|  |
| --- |
|  |

|  |
| --- |
| **Technical Report** |
| ICTWEB503 – Create Web-based Programs |
|  |
|  |
|  |
| **Sebastian Vowels** |
| **10/26/2021** |
|  |

Table of Contents

[Task 1 – Email 2](#_Toc86234667)

[Task 2 – Report 2](#_Toc86234668)

[Hypertext Transfer Protocol (HTTP) 2](#_Toc86234669)

[Limitations of HTTP 3](#_Toc86234670)

[Advantages of HTTP 3](#_Toc86234671)

[Web Service (API) 3](#_Toc86234672)

[Web Application (Front End) 4](#_Toc86234673)

[Task 3 – Development 6](#_Toc86234674)

[Task 4 – Perform Test 8](#_Toc86234675)

[Task 5 – Sign-off Sheet 10](#_Toc86234676)

# Task 1 – Email

Graphical user interface, text, application, email

Description automatically generated

# Task 2 – Report

### Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is a communication system utilised widely throughout the world, primarily on the internet. There have been different iterations of the standard of HTTP, namely HTTP 1 and HTTP 2. HTTP 1 was created in 1997 and its creators were unable to recognise the far stretching uses of the internet at the time. As such, standard provided in HTTP 1 was inefficient for what modern internet users required. HTTP 2 was created in 2015 and aimed to rectify a significant amount of these issues.

One of the biggest issues that HTTP 2 rectified related to the prioritisation of the resources being delivered. In HTTP 1, after a server receives a request from a client, they would send one resource at a time and await a response from the client confirming they had received the sent resource. Understandably, this process was highly inefficient as websites grew in size and complexity. When HTTP 1 was created a website consisted of a HTML page. Modern websites require HTML pages, CSS styling, JavaScript code, and multimedia assets galore. By forcing wait times for confirmation messages to be received, HTTP 1 was highly inefficient. HTTP 2 rectified this by allowing web servers to send all resources at once and provide context to the browser as to what should be rendered first as well as if any resources are missing. Should something be missing, the browser is then able to request just what they need. This concept is referred to as multiplexing and it substantially increased the efficiencies of the internet.

In regards to the FF web application, HTTP 2 should be utilised. Whilst in its current state there isn’t a great deal of information being transferred to the browser, HTTP 2 is the international standard and provides FF the opportunity to add functionality later on that might take better use of the improved standard.

### Limitations of HTTP

HTTP isn’t without its issues. Whilst being adopted world wide as the standard for internet communication there are some limitations of the protocol that should be kept in mind:

* Security
  + HTTP is sent across the internet as a plain text message. There is no encryption and anyone in between the client and the server is able to intercept the message.

### Advantages of HTTP

However, taking all of the limitations as previously mentioned into account, there is good reason as to why HTTP has been adopted worldwide:

* Connectionless
  + HTTP operates by sending messages, not having a face to face conversation. As a result, direct connections are not required between the client and server.
* Media Independent
  + Any type of data can be sent through HTTP as long as both the client and server have instructions as to how to unpack it.
* Addressing
  + HTTP included an addressing schema that enabled users to connect to different serves with improved accessibility. Before this, all connections utilised IP address which was difficult for users to interact with.

### Web Service (API)

The web application (front end) of the project will require a separate web service (back end) to manage and organise the data within the database. This back end will take on the form of an Application Programming Interface (API).

This API acts as an interface for the database. All data will pass through the API depending on the requests and inputs given to the API. The API will interact with the database on the client’s behalf and create/obtain data.

The web application will be directed to interact solely with the API for obtaining data from the database.

### Web Application (Front End)

The web application (front end) will be the component of the project that the users will interact with. The web application will be required to have the following functionality:

* Content: Surveys, Questions, Options
  + Create
  + Edit
  + Delete
  + View Details
* Authentication
  + Verify user details and log into application
  + Users can only interact with content that they create
* Session Management
  + Users can navigate between pages and receive customised content based on their authentication status (e.g. only viewing their surveys, username displayed on taskbar, etc).

Each of these components are fundamental to modern web application development. In particular, Session Management is crucial to allow individual users to access views that are tailored to them. This tailoring process takes on the form of providing the user with a customised web page that displays information that is most relevant to them. In regards to this project, this is exampled by a user logging in and being shown the surveys that they have created. Another advantage of this approach is that it by default provides a rudimentary level of Authorisation in that users are only allowed to edit and delete their own content as that is all that is displayed to them.

Session Management was implemented by first developing a Token controller within the API. This component is responsible for checking if a user’s details are present within the database and match the records. If so, a Token is generated and provided to the user via the HTTP response. This token is then what the user will provide to the front end application, which will then be passed onto the API when required. A valid Token is both proof that a user has been authenticated as well as who the user is.

