SWANSEA UNIVERSITY

MENG COMPUTING

CSM14 - INDIVIDUAL PROJECT

Narrative and Reflective Account

Luke Hammond - 522196

Project supervisor: Dr N A Harman

May 07, 2012

Contents

1 Introduction

The purpose of the Narrative and Reflective Account is to present a descriptive narrative of the process that was carried out to develop the Glimpse Web system. The Glimpse Web system that was developed comprises of three layers. These are the data layer, services layer and presentation layer. Each of these layers are discussed throughout this document.

Following this introductory section of the document, the entire project development process is described. Each of the important phases of the development process are discussed in relation to how they were completed. The first part of this section describes the process I undertook to formulate the requirements and specification items for the project. Following this, I discuss the development methodology I chose to use to help in the creation of the application. The penultimate phase that is discussed is the implementation process, which gives a discussion of exactly how I implemented the system. Finally, I outline the testing process that I used to make sure that each part of the system worked correctly as I was developing it and that the system met its specification.

The next section of this document describes a number of challenges that I faced while developing the Glimpse Web system. Some of the challenges that I talk about in this section include how I was unable to use the MVC framework properly at the start of the development phase of the project and how I overcome problems faced while developing WCF services when the message returned is too big. Fortunately, each of the challenges that I faced were overcome during the course of the project. Included with each challenge is a brief discussion of how it was overcome.

I then go on to discuss the technology that I decided to choose to develop the system. I explain why I chose the technologies that I did, along with why others were rejected.

The final section of this document reflects upon the entire project from start to finish. Any mistakes that were made during this time are mentioned here. An important product of making a mistake should be that lessons are learned from it. This was the case when I made mistakes and the lessons that I learned from doing so are provided along side each mistake. I close of the document with an evaluation of how well I think the project went.

2 Project Development Process

2.1 Requirements

At the very beginning of the life cycle of the project, one of the first major tasks was to develop a list of requirements for the system that was going to be developed. This list would then define the scope of the project by stating exactly what is to be produced by the end of the project. It was important that the set of requirements were achievable, given the time scale of the project, but it was just as important to make sure that a substantial amount of work would be required to complete the project.

The first stage of formulating a set of requirements saw me visit the client I would be developing the system for. Myself and Neal Harman payed a visit to Nick Becket at the headquarters of RNA Plant, in Capel Hendre, where Nick discussed his expectations for the project. Following this initial meeting, we arranged another meeting where Nick would demonstrate the current application they are using and we could discuss the project in more detail. During the second meeting we identified a list of essential features that must be present in the final system.

After meeting with Nick on two previous occasions, I had a better understanding of the scope of the project. From here, I created a list of formal requirements that covered each of the sections of the application in detail. The list not only included functional requirements, but it included non-functional requirements as well. I made sure that the list was complete, concise and each requirement was unambiguous. It was important that the list of requirements had the three features mentioned as it meant that the client and myself had the same understanding of what the system is required to do.

Once I had finished the list of requirements I was then able to move on to creating the specification document. Each requirement had one or more specification item relating to it, which stated how that requirement would be implemented.

2.2 Methodology

At the start of the project, it was necessary to choose methodology to follow to develop the system. In the requirements document several methodologies were outlined along with their advantages and disadvantages. After carefully analysing each of the methodologies, it was decided that the Scrum methodology would be the most suitable for this project. The flexible nature of the Scrum methodology makes it perfect for this project. Also, as working software will be produced frequently, it will allow the client to provide feedback stating whether what is being produced is correct or if changes need to be made.

As the Scrum methodology has been designed with the assumption that a team will be present, it was necessary for slight modifications to be made in how the process is followed. The three roles will essentially be reduced to two. The role of product owner was assigned to Nick Beckett of RNA Plant but he was not be able to add any major requirements to the product backlog once work had commenced, only slight modifications. The roles of ScrumMaster and development team were undertaken by myself. The other part of the process that needed modification was in relation to the meetings. As I was developing the software on my own, daily scrum meetings were impossible as there were no other team members to inform about the progress that had been. Instead of this I made personal notes of what I had done and what I was going to do. The sprint planning meeting was modified in such a way that it involved me choosing what will be added during the next sprint. On some occasions I contacted Nick Beckett to ask if he had a preference of what gets added next. At the end of each sprint, there was no need for a sprint review meeting but I put back into the product backlog any items that were not implemented during the sprint. The sprint retrospective meeting is held to discuss what went well during the last sprint and what could be improved. Instead of a discussion, I just thought about what went well and what could have been improved.

