FACULTY OF SCIENCE & TECHNOLOGY

BSc (Hons) Computing
May 2019

IMPROVING EXTENSIBILITY AND CODE QUALITY OF A KNOWLEDGE SHARING SYSTEM

by

Uchenna Okafor

Faculty of Science & Technology

Department of Computing and Informatics

Final Year Project

Abstract

Ladebug (pronounced Ladybug) is a debugging system that aims to make debugging errors easier for web developers. A proposal by the client was made to expand the scope of the system, however, the codebase of the system had undesirable software characteristics such as rigidity, fragility and immobility that made it expensive (time and effort required) to extend or maintain the application.

This study investigated the programming design principles and architectural patterns that aided the development of maintainable and extensible code. The study explored the SOLID design principles of object-oriented programming as a low-level approach to building an extensible system and the Onion architecture as a high-level approach to structuring the code of an extensible system. The main objective of the project was to lay the groundwork for future development of the system.

In conclusion, the study evaluated the implementation of the new codebase against the old codebase using code metrics, code review and code evaluation, and has found significant improvements in maintainability, extensibility and code quality. The artefact met its core objective as the new implementation made it easier for the system to be extended in ways that it was not planned for and laid the groundwork for future development.

With further work, the system would have the potential to become a public question and answer platform that could rival its competitors.

Dissertation Declaration

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

Copyright

The copyright for this dissertation remains with me.

Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

Client
The client of this dissertation is Sam Mullins, a web developer at Redweb.

Signed: Uchenna Okafor

Name: Uchenna Okafor

Date: 09/05/2019

Programme: BSc (Hons) Computing

Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed: Uchenna Okafor

Name: Uchenna Okafor

Date: 09/05/2019

Acknowledgments

Special thanks to Gernot Liebchen for supervising me throughout the journey of this dissertation and answering the millions of questions I asked him each week.

I'd also like to thank my client, Sam Mullins for all his help, support and direction throughout the duration of this project, without his expertise, contributions and cooperation this project would not be of the quality it is. Additionally, I'd like to thank the developers at Redweb for beta testing the system and providing useful feedback.

I'd also like to thank Tinodaishe Moyo for providing moral support and clarifying aspects of the project when I was unsure.

Finally, I'd like to thank my girlfriend for the moral support and encouragement throughout the duration of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 BACKGROUND AND CONTEXT

Background

In the context of web development, unhandled exceptions are typically displayed in the browser as an HTML error page rather than in the IDE (like in non-web development environments). Ladebug (pronounced Ladybug) is a debugging system that aims to make debugging errors easier for web developers. It shares similarities with question and answer platforms; the difference is that the questions are error pages and the answers are steps taken to fix the error.

Ladebug is comprised of two parts, a web app and a companion chrome extension.

- The role of the chrome extension is to detect error pages in the user's browser and display a list of suggested solutions if the same or similar error has been encountered and logged by another person.
  Once the user has fixed their error, the Chrome extension will ask how it was fixed. The user can either select an item from the list of previously suggested solutions or log alternative steps taken to fix the error. This feedback, along with the error page is stored in the backend server and is used in improving the future suggestions made by the system to other users with similar errors.

- The role of the web app is to act as the backend and as a repository of all the contributions made to the system through the chrome extension. Users can view and manually search for the items in the data stores.

See Appendix F for diagrams to better understand and visualize Ladebug conceptually. Additionally, the implementation of Ladebug can be seen in Appendix H.

Context

Ladebug was conceived by Sam Mullins (web developer at Redweb and the client of this project) and implemented by me during my placement year at Redweb (a digital agency focused on web development). It was designed to solve the problem of developers not being able to search for solutions to bespoke project errors. Also, when a developer fixed an error, there was no practical way of collecting that information, which became problematic when that developer had left the company, and the error they had fixed had reoccurred. Therefore, Ladebug was created to capture knowledge and to decrease the time it takes to debug errors.

Ladebug has been in use within the company for six months in the ASP.NET teams, and Redweb would now want to roll out Ladebug to more than one team. As they offer onsite services, they

want their externally based contractors to be able to use Ladebug even though they may not have access to the local Redweb network. As the proposed system is aimed to work for multiple teams, both inside and outside the local network, there is no reason why it could not be extended to other companies and teams other than Redweb in the future.

1.2 PROBLEM DEFINITION

Ladebug is currently hosted on Redweb's on-premise servers that are only accessible within Redweb's local network. Also, some functionality was explicitly built to work on Redweb's infrastructure. For example, the web app leveraged the company's active directory domain service for user authentication. This had the benefit of users not having to sign up and as they could use their existing work credentials to log in, but the downfall, is that the current implementation of Ladebug could not be set up to work outside of the local environment it was built for without a significant rework.

In addition, the current system was designed to only work with ASP.NET MVC error pages and did not have an easy way of adding support for new error pages for different web frameworks. This thereby limited Ladebug to the teams that work on ASP.NET MVC projects. Therefore, other teams within Redweb like the PHP team that uses the Drupal web framework could not use Ladebug. Finally, the existing system's current codebase is very rigid, tightly coupled and was not built with extensibility in mind, thus making it extremely difficult to extend and maintain without causing a breaking change.

1.3 PROPOSED SOLUTION AND RATIONALE

Make the existing system more globally accessible by utilising a cloud computing provider and leverage cloud computing technologies where possible, as this provides broader access to the system. Additionally, redesign and refactor the existing system with extensibility, maintainability and flexibility as the main priority.

1.3.1 Rationale

Requirements are rapidly changing, and technology is rapidly evolving faster than we can keep up. "It is a myth that we can get systems "right the first time." Instead, we should implement only today's stories, then refactor and expand the system to implement new stories tomorrow" (Martin, 2008, p. 158), therefore, designing software with maintainability in mind early on in the software development life cycle is crucial in ensuring its long-term success (Razina and Janzen, 2007).

1.4 Aim and Objectives

The core aim of this project is to refactor and redesign the existing system to be maintainable, extendible and flexible. The artefact will be a minimum viable product that lays the groundwork for future development.

The project's objectives are broken down into SMART objectives to ensure they are achieved

- Elicit core requirements from the client
- Refactor the existing system with a focus on extensibility, maintainability and flexibility
- Adhere to professional programming standards during implementation
- Lay the groundwork for the future development of the product
- Make the system more globally accessible

1.4.1 Success Criteria

The following criteria need to be met for the artefact to be successful

ID	Description
SC1	Obtain system requirements from the client
SC2	Research into the design patterns, programming practices and architectural patterns that enables building maintainable, extendible and flexibility software
SC3	Migrate the existing system from a local environment to a cloud computing provider
SC4	Refactor the existing system and follow industry standard practices for software development
SC5	Implement the features in the system requirements
SC6	Obtains and evaluate feedback from the users that will test the new system
SC7	Obtain critical feedback from experienced developers who will review the code quality of the artefact and have them gauge their level of confidence in extending parts of the system without fear of breaking the entire system
SC8	Receive feedback from the client on artefact and suggestions on how the system could be improved and the future direction of the product

Table 1 - Success criteria

1.5 Risk Analysis

Evidence of risk analysis can be found in the project proposal form in Appendix A.

2 BUILDING EXTENDIBLE SOFTWARE

2.1 OVERVIEW

Building extendible and maintainable software is very complex and requires careful consideration of the engineering practices used. This section will discuss the development of extensible software, with a focus on the appropriate design and architectural patterns. The design patterns serve as a low-level approach to writing extendible code whereas architectural patterns serve as a high-level view of the structure of the codebase.

2.2 PRINCIPLES OF GOOD SOFTWARE DESIGN

Good design practices need to be established from the beginning of the software development lifecycle (SDLC) because 70% of software costs is in its long-term maintenance (Lee et al. 2013). Therefore, using good design principles to enable the engineering of maintainable software at the beginning of the SDLC is a top priority. Otherwise, it becomes increasingly costly to maintain poorly designed code.

According to Martin (2002), there are three characteristics of bad software design that should be avoided

- Rigidity - making changes in one part of the code breaks another part of the system
- Fragility - code can break and in places that have no relationship with the modified code
- Immobility - code cannot be easily reusable because it is tightly coupled

The current system has all these bad characteristics, for example, the code snippet in Figure 1 shows a class that is tightly coupled to the implementation of elastic search, and this makes the code immobile. The `ErrorsRepository class` instantiates the `ElasticSearchSevice,` but if the `ElasticSearchService` class were modified to have a mandatory constructor argument, every class that creates an instance of `ElasticSearchSevice` would break, which is undesirable as it means the code is rigid.

```
namespace Ladebug.Repositories
{
    10 references | Uchenna Okafor, 19 days ago | 1 author, 1 change
    public class ErrorsRepository : Repository
    {
        private readonly ElasticSearchService _esService;

        5 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public ErrorsRepository()
        {
            _esService = new ElasticSearchService();
        }
```

Figure 1 – Example of rigid and immobile code from the existing system

This subsection will explore programming practices that resolve these bad characteristics by exploring how to write code that is more maintainable, extendible and flexible.