Within the front end application the HttpContext class within the AspNetCore library was utilised to enable the sessions state management. This class enables the storing of session dictionary values which were used to store the Token and username.

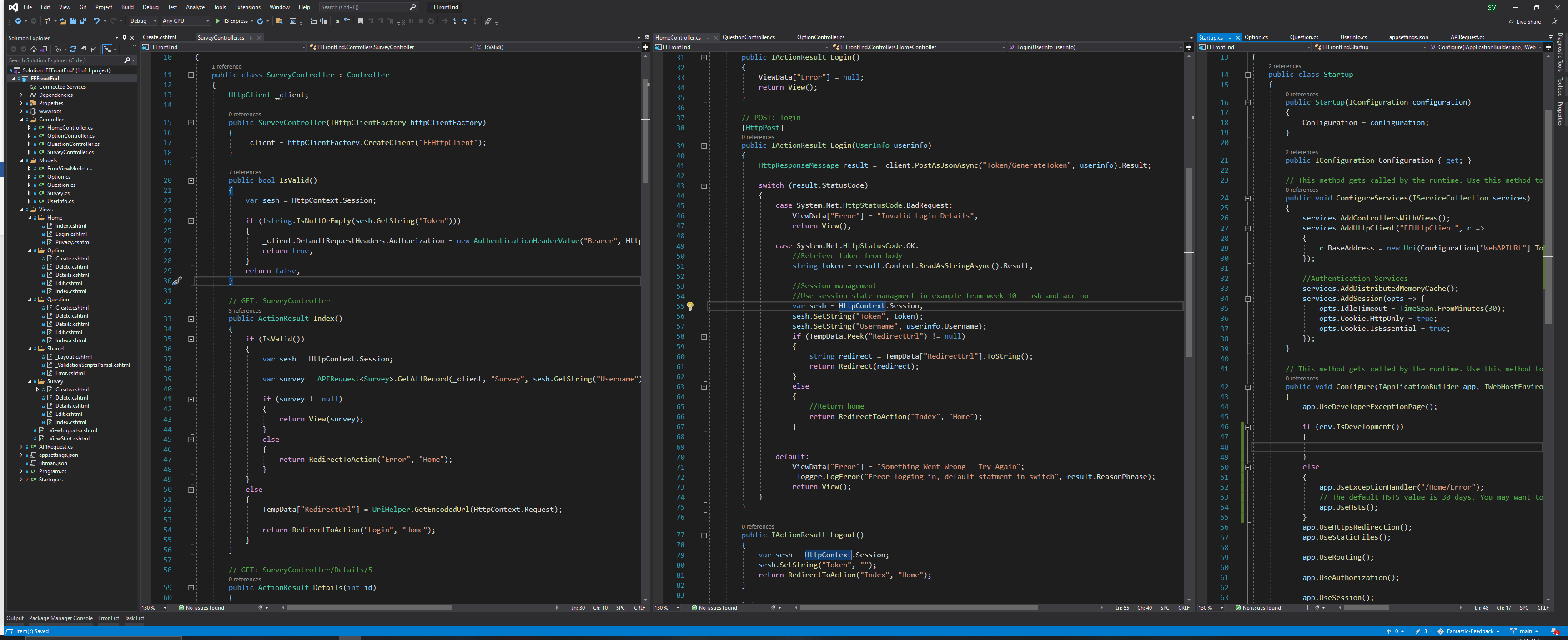


Figure 1- Screenshot displaying use of session management within project

In Figure 1, the middle code (HomeController) displays the Login method which is responsible sending a request to the API with the users log in details. Should this be successful the returned Token and username are saved into a session state dictionary pair. In the left most code (SurveyController), the IsValid() is responsible for checking if the session state dictionary pair is present or assigned anything. If it is, it provides the HTTP client with the token to be used in a later request. The right most code (Startup), the service for session management can be seen within the ConfigureServices().

# A picture containing text, monitor, indoor, computer Description automatically generatedText Description automatically generatedTask 3 – Development

Figure 3 - Screenshot of HomeController (Left) and SurveyController (Right)

Figure 2 - Screenshot of Solution Explorer of Front-End Project

A picture containing text, monitor, black, screen

Description automatically generatedA picture containing text, monitor, screenshot, black

Description automatically generatedA picture containing text, monitor, computer, indoor

Description automatically generated

Figure 6 - Screenshot of TokenController (Left) and QuestionController (Right) (API)

Figure 5 - Screenshot of APIRequest (Left) and Startup (Right)

Figure 4 - Screenshot of QuestionController (Left) and OptionController (Right)