The reason the waterfall model was not used for this project was because of its inflexible nature. Also, the spiral model was not chosen because of the need for intensive risk analysis to be carried out throughout the development. This seems unnecessary for this project and would have used up valuable development time.

2.3 Implementation

Once I had developed the requirements and specification documents it was clear that there were three main parts to the system. These parts, or layers as I have been calling them, have been referred to throughout all of the documents for this project. The three layers are:

- Data layer.
- Services layer.
- Presentation layer.

The first of the layers, listed above, that I implemented was the data layer. To do this I created an ADO.NET Entity Data Model using the database schema that is currently being used with the desktop application, Glimpse Desktop. Several problems were encountered during this process, all of which will be discussed in the following section of this document. However, all of these problems were resolved and an entity data model was created successfully.

The remaining two layers of the application were developed in parallel. It was clear that the application had several different sections. For example there is an invoicing section, maintenance section and a mapping section, to name a few. What I decided to do was to pick one of the sections at a time and develop the WCF services related to that section. As I was developing the services for a section, I made sure I used the 'WCF Test Client', provided by Visual Studio, to make sure that what I written was correct and produced the desired result. When I was sure that the services were correct, I moved on to developing the corresponding section in the presentation layer. By developing in this manner, I made sure that any functionality exposed by the services was not forgotten about accidentally. It also meant that I had something I could show to the client if they wanted to see what I had done. This would not have been possible if I developed the entire services layer first and then moved on to develop the presentation layer.

2.4 Testing

Throughout the development period of this project, testing played a vital role in making sure that what I was writing was correct. The methods I used to test the code in the services layer and the presentation layer differed. For the services layer I was able to take advantage of the 'WCF Test Client' to test the services I had written. But a test client was not available to test the controllers in the presentation layer.

As previously mentioned, I utilised the 'WCF Test Client' to test my web services. The test client is provided with Visual Studio is a GUI tool that enables a developer to test a service by inputting the required parameters and submitting the request. When the service responds, the developer is presented with the response returned. This tool made testing the WCF services I had written really simple. I was able to test each publicly exposed method with a variety of different parameters and view the response to check that they functioned correctly.

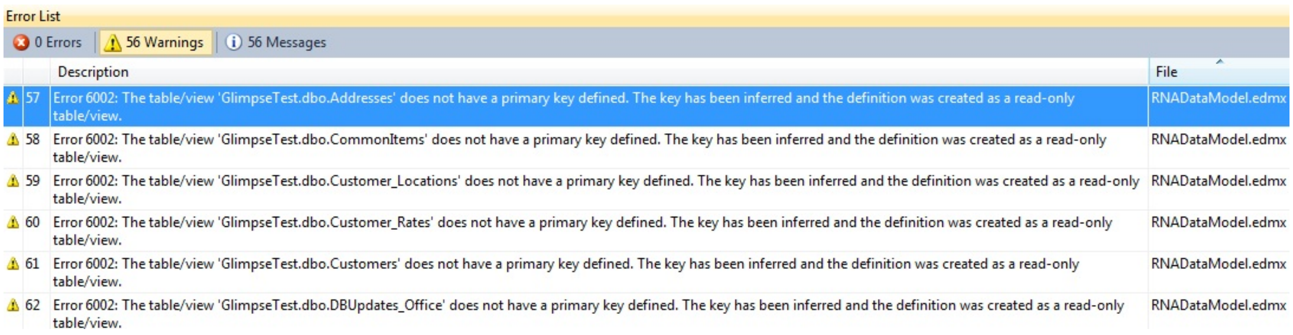
To test the presentation layer of the application, I mainly used manual testing. What I mean by this is, I would interact with the application through the user interface and make sure the expected result was achieved. I was able to do this because the services had been tested so I knew they were correct, all I had to do was make sure that the presentation layer was communicating correctly with the services.

Once the system had been developed and was deemed to be complete, I undertook thorough acceptance testing. Performing acceptance testing was required to make sure that the system that has been developed matches up to the specification. This was the most important testing phase because, if the system did not meet the specification then the project would not be classed as a success. To carry out the acceptance testing, I developed an extensive test suite that was designed to cover to each specification item. Each test case in the test suite has been given an identifier, a list of specification IDs stating which specification items the test covers, a description, acceptance criterion and a *PASS* or *FAIL* outcome. Regarding the tests that failed, a description was provided explaining why the test case failed. Out of all of the test cases, only 4 failed. The reasons behind the failure of these tests can be seen in the *Testing Document*. However, the failing tests had no detrimental impact on the overall system.