SOLID Design Principles

The SOLID design principle first introduced by Robert Martin (2008) lists fives design principles for writing maintainable, extendible and robust code in object-oriented programming languages. SOLID stands for Single Responsibility, Open/Close, Liskov's Substitution, Interface Segregation, and Dependency Inversion. The five design principles aim to address the issues of rigidity, fragility and immobility that leads to bad software design.

2.2.1 Single Responsibility Principle

The single responsibility principle (SRP) states that programming constructs such as a class, module or function should have a single responsibility and should never try to do more than one thing. This principle assures developers that a class, module or function they are using only does what it is supposed to do well. The code snippet in Figure 2 violates this principle as the `Encrypt` method is trying to read from a file, encrypt it and write the encrypted string back to the file. The class is doing more than its method name "`Encrypt`" suggests, which is a problem as it does not allow data source flexibility, as the developer may want the plain text to be read from a database or an external service.

```
public class EncryptionService
{
    public void Encrypt(string sourceFileName, string targetFileName)
    {
        // Read content
        byte[] content;
        using(var fs = new FileStream(sourceFileName, FileMode.Open, FileAccess.Read))
        {
            content = new byte[fs.Length];
            fs.Read(content, 0, content.Length);
        }

        // encrypt
        byte[] encryptedContent = DoEncryption(content);

        // write encrypted content
        using(var fs = new FileStream(targetFileName, FileMode.CreateNew, FileAccess.ReadWrite))
        {
            fs.Write(encryptedContent, 0, encryptedContent.Length);
        }
    }

    private byte[] DoEncryption(byte[] content)
    {
        byte[] encryptedContent = null;
        // put here your encryption algorithm...
        return encryptedContent;
    }
}
```

Figure 2 – A code snippet of (Schenker, 2009)

2.2.2 Open-Closed principle

The open-closed principle (OCP) states that classes, modules or functions should be open for extension but closed for modification. This principle encourages the development of high-quality and maintainable code, as it requires that classes follow the single responsibility principle and never have any need to change because their implementation should be final. This means that as the requirements of a project changes, it becomes easy to maintain the code as new code is written on top of the existing code, rather than introducing bugs by modifying the existing code.

2.2.3 Liskov Substitution Principle

The Liskov substitution principle (LSP), first coined by Barbara Liskov (1988) states that subtypes must be consistent and substitutable with their base types (Dooley, 2011) and this is important because it requires that subtypes of a derived class should be consistent and should behave as the subtype infers. This principle encourages code to be more flexible.

2.2.4 Interface Segregation Principle

The interface segregation principle (ISP) states that no client should be forced to depend on methods it does not use (Martin and Martin, 2002). The Interface segregation principle also extends the single responsibility principle as ISP requires that interfaces with multiple responsibilities be broken down into smaller more specific interfaces. The goal is to reduce the impact of code changes, which makes the code more flexible as functionality is more granular, thus easier to maintain and refactor.

2.2.5 Dependency Inversion principle

The dependency inversion principle (DIP) states that both high-level and low-level modules should not depend on concrete implementations but on abstractions. The rationale is that depending on concrete implementations means depending on volatile functionality; instead, software architecture should favour the dependency of stable abstract interfaces (Martin, 2017, p. 100). This is advantageous as code can be extended and reused in ways not explicitly planned for (Seeman, 2011, p. 16). Software requirements are continually changing, so this principle ensures the future extensibility and maintainability of software.

2.2.5.1 Dependency Injection

The dependency inversion principle is commonly achieved using dependency injection. Dependency injection is a technique that enables loose coupling between different parts of an application (Smith, 2019). Looser coupling is advantageous because it allows for the isolation of code and means changes in one part of the code will not break another part of the application.

Figure 3 shows a refactored code snippet of Figure 2. The refactored code shows the dependency of the read and write functionality is inverted towards the `IReader` and `IWriter` interfaces. This can be further visualized in Figure 4 whereby the `EncryptionService`, `FileReader`, `DatabaseReader`, `FileWriter` and `WebServiceWriter` classes are all dependent on the stable abstractions. Using dependency injection, concrete implementations can be injected in place of the abstractions, and this is desirable as it allows for modules to be easily swapped or modified without having any unexpected consequences on other parts of the application, therefore resulting maintainable, extendible and flexible code.

```
public class EncryptionService
{
    public void Encrypt(IReader reader, IWriter writer)
    {
        // Read content
        byte[] content = reader.ReadAll();

        // encrypt
        byte[] encryptedContent = DoEncryption(content);

        // write encrypted content
        writer.Write(encryptedContent);
    }

    // rest of code omitted for brevity...
}
```

Figure 3 – A code snippet illustrating the dependency inversion principle (Schenker, 2009)

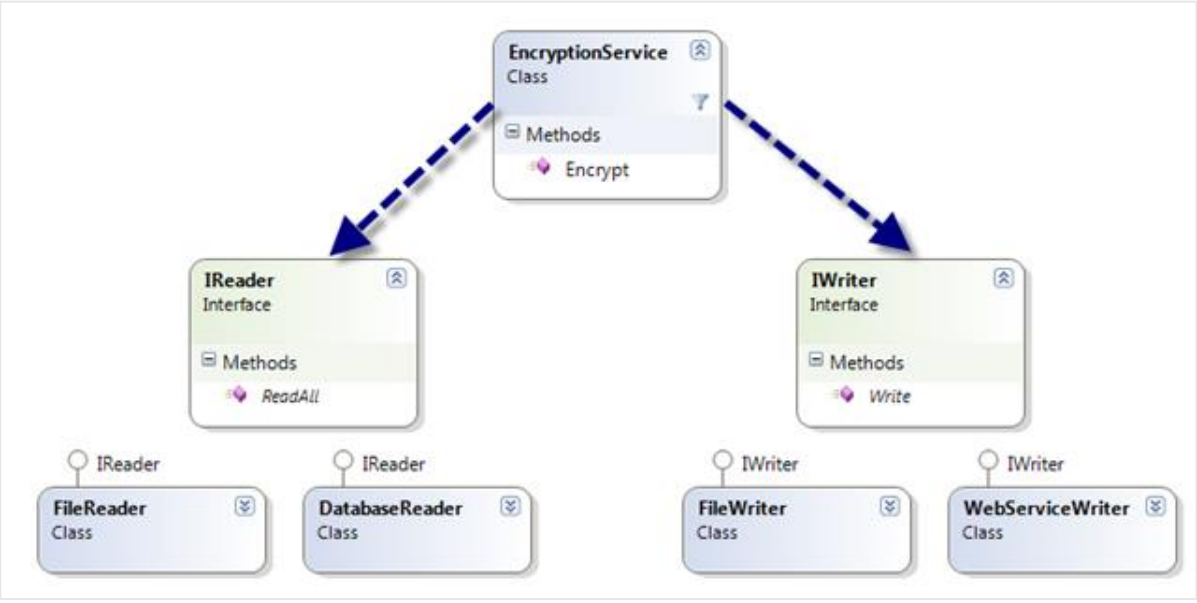


Figure 4 - Class diagram illustrating the dependency inversion principle (Schenker, 2009)

2.2.6 Conclusions

In conclusion, the SOLID design principles provide a set of guidelines for writing code that is maintainable, extendible, flexible and testable. A core theme around of the proposed system is to lay the groundwork for the future development of the new system, and the SOLID design principles offer programming practices that enable the development of code that is loosely coupled, highly maintainable and highly extendible.

2.3 Architectural Patterns

2.3.1 Overview

The design principles mentioned in the previous chapter serves as a micro overview of how the code will be written and implemented. Architectural patterns serve as a macro overview of how the codebase will be logically structured.

The following section will be discussing suitable architectural patterns, and how they contribute to the maintainability and extendible of the proposed system. Finally, followed by a selection of the most appropriate with rationale and justifications.

