must be interactive, allowing the user to type the SQL command in a textbox and view the result of the query.
F4	User Profile\Dashboard	The application must contain a user profile, with a dashboard so the user can view their current progress, lesson and current rank\achievement (Behaviorism). A continue button must allow the user to continue to the current\next lesson.
F5	Resume lesson at specific points	Each lesson should be designed in sequences which is defined in a database, so a user can resume the lesson at a specific point.
F6	Achievements and Skill Levels	Users must receive an achievement, or increase in skill level at the end of each exercise. This feature will add an element of positive reinforcement to enhance learning.
F7	Points system	To add an element of gamification, users receive a defined amount of points (Query Points) upon the successful completion of each exercise.
F8	Unlockable Lessons	In addition to F6, users cannot proceed forward to another level unless the current level is completed (Lessons are done in sequence). Each lesson checks if the previous was completed by querying the database, before it can be accessible.
F9	Feedback mechanism	A feedback mechanism is used to inform the user of what is happening during each lesson in the form of messages displayed to the user.
F10	Responsive	The display of each window should adapt to varying screen sizes with readable content changing layout to suit the screen and maintain functionality

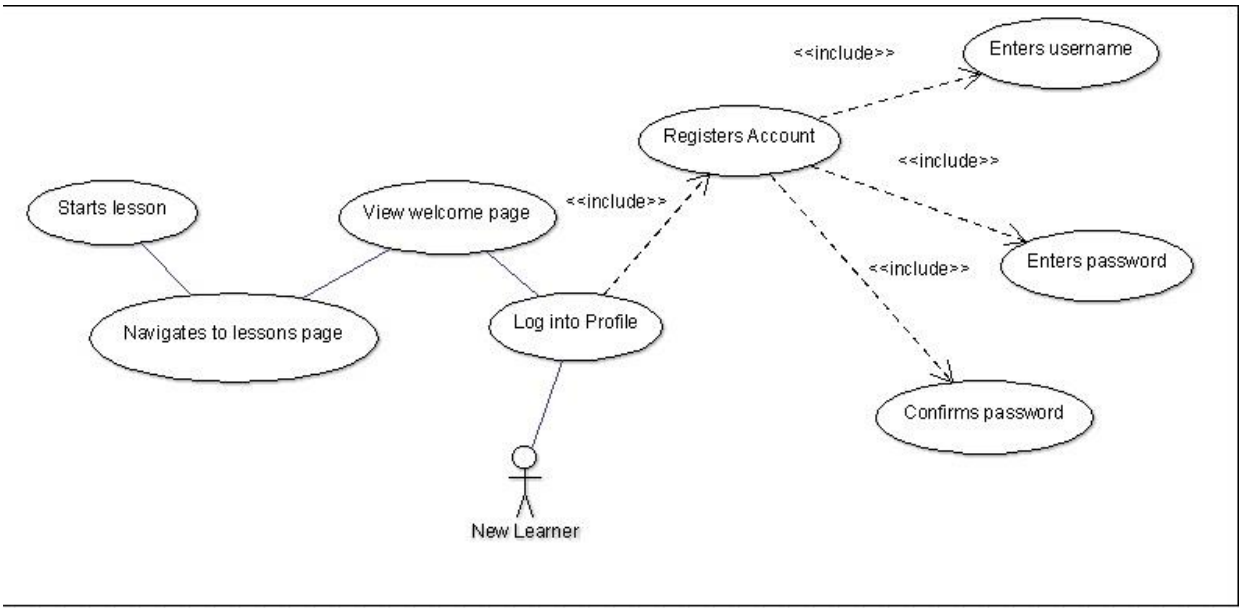
3.4 Non-Functional Requirements

Number	Requirement	Rationale
NF1	Database Security	Database and contained data must be secure. Users are not allowed to directly query the database.
NF2	Flexibility	The application must be flexible enough to support the needs of the user. Learning content is presented in various forms: Interactive lessons, videos and a quick reference guide to reduce cognitive overload and caters for the returning user who just wants to refresh their memory.

NF3	Maintainability	The application should be dynamic so that lessons can be easily updated. This should be done by supplying the learning content from a database.
NF4	Consistency	The appearance and delivery of the learning content should be consistent to reduce cognitive overload. Layouts, buttons and the positioning of key elements should be consistent in each page. (Nielsen's Heuristics)
NF5	Usability	The system must be easy to use and navigate using menu items and navigation buttons directing the user to the desired web page. The user must be able to identify the page to which they are currently viewing (Nielsen's Heuristics)

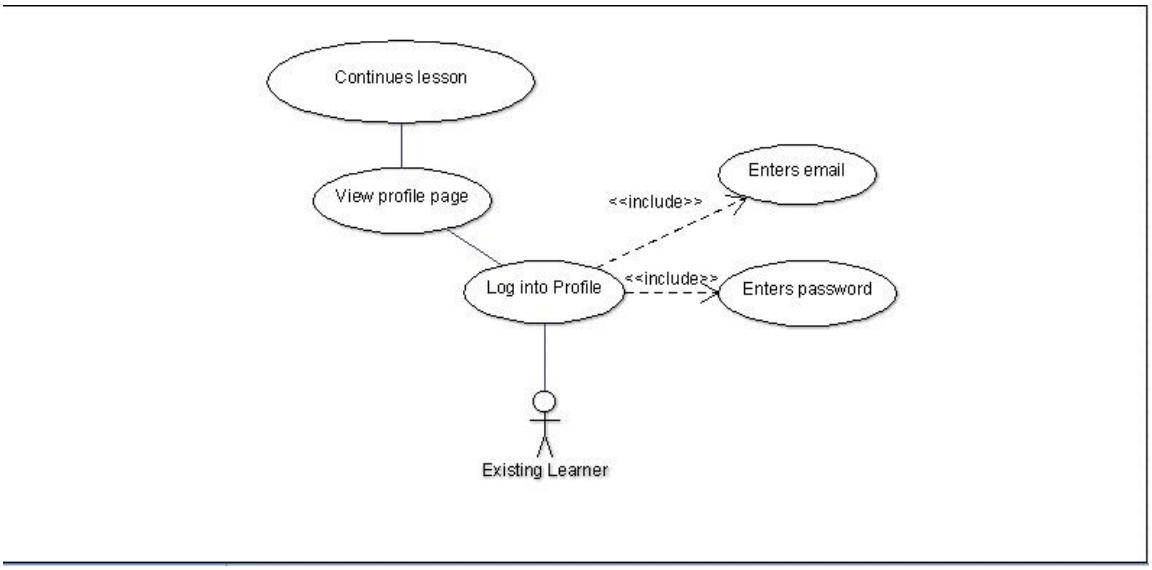
From the list of user requirements, Use Case diagrams were drafted to visualize the process in which a user interacts with the application, based on a few scenarios. This process aids in determining the main pages required for the project.

3.5 New learner use case



New Learner Use Case details: A new learner navigates to the login page and selects the register account link and is redirected to the registration page. Here, he\she enters a username, password and confirms password. Upon registration, he\she is then redirected to a welcome page, with the option to start learning. Upon navigation to the lessons\missions page, they proceed to begin a mission.

3.6 Existing Learner



Existing Learner Use Case details:

An existing learner logs in, navigates to the profile page and continues the last lesson previously attempted.

Chapter 4. Design

4.1 Introduction

The aim of this chapter is to show the design process of the eLearning application. Software design is a critical phase in the software development process and this chapter seeks to outline the steps taken to create a conceptual model of the application.

4.2 Database Design

The first step in the design process involved collating a list of the data, which the eLearning system recorded, according to the user requirements. The requirements analysis provided most of the information for this phase in the development process, as functional requirements formed the basis of the data required.

From the requirements analysis, the following list was gathered:

Feature	**Characteristic**
Login\Users:	Username, Password
Profile:	Rank, rank image (Badge), points
Lessons:	Lesson name, description, video

The list was then translated into an Entity Relationship Diagram to determine how each entity related to the other.

The first entity relationship diagram revealed two (2) Many-to-Many relationships, which is not in accordance with good relational database design. These were the relationships between Users\Ranks and Users\Lessons.

4.3 Normalization from UNF to 3NF

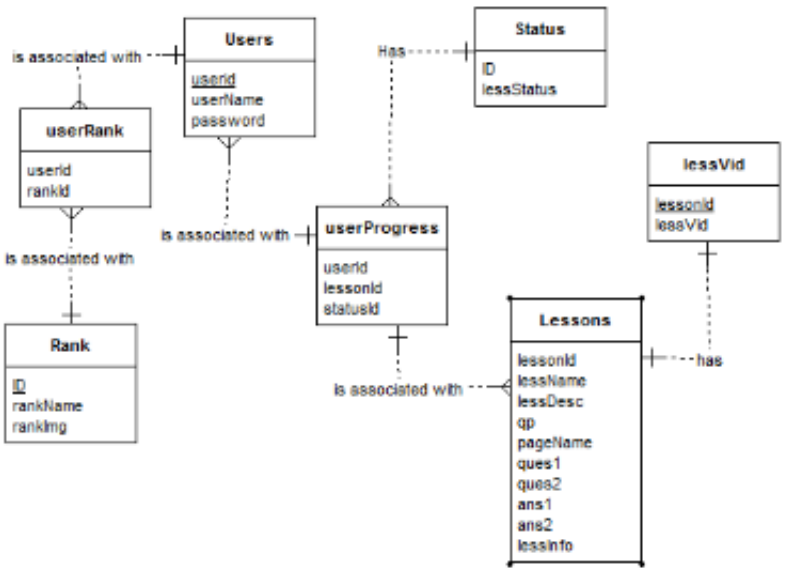
Using the normalization process, the Many-to-Many relationships were eliminated.

UNF	1NF	2NF	3NF	Tables
userId username passWord rankId rankName rankImg	<u>userId</u> username passWord <u>userId*</u> <u>rankId</u> rankName	<u>userId</u> username passWord <u>userId*</u> rankId*	<u>userId</u> username passWord <u>userId*</u> rankId*	Users userRank

lessonId lessName lessDesc lessStatus lessonVid qp pageName ques1 ques2 ans1 ans2 lessInfo	rankImg userId* lessonId lessStatus lessName lessDesc lessonVid qp pageName ques1 ques2 ans1 ans2 lessInfo	rankId rankName rankImg userId* lessonId* lessStatus lessonId lessName lessDesc lessonLevel lessonVid qp pageName ques1 ques2 ans1 ans2 lessInfo	rankId rankName rankImg userId* lessonId* statusId* statusId lessStatus lessonId lessName lessDesc qp pageName ques1 ques2 ans1 ans2 lessInfo lessonId* lessonVid	Rank userProgress Status Lessons lessVid

Following the normalization process, the two (2) Many-to-Many relationships were eliminated, creating two new tables: userProgress and userRank.

Fig 1. Final Entity Relationship Diagram



Users – The intention of this entity is to record user information, primarily the username and password.

UserRank – The intention of this entity is to capture the rank associated with each user. The inclusion of this entity was because of the normalization process, separating Many-to-Many relationships.

Rank – The intention of this entity is to store the rank information, primarily the rank name and the image associated.

Userprogress – The userProgress entity captures the progress of each user, primarily the user id, current lesson id and the status id. The inclusion of this entity was because of the normalization process. Querying this table determines the lesson a user is currently attempting.

Status – This entity captures the basic status information. Each status has a unique ID and a name

Lesson – The lesson entity records all the information relevant to each lesson.

LessVid – The lessVid entity is responsible for recording all the information relevant to each video, primarily a unique ID, the URL of the video file and the title.

Following the completion of the normalization process, a logical model of the database was constructed.

4.4 Logical Model (Schema)

Users (<u>userId</u>, username, passWord)

UserRank (<u>userId*</u>, rankId*)

Rank (<u>ID</u>, rankName)

userProgress (<u>userId*</u>, <u>lessonId*</u>, statusId*)

Status (<u>ID</u>, lessStatus)

Lesson (<u>lessonId</u>, lessName, lessDesc, qp, pageName, ques1, ques2, ans1, ans2, lessInfo)

LessVid (<u>lessonId</u>*, lessonVid)

4.4 Data Dictionary

Table:	**Users**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
userId	AutoNumber	Max	No	PK
rankId	Number	Max	No	FK
Email	Short Text	Max	No	Unique

Table:	**userRank**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
userId	Number	Max	No	FK

rankId	Number	Max	No	PK

Table:	**Rank**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
ID	AutoNumber	Max	No	PK
rankName	Short Text	Max	No	FK

Table:	**userProgress**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
ID	AutoNumber	Max	No	PK
userId	Number	Max	No	FK
lessonId	Number	Max	No	FK
lessonStatus	Number	Max	No	FK
seqId	Number	Max	No	

Table:	**Status**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
userId	AutoNumber	Max	No	PK
rankId	Number	Max	No	FK
Email	Short Text	Max	No	Unique

Table:	**Lessons**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
ID	AutoNumber	Max	No	PK
lessName	Short Text	Max	Yes	
lessDesc	Short Text	Max	Yes	
lesRank	Number	Max	Yes	
Qp	Number	Max	Yes	
pageName	Short Text	Max	Yes	
Ques1	Short Text	Max	Yes	
Ques2	Short Text	Max	Yes	
Ans1	Short Text	Max	Yes	
Ans2	Short Text	Max	Yes	
lessInfo	Short Text	Max	Yes	

Table:	**lessVid**			
Field	**Data Type**	**Length**	**Allow Nulls?**	**Constraint**
lessonId	AutoNumber	Max	Yes	PK
lessVid	Short Text	Max	Yes	

4.5 Application Design

In an attempt to present a sense of familiarity to the learner, the basis of the eLearning application centered on a Metro UI style interface (Windows 10). A flat design is used as the main theme, throughout the

eLearning website. The inspiration for each page layout came from the research conducted on similar eLearning websites. Using Balsamic, mockups were created to visualize the layout of the major pages.

4.5.1 Missions page design

The intention of the mission's page is to present a learner with all the missions in a tile layout. The SQL mission begins at stage one. The learner is required to complete the lesson before lesson 2 is unlocked and so on. The learner earns query points upon completion of each lesson and a pre-defined amount is required to unlock the bonus stage.