3 Challenges

3.1 Working with the Existing Database Schema

At the start of the implementation period for this project, it was necessary to decide how the application was going to interact with the database. It did not take long to decide that the best way forward was to use the Microsoft ADO.NET Entity Framework. Once it was decided how the application will communicate with the database, the next phase was to create an ADO.NET Entity Data Model. During the creation of the data model a problem was encountered that I was previously unaware of. The problem was that, the existing database schema that is currently being used by the legacy application contained tables that do not have any primary keys defined. The screenshot provided in Figure 1 shows the warnings that were produced which clearly state that the tables do not have primary keys defined.


Error List

0 Errors | 56 Warnings | 56 Messages

		Description	File
⚠	57	Error 6002: The table/view 'GlimpseTest.dbo.Addresses' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx
⚠	58	Error 6002: The table/view 'GlimpseTest.dbo.CommonItems' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx
⚠	59	Error 6002: The table/view 'GlimpseTest.dbo.Customer_Locations' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx
⚠	60	Error 6002: The table/view 'GlimpseTest.dbo.Customer_Rates' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx
⚠	61	Error 6002: The table/view 'GlimpseTest.dbo.Customers' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx
⚠	62	Error 6002: The table/view 'GlimpseTest.dbo.DBUpdates_Office' does not have a primary key defined. The key has been inferred and the definition was created as a read-only table/view.	RNADataModel.edmx

Figure 1: Warnings produced when creating a data model from a SQL database where no primary keys exist.

When a primary key is not defined on a table, Entity Framework infers a primary key and doesn't allow update operations to be performed on rows contained within said table. Thus, making it a read only table. Using read only tables for this application would not be possible as it must be possible to update records held in the database as well as retrieving them. So, to overcome this problem, I had to work my way through all of the tables and add primary keys where required.

After I had added the primary keys to the tables, I tried to update the data model from the database so that it would include the primary keys. However, instead of removing the existing inferred primary keys and adding the keys I had created, the update only added the keys I created to the keys that were inferred. By doing this, it meant that the tables in the database did not match the tables in the Entity Framework data model. This was expressed by the framework by displaying error messages similar to the one shown in Figure 2. The solution to this problem was either going through the data model and removing the inferred primary keys by deselecting the 'Entity Key' property, or deleting the entire data model and creating a new model from scratch. I settled on the latter as it seemed the simpler way to do it and would result in no mismatches between the model and the database.

Error 4 Error 3002: Problem in mapping fragments starting at line 2057:Potential runtime violation of table JobsToDelete's keys (JobsToDelete.JobNo): Columns (JobsToDelete.JobNo) are mapped to EntitySet JobsToDeletes's properties (JobsToDeletes.JobNo) on the conceptual side but they do not form the EntitySet's key properties (JobsToDeletes.DriverNo, JobsToDeletes.JobNo).

Figure 2: Update model error.

The graphical representation of the data model that was produced is quite large as there is over 50 tables in the database. In Figure 3, a section of the model has been included to provide an example of what gets produced by the Entity Framework. It is clear from the image that the primary keys are now present on the tables.