2.3.2 Problem with the existing architecture

The problem with the architecture of the existing system is that it is monolithic. All the classes, functions and modules are placed under one project inside the Visual Studio solution (a solution can contain many projects), like the solution structure in Figure 5. This monolithic structure makes it hard to maintain as the separation of concerns is achieved using folders which is very limiting (evidence can be found in Appendix S). This approach isn't necessarily wrong as it works well for small projects but can be an issue for larger projects, because as the project grows the codebase is buried deep into many folders and there is not a clear distinction on which folders depend on each other, and this unorganized structured gives rise to spaghetti code (Smith, 2019).

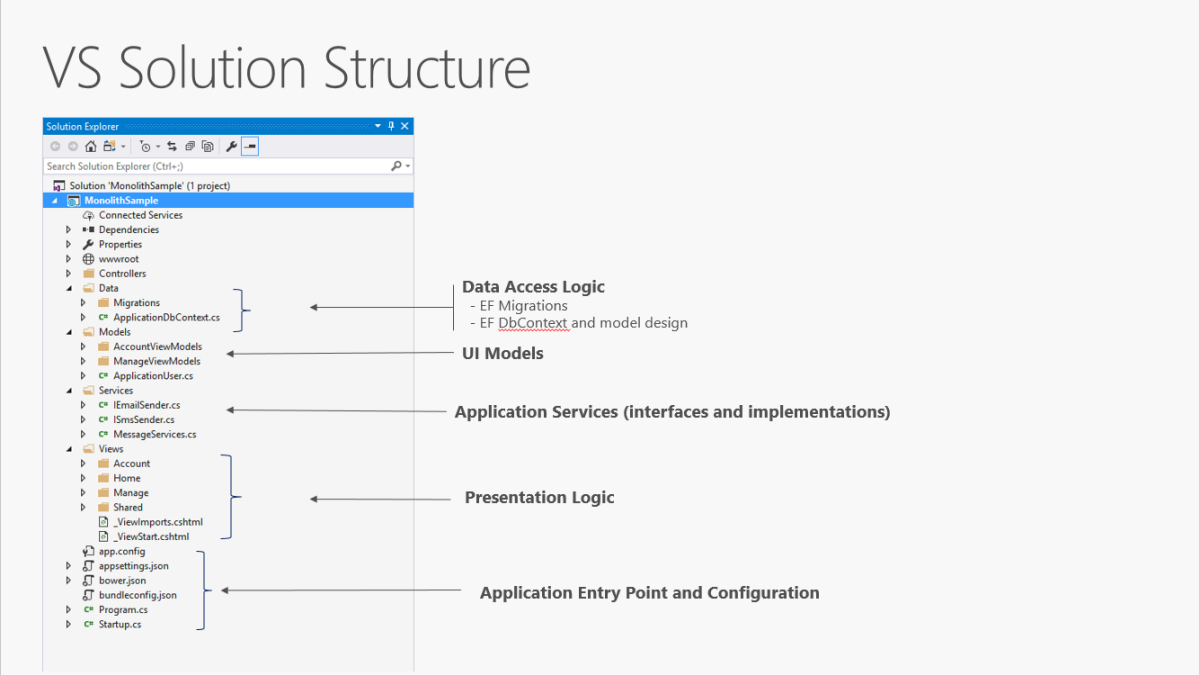


Figure 5 - Default visual studio solution structure (Smith, 2019)

2.4 LAYERED ARCHITECTURES

Layered architecture is an approach that involves splitting up a single project solution into a multi-project solution in order to address the issues presented by a monolithic application structure. Layers help keep growing codebase organised, making it easier to maintain and design.

2.4.1 3-Tier Application Architecture

The 3-Tier application architecture commonly referred to as the N-Tier architecture is a traditional architectural approach in web development and it consists of three layers as shown in Figure 6.

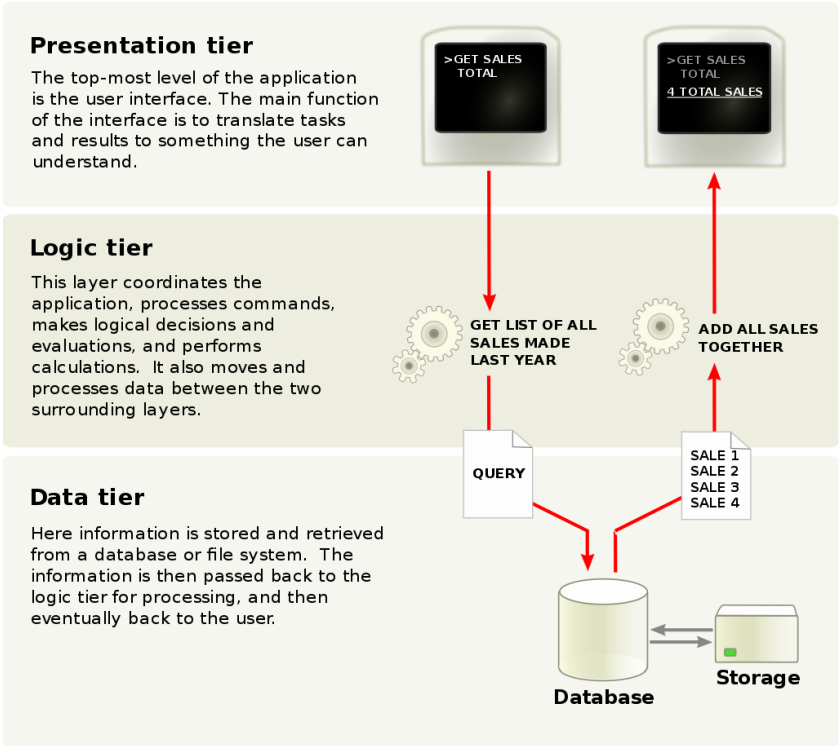


Figure 6 - 3-Tier Application Architecture (Wikipedia, n.d.)

Presentation Layer

The role of the presentation layer is to transform data from the business logic layer into information that can be understood by the user. This layer is typically the layer where the HTML content is developed.

Business Logic Layer

The role of the business logic layer is to perform calculation, processing and logical decisions that are relevant to the application domain. It also acts as an intermediate medium between the presentation and data layer. It takes in data from the data layer, processes it so that the presentation layer can process the information.

Data Layer

The role of the data layer is to store and retrieve persistent data. This is typically a class that accesses the database, performs queries and returns the result to the business logic layer for further processing.

The 3-tier application architecture is advantageous as its layered approach makes it easier to separate the different concerns of the application into their own logical layers. This potentially allows developers to work on different tiers simultaneously without impacting the other tiers. In conclusion, this architectural pattern offers logical separation of application functionality, which makes it easier to maintain and manage code. Major drawbacks of this architecture are the high setup costs and complex structure (Kambalyal, 2010).

2.4.2 Onion Architecture

Onion architecture first introduced by Jeffrey (2008) is an architectural pattern that is inspired by the layers in an onion as shown in Figure 7.

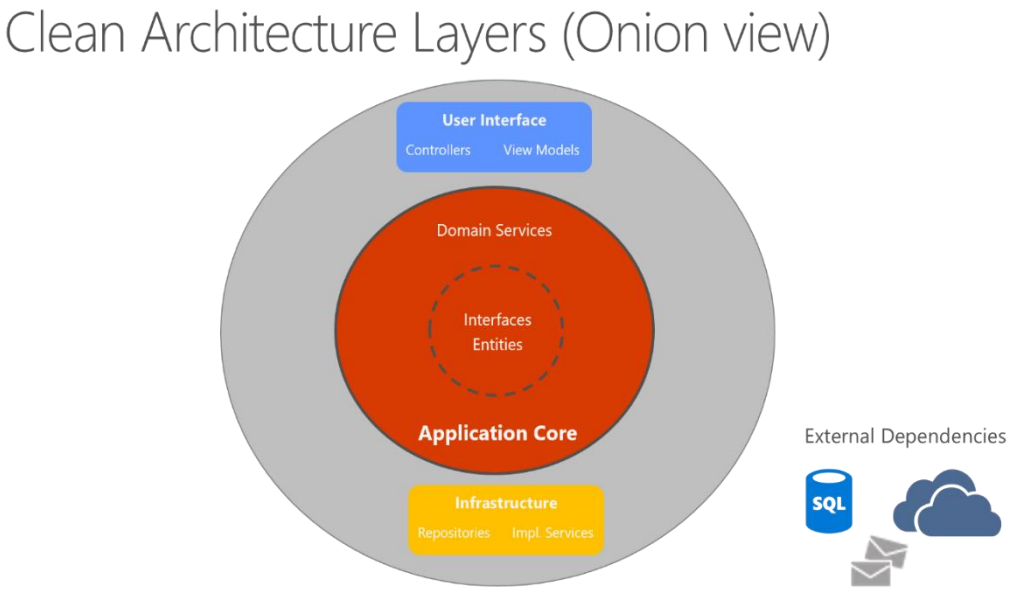


Figure 7 - Onion Architecture Layers (Smith, 2019)

The layers are usually depicted as four layers, but there are no guidelines on how the onion architecture is implemented as it is different per application. The general layers are as follows.