Fig. 2 Missions page design

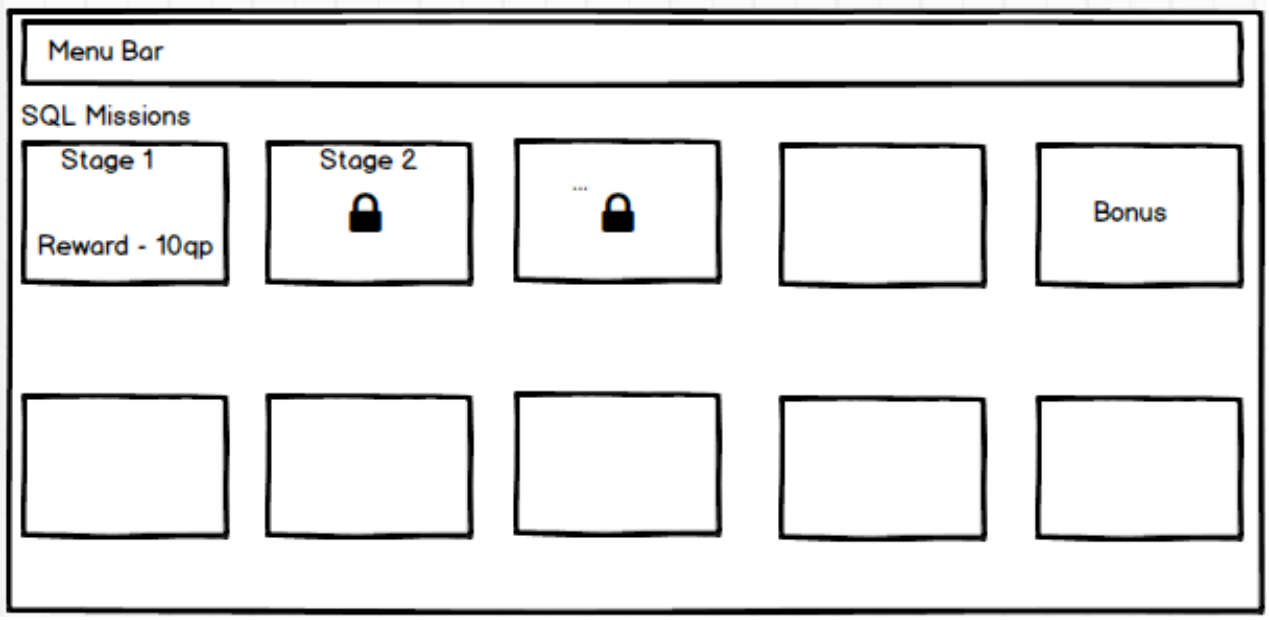


Fig 3 Profile page design

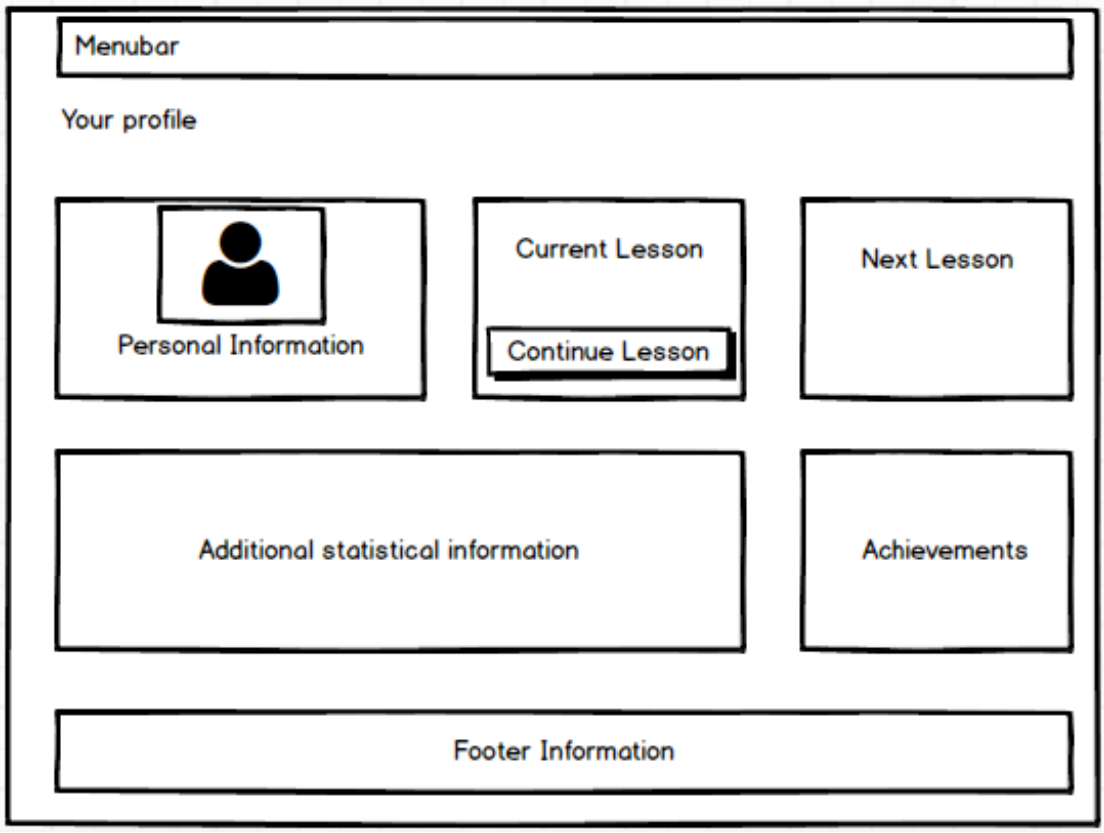


Fig. 4 Main mission page design

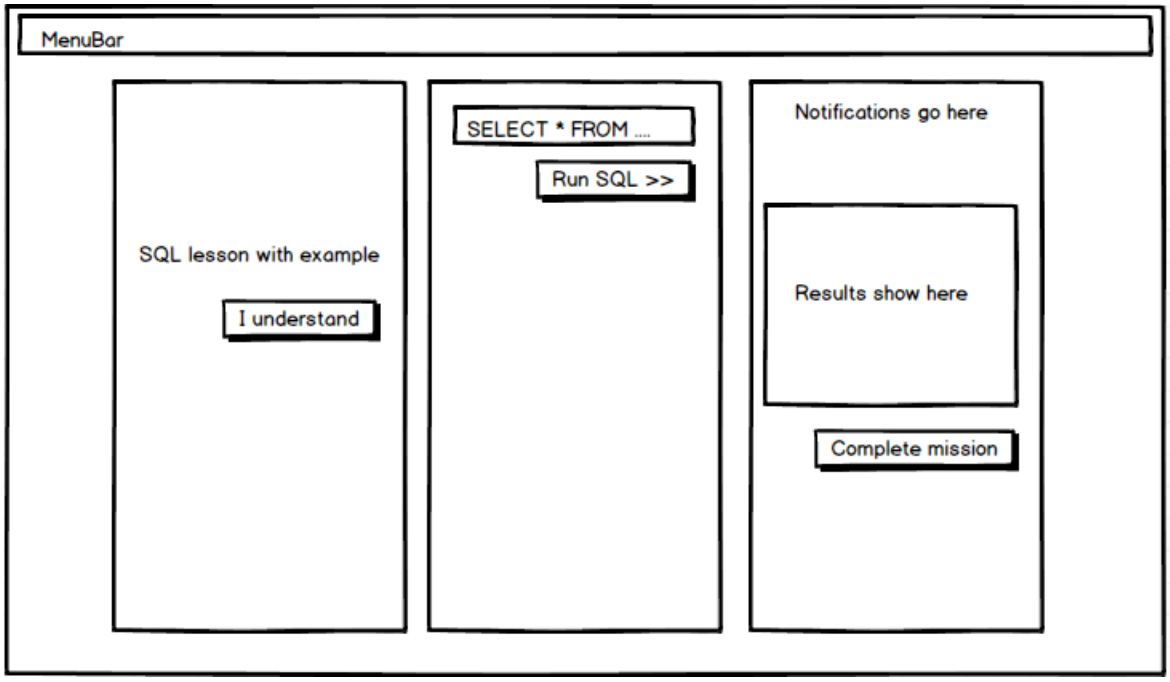
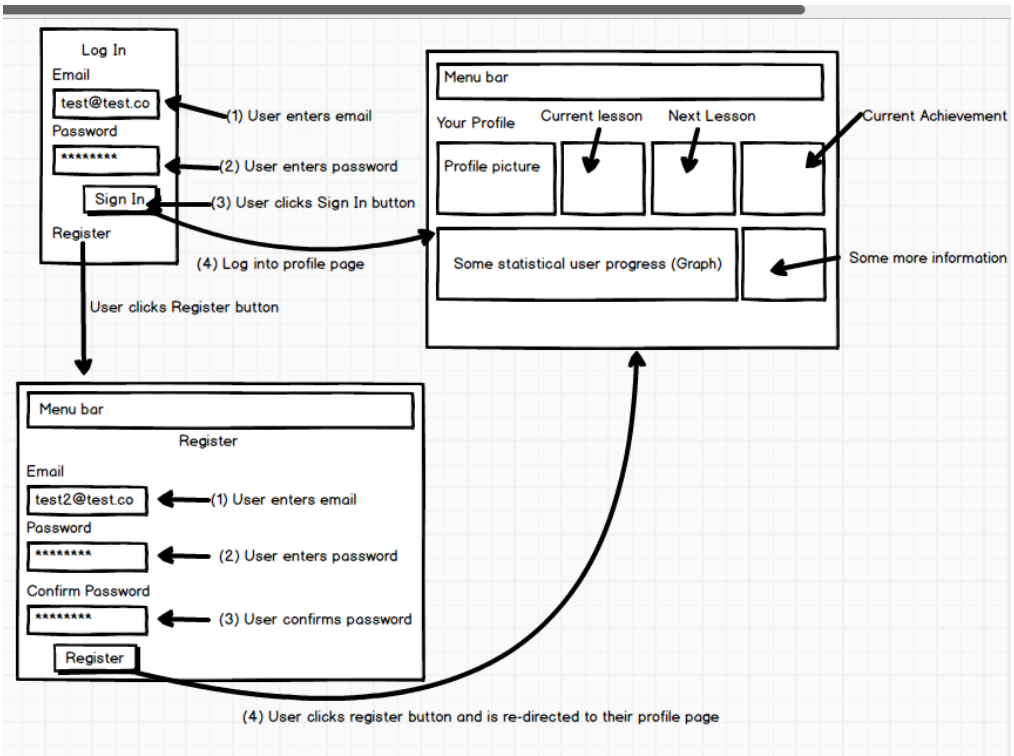


Fig. 5 Story Board for login



Chapter 5. Software Implementation

5.1 Introduction

The intention of the software implementation chapter is to present a walkthrough of the development process. It will demonstrate how the research, requirements and design translates into a working prototype of the eLearning system. The implementation phase begins with a development plan, which outlines the stages undertaken to development core functionality, leading into other advanced features.

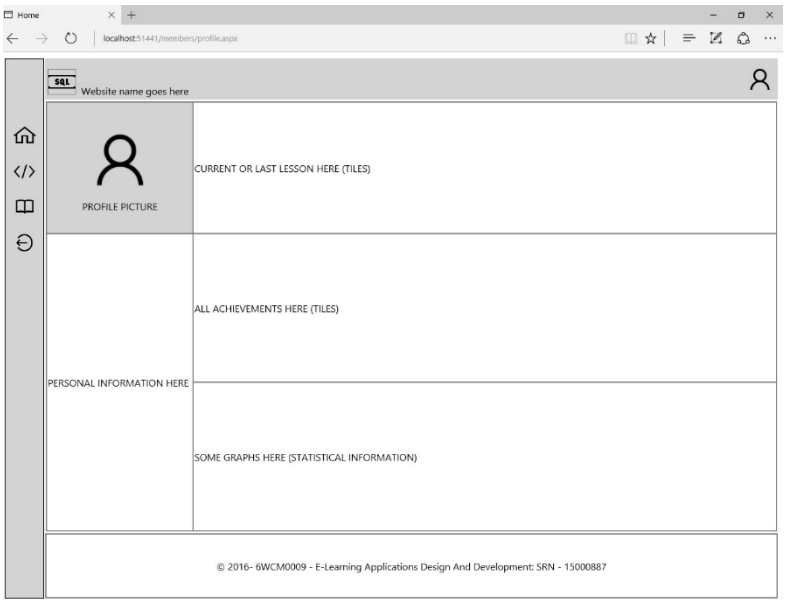
<u>Development Plan</u>

1. Create web form template
2. Create database
3. Plan profile page design
4. Create profile page
5. Create lessons page (Metro UI)
6. Link application to database
7. Add registered users to database
8. Retrieve lesson information
9. Create widget with each tile to display user information
10. Create mission pages

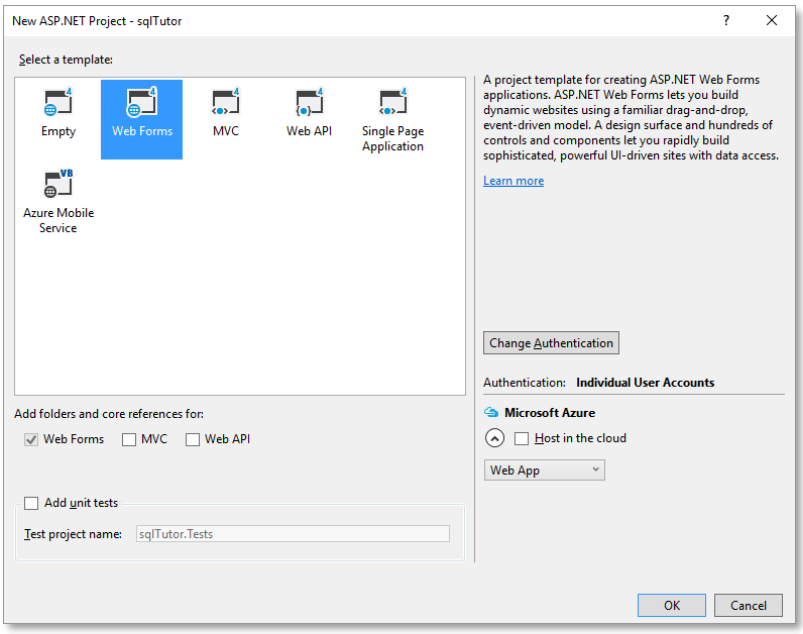
5.1.1 Create web form template

The original design of the prototype application started from a blank web form and included tables. These tables formed the foundation of the main layout. However, as development progressed, too much time was being spent building basic functional requirements. In addition, the table layout did not provide sufficient responsiveness, therefore it was decided to use the default template provided by Visual studio 2015.

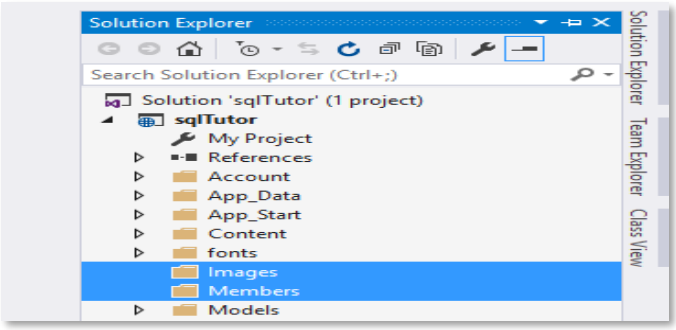
Fig 6 Initial design



The second iteration of the development process started by using a default web forms template provided by Visual Studio 2015.



This design decision was made since the default template provided by visual studio 2015 contained most of the basic functionality that was required. This enabled more time to be spent on presenting the teaching content and less time on functional development. The main framework at this stage of development consisted of one main master page. To add some structure to the project hierarchy, some additional folders were added for member pages and images.



Once this was completed, verification of basic functionality was confirmed within the Microsoft Edge web browser. In evaluating the default template, an email address was used to authenticate the logged in user. Therefore, the Access database Users schema was edited to accept an email address, instead of a user name.

5.2 Create Database

A database was created using Microsoft Access, which was already installed on the workstation used. This database will be responsible for handling the lesson information, in relation to each user.

Microsoft Access was chosen for its ease of use and portability, however in a fully developed application, SQL server will be more suitable for better remote management, backup capability and increased performance.