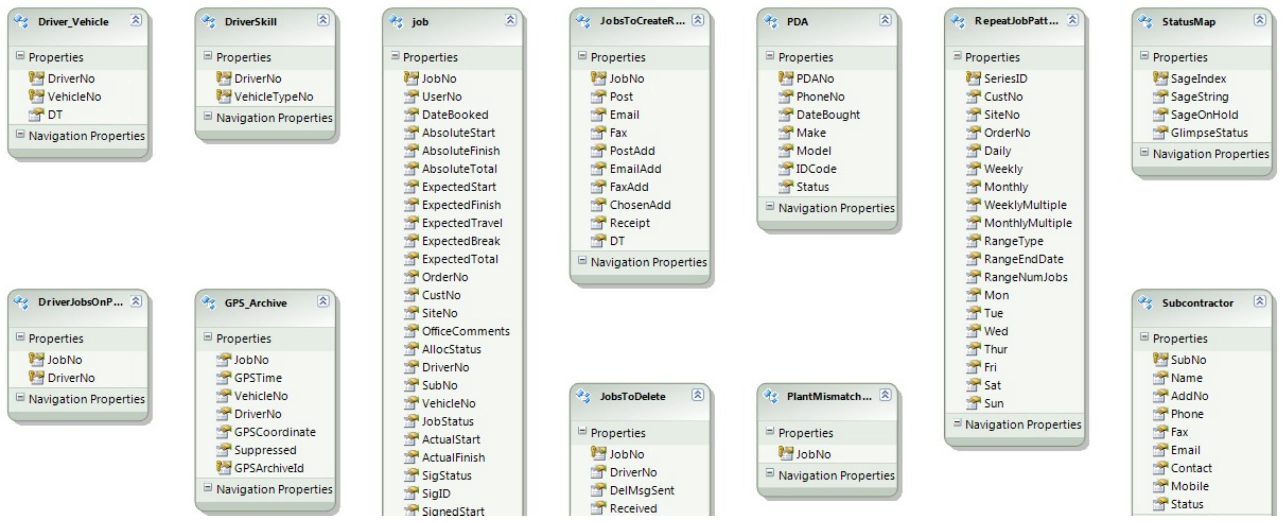


Figure 3: A section of the data model produced by Entity Framework.

Apart from the minor problems discussed above, using the Entity Framework to handle communication between the application and the database has been straight forward. It has allowed me to use LINQ to issue queries to the database which returns strongly typed objects that can be manipulated and saved back to the database. This is in contrast to what I have done in past projects where it has been necessary to write the SQL statements to run against the database and do not return strongly typed objects. By using the Entity Framework and LINQ, I believe that it has enabled me to be more productive and to write much cleaner code.

3.2 Using the MVC Framework

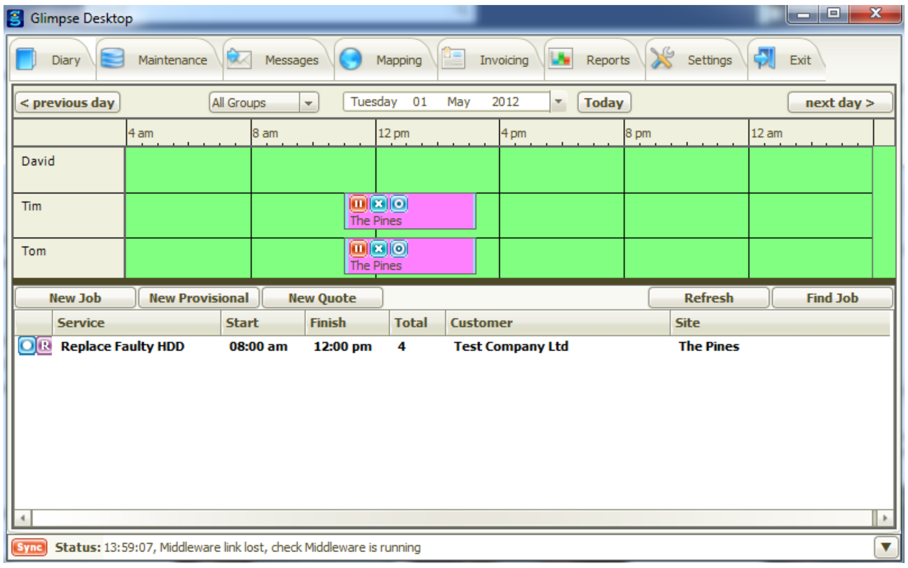
During the creation of the requirements and specification documents, it was decided that the presentation layer would be written using the ASP.NET MVC framework. Up until the start of the development phase of this project I have had limited experience working with MVC. Due to my lack of experience with the framework I ended up using it in a way that it was not intended. The functionality that I implemented at that time worked correctly but I was putting too much input logic into the views. The input logic should reside within the controllers and not the views. I was having to use a significant amount of JavaScript to get a view to display as I required, depending on the model that was passed to it and the action from which the view was called.

After gaining more experience using the MVC framework and undertaking more research on the subject, I came to understand how to work with the framework properly. The aim of the framework is to help a developer create applications where the different aspects of the application are kept separate and with loose coupling between them. These aspects are input logic, business logic and UI logic. The input logic should reside in the controllers, the business logic belongs to the models and the UI logic belongs to the views. From this point onwards, I put as much of the input logic as I could into the controllers and less into the views. In quite a few cases the view contained no input logic. By doing this I was able to develop the application a lot quicker with much cleaner code that was easier to understand.