Domain Layer

This is the innermost layers, and it is where all core domain objects and abstractions are defined, which will be later implemented by the outer layers.

Services Layer

This layer implements the functionality pertaining to the business logic of the application.

Persistence Layer

This layer implements the functionality related to data persistence. Typically, this is where database related code and configurations are implemented.

UI Layer

This is the outermost layer, and it is the layer that depends on all the inner layers and uses dependency injection to build a loosely coupled relationship with the inner layers through interfaces defined in the core domain layer.

The general concept behind the onion architecture is in the separation of concerns using different layers. The further up the layers, the higher level the code is. This means, the innermost layers, present the abstractions and core domain logic, whereas the outer layers implement that domain logic. This results in dependency only pointing inwards, which prevents circular dependencies. This is desirable as the effects of modifying or deleting a layer will not be as impactful if the dependencies were circular as in the 3-Tier application architecture.

The disadvantage in the onion architecture is in its extremely high costs in setup due to the complexities of the layers (Pal, 2018).

2.4.3 Conclusion

The research into architectural patterns has revealed that layered architecture in comparison to monolithic architecture is more complicated. There is a high cost in setup. However, simplicity is sacrificed when designing complex and extensible applications.

The 3-Tier application architecture seems to be very limiting with only providing three that are all dependent on each other. This creates a circular dependency that leads to tangled code that is difficult to maintain. Whereas the onion architecture provides a different approach to structuring the

layers whereby each layer is independent, which also makes them very testable and maintainable as they can be changed without a significant effect on rest of the application.

Therefore, the onion architecture will be chosen as it offers a testable and maintainable architecture style due to the independent nature of the layers and due to the integration with SOLID design principles. This architecture will help in structuring the new system to facilitate the future lifecycle of the system.

3 METHODOLOGY

This section will explore the different software development methodologies suitable for building high quality and extendible systems, evaluate their appropriateness for the project, followed by a selection of the most suitable methodology.

3.1 DEVELOPMENT METHODOLOGY

3.1.1 Rapid Application Development Model

Rapid Application Development model (RAD), involves functionality being implemented as individual components by different team members in parallel, then later integrated into a final working prototype (Paul et al. 1999). This approach is advantageous as it results in the development of systems that are high quality and highly reusable. As every developer is working on an individual part without knowledge of what others are doing, the system must be designed with a high level of abstraction and reusability by principle. Consequently, there is an increased level of complexity in development, as the design must be carefully planned. Therefore, highly skilled developers are required for RAD to work.

3.1.2 Prototyping Model

The prototyping methodology aims to refine a product until it satisfies the client (Carr and Verner, 1997). The prototyping model advocates a throwaway prototype that is built and tested, then shown to the client in order to gauge the look and feel of the system. This process is repeated until the desired prototype is reached as shown in Figure 8 and once the client is satisfied with a prototype, the final product is subsequently built from the approved prototype. The heavy involvement of users in this model, means, there is a rich amount of feedback. Therefore, unclear aspects of the system and missing functionality are identified earlier, and that can help in designing a system that will satisfy the client (Centers for Medicare & Medicaid Services (CMS), 2005). However, this allows for the scope of the initial system to rapidly increase, and that can be expensive.

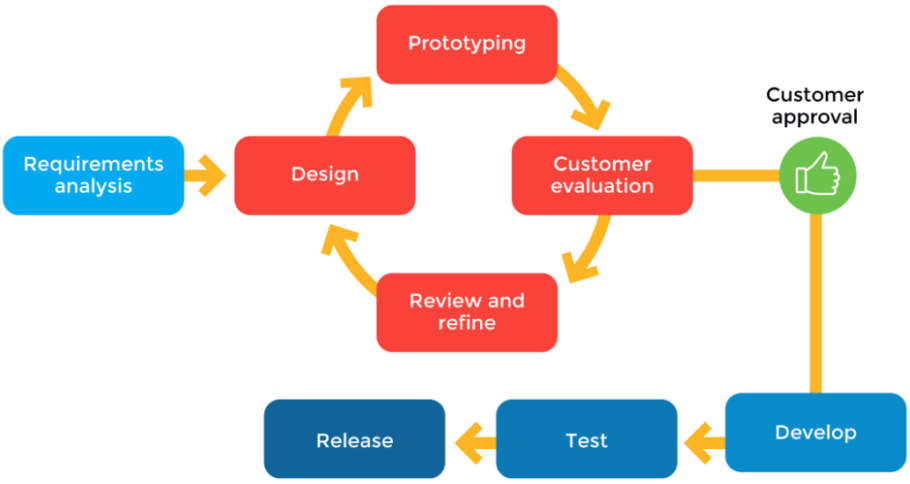


Figure 8 - Prototype model (Fronczak, 2019)

3.1.3 Extreme Programming Model

The Extreme programming (XP) model, established by Beck (1999) is an agile approach (illustrated in Figure 9) that emphasizes frequent releases in short development cycles, with a focus on code quality and limiting the project's scope to the minimum required functionality. XP is heavily focused on refactoring due to the importance of quality. This focus on code quality is achieved through the four main activities in XP, which are coding, testing, listening and designing. This continuous feedback from the client and testing that drives the production of higher quality code. However, XP is very dependent on the developers involved. Therefore, developers involved need to be on the same level for XP to succeed.

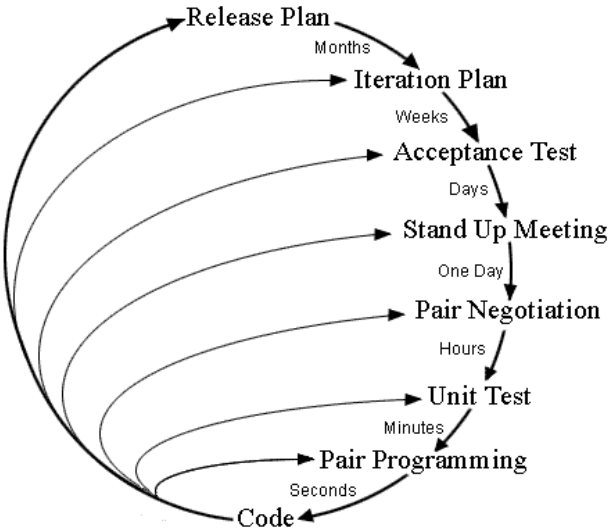


Figure 9 - Extreme Programming Model (Wells, 2001)

3.1.4 Chosen methodology

As the core requirements are to refactor the existing system, a methodology that favours incremental development will be ideal. This immediately dismisses RAD as a suitable methodology, although RAD offers high-quality systems and high reusability through isolated parallel development, the disadvantage of it not being suitable for a single developer projects makes it an unsuitable methodology choice.

The prototyping model on the other hand results in frequent prototypes being produced, this would enable the objective of building the minimum viable product. This is advantageous as it provides an early feel of the system before further development work. This would suggest that the prototyping model would be a suitable choice. However, the prototyping model is most suited when project objectives are unclear (Centers for Medicare & Medicaid Services (CMS), 2005). Therefore, this was not the chosen methodology as the project requirements are well defined.

Finally, XP consists of short incremental release plans where the system is viewed by the client and tested by developers in order to generate feedback, before moving onto the next iteration with a bias in implement high priority requirements. The flat hierarchical structure of XP makes it suitable for small teams; although XP requires teams of more than one, it can also work with a single person team (Agarwal and Umphress, 2008), especially if the developer knows enough to stand in for the client. Considering all the advantages, disadvantages and project requirements, XP seems to be the most appropriate methodology and therefore is the methodology of choice.

4 REQUIREMENTS AND ANALYSIS

The researcher has in-depth background knowledge and understanding of the project due to heavy involvement throughout the existing project's lifecycle. Therefore, a detailed formal interview process to elicit the requirements is not necessary as the researcher has enough knowledge to stand in for the client.

4.1 REQUIREMENTS ELICITATION

Semi-formal emails were exchanged with the client, which discussed the general direction of the project in context to this dissertation and a list of core feature requirements (evidence of exchanges can be found in Appendix D).