Fig. 7 Database creation

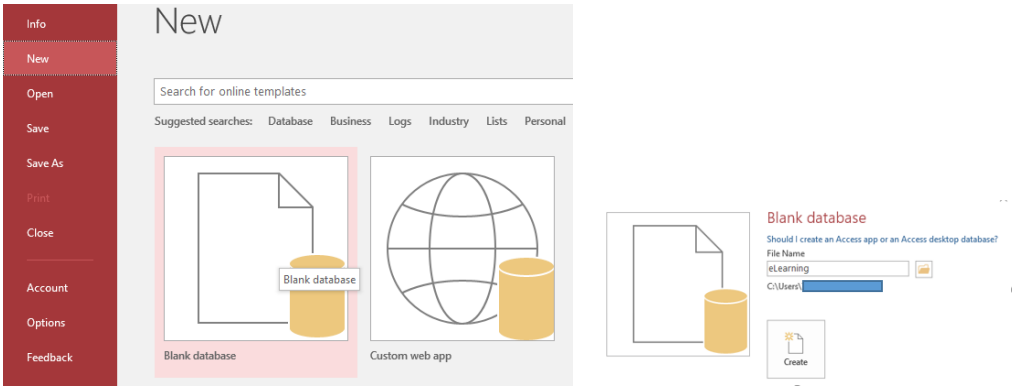


Fig 8. List of tables

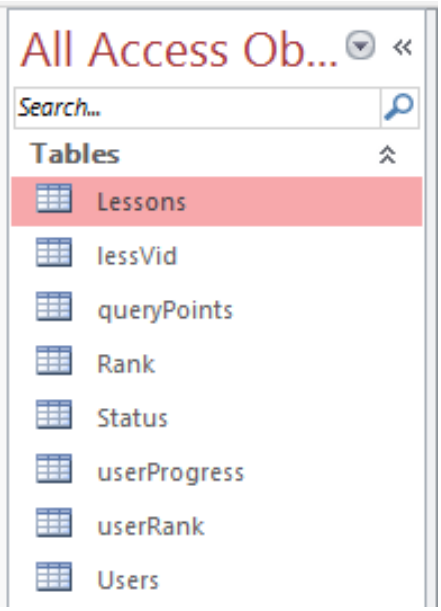


Fig. 9 Initial lessons information

Users | Lessons

ID	lessName	lessDesc	lesRank
7	INTRODUCTION	This is where we start our SQL journey	8
8	SELECT ALL STATEMENT	In this lesson, you will learn how to SELECT all records in a table	4
9	SELECT STATEMENT ONE TABLE	Learn how to SELECT data from one table	4
10	SELECT STATEMENT - MULTIPLE TABLES	Learn how to SELECT data from multiple tables	5
11	CREATE TABLE STATEMENT	Learn how to CREATE a table	1
12	INSERT STATEMENT	Learn how to INSERT a record into a table	2
13	UPDATE STATEMENT	Learn how to UPDATE existing records in a table	3
14	DELETE STATEMENT	Learn how to DELETE a record from a table	3
15	BONUS STAGE	Earn additional rewards	6

5.3 Profile page design

At this stage, all the pages required for the web application were created, starting with the user profile page, which was created using the already existing "/Account/Manage.aspx page".

Based on some initial research using Google and YouTube, the flex box CSS function seemed capable of providing the design required (YouTube, 2016).

5.4 – 5.5 Create profile page, Create lessons page (Metro UI)

The flexbox function was implemented by creating the following CSS code:

```
.parent{
        font-family:'Segoe UI';
        font-weight:600;
        display:flex;
        height:calc(100vh - 350px);
        flex-wrap:wrap;
        color:white;

}

.item{
        width:20%;
        height:50%;
        background-color:deepskyblue;
        margin:5px;
        text-align:center;
        vertical-align:central;
}
```

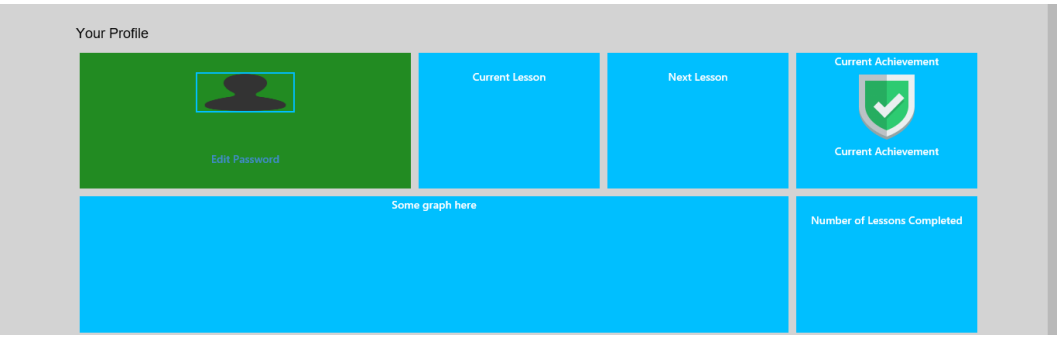
The class "parent" is associated with the div element for the entire page. The class "item" is associated with a div element, representing a tile. The dimensions, background color, margin and alignment were defined, as well as the font-family Srgoe UI, which is used by windows 10.

Individual div elements were manipulated in CSS to achieve the desired result.

```
/*Adjust the size of individual Tiles*/

.item:nth-child(1){

    background-color:forestgreen;
    flex-grow:2}

.item:nth-child(5){
    flex-grow:1;
    background:white;
}
```

Result

Additional controls were added to the page to facilitate the functionality, based on the design plan. This included labels, image boxes and another div element to hold the profile picture.



The same concept was applied to the lessons page to achieve the desired result. Div. elements were created, and the parent div was given an ID of “content” and it’s display property was set to Flex.

CSS snippet

```
#content {
    text-decoration: none;
    list-style: none;
    display: flex;
    /*Flexbox*/
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
}
```

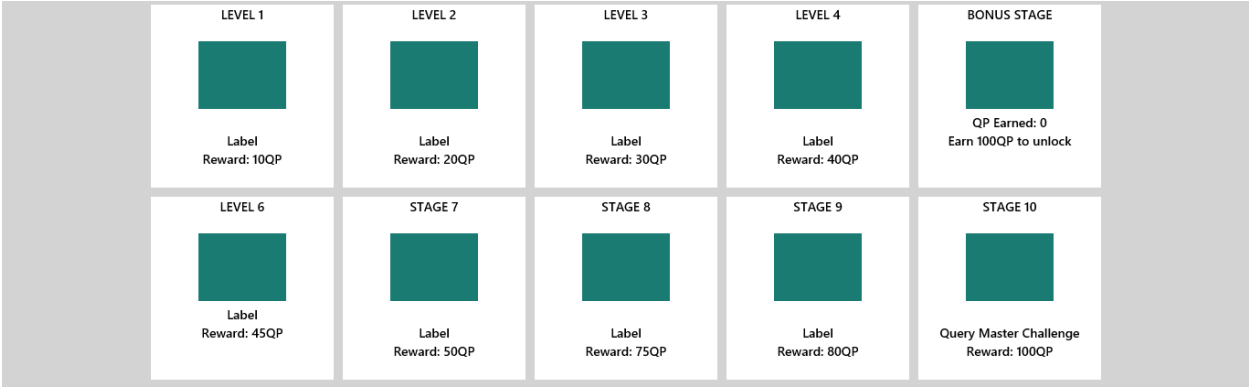
```
/*Lesson Tiles*/
.tile {
    flex-direction: row;
    width: 200px;
    height: 200px;
    margin: 5px;
    background-color: white;
    vertical-align: middle;
    text-align: center;
}
```

HTML snippet

```
<div id="content">
    <div class="tile">
    </div>
    <div class="tile">
    </div>
    <div class="tile">
    </div>
    <div class="tile">
    </div>
```

Result

Additional asp controls were added to define each tile according to the design plan.



5.6 Link application to database

A separate class named “connections” was created to handle the communication to the database. This decision was made by applying the programming principle “Seperation of conserns”. Therefore any problems connecting to the database can be easily diagnosed and all code logic can be centrally updated.

Declarations:

```
Public Class connections
    Dim conn As New OleDb.OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\watcht
    Dim cmd As New OleDb.OleDbCommand
    Public da As OleDb.OleDbDataAdapter
    Public ds As DataSet
    Public params As New List(Of OleDb.OleDbParameter)
    Public count As Integer 'To capture the amount of records for statistics
```

Query Method:

```
Public Sub queryData(Query As String) 'Method to query the database
    Try
        conn.Open() 'Opens the connection
        cmd = New OleDb.OleDbCommand(Query, conn) 'Defines the SQL command (using the Oledb connection and the Query String)
        params.ForEach(Sub(x) cmd.Parameters.Add(x)) 'Adds the parameters of the query
        params.Clear() 'Clears the parameters
        ds = New DataSet 'Instance of a new Dataset
        da = New OleDb.OleDbDataAdapter(cmd) 'Instance of a new DataAdapter
        count = da.Fill(ds) 'Retrieves the result count and passes the value to the count variable (For statistical use)
        conn.Close() 'Closes the connection
    Catch ex As Exception
        'Handles any exceptions
    End Try

    If conn.State = ConnectionState.Open Then
        conn.Close() 'Ensures that the connection is closed after method execution
    End If

End Sub
```

The count variable is used to retrieve the numerical amount of the query result. For example, if the query returns 5 records, the count variable will be set to 5.

5.7 Add registered users to database

The above logic was then adapted to create a method for handling user registration. The Microsoft template created a separate ADO.NET database for login, however this connection made a seperate entry into the Access database. The default value of 7 was passed as a one parameter, corresponding with the default rank of “New SQL Learner”. The email address of the user was used as a second parameter.

```
Public Sub register(ByVal email As String) 'Method to insert registered user into the database.

    cmd.Connection = conn 'assigns the OleDb connection string as the OledbCommand connection
    Try
        conn.Open() 'Opens the connection
        If conn.State = ConnectionState.Open Then 'If the connection is open
            cmd.CommandText = "INSERT INTO Users (rankId, email) VALUES(7, '" & email & "')" 'Defines SQL Query
            cmd.ExecuteNonQuery() 'Executes SQL Query
            conn.Close() 'Closes the connection
        End If
    Catch ex As Exception
        'Handles any exceptions
    End Try

    If conn.State = ConnectionState.Open Then
        conn.Close() 'Ensures that the connection is closed after method execution
    End If

End Sub
```

Register implementation – Register.aspx page:

```
Dim c As New connections
c.register(Email.Text) 'Email.Text = Value from the Email textBox
'*****************************************************
```

The queryData method was implemented within the lessons.aspx page by declaring an instance of the connections class to a variable named queryConn. A variable named userId was also declared to hold the user id of the logged in user, retrieved from the database.

Declarations:

```
Imports Microsoft.AspNet.Identity

1 reference
Public Class lessons
    Inherits System.Web.UI.Page
    Dim queryConn As New connections
    Dim userId As Integer
```

Method:

```
Private Sub loadLesson1()

    Try
        queryConn.queryData("SELECT * FROM Lessons WHERE ID = " & 7)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbLess1.Text = r("lessName")
            imgLess1.ToolTip = r("lessDesc")
        Next
    Catch ex As Exception

    End Try

End Sub
```

5.8 Retrieve lesson information

With the above illustrated code, all lesson information associated with the specified lesson ID can be retrieved and the values assigned to the controls. In the example shown below, the SQL query retrieves all the information for the lesson with ID 7. The lesson name value is assigned to a label and the lesson description is assigned to the image tool tip. The method was then placed in the Page Load event.

```
0 references
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    queryConn.queryData("SELECT UserId FROM Users WHERE Email = '" & User.Identity.GetUserName() & "'")
    For Each r As DataRow In queryConn.ds.Tables(0).Rows
        userId = r("userId")
    Next
    loadLesson1()
End Sub
```

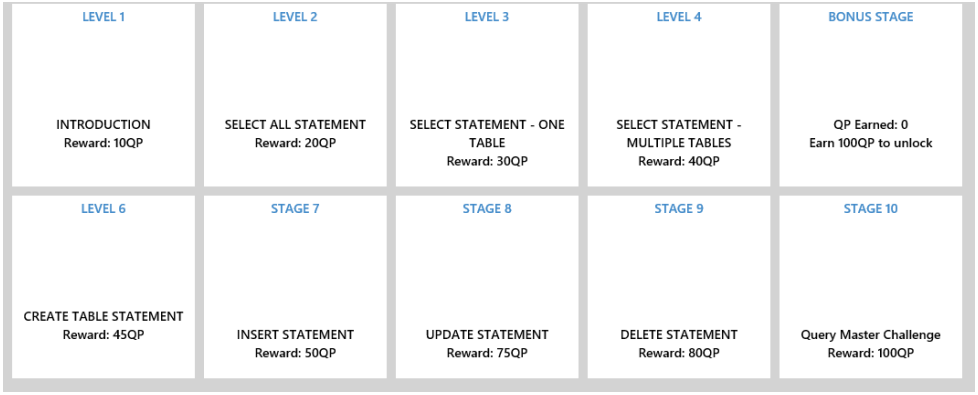
Result:

The same concept was applied to the remaining tiles, creating a separate method for each lesson, changing the ID value to the desired lesson ID. For example, the loadLesson2 method was created using lesson ID 8 and placed in the Page Load event. This concept was applied for the remaining lessons, creating a new method and changing the lesson ID.

```
0 references
Private Sub loadLesson2()
    Try
        queryConn.queryData("SELECT * FROM Lessons WHERE ID = 8")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbLess2.Text = r("lessName")
            imgLess2.ToolTip = r("lessDesc")
        Next
    Catch ex As Exception
    End Try
End Sub
```

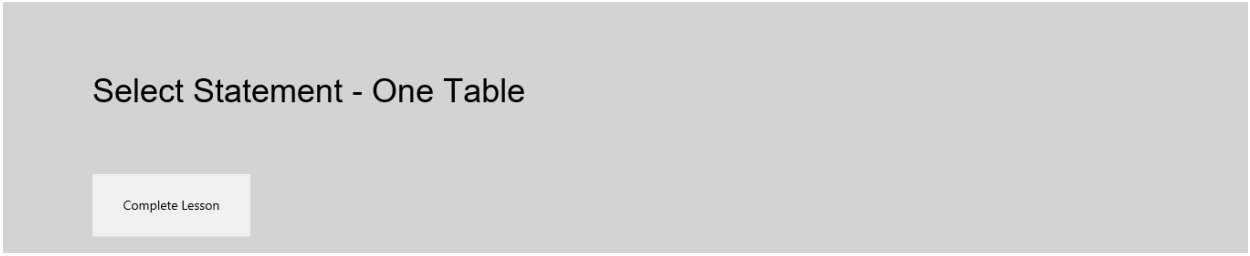
```
0 references
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    queryConn.queryData("SELECT UserId FROM Users WHERE Email = '" & User.Identity.GetUserName() & "'")
    For Each r As DataRow In queryConn.ds.Tables(0).Rows
        userId = r("userId")
    Next
    loadLesson1()
    loadLesson2()
End Sub
```

Result:



Completing a lesson:

The pages for each section was created with a button to simulate the completion of a lesson.