Also, during the start of the implementation phase of the MVC application I decided not to create models. The reason for this was that I had already created data contracts during the development of the WCF services and these could be used to create strongly typed views. I also felt that creating models would be redundant as they would be an exact copy of the data contracts. This decision turned out to be the wrong thing to do for a number of reasons. As I got further into the development of the application I realised that I needed other properties in the view model, that would be used by the view, which weren't necessarily required in the data contracts. Also, by using the data contracts instead of models, I was not able to take full advantage of the validation annotations which are included with MVC and make validating user input extremely simple. For the reasons mentioned, I continued developing the MVC application using models as I was able to add additional properties that my views could use and I could validate user input with ease.

3.3 The Diary Screen

In the current desktop application being used by RNA Plant, Glimpse Desktop, there is a diary screen. This screen provides a lot of functionality regarding the allocation of jobs and the details of jobs. It allows a user to allocate a job to an operator or a subcontractor, unallocate a job and change the start time of a job dynamically through the use of a drag and drop feature. A user of the application can use the diary screen to get an overview of what jobs have been allocated, who they have been allocated to and what jobs have not yet been allocated for a particular day. The screen is shown in Figure 4.



Service	Start	Finish	Total	Customer	Site
Replace Faulty HDD	08:00 am	12:00 pm	4	Test Company Ltd	The Pines

Status: 13:59:07, Middleware link lost, check Middleware is running

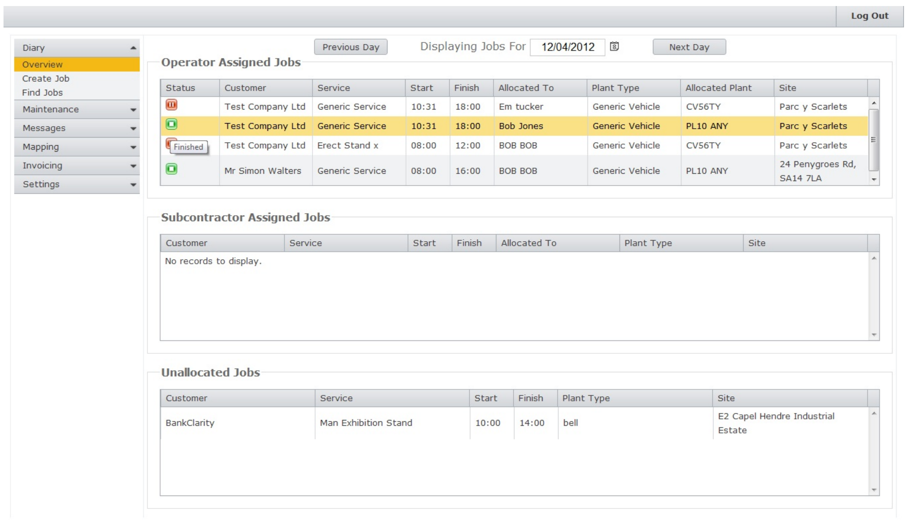
Figure 4: The diary screen that is currently being used in Glimpse Desktop.

At the start of the implementation period, one of the main sections of the application that was focused upon was the diary section. Although it was not specified in the specification or requirements document, I really wanted to provide a diary screen that was similar to the one being used in Glimpse Desktop. The main features I wanted to include were as follows:

- To be able to drag and drop a job between operators and subcontractors.
- Unallocate a job by dragging and dropping it into an unallocated job list.
- To be able to drag and drop a job along a timeline to update the start and finish times for a job.
- To be able to get an overview of what jobs have been allocated, who they have been allocated to and what jobs have not yet been allocated for a particular day just by looking at the screen.

I came to the conclusion that creating this component myself, from scratch, would not be possible. The main reasons for this are that it would require a considerable amount of time, I have not had much experience in developing custom components for the web and was outside the scope of this project. Following this realisation, I decided to search the internet for software developing companies that develop web-based controls for MVC. I managed to find a few companies that provided a scheduling component that were able to provide the functionality I require. However, after creating an example project to test the components, the results were not as good as expected. After finding out that the ready made components did not live up to my expectations, I moved on to develop the other sections of the application but I continued to look for a scheduling component at the same time. During this time I made sure that I devised an alternative solution for the diary section of the application if I could not find a suitable component.