From the elicitation of core requirements from the client, a fuller set of requirements were engineered. The requirements were split into functional, non-functional and technical requirements and were prioritized using the MoSCoW prioritization technique (Clegg and Barker, 1994) to ensure that implementation of higher priority features. Evidence of the requirements can be found in Appendix E.

4.2 PROJECT MANAGEMENT

GitHub's project board was leveraged as a project management tool to create tasks based on requirements (see Figure 10), so their progress during development can be tracked.

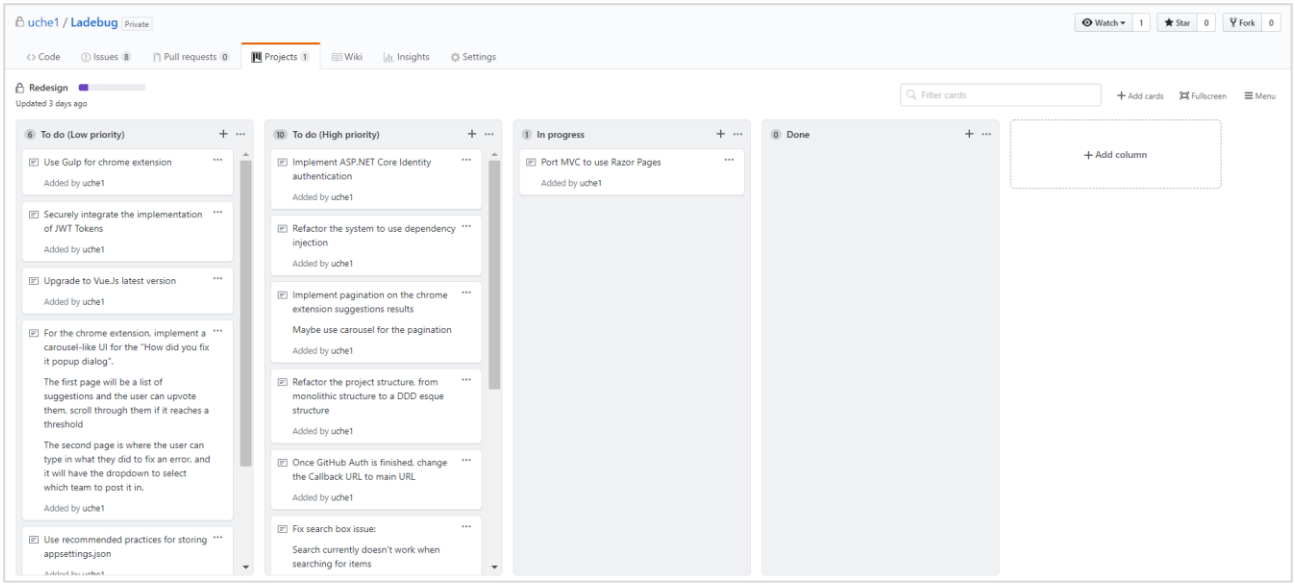


Figure 10 – Initial project board for the Ladebug in GitHub

4.3 EVALUATION

The final artefact will be evaluated in the following ways

- Measure how well the project meets the success criteria and elicited requirements
- Static code analysis of the existing and new codebase
- Conduct a code review and evaluation of both the existing and new codebase in order to determine the extent the core theme of the dissertation has been met.
- Having a group of developers at Redweb beta test the final artefact and report feedback

5 DESIGN

This section will discuss the common code related design pitfalls of the existing system, drawing on the themes mentioned in the previous sections as a solution. Additionally, non-code related design choices will be detailed and justified.

5.1 PITFALLS OF EXISTING SYSTEMS DESIGN

The common issue surrounding the existing code was in its tightly coupled implementation, which negatively affected the extensibility and flexibility of the codebase as mentioned in chapter 2.2. This chapter will build on the previously discussed themes and discuss improved designs choices and changes to the web app and Chrome extension.

5.2 USE OF INTERFACES, DEPENDENCY INVERSION AND DEPENDENCY INJECTION

Interfaces (object-oriented programming concept) and abstract classes will be used throughout the refactoring process of the existing code as they enable the inversion of dependency, which will be crucial in writing better and loosely coupled code. In tandem with the use of interfaces and dependency inversion, dependency injection will play a significant role in tying together the interfaces and their concrete implementations at runtime. Thus, the use of dependency injection will be vital in building a loosely coupled application (Wenzel et al. 2019).

5.3 PROPOSED REDESIGN OF THE EXISTING MODULES

5.3.1 Refactored search service

An existing code snippet of the existing system as shown in Figure 11 shows the `ErrorSearchService` class, which is responsible for searching using Elastic Search. The problem with the code is that it is tightly coupled to the implementation of Elastic Search, therefore, making it difficult to replace or swap the search engine provider (Elastic Search) without breaking the code.

```
public class ErrorSearchService
{
    private readonly ElasticSearchService _esService;

    public ErrorSearchService()
    {
        _esService = new ElasticSearchService();
    }
```

Figure 11 - Code snippet of the existing system code design

The new, improved design would be to abstract the implementation of the search engine provider using an Interface and have the `ErrorSearchService` depend on a stable abstraction/interface as

shown in Figure 12. Therefore, allowing for the implementation of the search engine provider to be flexible, and configured at runtime using dependency injection.

```
public class ErrorSearchService
{
    private readonly ISearchProvider _searchProvider;

    public ErrorSearchService (ISearchProvider searchProvider)
    {
        _searchProvider = searchProvider;
    }
```

Figure 12 - Improved code design of snippet in Figure 11

5.3.2 Redesign of the error parser

In the context of this project, error parsing is the process of extracting and displaying relevant information from the HTML document of an error page, as a JSON object (evidence of this process can be found in Appendix G). This process enables high precision and fast search queries, as the lightweight JSON object does not require extensive resources from the search engine (Elastic Search) to process.

The implementation of the existing error parser as shown in Figure 13 is a self-contained static class that was just called from a single line as shown in Figure 14. The existing implementation is arguably maintainable due to its simplicity; however, it lacks extensibility. The current implementation will change to facilitate the change in requirement for Ladebug to support multiple web frameworks.

```
public static class ErrorParser
{
    private const string YellowScreenPattern = "^Server Error in '(.*)' Application.$";
    private const string LineNumberPattern = @"^Line \d+:";

    public static Error ParseHtml(string html)
    {
        var document = new HtmlParser().Parse(html);

        if (!IsErrorPage(document))
        {
            return null;
        }

        var font = document.QuerySelector("font");
```

Figure 13 - Code snippet of the existing systems error parser

```
var parsedError = ErrorParser.ParseHtml(htmlErrorPage);
```

Figure 14 - How the error parser is called in the existing system

5.3.2.1 New design

The redesigned and refactored error parser framework draws upon concepts from object-oriented programming such as inheritance and polymorphism and all the SOLID design principles, in particular, the dependency inversion principle as mentioned in section 2.2.5 as a solution to solving the problems of extendibility, flexibility and maintainability. Figure 15 shows a class diagram of the new design of the error parsing framework, which consists of two major components.