As an additional separation of concern, a class named “record” was created to handle updating the user information. The first functionality created was designed to insert a record into the database, on the Complete Lesson button click event. In order to easily track which lessons were completed, a Complete table was added to the database. This made retrieving each lesson a lot simpler, instead of querying the lesson status based on the original design.

```
'Accepts two parameters and updates the 'Complete' table
'@user The userId, @lesson The lessonId
8 references
Public Sub completeLesson(ByVal user As Integer, ByVal lesson As Integer)
    Try
        queryConn.queryData("SELECT * FROM complete WHERE userId =" & user & " AND lessonId = " & lesson)
        If queryConn.count = 0 Then 'Only if the query does not return any results
            queryConn.queryData("INSERT into complete (userId, lessonId) VALUES (" & user & ", " & lesson & ")") 'Insert a new record with the supplie
        End If
    Catch ex As Exception

    End Try
End Sub
```

The objective of this method is to update the complete table, only if the lesson has not yet been completed, by referencing queryConn.count. The code was then implemented by using the following:

```
Public Class lesson1
    Inherits System.Web.UI.Page
    Dim queryConn As New connections 'New instance of the connections Class
    Dim r As New record 'New instance of the record Class
    Dim userId As Integer 'Variable to store the user id
    Dim lessId As Integer 'Variable to store the lesson id
```

The same logic was then applied to the click event on each lesson page and tested for functionality. An additional method was created to redirect the user to the lessons page after completion.

```
Public Sub redirect()
    Response.Redirect("/Members/lessons.aspx")
End Sub
```

```
0 references
Protected Sub btnComplete_Click(sender As Object, e As EventArgs) Handles btnComplete.Click
    r.completeLesson(userId, lessId)
    redirect()
End Sub
```

Updating the user progress:

The design of each lesson was intended to track the progress within each lesson. According to the user requirements, if a user navigates away from a lesson before completion, they are able to return to that point, within the lesson. To achieve this, each lesson was divided into sequences associated with a specific point in the lesson.

The method was created using the connections class, standardizing the variable name queryConn.

```
Public Sub updateLesson(ByVal less As Integer, ByVal user As String, ByVal progress As Integer)
    Try
        queryConn.queryData("SELECT * from userProgress WHERE userID = " & user) 'Queries the userProgress table using the userId as a parameter
        If queryConn.count = 0 Then 'If no results are found
            'Inserts a new record with the supplied parameters
            queryConn.queryData("INSERT INTO userProgress (userId, lessonId,
            lessonStatus, seqId) VALUES (" & user & "," & less & ", 1, " & progress & ")")
        Else
            'If not, updates the user record with the supplied parameters
            queryConn.queryData("UPDATE userProgress SET lessonId = " & less & ",
            lessonStatus = 1, seqId = " & progress & " WHERE userId = " & user)
        End If

    Catch ex As Exception
    End Try
End Sub
```

Query points implementation:

The query points system was easily implemented with the use of the connections class, creating a method with the desired query.

```
Public Sub addQp(ByVal id As Integer, ByVal qp As Integer)
    Try
        queryConn.queryData("INSERT INTO queryPoints (userId, queryPoints) VALUES (" & id & "," & qp & ")")
    Catch ex As Exception

    End Try
End Sub
```

The above method accepts the user ID and query point amount as parameters and inserts the record into the queryPoints table.

```
  Protected Sub btnTask1_Click(sender As Object, e As EventArgs) Handles btnNext.Click
      r.addQp(userId, 2)
  End Sub
```

However, an issue was encountered whereby the queryPoints table was updated each time the method was called, even though the lesson was already completed.

To resolve this issue, a pre-condition was developed to first check if a lesson existed in the complete table. The result of the pre-condition was then passed to a Boolean variable. With that done, the Boolean value can now be checked before updating the queryPoints table.

Declaration in the record class:

```
Public Class record
    'This class holds the methods to update the Lesson and User records.
    Dim queryConn As New connections
    Public complete As Boolean
```

The getComplete method:

```
Public Sub getComplete(ByVal id As Integer, ByVal lessid As Integer)
    Try
        queryConn.queryData("SELECT * FROM complete WHERE userId = " & id & "AND lessonId = " & lessid)
        If queryConn.count = 0 Then
            complete = False
        Else
            complete = True
        End If
    Catch ex As Exception
    End Try

End Sub
```

Implementation of getComplete method:

```
Private Sub introduction_Load(sender As Object, e As EventArgs) Handles Me.Load

    Try
        queryConn.queryData("SELECT Users.userId, userProgress.lessonId, seqId
                             From Users, userProgress
                             Where (Users.email = '" & User.Identity.GetUserName & "') AND (userProgress.lessonId = 7)")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            userId = r("userId")
            lessId = r("lessonId")
            seq = r("seqId")
        Next
    Catch ex As Exception
    End Try
    r.getComplete(userId, lessId) 'Calls a method of the record class to check if the lesson was already completed
    If r.complete = True Then
        complete = True 'Sets the complete variable to true if Method returns true otherwise, the value is false
    End If
End Sub
```

The getComplete method was implemented in the page load event, passing the result to a local Boolean variable named “complete”. The value of this variable is then checked before updating the queryPoints table.

```
Protected Sub btnTask1_Click(sender As Object, e As EventArgs) Handles btnNext.Click
    If complete = False Then
        r.updateQp(userId, 4)
    End If
End Sub
```

5.9 Create widget with each tile to display user information

The main concept of this process is to query all the user specific data, and pass the values to the controls on the profile page. By utilizing the connections class, a method called getLessons was developed to query the database, using the logged in user as the identifier.

```
Private Sub getLessons()
    Try
        queryConn.queryData("SELECT Lessons.lessName,Lessons.lessDesc,Status.lessStatus, Lessons.ID, Users.email,
                             queryPoints.queryPoints, Lessons.lesRank, Rank.rn
                             FROM Users, Lessons, userProgress , Status, queryPoints, Rank
                             WHERE (Users.email = '" & User.Identity.GetUserName & "' AND
                             UserProgress.lessonId = Lessons.ID AND
                             Lessons.lesRank = Rank.ID
                             AND userProgress.lessonStatus = Status.ID AND Users.userId = queryPoints.userId )") 'SQL Query
        If queryConn.count = 0 Then 'If no result is found.
            newUser = True
            btnContinue.Text = "Start Learning SQL"
            lbNextLesson.Text = "You have not started any lessons yet"
        Else
            For Each r As DataRow In queryConn.ds.Tables(0).Rows
                lbCurrentName.Text = r("lessName") 'Sets the results of the query as the values for the respective controls
                lbCurrentDesc.Text = r("lessDesc")
                lbCurrentStatus.Text = r("lessStatus")
                lbQp.Text = r("queryPoints")
                lbRank.Text = r("rn")
                lbEmail.Text = "Email: " & r("email")
                nextLess = r("ID") + 1 'Retrieves the ID from the current lesson, adds 1 and sets the value to the nextLess variable
            Next
        End If
    Catch ex As Exception

    End Try

End Sub
```

To the determine the next lesson, the value of one is added to the current lesson ID and passed to a local Integer variable named nextLess. The value of this variable is then used by the getNextLesson method as a parameter, to retrieve the associated information with an SQL query.

```
Private Sub getNextLesson()
    Try
        queryConn.queryData("SELECT lessName, lessDesc, pageName FROM Lessons WHERE ID = " & nextLess)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbNextLesson.Text = r("lessName")
            lbNextDesc.Text = r("lessDesc")
            nextLess = 0
            navPage = r("pageName")
        Next
    Catch ex As Exception

    End Try
End Sub
```

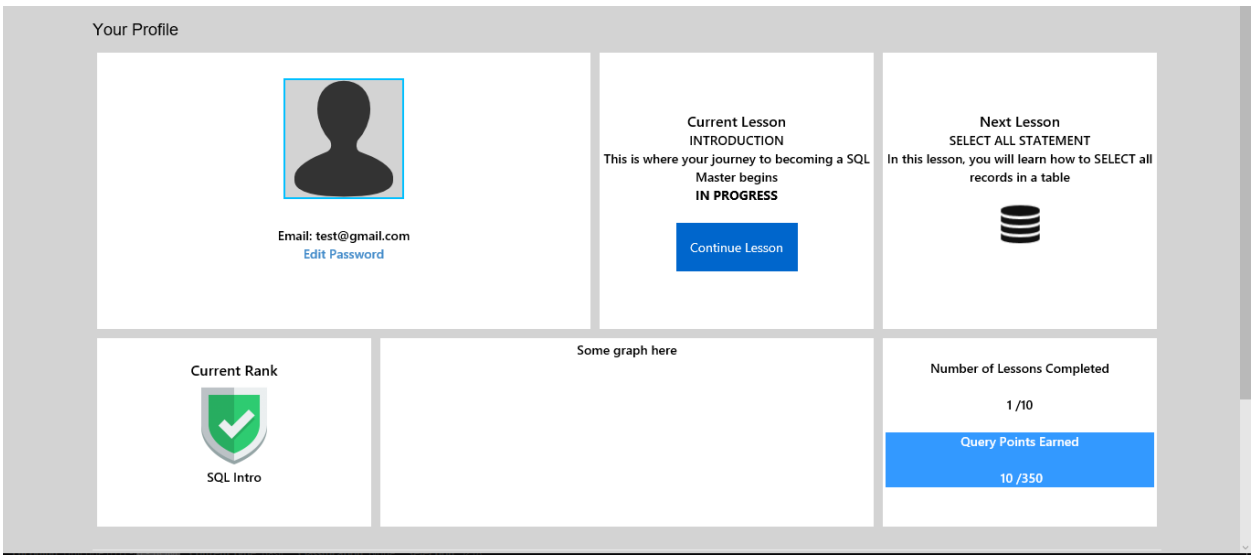
The total amount of completed lessons was determined by querying the complete table. The amount of records within the complete table, associated with the logged on user is passed to a label.

```
Private Sub getCount()
    Try
        queryConn.queryData("SELECT * FROM complete, Users WHERE Users.email = '" & User.Identity.GetUserName & "' AND Users.userId = complete.userId")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbAmount.Text = queryConn.count
        Next
    Catch ex As Exception

    End Try
End Sub
```

All three methods were then placed in the page load event.

```
Protected Sub Page_Load() Handles Me.Load
    Dim manager = Context.GetOwinContext().GetUserManager(Of ApplicationUserManager)()

    HasPhoneNumber = String.IsNullOrEmpty(manager.GetPhoneNumber(User.Identity.GetUserId()))

    ' Enable this after setting up two-factor authentientication
    'PhoneNumber.Text = If(manager.GetPhoneNumber(User.Identity.GetUserId()), String.Empty)

    TwoFactorEnabled = manager.GetTwoFactorEnabled(User.Identity.GetUserId())

    lbEmail.Text = User.Identity.GetUserName
    LoginsCount = manager.GetLogins(User.Identity.GetUserId()).Count

    Dim authenticationManager = HttpContext.Current.GetOwinContext().Authentication

    getCount()
    getLessons()
    getNextLesson()


End Sub
```

With some additional CSS styling, the following result was achieved in the profile page.



Lock\Unlock:

This functionality was achieved by disabling the image button for each lesson. Then, the loadLesson method was edited to query the complete table. Once the previous lesson was found in the complete table, the next lesson image button and image will be changed and enabled.

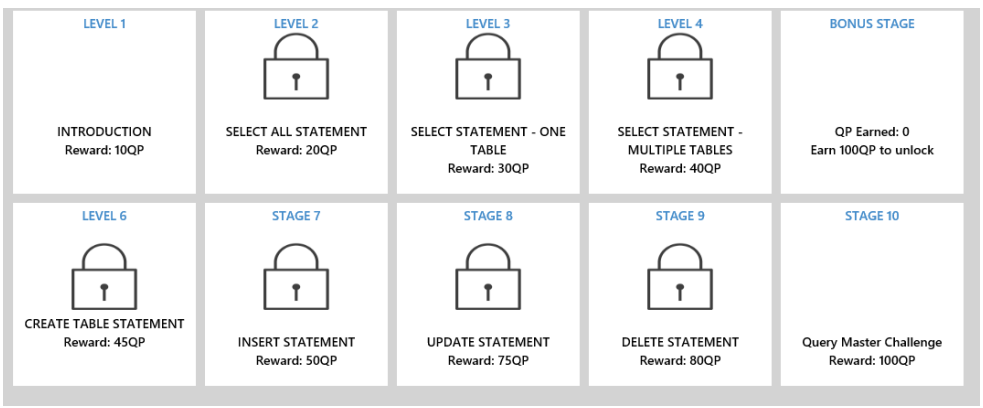
Declarations:

```
Public Class lessons
    Inherits System.Web.UI.Page
    Dim queryConn As New connections
    Dim userId As Integer
    Dim imgUrl2 As String = "~/Images/locked-icon.png"
    Dim imgUrl As String = "~/Images/database.png"
    Dim imgUrl3 As String = "~/Images/star.png"
```

Code:

```
Private Sub loadLesson2()
    Dim l As Integer = 7
    Try
        queryConn.queryData("SELECT * FROM Lessons WHERE ID = " & l + 1)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbLess2.Text = r("lessName")
            imgLess2.ToolTip = r("lessDesc")
            imgLess2.ImageUrl = imgUrl2
        Next
        queryConn.queryData("SELECT * FROM complete WHERE lessonId = " & l)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            If queryConn.count > 0 Then
                imgLess2.ImageUrl = imgUrl

                imgLess2.Enabled = True
            End If
        Next
    Catch ex As Exception
    End Try

End Sub
```

Result:



After duplicating the code to retrieve all the relevant lesson information for each lesson, the class became very inefficient. This became evident as the same algorithm appeared multiple times with a few minor variables. Although performance did not seem to be affected, it definitely made code management difficult.

As a result, a more innovative and efficient method of achieving the same result was conceptualized. The following code is the result:

```
'Accepts an integer (lesson ID), label (Lesson Label) and image (Lesson Image) as parameters
'Queries the Lessons table with the lesson ID and passes the lesson information to the label and image
'Queries the complete table and unlocks (Enables controls) the next lesson if the previous lesson ID is found
7 references
Public Sub populate(ByVal l As Integer, ByVal lb As Label, ByVal img As Image)
    Try
        queryConn.queryData("SELECT * FROM Lessons WHERE ID = " & l + 1)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lb.Text = r("lessName")
            img.ToolTip = r("lessDesc")
            img.ImageUrl = imgUrl2
        Next
        queryConn.queryData("SELECT * FROM complete WHERE lessonId = " & l & " AND userId = " & userId)

        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            If queryConn.count > 0 Then
                img.ImageUrl = imgUrl
                img.Enabled = True
            End If
        Next
    Catch ex As Exception

    End Try
End Sub
```

The populate method accepts an Integer, label and image as parameters, which is substituted into the loadLesson algorithm. Therefore, the previously repeated code only had to be defined once.

Implementation of the populate method:

```
'Loads all the lessons
0 references
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    'If the user is anonymous
    If User.Identity.GetUserName = "" Then
        Response.Redirect("/loggedOut.aspx") 'Redirects anonymous users to the loggedOut page
    Else
        queryConn.queryData("SELECT UserId FROM Users WHERE Email = '" & User.Identity.GetUserName() & "'")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            userId = r("userId") 'Sets the user id to the userId variable
        Next
        loadLesson1() 'calls the loadLesson method
        'Calls the populate method to load each lesson, reuses the code
        populate(7, lbLess2, imgLess2)
        populate(8, lbLess3, imgLess3)
        populate(9, lbLess4, imgLess4)
        populate(10, lbLess5, imgLess5)
        populate(11, lbLess6, imgLess6)
        populate(12, lbLess7, imgLess7)
        populate(13, lbLess8, imgLess8)
    End If

End Sub
```

The progress bar:

A progress bar was added to the lessons page, to represent the overall completion progress of the user. This was achieved by a Div. element inside another Div. element (YouTube 2016).

```
<div id ="outer">

<div id="inner">

</div>

</div>
```

Each Div. element was then defined using CSS.

```
#outer {
    width:100%;
     height: 20px;
     background-color: gray;
}

#inner {
     width: 0px;
     height: 100%;
     background-color: blue;
     color: white;
     text-align: center;
}
```

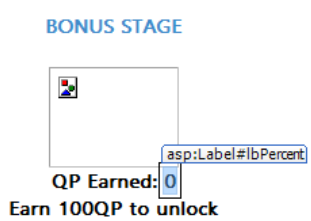
Now by implementing some client side javaScript, the inner Div. width was animated, in relation to the percentage of query points earned. However, this value had to be expressed as a percentage of the total amount of the query points earned.