Once I had developed the rest of the application I came back to implementing the diary screen. Unfortunately I did not manage to find an MVC scheduling control so I had to implement my alternative solution for the diary screen functionality. The solution I devised was to have three grids; one for operator allocated jobs, another for subcontractor allocated jobs and the final grid for unallocated jobs. It was not possible for me to implement drag and drop functionality between the grids but, selecting a job in any of the grids would direct the user to a screen showing the details of the job. Then from the job details screen the user is then able to update who the job is allocated to as well as its start and finish times. I believe that the solution I developed provides the features previously mentioned but without the ability to drag and drop jobs. The solution I developed is shown in Figure 5.



Operator Assigned Jobs

Status	Customer	Service	Start	Finish	Allocated To	Plant Type	Allocated Plant	Site
🟥	Test Company Ltd	Generic Service	10:31	18:00	Em tucker	Generic Vehicle	CV56TY	Parc y Scarlets
🟩	Test Company Ltd	Generic Service	10:31	18:00	Bob Jones	Generic Vehicle	PL10 ANY	Parc y Scarlets
🟥 Finished	Test Company Ltd	Erect Stand x	08:00	12:00	BOB BOB	Generic Vehicle	CV56TY	Parc y Scarlets
🟩	Mr Simon Walters	Generic Service	08:00	16:00	BOB BOB	Generic Vehicle	PL10 ANY	24 Penygroes Rd, SA14 7LA

Subcontractor Assigned Jobs

Customer	Service	Start	Finish	Allocated To	Plant Type	Site
No records to display.						

Unallocated Jobs

Customer	Service	Start	Finish	Plant Type	Site
BankClarity	Man Exhibition Stand	10:00	14:00	bell	E2 Capel Hendre Industrial Estate

Figure 5: The diary screen that has been developed for Glimpse Web.

Even though I did not manage to find an adequate scheduling control for MVC, I did find a software developing company that can provide a scheduling control targeted at ASP.NET AJAX. The name of the company is Telerik and they have a component named ‘RadScheduler’. I also believe that, from researching on the internet, it will be possible to integrate this control into the MVC application. As the control is not targeted at the MVC framework it is not possible for the control to interact with the database through actions within MVC controllers. Interaction between the control and the database can be accomplished through the use of WCF web services. The reason I decided against using the ‘RadScheduler’ was because it requires a licence fee and I found it towards the end of the implementation period and therefore did not have the time.

3.4 Utilising the Google Maps API with MVC

One of the main sections of the desktop application, Glimpse Desktop, is that it allows you to track resources, view job routes and address locations on a map. This functionality was also required in the web application that I developed, Glimpse Web. The desktop application is utilising Microsoft MapPoint for the mapping section of the application. It was not possible for me to use Microsoft MapPoint in the web application as the MapPoint web service is deprecated and was retired from November 18, 2011. The solution to this was to use Google Maps. I decided to use Google Maps for two main reasons. The first being that there is a lot of support online regarding using the Google Maps API in a web application. Secondly, the majority of users will know how to use Google Maps, with regards to navigation and zooming, as they will have already had some experience using Google Maps.

To integrate Google Maps into my application I would have to use the Google Maps JavaScript API. This proved to be quite challenging for me as I had never created an application that interacts with the Google Maps JavaScript API and I had not had a considerable amount of experience using JavaScript. To overcome this challenge I made a few test projects that mimicked the functionality that I required for the mapping section of Glimpse Web. I also searched for tutorials on the internet where the Google Maps JavaScript API was being used in an MVC application. After gaining more experience developing with the Google Maps JavaScript API, I was then able to integrate it into Glimpse Web.