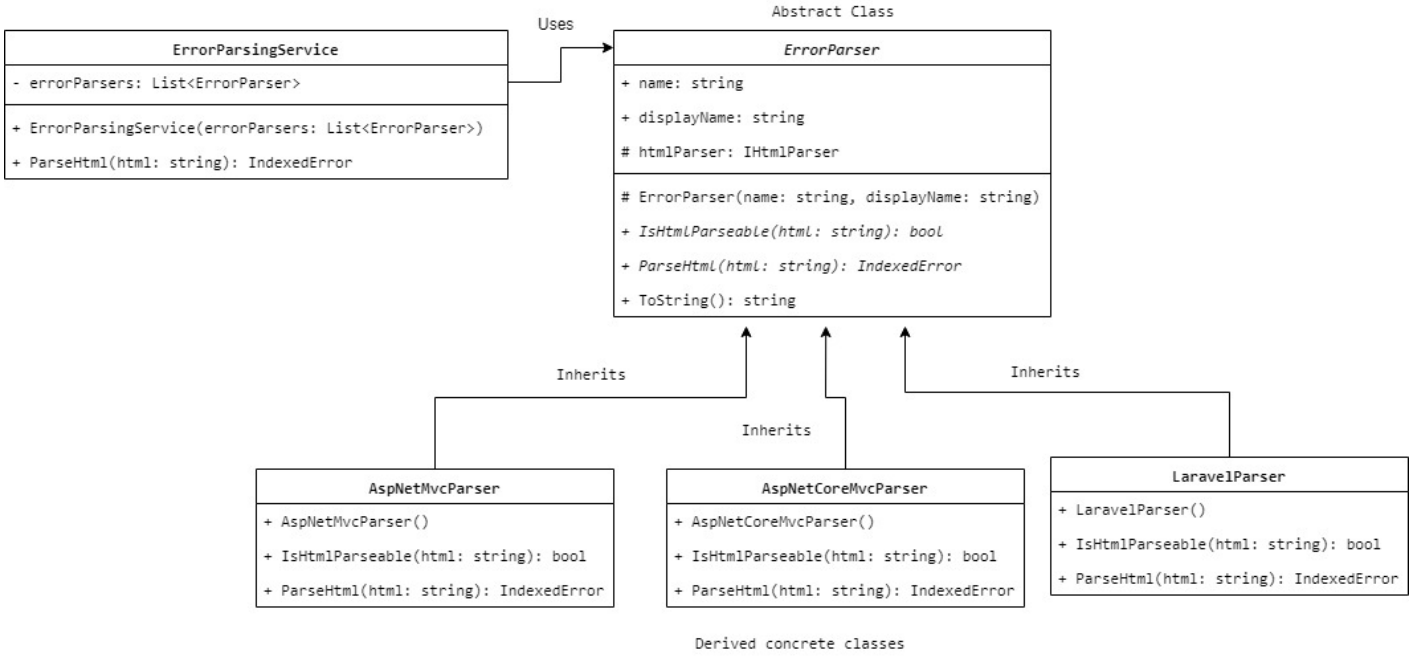


Figure 15 - Conceptual class diagram of the redesigned and refactored error parsing framework

The first component is the `ErrorParser` class, which is the base class that all error parser modules derive from. The `ErrorParser` class is an abstract class that defines the common characteristics and contracts that all error parser modules must agree to implement. The two most imperative contracts are.

1) The `IsHtmlParseable` function, which returns a Boolean value indicating if the HTML document passed in the parameter is a valid error page for that error parser module. For example, if the HTML document of an ASP.NET error page was to be passed to a Laravel parser, it will return false as that module would not recognize its HTML content.
2) The `ParseHtml` function, which will parse the HTML document of an error page and return an object

The second component of the proposed error parsing framework is the `ErrorParsingService` class. This class is a container that contains an array of the abstract `ErrorParser` modules. Based on this conceptual design, an HTML document of an error page can be parsed by calling the `ParseHtml` method in the `ErrorParsingService`, which iterate through the array of `ErrorParser` modules, and the first module to return true on the `IsHtmlParseable` method will parse the HTML document.

This new design solves the problem of extendibility and maintainability the existing error parser suffered from. It is extendible and maintainable as each `ErrorParser` module is a self-contained class and it is extendible as the `ErrorParsingService` class is loosely coupled, it only depends on the `ErrorParser` class. Dependency injection will be used to inject the implementations of the `ErrorParser` modules (i.e. `LaravelParser`, `AspNetMvcParser`) to the constructor of the `ErrorParsingService` at runtime.

5.3.3 Redesign of the existing system architecture

The new architecture will consist of multiple layers which draw upon the onion architecture discussed in section 2.4.2. The separation of concerns will be the key focus behind the use of a layered architecture (Wenzel et al. 2019), which can make the system easier to maintain as functionality is intuitively and semantically separated. The diagram in Figure 16 shows a conceptual design of the new layered architecture design.

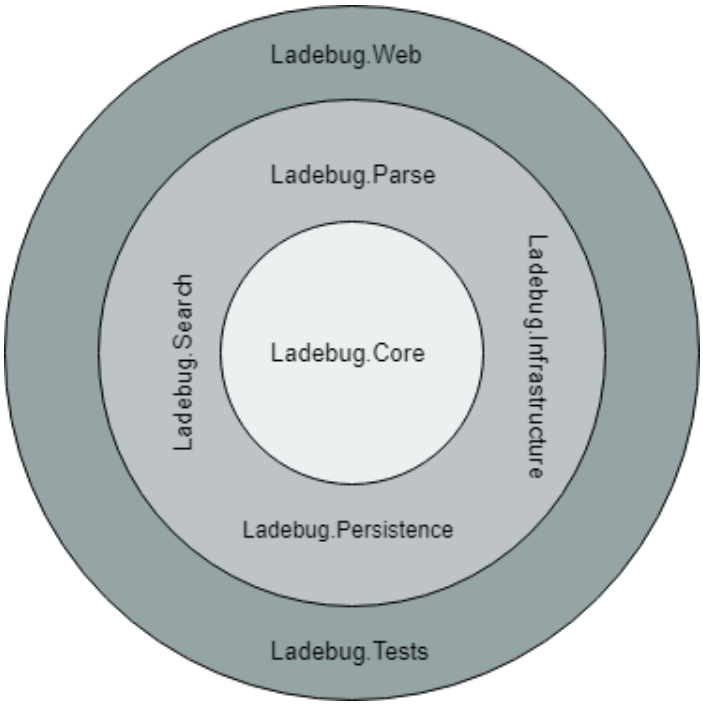


Figure 16 - Onion view of the proposed systems architecture

5.3.4 The project layers

Table 2 describes each layer and its purpose within the context of the system.

Ladebug.Core
This layer is the core of the application, and it will only contain core domain objects and abstractions as discussed in sections 2.4.2, 5.1 and 5.3. The interfaces/abstractions defined in this layer will be implementations in the other layers. This layer cannot depend on any of the other layers as that would violate its no external dependency rule.
Ladebug.Infrastructure
This layer will contain generic application-wide implementations and helper functions.
Ladebug.Persistence
This layer will contain the implementation of all database related code. For example, this is where Entity Framework model configurations and repositories will be implemented.
Ladebug.Search
This layer is where all the answer searching related code will be implemented. For example, the search engine search service and the StackOverflow search service will be implemented in this layer.
Ladebug.Parse
This layer will contain all parsing related code. This is where the different error parser modules will be implemented.
Ladebug.Tests
This is where all the unit tests will be written and executed
Ladebug.Web
The web layer is the outermost layer, and it is where the UI of the MVC application will be implemented. This layer will call the functions and modules in the inner layers to orchestrate the implementation of the web app, in addition, it will also be where the dependencies are injected/registered (composition root).

Table 2 - Explanation of the proposed architecture layers

5.4 Web App Design Choices and Rationale

5.4.1 Entity Relationship Diagram

As the requirement of the existing system has changed, so will the ERD. The new ERD has mainly accommodated for the addition to teams. Evidence of the existing ERD and the new proposed ERD diagrams can be found in Appendix I.

5.4.2 Low Fidelity Designs

Low fidelity screen designs were created from the requirements to gauge a better visual understanding of the look and feel of the final artefact (Chrome extension and web app). Evidence of low fidelity designs can be found in Appendix J.

5.4.3 Redesigning Authentication

Authentication for the current system was limited to Redweb's internal network infrastructure. It leveraged Redweb's active directory services to authenticate users; this meant users could log in to Ladebug using their work account without having to sign up. As the requirement of the project has changed, the current authentication mechanism must be redesigned.

The redesigned system will allow users to create an account and use it to log in. In addition to a local account, the option for users to be able to register and log in with an external account using OAuth would assist in providing more authentication options. According to a report by Bitglass (2018), Office 365's adoption reached 56.3% in 2018 for all companies globally. This means many companies use Microsoft's cloud services (including Redweb). As mentioned in section 1.2, Ladebug could be used by organizations and companies in the future with the new addition of teams, therefore justifying the use of Microsoft as an external login choice.
Also, GitHub and StackOverflow will also be added as an option for external login choice, because both platforms are used by developers, which is the target audience for Ladebug.

5.4.4 Upgrade to ASP.NET Core

ASP.NET MVC 5.2 (Microsoft, 2014), a web application framework by Microsoft released in 2014 was used to build the existing system. However, a newer redesign of the framework, ASP.NET Core MVC 2.2 (Microsoft, 2019) which offers many improvements over its predecessors is now available. ASP.NET Core is a cross-platform, open-source framework for building modern cloud-optimized web applications.

ASP.NET Core will be the framework of choice for the implementation of the project, due to its new features and how they assist the development of the project requirements and theme. Below is a list of characteristics that make ASP.NET Core a suitable framework to use.

Modular

ASP.NET Core introduces a more modular approach to development with the framework being split into packages which can be downloaded as needed. This results in less software bloat and means unnecessary dependencies do not cause an issue in the codebase.