To achieve this, a method was created to retrieve the amount of query points associated with the user.

```
Private Sub getQp()
    Try
        queryConn.queryData("SELECT queryPoints FROM queryPoints WHERE userId = " & userId)
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbPercent.Text = r("queryPoints")
        Next
    Catch ex As Exception

    End Try

End Sub
```

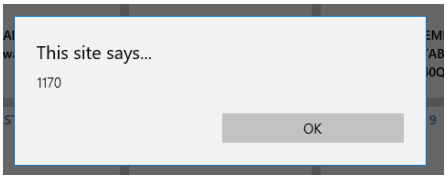
This method was then placed in the page load event.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If User.Identity.GetUserName = "" Then
        Response.Redirect("/loggedOut.aspx")
    Else
        queryConn.queryData("SELECT UserId FROM Users WHERE Email = '" & User.Identity.GetUserName() & "'")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            userId = r("userId")
        Next
        loadLesson1()
        loadLesson2()
        loadLesson3()
        loadLesson4()
        loadLesson5()
        loadLesson6()
        loadLesson7()
        loadLesson8()
    End If

    getQp()
End Sub
```



JavaScript

As a diagnostic, an alert was placed in the javaScript to display the current outer Div. width. This value was then used to equate the desired value on a calculator. The following was the result.

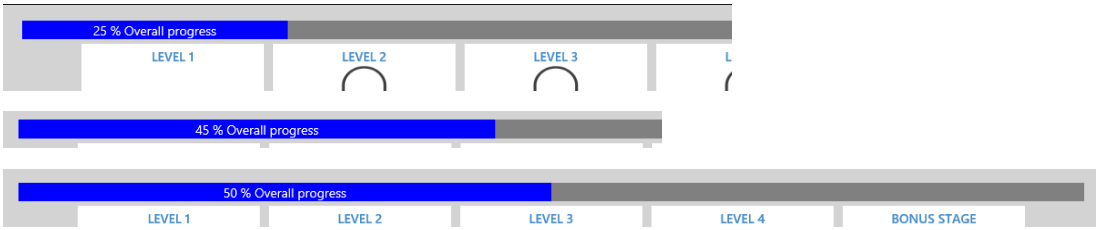


The amount displayed was 1170 pixels. This value was then equated into the calculation, which resulted in the following javaScript. The value of the label containing the amount of query points was retrieved by using the getElementById function, within the javaScript (Stackoverflow.com, 2016).

```
<script type="text/javascript">
    $(document).ready(function () {
        amount = document.getElementById('<%= lbPercent.ClientID %>').innerHTML
        animateProgressBar(amount);
        function animateProgressBar(p)
        {
            val = p / 350 * 100 /*points gained / total points * 100 to display text as a percentage of the total points*/
            oW = $("#outer").width();/*The current with of the outer Div element of the progress bar*/
            alert(oW)
            $('#inner').animate({
                'width': (val / 100) * oW
            }, 3000);

            $({ counter: 1 }).animate({ counter: val }, {
                duration: 3000,
                step : function()
                {
                    $('#inner').text(Math.ceil(this.counter) + ' % Overall progress');
                }
            })
        }
    });

</script>
```

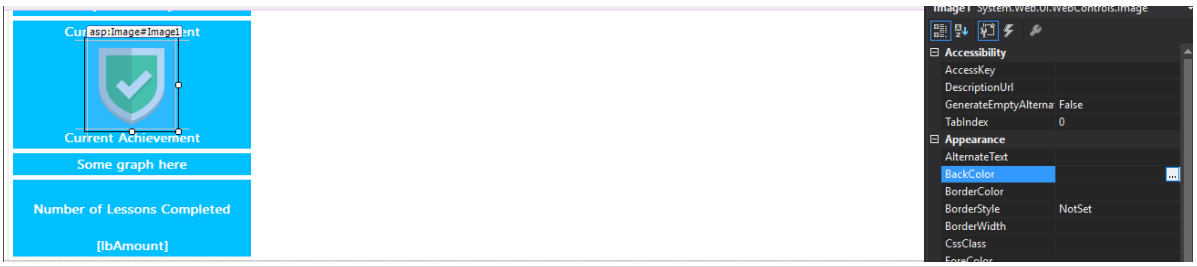
A manual entry was made into the queryPoints table to test the functionality of the progress bar. Inserting 175 in the queryPoints column should produce a 50% progress bar, which was tested and proved to be successful.

The Progress bar animation:

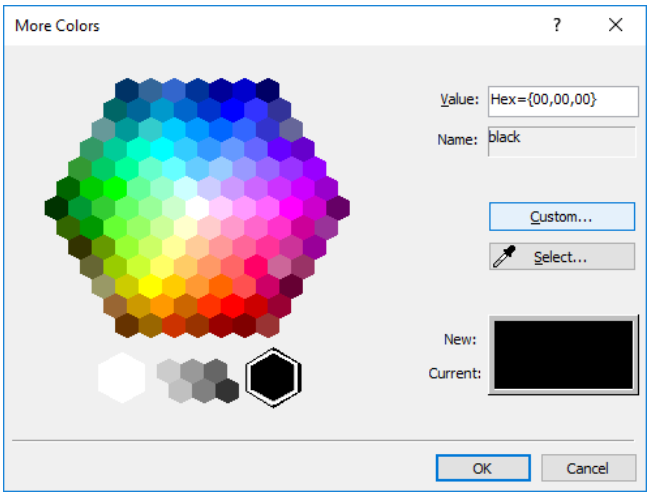


Styling:

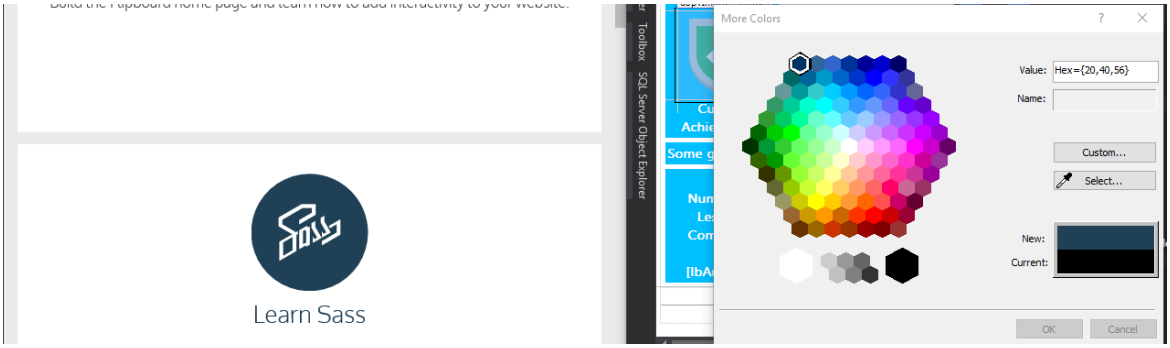
Using the following technique, colors were grabbed from the specific websites to form a color scheme



The back color property of a random control was selected.



The select tool was used then to grab colors from appealing websites (codeCademy), which was applied to the tiles.



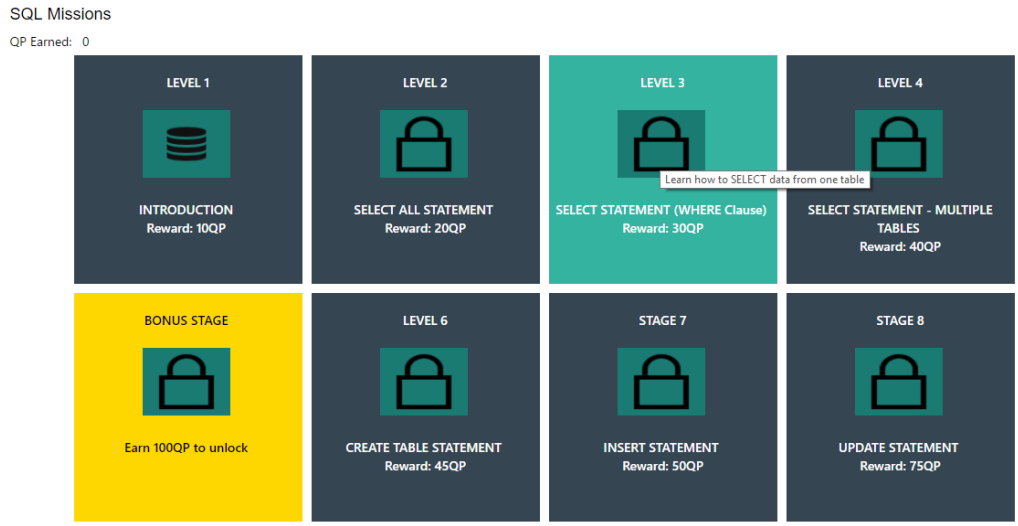
The hex value was then used in the CSS sheet for the associated element.

```
.tile {
    flex-direction: row;
    width: 300px;
    height: 300px;
    margin: 5px;
    background-color: #354551;
    vertical-align: middle;
    text-align: center;
    color: white;
    font-weight: 600
}
```

Example: center

Final lesson page result with hover effect.

```
.tile:hover {
    background-color: #34B3A0;
    color: white;
}
```



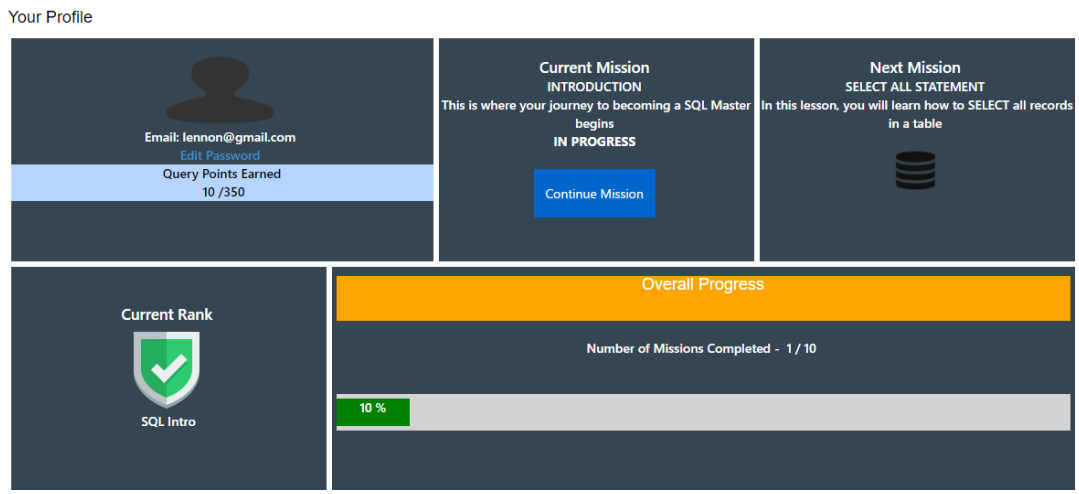
To complete the profile page, a design decision was made to relocate the progress bar and use the completed missions to calculate user progress, instead of the query points. This made future development easier, in the event that the reward scheme was changed.

This was accomplished by editing the JavaScript to retrieve the completed missions value for the progress calculation.

Code:

```
<script>
    $(document).ready(function () {
        amount = document.getElementById('<%= lbAmount.ClientID %>').innerHTML
        total = document.getElementById('<%= lbTotal.ClientID %>').innerHTML
        highlight();
        animateProgressBar(amount);
        function animateProgressBar(p)
        {
            val = p / total * 100 /*lessons completed / total lessons * 100 to display text as a percentage of the total lessons*/

            oW = $("#outer").width();/*The current with of the outer Div element of the progress bar*/
            /**alert(val) Used to verify the width of the outer div element**/
            $('#inner').animate({
                'width': (val * oW) / 100

            }, 3000);


            $({ counter: 1 }).animate({ counter: val }, {

                duration: 3000,
                step : function()
                {
                    $('#inner').text(Math.ceil(this.counter) + ' %');
                }
            })
        }
```

Using a consistent styling in accordance with the non-functional requirements, the following result was achieved.



5.10 Create mission pages

Using the design plan the layout of the lessons area was developed by following the same flexbox principles. Three DIV elements were created within a parent DIV element and the parent DIV's display property was set to Flex.

```
/****************************Lesson page Layout style*******************************************/
#lessonBody {
    display: flex;
    justify-content: center;
    padding-top:100px;
}
```

The flex direction of the contents was set to row. Additional div elements were then added, according to the design plan.

Loading lesson data:

The connections class was used to retrieve the necessary lesson information, which was stored in the database. Using the already defined queryConn method, the relevant lesson information was retrieved, using the query in the page load event.

```
otected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load


 Try
     queryConn.queryData("SELECT Lessons.ans1, Lessons.ans2, Lessons.ques1, Lessons.ques2, userProgress.seqId, Users.userId
                          FROM (userProgress INNER JOIN
                          Users ON userProgress.userId = Users.userId), Lessons
                          WHERE (Users.email = '" & User.Identity.GetUserName & "') AND (Lessons.ID = " & lesson & ")")
     For Each r As DataRow In queryConn.ds.Tables(0).Rows
         userId = r("userId") 'User ID
         lessId = lesson 'The lesson ID
         seq = r("seqId") 'The lesson sequence ID
         ques1 = r("ques1") 'The first question
         ques2 = r("ques2") 'The second question
         ANS1 = r("ans1") 'The first answer
         ANS2 = r("ans2") 'The second answer
     Next
 Catch ex As Exception
 End Try
```

The result of the query was then passed to locally declared variables, to be used in the logic for the rest of the lesson. Separate methods were created for each function, to ensure that each concerns are isolated and methods are restricted to one purpose.

The following outlines the pseudocode used for the lesson execution and the associated VB code.

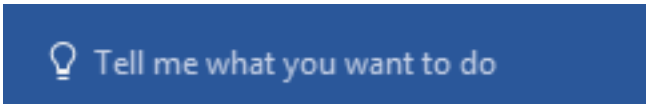
The first task is loaded with the page. If the user input matched the value of ans1 retrieved from the database, a feedback message is displayed and the database is updated with the associated sequence.

```
'executes a specific method, based on the lbTask text
1 reference
Public Sub loadLesson()
    If lbTask.Text = task1 Then
        execute1() 'Calls the execute1 method

    ElseIf lbTask.Text = task2 Then
        execute2() 'Calls the execute2 method

    End If
End Sub
```

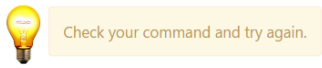
Execute1 matches the user input with the ans1 value, retrieved from the database. If the result is false, an image, along with an error message is displayed to the user.

If the result is true, a success message is displayed. Execute2 follows the same logic, but instead matches the user input with the ans2 variable.

For the feedback mechanism required in the user requirements, an image was used to represent the object displaying a message. This feature was conceptualized using the help image within Microsoft products.



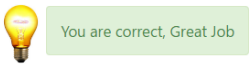
(Microsoft Office 2016)



(image by noBacks.com)

To customize the messages, VB.NET code was used to change the CSS class of the label (lbResult) displaying the message.