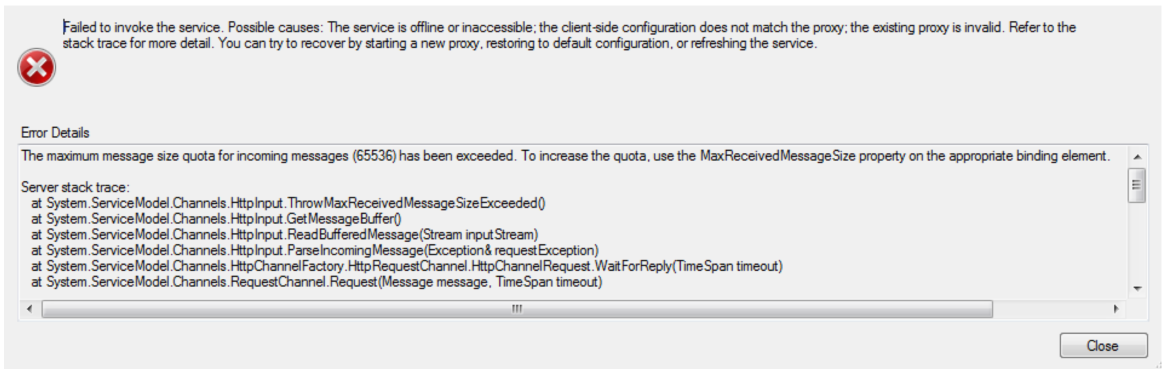
3.5 JavaScript Development

At the start of the implementation period of the project, my skill level of using JavaScript was quite low. Mainly because I had never undertaken web development that required me to use JavaScript. This proved to be a problem for me to begin with as I was not familiar with the JavaScript API. As this was the case I spent a lot of my time searching through the API for what I needed. Further on in the development period, my experience with JavaScript had significantly increased. Because of this, I was able to write JavaScript much more efficiently without having to search through the API for what I needed.

Another challenge I faced when using JavaScript was how to debug JavaScript code. The code resides in my views which are '.cshtml' files and it is not possible to place a breakpoint for the JavaScript in these files. To overcome this problem I undertook some research on the best way to debug JavaScript and found out that Google Chrome provides developer tools, one of which allows you to place breakpoints on JavaScript code. This allowed me to see the state of my variables to help me find the errors in my code.

3.6 Message Limit with WCF Framework

While I was developing a WCF web service to expose functionality relating to the diary/jobs section of the application, I encountered a problem. The problem was that WCF sets a default maximum message size of 65,536 bytes. One of the methods exposed by the service I created returns all of the jobs in the database. While testing this method, using the WCF Test Client, I was made aware that the message size returned by this method exceeds the default maximum message size. The error message I was presented with is shown in Figure 6.



```
Failed to invoke the service. Possible causes: The service is offline or inaccessible; the client-side configuration does not match the proxy; the existing proxy is invalid. Refer to the
stack trace for more detail. You can try to recover by starting a new proxy, restoring to default configuration, or refreshing the service.

Error Details
The maximum message size quota for incoming messages (65536) has been exceeded. To increase the quota, use the MaxReceivedMessageSize property on the appropriate binding element.

Server stack trace:
  at System.ServiceModel.Channels.HttpInput.ThrowMaxReceivedMessageSizeExceeded()
  at System.ServiceModel.Channels.HttpInput.GetMessageBuffer()
  at System.ServiceModel.Channels.HttpInput.ReadBufferedMessage(Stream inputStream)
  at System.ServiceModel.Channels.HttpInput.ParseIncomingMessage(Exception& requestException)
  at System.ServiceModel.Channels.HttpChannelFactory.HttpRequestChannel.HttpChannelRequest.WaitForReply(TimeSpan timeout)
  at System.ServiceModel.Channels.RequestChannel.Request(Message message, TimeSpan timeout)
```

Figure 6: The error raised when the maximum message size is exceeded.

To overcome this problem I had to edit the 'maxReceivedMessageSize' value in the configuration file for the service. While using the 'WCF Test Client' I was able to access the 'SvcConfigEditor' which allowed me to edit the configuration file for the service. After increasing the 'maxReceivedMessageSize' value in the configuration file the method was able to return all of the jobs in the database without raising the error previously mentioned.

I also had to edit the ‘Web.config’ file in my MVC application to ensure that the bindings for the ‘JobsService’ were correct. I increased the ‘maxReceivedMessageSize’ value for the ‘JobsService’ to match the value I used with the ‘WCF Test Client’.

4 Technology Choices

4.1 Data Layer

The data layer of the application handles the communication between the services layer and the database. It was decided that the best technology to use for this was the combination of Microsoft ADO.NET Entity Framework and LINQ. The reason behind the decision was that it makes interacting with a data source extremely easy and requires very minimal to zero plumbing code. By not having to write any plumbing code, this has made my code a lot easier to read and understand. It also allowed me to concentrate mainly on the business logic of the application and less so on the data access logic. Using LINQ has allowed me to issue queries to the database and have strongly typed objects returned that can be manipulated and saved back to the database. The data layer was the first layer of the application that was implemented.

The alternative to using the Microsoft ADO.NET Entity Framework and LINQ, to communicate with the database, would require explicitly writing the SQL queries. This was decided against as it could potentially make my code slightly harder to read and understand. As these problems are solved by Entity Framework it seemed like the logical decision was to rule out using explicit SQL queries to communicate with the database. Also, the SQL queries would not return strongly typed objects.