Built-in dependency injection

In ASP.NET MVC 4/5 a separate DI container had to be configured manually by the developer as an external package which can be time-consuming. Whereas, ASP.NET Core is built around and has built in support dependency injection. This means the framework is very loosely coupled and extendible by design and it supports makes ASP.NET Core a suitable framework for implementing layered architecture (Smith, 2019) and loosely coupled code.

Identity

ASP.NET Core introduced ASP.NET Core Identity, a fully-fledged customizable membership system with built-in support for configuring authentication for external OAuth providers. This will make it easier to implement and integrate the authentication specifications mentioned in section 5.4.3 as the framework already does the heavy lifting.

6 IMPLEMENTATION

This section will highlight the implementation of the artefacts, its limitations and a handful of refactored code changes that were discussed in the previous chapters.

6.1 OVERVIEW

6.1.1 System Overview

The refactor and redesign of the existing system, in addition to the implementation of the user requirements has led to some overall changes to the macro processes of the system. Figure 17 shows a macro overview of the existing system, while Figure 18 shows the macros process of the new implementation. The macro process diagrams illustrate how the Chrome extension interacts with the backend APIs to retrieve answers.

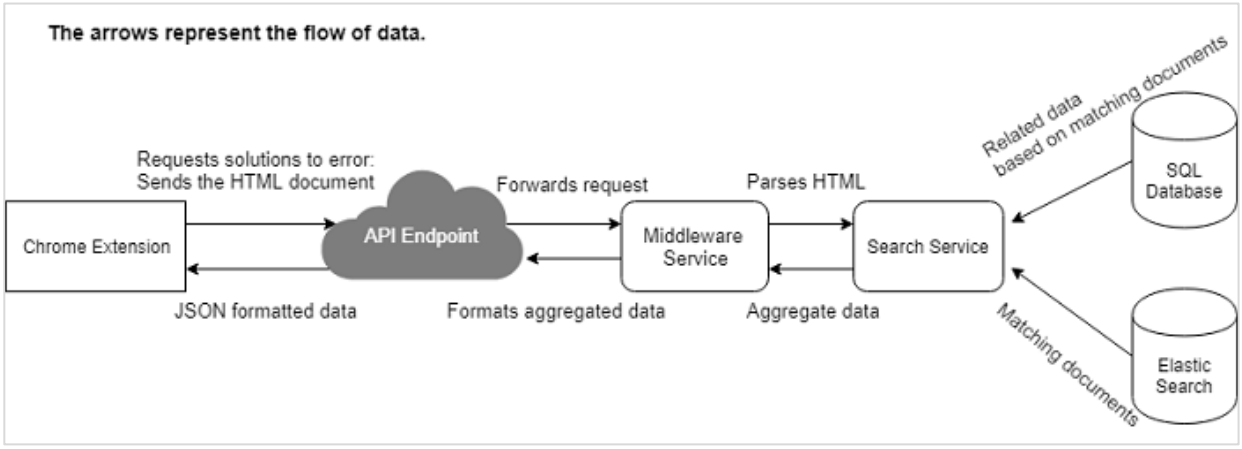


Figure 17 – An overview diagram of how the existing system works (Okafor, 2018)

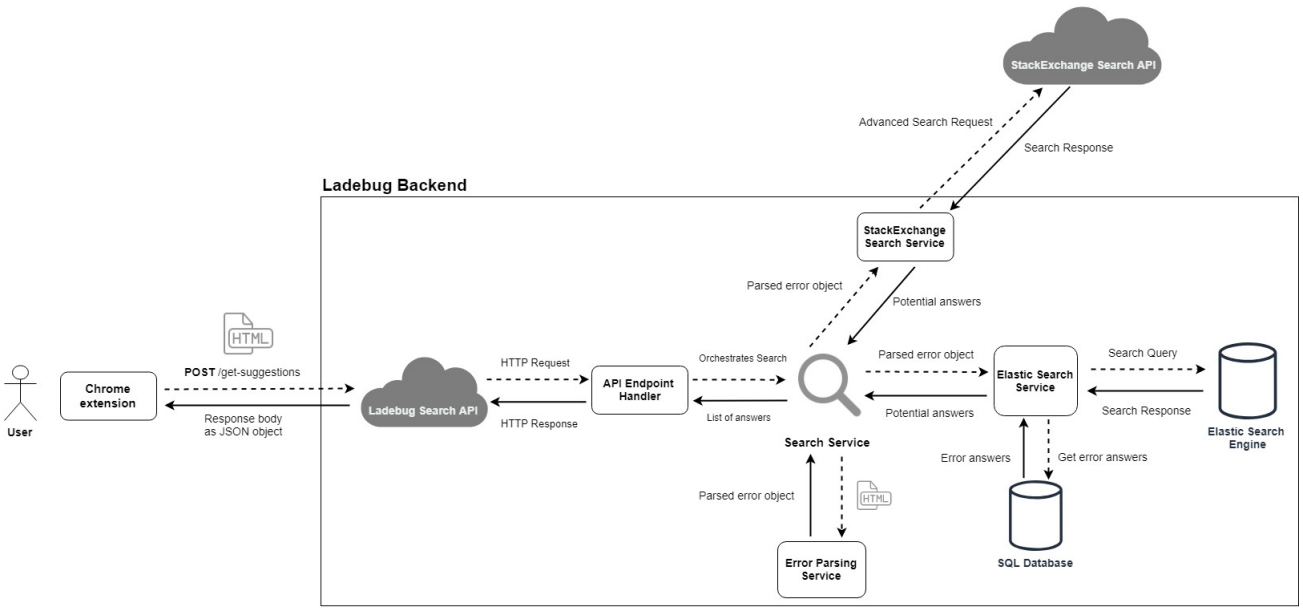


Figure 18 - An overview diagram of how the new system works

More overview diagrams of the old system can be found in Figure 46, and Figure 47 of Appendix F and more overview diagrams of the new system can be found in Appendix L.

6.1.2 Third-party integration

The new implementation has introduced the integration of StackOverflow through their Search API, which has enhanced the suggestions provided to the user by the system, as it is now able to look in the internal search engine and StackOverflow for answers as shown in Figure 18.

6.1.3 Web App And Chrome Extension Changes

The final user interface of the artefacts has changed to reflect the designs made section 5.4.2. Evidence of the final user interface of the artefacts can be found in Appendix K.

6.2 Architecture Implementation

The project was restructured from a monolith architecture to a layered architecture as discussed in section 5.3.3. Evidence of the layers and its full structure can be found in Appendix T.

The project dependency diagram in Figure 19 shows the implementation of layers as discussed in chapter 5.3.3. The important takeaway in the diagram is that the core layer is not dependent on any other layer, and the other layers does not reference the Web layer. Concerns are separated into separate layers/assemblies, which makes it more isolated and easier to maintain.