```
2 references
Public Sub correctClass()
    lbResult.Attributes.Add("class", "alert alert-success")
End Sub

2 references
Public Sub wrongClass()
    lbResult.Attributes.Add("class", "alert alert-warning")
End Sub
```



The image is also changed whenever a message is not displayed.

```
3 references
Private Sub bOff()
    imgBulb.ImageUrl = bulbOff
End Sub

'Changes the source image of the bulb to the bulbOff variable value
4 references
Public Sub bOn()
    imgBulb.ImageUrl = bulbOn
End Sub
```

```
Dim bulbOn As String = "~/Images/bulb.jpg" 'Bulb image illuminated
Dim bulbOff As String = "~/Images/bulbOff.jpg" 'Bulb image not illuminated
```

The basis of each lesson execution was based on manipulating the visibility of controls, when certain buttons are clicked. The change in color was achieved by editing the image in Photoshop Express.

```
reference
ublic Sub execute1()
    lbResult.Visible = True 'Sets the bulb message text to visible
    If txtRunSql.Text = ANS1 Then 'If the user's entered text and the answer1 matches
        panRun.Visible = False 'Sets the visibility of the run sql panel to false
        correctClass() 'Calls the correctClass method
        lbResult.Text = feedback 'Shows the feedback message
        bOn() 'Calls the bOn method which changes the bulb image
        txtRunSql.Text = "" 'Resets the txtRunSql control text to Blank
        btnNext.Visible = True 'Enables the btnNext control
        btnNext.BackColor = Drawing.Color.Blue 'Changes the btnNext control background to blue
        panLess1.Visible = True 'Sets the visibility of the lesson 1 panel to true (Displays the result table)
        imgCorrect.Visible = True 'Displays the correct image (Check)
        lbPercent.Text = 32 'Sets the percentage complete to 16
        r.updateLesson(lesson, userId, 32) 'Calls the updateLesson method from the record class, updates the database with the

    Else
        wrongClass() 'Calls the wrongClass method
        lbResult.Text = error1 'Displays the error message label
        bOn() 'calls the bOn method
        btnNext.Visible = False 'Changes the visibility of the btnNext button
        txtRunSql.Text = "" 'Clears the text of the txtRunsql text box
        addWrong() 'Calls the addWrong method
    End If
```

Tracking an incorrect answer:

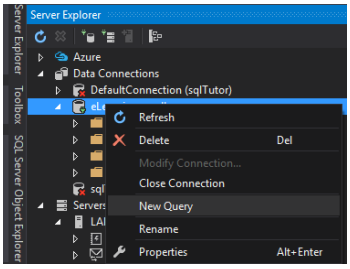
A method was developed to increase the value of a hidden field control by 1. Based on the hidden field value, pre-defined messages can be displayed.

```
'Tracks the amount of incorrect answers and displays a message at specific count
'Passes the count to a Hidden Field
2 references
Public Sub addWrong()
    wrongAns.Value += 1 'Adds 1 to the hidden field value
    If wrongAns.Value = 2 Then
        lbResult.Text = feedBack2 'Shows the feedback2 text if the hidden field value is 2
    ElseIf wrongAns.Value = 3 Then
        lbResult.Text = feedBack3 'Shows the feedback3 text if the hidden field value is 3
    ElseIf wrongAns.Value > 4 And lbTask.Text = task1 Then
        lbResult.Text = help1 'Shows the help message if the hidden field value is 4
        txtRunSql.Text = "" 'Clears the sql text box
        txtRunSql.Text = ANS1 'Places the answer in the txtRunSql textbox
        txtRunSql.BackColor = Drawing.Color.DeepSkyBlue 'Changes the sql textbox background color
        wrongAns.Value = 1 'Resets the hidden field value to 1
    ElseIf wrongAns.Value > 4 And lbTask.Text = task2 Then 'If during task 2 and the hidden field value is 4
        lbResult.Text = help 'Shows the help message
        btnShowAns.Visible = True 'Makes the show answer button visible
        btnRun.Visible = False 'Makes the run sql button visible
        lbAnswer.ForeColor = Drawing.Color.DeepSkyBlue 'Changes the answer label color
    End If
End Sub
```

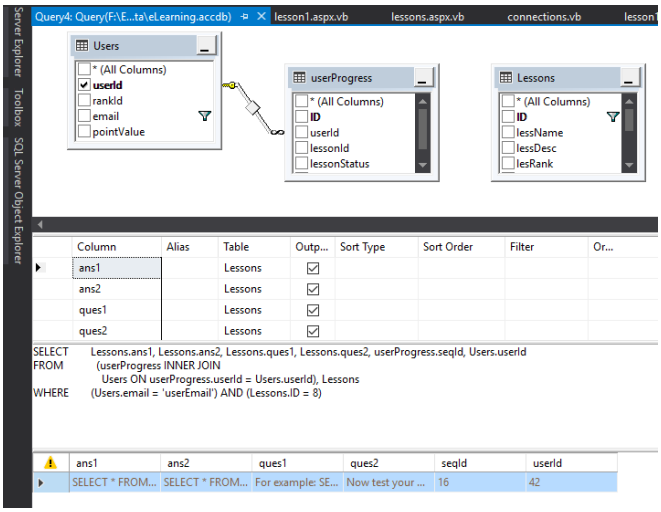
The algorithm for each lesson was repeated based on the major methods outlined above. The ans and ques variables stored the defined questions and answers retrieved from the lessons table, within the database. This conformed with the maintainability aspect of the user requirements, allowing the tutor to update the lesson content, without affecting the application code.

Developing queries:

The built-in DBMS within Visual Studio 2015 was used to optimize queries.



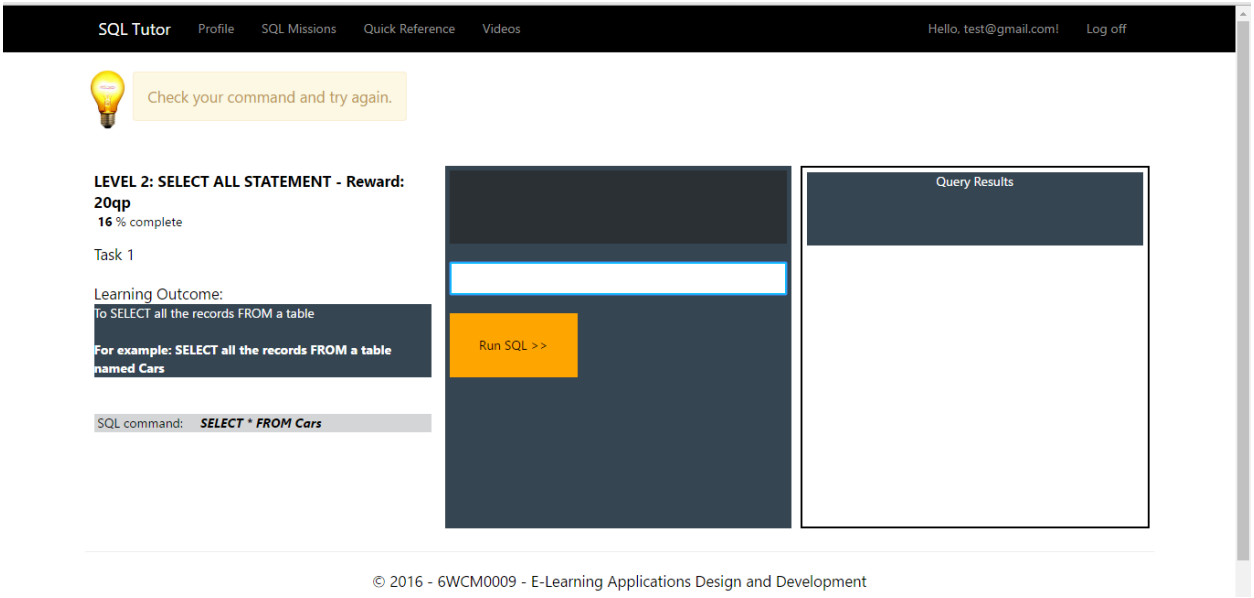
Next, the required tables were selected to form the required query.



The query was then copied and translated into the method.

```
0 references
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Try
        queryConn.queryData("SELECT Lessons.ans1, Lessons.ans2, Lessons.ques1, Lessons.ques2, userProgress.seqId, Users.userId
                             FROM (userProgress INNER JOIN
                             Users ON userProgress.userId = Users.userId), Lessons
                             WHERE (Users.email = '" & User.Identity.GetUserName & "') AND (Lessons.ID = " & lesson & ")")
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            userId = r("userId")
            lessId = lesson
            seq = r("seqId")
            ques1 = r("ques1")
            ques2 = r("ques2")
            ANS1 = r("ans1")
            ANS2 = r("ans2")
        Next
    Catch ex As Exception
    End Try        lbPercent.Text = seq
```

Combining all the elements within the mission's page, the following result was achieved.



Landing page:

The progress bar was integrated into the landing page to display the current lesson sequence. However, to ensure that it is visible only if there is a mission in progress, the progress bar DIV elements were placed inside a panel and the visibility was manipulated, based on the mission sequence.

Retrieve current lesson information:

```
'Query to retrieve the logged on user's current lessons and the page name value for navigation and percentage complete
'The pageName variable is used to hold the page name so the user can navigate to the lesson page when they click the continue button
1 reference
Private Sub getLessons()
    lbUser.Text = User.Identity.GetUserName
    Try
        queryConn.queryData("SELECT Lessons.lessName,Users.email, Lessons.pageName, userProgress.seqId
                             FROM Users, Lessons, userProgress
                             WHERE (Users.email = '" & lbUser.Text & "' AND
                             userProgress.lessonId = Lessons.ID
                             AND Users.userId = userProgress.userId)") 'SQL Query
        For Each r As DataRow In queryConn.ds.Tables(0).Rows
            lbTopic.Text = "Currently Viewing: " & r("lessName")
            pageName = r("pageName")
            lbPercent.Text = r("seqId")
            progress = r("seqId")

        Next
    Catch ex As Exception
    End Try


End Sub
```

The key functionality within this method uses the pageName value from the database to allow the user to navigate to the current lesson's page. A label holds the sequence value to determine the progress made within the mission. This information is retrieved from the userProgress table.

Once the sequence is below 2, the progress bar panel's visibility is set to false.

```
'If the progress percentage is less than 2, the panel the progress bar is contained in will be hidden, else, visible
1 reference
Public Sub showProgressBar()
    If lbPercent.Text < 2 Then
        panelProgress.Visible = False
    Else
        panelProgress.Visible = True
    End If
End Sub
```

For the progress bar animation, the same principle used in the first implementation was repeated.

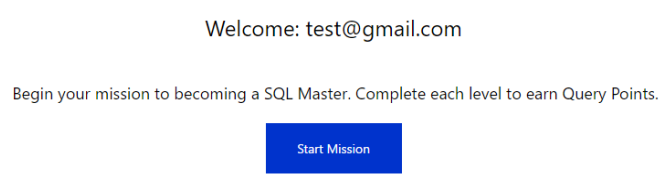
```
<script type="text/javascript">

    $(document).ready(function () {

        amount = document.getElementById('<%= lbPercent.ClientID %>').innerHTML

        animateProgressBar(amount);

        function animateProgressBar(p) {
            oW = $("#outer").width();
            $('#inner').animate({
                'width': (oW * p) / 100
            }, 3000);

            $({ counter: 1 }).animate({ counter: p }, {

                duration: 3000,
                step: function () {
                    $('#inner').text(Math.ceil(this.counter) + ' %');
                }
            })

        }
    });

</script>
```

In addition to hiding\showing the progress bar, if there is no user associated information found in the Complete table within the database (a new user), a customized welcome message is displayed.

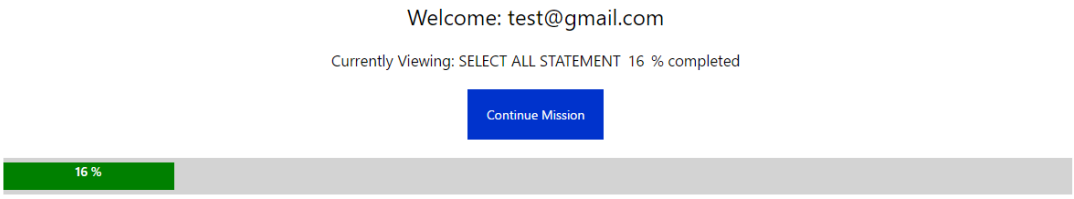
Code:

```
'Query to retrieve the completed lessons for the logged on user and hides the progress bar and percentage completed if
'Displays a second panel with welcome information
1 reference
Public Sub getComplete()
    Try
        queryConn.queryData("SELECT * FROM Complete, Users WHERE Users.email = " & User.Identity.GetUserName & " AND
                            Users.userId = Complete.userID")
        If queryConn.count = 0 Then
            panNewUser.Visible = True
            lbcompleted.Visible = False
            lbPercent.Visible = False
        Else
            panNewUser.Visible = False
            lbcompleted.Visible = True
            lbPercent.Visible = True
        End If

    Catch ex As Exception
    End Try
End Sub
```

New learner result:



Returning learner result:



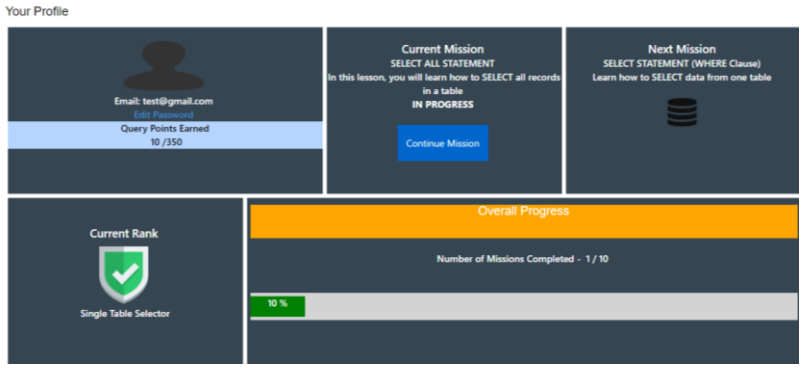
Advanced requirement - Responsive design implementation

Relative instead of absolute sizes were used for the layout. Tile widths and heights were defined using pixels, however the sizes used were set to the minimum screen size of 250px. Since the flexbox was already implemented for the main page layouts, responsiveness was easily achieved using CSS media queries. Once the page size was changed, the relevant media query is triggered and the desired div element on the page is manipulated. For example:

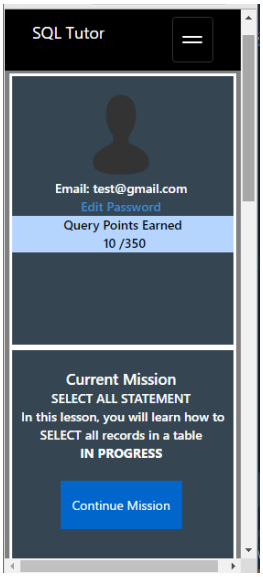
```
@media only screen and (max-width : 500px) {
    body{
        background-color: gray;

    }

    .parent{
        flex-direction:column;
    }
    .item, .iconItem{
    height:300px;
    width:100%;
}