4.2 Services Layer

I decided to use the WCF framework to implement the services layer of the application. This decision was made towards the start of the project as I was creating the requirements and specification documents. One of the main reasons for choosing to use WCF was that a colleague highly recommended I look into it after informing them about the system that I had to develop. They have had a significant amount of previous experience using the framework and suggested that it would be suitable for the needs of the project.

The language choices available to me to write the WCF services were C# and Visual Basic. I felt that choosing C# would be a better option because I have had more experience using the language and it is more familiar to me. This has allowed me to concentrate on writing the application rather than also having to learn a new programming language.

4.3 Presentation Layer

For the presentation layer of the application I chose to use ASP.NET MVC 3. This was decided during the creation of my specification document. There were several reasons for choosing to use the MVC framework to develop the presentation layer. The main one being that the user interface logic is kept separate from the input logic. This makes it possible to change one of the components without having to make any changes to the other. Several other reasons for choosing to use MVC were that it makes validating user input and authenticating users relatively simple to achieve. It also allows restrictions to be placed on certain functionality based on a users rights level. The three previously mentioned items are three things that are required for this application.

The language choices available to me to write the controllers and models were C# and Visual Basic. I felt that choosing C# would be a better option for the same reasons mentioned in the previous section about the language choice for the services layer. That is, I have had more experience using the language and it is more familiar to me. Also, by already having experience with the language I have been able to concentrate on writing the code rather than having to learn a new programming language.

5 Reflection

5.1 Mistakes Made

From the start of the project to the finish, I feel that no major mistakes were encountered. By major mistakes I mean ones that were detrimental to the progression and overall success of the project. However, a few minor mistakes were encountered along the way.

The first of these mistakes is with regards to the amount of work I committed to. Due to my lack of experience in developing large applications, I completely underestimated the amount of work I could do within the time that was available. To compensate for this, I have had to work on the project for longer hours than I initially planned to which reduced the amount of time I could spend on other coursework and activities. By doing this I was able to get to a position where the majority of the application was finished with plenty of time to spare. The spare time allowed me to implement the remaining functionality of the application. I have learned a lot from this mistake and in the future I will be sure that I set myself a more realistic target taking into account the time frame available. If I were to undertake this project again I think I would the specification slightly.

I made two other mistakes at the beginning of the implementation phase of the MVC application. These mistakes are mentioned in more detail in the *Challenges* section of this document. They are related to not using the MVC framework in the way it was intended. The first mistake was that I was putting too much input logic into the views. Secondly, I decided not to use WCF data contracts for strongly types views instead of MVC models. It did not take long for me to realise where I was going wrong and these mistakes were corrected not long after they were encountered. By encountering these mistakes early on in development, it prevented me from potentially having to rewrite a lot of code. If I were to develop a web application utilising the MVC framework in the future I certainly would not make these mistakes again. By not doing so, the progress of the application would advance quicker than this application did at the start. If I were to undertake this project again, I would make sure that used the MVC framework as it was intended and used MVC models even if I thought a different solution may be more convenient.

5.2 Evaluation

Now that I have reached the end of the project I can look at what I have developed and determine whether the project was a success or not. The success of the project is determined by whether it meets the requirements that were set out at the beginning of the project. I believe that the project has been a success. There are only a few requirements that were not satisfied but they do not affect the overall functionality of the application. The requirements that were not met were originally classed as ‘nice to have’ items at the start of the project.

I believe that the technologies that I chose to use for this project were suitable for the system that was developed. They allowed me to develop the application to a high standard with the functionality that was required. At no point during the implementation of the system did I find myself thinking that I chose the wrong programming language or framework.

Throughout the project life cycle I faced several challenges. These challenges have been mentioned in this document and each of them were overcome. By overcoming these challenges, I have gained invaluable experience in dealing with such situations. This experience has stood me in good stead for when I ever face a similar situation in the future.

Most of all, this project has given me experience in developing a large scale application for a real-world client. This is something that I had never done before. Having undertaken such a task, I feel more confident that if I were to do something similar in the future I would be able to do it successfully and to a high standard. I also feel that my project management skills have increased significantly. This will allow me to estimate the time required to complete a certain task more accurately and ultimately be able to work out how much can be achieved in a given time frame. As well as what I have already mentioned, I have become a more proficient programmer.