# Task 4 – Perform Test

|  |  |  |
| --- | --- | --- |
| **Test Performed** | **Expected Results** | **Actual Results** |
| Login process:   * Navigate to login page * Enter correct login details | User should be authenticated successfully and navigated to survey index page. | User was able to be authenticated and logged into web application. |
| Login error process:   * Navigate to login page * Enter incorrect login details | User should be provided with an incorrect login attempt message within the login page. | User provided with “Invalid Login Details” warning and repromoted to login. |
| Create process:   * Successfully Login * Navigate to Survey Index page * Create 1 new Survey * Create 1 new Question within Survey * Create 1 new Option within Question | All items should be created without errors and be able to be displayed on survey details page. All should be accessible within the same login. | User was able to create all items successfully. Each item is displayed correctly within the relevant pages along the process. |
| View Details process:   * Successfully Login * Access survey details page * Access question details page * Access option details page | Each details page should provide the correct details for the accessed item. | User was able to view the details page for the selected survey and option. The question details page, whilst built and functional, hasn’t been implemented from the survey details page yet. This will be discussed below. |
| Edit process:   * Successfully Login * Access survey details page * Edit 1 aspect of the survey details * Edit 1 aspect of a question within the survey * Edit 1 aspect of a option within a question | All edited details should retain their new value and be displayed correctly within their respective details view. | User was able to edit the details of a survey, a question, and an option successfully. |
| Delete process:   * Successfully Login * Access survey details page * Delete 1 option from a question within the survey * Delete 1 question from within the survey * Delete the survey (from index page) | At each stage of deleting, when returning to the survey details page the deleted item should no longer be present. Once the survey is deleted it should no longer be present within the survey index page. | User was able to delete an option, a question, and a survey successfully. |
| Authorisation Testing:   * Successfully Login * Obtain from DB ID number of Survey that was created by a different user than the current one logged in * Navigate to the details page for that survey using the ID obtained (localhost:xxxxx/Survey/Details/{id of different survey} | User shouldn’t be able to access foreign survey. | Error presented to user stating that the survey could not be found. Error handling not implemented however test successful as user was not able to access a foreign survey. |
| Session State Testing:   * Successfully Login * Open a new browser and navigate to home page * Successfully Login with different login * Navigate to survey index page in both browsers | Both users should be able to login and display differing content based on their login (i.e. the webpages should display different surveys that are relevant to the user logged in). | User was able to login with two separate logins in two separate browsers (Firefox and edge). Each browser only displayed content that was applicable to them. |
| Logout process:   * Successfully Login * Press the Logout button in the nav bar * Navigate to the home page * Navigate to survey index page | The user should be able to logout without issues and upon attempting to return to the survey index page should be redirected back to logging in. The username in the nav bar shouldn’t be present either. | User was able to logout successfully and upon navigating to the survey index page whilst logged out they were redirected to the login page. |
| API Dependency:   * Turn off API * Attempt login * Attempt accessing survey index page | User shouldn’t be able to access any content nor be authenticated. | User was unable to be authenticated and therefore couldn’t attempt to access survey index page. |

Overall the majority of tests were completed successfully and as intended. The Login and Login error process worked as intended and provided the user with feedback upon an unsuccessful login. The creation process worked as intended and the information was able to be accessed. The Edit process worked as intended and the modified data was able to be accessed at a later point. The Delete process worked as intended and the data was no longer able to be accessed after this.

The View Details process was a moderate success. The Survey and Options details views were able to be accessed and displayed the correct data in the intended format. The Question details view is able to be accessed by navigating directly to that page and does display the correct data in the intended format. Unfortunately the Html.ActionLink() web link to the Question details view from the Survey details view is as of writing not implemented. In the future build this will be implemented.

The Authorisation test was a moderate success. The user was unable to access another users surveys. This is by design as a check is made within the API to verify that the requesting user has a survey that matches the given ID. However the error presented is not handled by the front end and should be. This will be implemented in a future build.

Session State testing was successful as the user was able to log into two separate accounts in two browsers and each presented the user with a view customised to the logged in account.

Logout process worked as intended and the user was redirect to login successfully when required. API Dependency was established. This test highlights the web applications reliance on the API. This cannot be mitigated by design, but was included to highlight the need for redundancy and fallbacks to be implemented in production.

# Task 5 – Sign-off Sheet

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name:** | Fantastic Feedback Web Application and Web Service | | |
| **Project Manager:** | Shaun O’Sullivan | | |
| **Start Date:** | 30/08/2021 | | |
| **Completion Date:** | 29/10/2021 | | |
| **Project Deliverables:** | | | |
| Web Application - FFFrontEnd Solution and Build | | | |
| Web Service – API-FantasticFeedback Solution and Build | | | |
| Technical Report | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| ***I acknowledge that I have submitted all the stated deliverables as per the Project Requirements and your instructions.*** | | | |
| Sebastian Vowels | |  | 27/10/2021 |
| **Programmer’s Name** | | **Signature** | **Date** |
| Shaun O’Sullivan | |  |  |
| **Project Manager** | | **Signature** | **Date** |