}
```

In the above example, the flex direction of the parent DIV element is changed to column, so the overall orientation of each tile is changed from horizontal to vertical.

Profile page – Before:



Profile page – After:



Navigation:

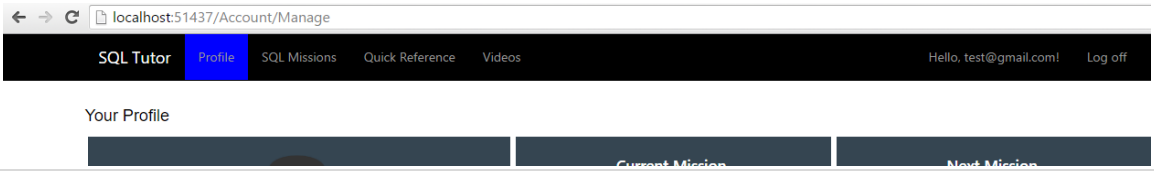
To ensure the user is aware of which page they are currently on, a JavaScript function was added to each page, which would change the background color of the associated menu item on page load (Nielsen's Heuristics). In order for the function to work, each menu item was assigned an ID, which was then referenced in the JavaScript function, for example:

```
<ul class="nav navbar-nav">
    <li><a runat="server" id="main" href="~/Account/Manage.aspx">Profile</a></li>
    <li><a runat="server" id="lessons" href="Members/lessons.aspx">SQL Missions</a></li>
    <li><a runat="server" id="aricles" href="Members/articles.aspx">Quick Reference</a></li>
    <li><a runat="server" id="videos" href="Members/videos.aspx">Videos</a></li>
</ul>
```

Function:

```
    function highlight() {
        document.getElementById('main').style.backgroundColor = "blue";
    }
});
```

Result



Additional styling was added to the default page which a user first navigates to. In following some of the elements from Khan Academy, a full page image background was placed on the landing page (Stackoverflow.com, 2016).

Version control:

This feature in the development process was easily implemented by adding the GitHub extension in Visual Studio 2015. Next, a GitHub account was created and once this was completed, a repository was created to link the project to.

For each major development point, the changes are committed to the local repository, and then pushed to the external server. Relevant comments are added to describe development at that specific point. Changes are displayed in a toolbar located at the bottom of the visual studio 2015 work space.

Chapter 6. Software Testing

A test plan was developed based on the requirements defined in the analysis phase of the project. Each test plan is separated into both functional and non-functional testing. The following presents each test description and the corresponding result, as it relates to the final e-learning project.

6.1 Functional testing

Test 1: User Registration		
#	Test description	Result
1.1	The new user must be able to access the registration page from a landing page	Success
1.2	Each register hyperlink must navigate the user to the registration page	Success
1.3	The user must enter a unique email address. If the user name is not unique, an error message should be displayed	Success
1.4	The password should be displayed using secured text	Success
1.5	The password should be entered twice for confirmation. If passwords do not match, an error message should be displayed.	Success
1.6	Passwords must conform to a defined level of complexity. If not, an error message must be displayed.	Success
1.7	Upon successful registration, users should be directed to a welcome page.	Success

Test 2: User Login		
#	Test description	Result
2.1	The registered user must be able to access the login page from a landing page	Success
2.2	The registered user must supply the associated email address and password.	Success
2.3	Upon successful login, the user must be directed to a welcome page	Success
2.4	Incorrect credentials should present a relevant error message to the user	Success

Test 3: Start a lesson (New user)		
#	Test description	Result
3.1	Upon successful login, a user must be able to click the start mission button to navigate to the Missions page	Success

3.2	On the lessons page, a new user must only be able to access the introduction mission. All other lesson controls must be disabled.	Success

Test 4: Lesson Resumption		
#	Test description	Result
4.1	A user must be able to start a mission. If the user navigates away from the page before completion, upon return to the lesson, the page should load at that specific point	Success

Test 5: Lesson interaction		
#	Test description	Result
5.1	Users can type directly into a text box on the mission page	Success

Test 6: Achievements and points		
#	Test description	Result
6.1	Users should receive a new rank upon completion of a mission	Success
6.2	Users should receive a specific amount query points defined for each mission upon completion.	Success
6.3	Users should only receive query points upon initial completion of the mission. A second attempt should not increase the query point amount.	Success

Test 7: Feedback mechanism on the mission page		
#	Test description	Result
7.1	A custom message is displayed based on user input. An error message is shown for incorrect input. A success message is shown for correct input.	Success

Test 8: Responsiveness		
#	Test description	Result
8.1	The orientation of each page should align vertically on small windows. Page elements should automatically adapt to the screen size.	Failed on the videos page

6.2 Analysis of test results

All twenty tests were conducted and proved to be successful. Although one page failed the responsiveness test, this only applied to the embedded YouTube link and not the video control. All videos would be saved on a server in future development. In doing so, the application would not be susceptible to broken external links.

The responsiveness function was tested using the developer tools within the Google Chrome web browser. These tools allowed the application to be viewed with varying screen sizes. The series of images relating to these tests can be found under the responsiveness section of the appendix.

Chapter 7. Software Evaluation

The application proved to meet most the requirements, based on the original plan. Several deviations were made from the design plan, as features of the Web Forms template were utilized, so additional alterations to the code would not be necessary. For example, the default.aspx page was converted to a welcome page, since this is the page the template redirected to on log out. This page is also the first page the user navigates to on startup.

Other design decisions included simulating a database table instead of allowing the user to query a live database. In future development, a more dynamic design would link the application to a sample data table and allow the user to run live queries. In this way, the learning content can be dynamically updated, since lessons are already retrieved from the database. Although the current design allowed for these lesson questions to be changed in the database, a constraint still existed, since the table showing the result of the query was hard coded into the application.

In evaluating responsiveness, vertically aligned lessons presented another constraint. Since the lesson workspace was separated into three segments, whichever segment the user is intended to focus on should scroll into view when the user completes a task. This feature was not fully developed due to time constraints and the necessary knowledge. The goal in future development is to vertically scroll the window during each point of interest, as well as scroll back to the top when an error message is displayed.

The intention was also to fully develop a bonus stage, incorporation all the learning outcomes presented thus far. In order to add a challenging element to the lesson, the user would be asked to complete all the queries within a specified timeframe (Countdown). Other elements included reducing the points awarded for each incorrect answer. These features were not implemented due to limited knowledge, as well as insufficient research material to support these features.

Chapter 8. Discussion and Conclusion

8.1 Introduction

This chapter aims to discuss thoughts on the final project and the overall journey from start to finish.

8.2 Challenges

Many of the technologies demonstrated in this project was acquired from hours of research using google searches, web design forums and YouTube. These technologies included JavaScript\JQuery, querying a Microsoft Access database using VB.Net, Cascading Style Sheets, Html and Html elements. Although many other suitable and more applicable technologies existed, it did not seem practical to learn that much, given the time frame for project completion. With that said, the initial lack of experience in web development proved to be the greatest challenge.

8.3 Tasks Completed

Despite all of the challenges faced, the application was developed in accordance with the original requirements defined. All of the basic functions were developed according to plan, with the minor deviations, which were mentioned in previous chapters. The presentation of micro lessons was successfully accomplished, as well as maintaining the focus on specified learning outcomes.

8.4 Tasks not completed

In a complete application, all of the lessons along with the corresponding videos will be developed and added, but presented according to the project design.

8.5 Experience gained

This project has presented an opportunity to learn about the web and web technologies. I must say that I am a lot more knowledgeable in these areas, as compared to when I first started. I was also able to apply some of the knowledge acquired in earlier modules, which assisted tremendously on how code was structured and managed. As a working individual, it was certainly difficult to manage the time required to complete the project, however I feel very proud to say that I am satisfied with the final product and how the project was executed.

The project also allowed me to apply a structured approach to the development process, especially in the area of report writing. It has given me the opportunity to identify my strengths in problem solving and the manner in which programming languages are understood. Many of the examples shown on the internet had to be translated into the scenario applicable to the project. Therefore, I was required to take apart the logic, understand its concepts and apply to my own intention. I hope to use the experience gained in this project, to fuel the rest of my intended career in web development.

.

References

Impact, D. (2016). *Deans for Impact*. [online] Deansforimpact.org. Available at: http://deansforimpact.org/the_science_of_learning.html [Accessed 3 Sep. 2016].

YouTube. (2016). *Simple jQuery progress bar*. [online] Available at: https://www.youtube.com/watch?v=5bPUZFgbn3A [Accessed 3 Sep. 2016].

YouTube. (2016). *CSS FlexBox Essentials*. [online] Available at: https://www.youtube.com/watch?v=G7EIAgfkhmg [Accessed 3 Sep. 2016].

"Responsive Web Design: Key Tips and Approaches". *99designs Blog*. N.p., 2012. Web. 2 July. 2016.
Bryant, J. and Jones, M. (2012). Responsive Web Design. *Apress*, [online] pp.37-49. Available at: http://link.springer.com/chapter/10.1007/978-1-4302-4525-4_4#page-1 [Accessed 3 Jul. 2016].

"The U.S. Mobile App Report". *comScore, Inc*. N.p., 2016. Web. 2 Aug. 2016.

Stackoverflow.com. (2016). *Prevent redirect to site.mobile.master*. [online] Available at: http://stackoverflow.com/questions/22695356/prevent-redirect-to-site-mobile-master [Accessed 20 Aug. 2016].

Stackoverflow.com. (2016). *css perfect full screen image background*. [online] Available at: http://stackoverflow.com/questions/7259084/css-perfect-full-screen-image-background [Accessed 3 Sep. 2016].

Stackoverflow.com. (2016). *how to get value in javascript for asp Label control?*. [online] Available at: http://stackoverflow.com/questions/19547258/how-to-get-value-in-javascript-for-asp-label-control [Accessed 3 Sep. 2016].

Google Books. (2016). *Project Managing E-Learning*. [online] Available at: https://books.google.tt/books?id=mMqTAgAAQBAJ&pg=PA69&dq=track+progress+elearning&hl=en&sa=X&ved=0ahUKEwjuptjuzs7OAhXCHh4KHXAEDc4Q6AEIHDAA#v=onepage&q=track%20progress%20elearning&f=false [Accessed 3 Sep. 2016].

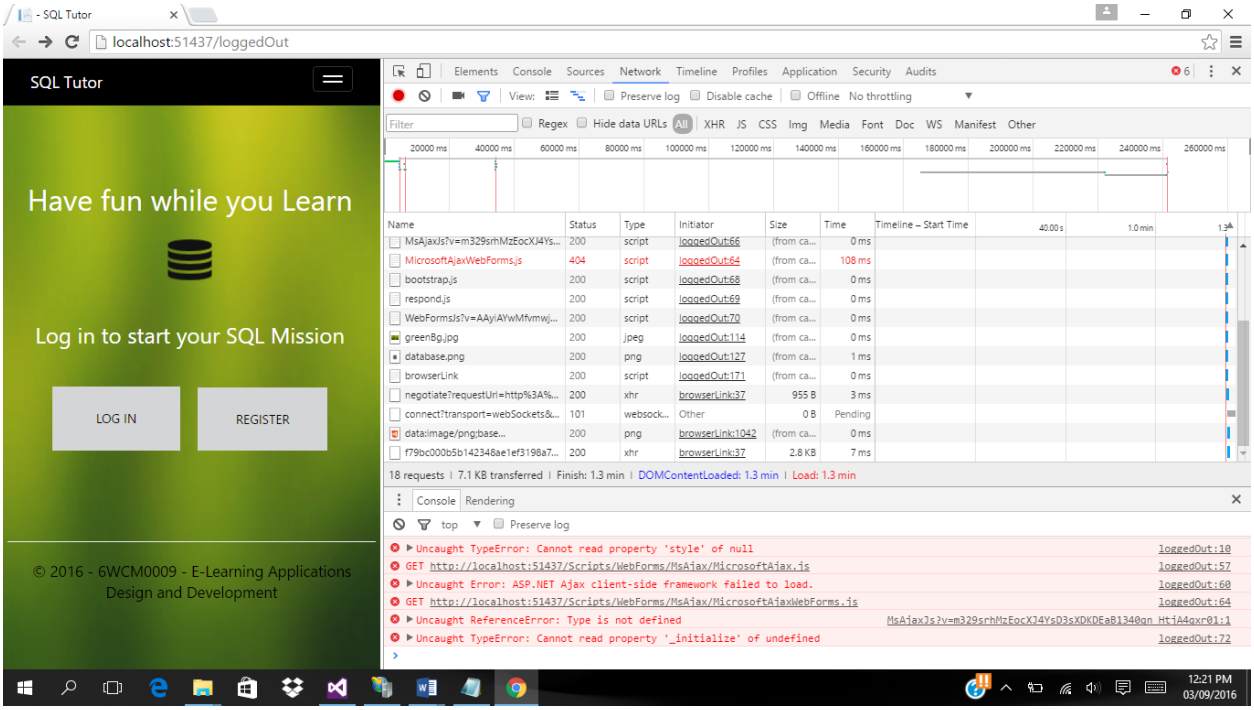
Khanacademy.org. (2016). [online] Available at: https://www.khanacademy.org/ [Accessed 3 Sep. 2016].

Codecademy. (2016). *Learn to code*. [online] Available at: https://www.codecademy.com/ [Accessed 3 Sep. 2016].
W3schools.com. (2016). *W3Schools Online Web Tutorials*. [online] Available at: http://www.w3schools.com [Accessed 3 Sep. 2016].

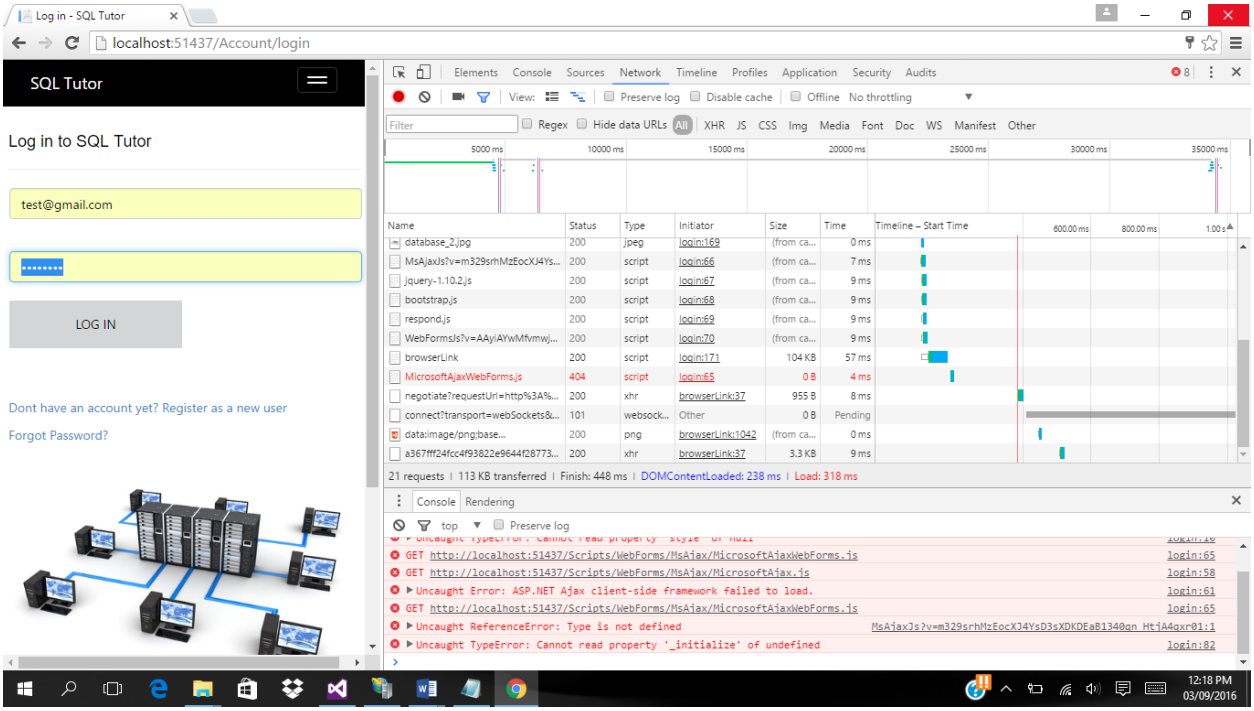
Appendix

Responsiveness

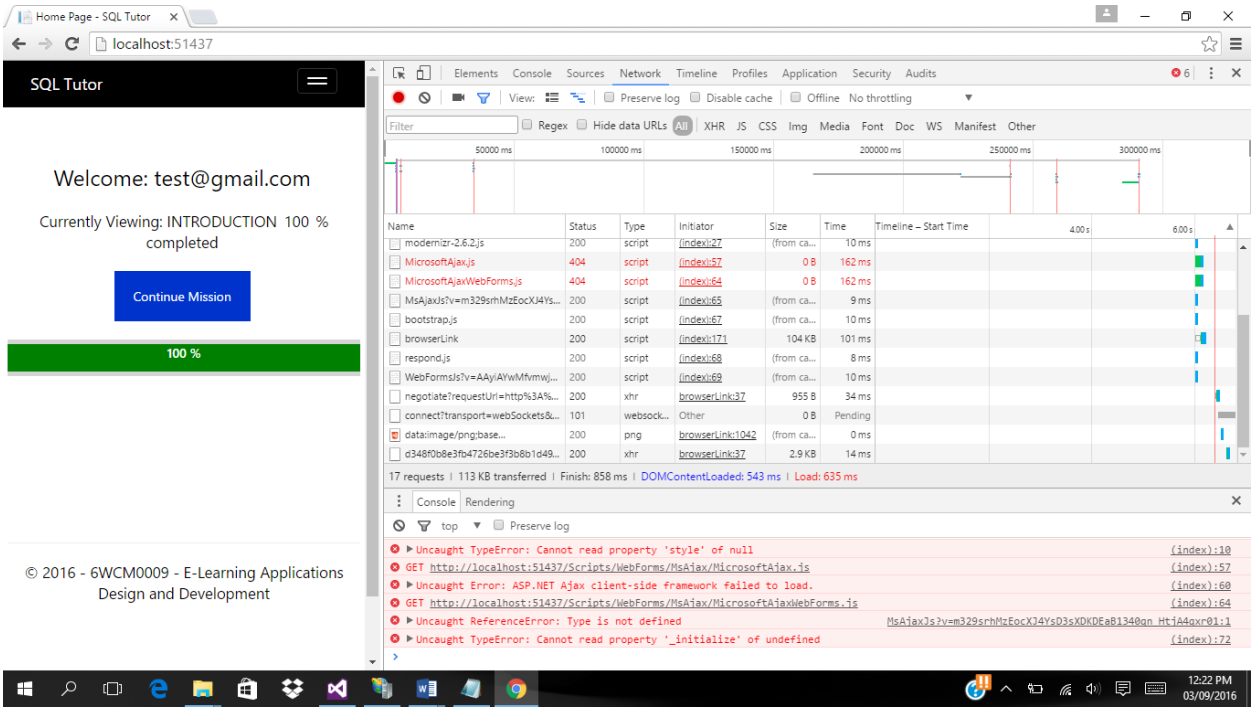
Landing page:



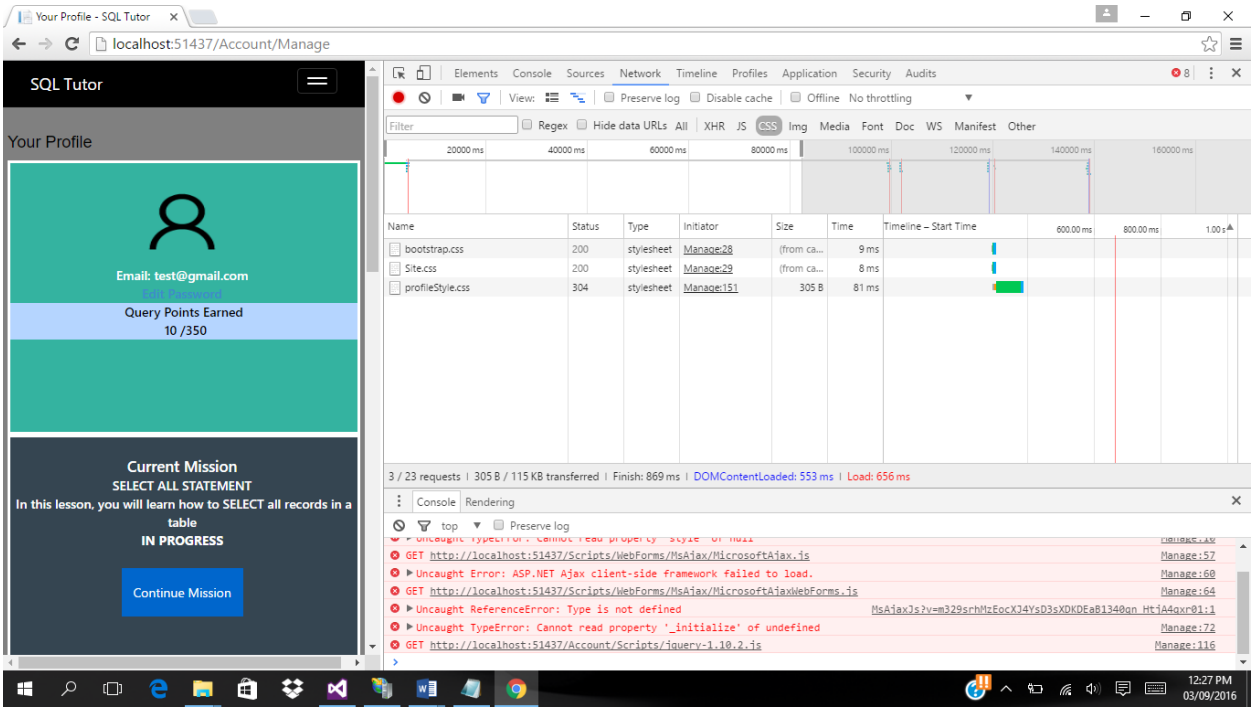
Login page



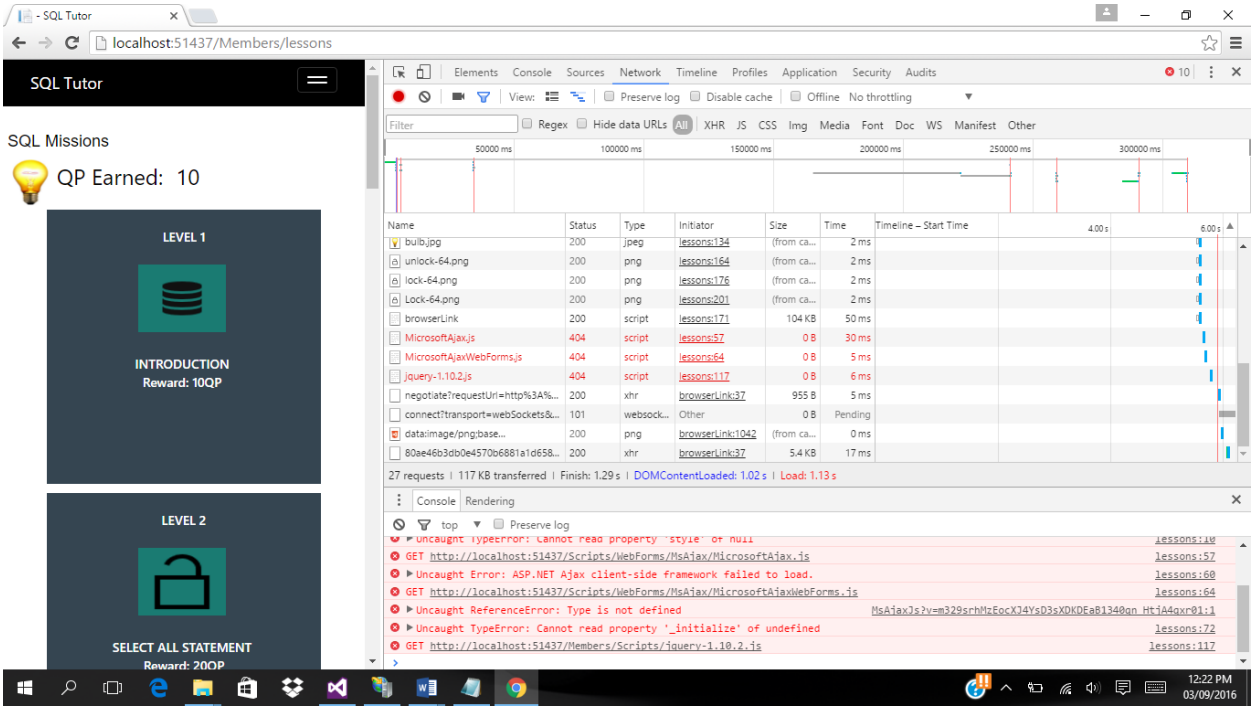
Welcome Page



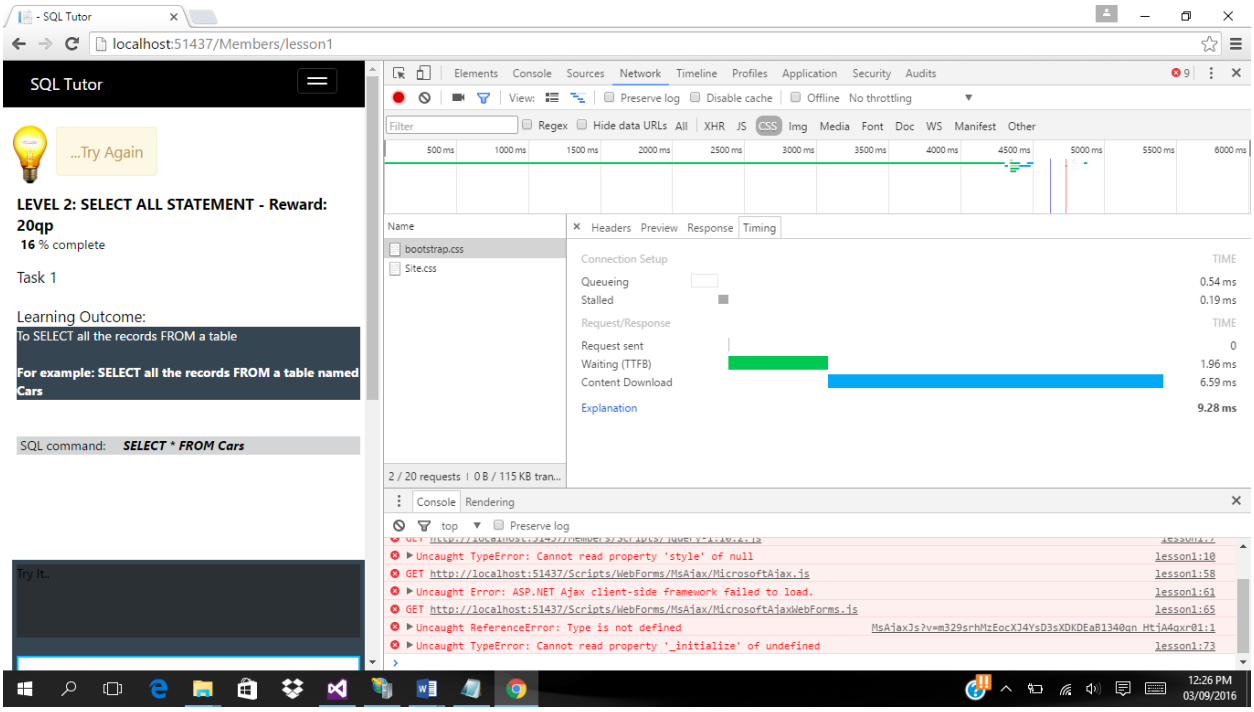
Profile Page



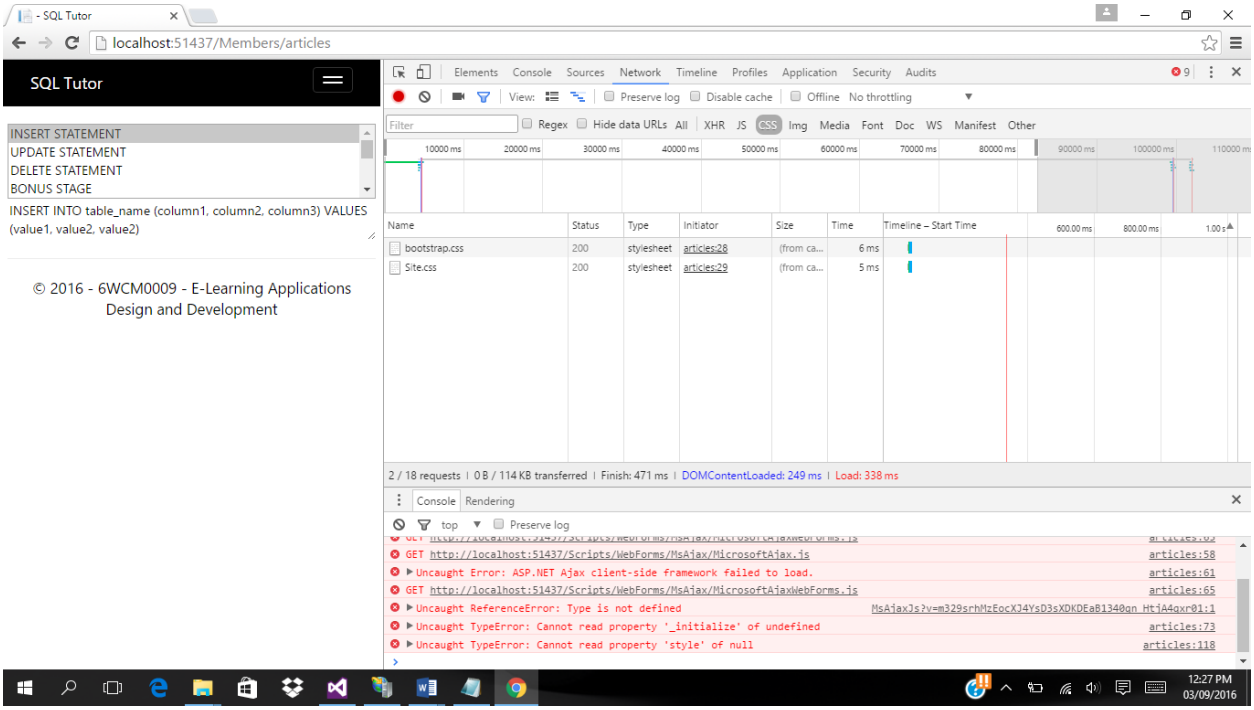
Missions page



Level 2 page



Quick reference page



Videos Page