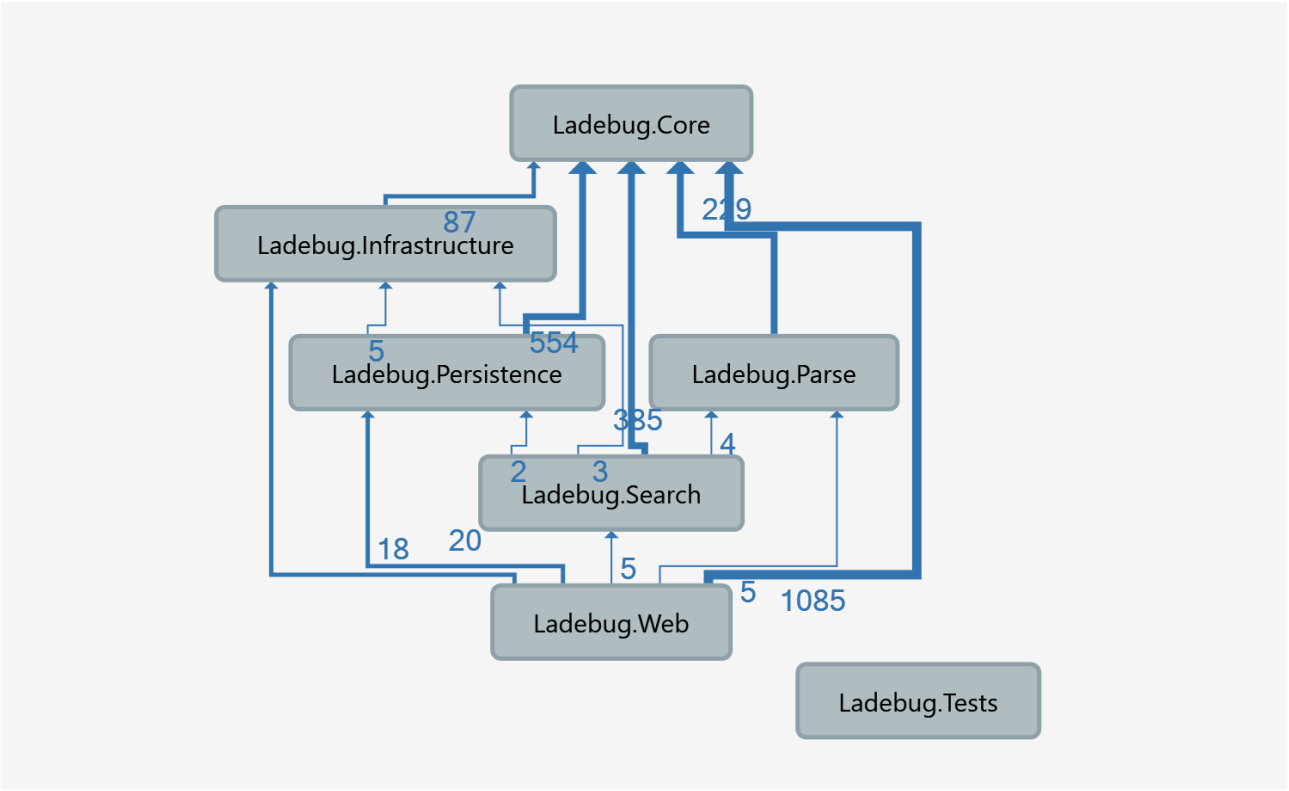


Figure 19 - Project dependency diagram (generated using visual studio 2017)

6.3 EXTENDIBLE CODE

6.3.1 Composition Root

In the context of programming, composition root is defined as the unique location in an application where modules are composed together (Seeman, 2011). In the web app, the composition root is where the runtime implementations and dependencies are composed together as shown in Figure 20 and Figure 21, which can be found in the Ladebug.Web layer. ASP.NET Core provides an easy to use, built-in support for registering/injecting dependencies as shown in the diagrams below.

```
1 reference | Uchenna Okafor, 14 days ago | 1 author, 9 changes | 0 exceptions
public static void ConfigureServicesDependencies(this IServiceCollection services, IConfiguration configuration, IHostingEnvironment env)
{
    services.AddSingleton<IActionContextAccessor, ActionContextAccessor>();
    services.AddSingleton<IUrlHelperFactory, UrlHelperFactory>();
    services.AddHttpContextAccessor();
    services.AddSingleton<IMiscellaneousLinkGeneratorService, MiscellaneousLinkGeneratorService>();

    services.AddSingleton<CustomCookieAuthenticationEvents>();

    services.AddScoped(typeof(IRepository<>), typeof(Repository<>));
    services.AddScoped<IUnitOfWork, UnitOfWork>();

    services.AddTransient<ICommentsRepository, CommentsRepository>();
    services.AddTransient<IErrorsRepository, ErrorsRepository>();
    services.AddTransient<ITeamRepository, TeamsRepository>();
    services.AddTransient<ITeamManager, TeamManager>();

    services.AddSingleton<IJwtTokenService, JwtTokenService>();

    services.AddSingleton<ErrorParser, AspNetMvcParser>();
    services.AddSingleton<ErrorParser, DotNetCoreMvcParser>();
    services.AddScoped<IParsingService, ErrorParsingService>();

    services.ConfigureSearchServices(configuration, env);
}
```

Figure 20 - The unique location where all application dependencies are registered/injected

```
1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
private static ConnectionSettings GetElasticSearchConnectionSettings(IConfiguration configuration, IHostingEnvironment env)
{
    ConnectionSettings settings;
    if (env.IsDevelopment())
    {
        settings = new ConnectionSettings();
    }
    else
    {
        var node = new Uri(configuration["Search:ElasticSearch:Instance:NodeUri"]);
        var username = configuration["Search:ElasticSearch:Instance:Username"];
        var password = configuration["Search:ElasticSearch:Instance:Password"];

        settings = new ConnectionSettings(node);
        settings.BasicAuthentication(username, password);
    }

    settings.RequestTimeout(TimeSpan.FromSeconds(15));
    settings.DefaultIndex(configuration["Search:ElasticSearch:Configurations:DefaultIndex"]);
    return settings;
}

1 reference | Uchenna Okafor, 13 days ago | 1 author, 8 changes | 0 exceptions
private static void ConfigureSearchServices(this IServiceCollection services, IConfiguration configuration, IHostingEnvironment env)
{
    services.AddScoped<IElasticClient>(sp =>
    {
        var settings = GetElasticSearchConnectionSettings(configuration, env);
        return new ElasticClient(settings);
    });

    services.AddScoped<ISearchProvider, ElasticSearchProvider>();

    services.AddScoped<ISearchManager, SearchManager>();
    services.AddScoped<ISearchService, SearchService>();
    services.AddScoped<IThirdPartyAnswerFinder, StackOverflowAnswerFinder>();
    services.AddScoped<IElasticSearchResultsFormatter, ElasticSearchResultsFormatter>();
    services.AddScoped<ISearchServiceSettingsService, SearchServiceSettingsService>();
}
}
```

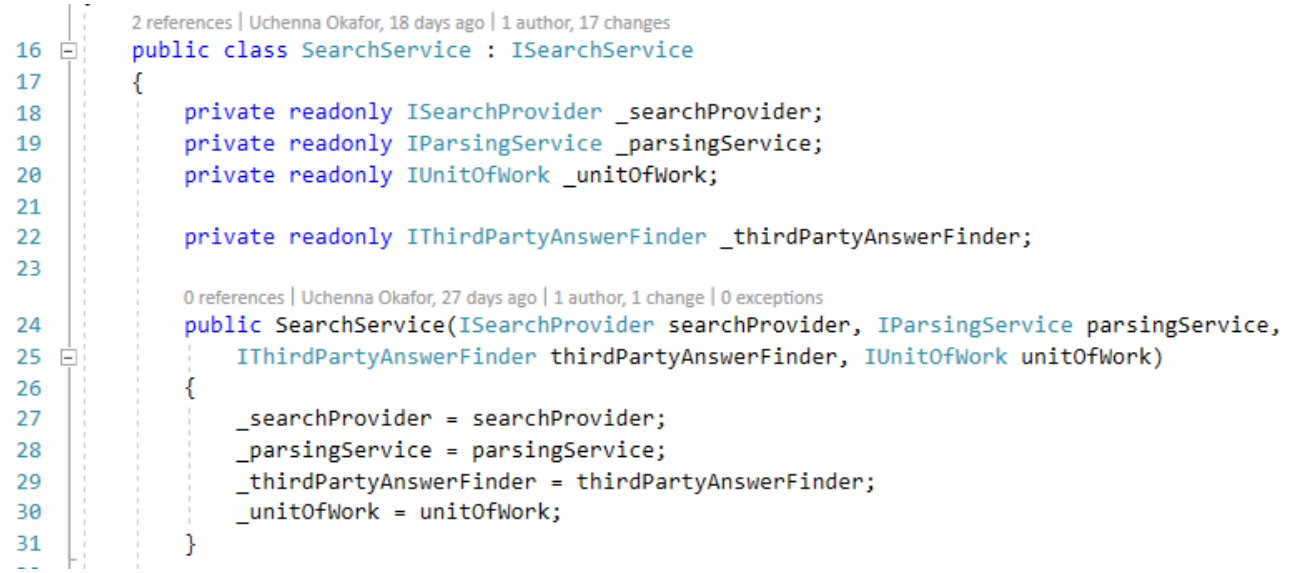
Figure 21 – The unique location where search related dependencies are constructed and registered/injected

6.3.1.1 Dependency Injection

The code snippet in Figure 20 and Figure 21 shows concrete implementations being registered to their abstractions using dependency injection. The code is loosely coupled as it allows for implementations to be swapped without breaking existing code. As mentioned in chapter 2.2.5.1, lines 89 and 101 in Figure 22 shows different implementations to the search provider being registered to the application which is then injected to the constructor of the `SearchService` class in Figure 23 at runtime by ASP.NET Core's inbuilt DI container.

```
1 reference | Uchenna Okafor, 13 days ago | 1 author, 8 changes | 0 exceptions
79  private static void ConfigureSearchServices(this IServiceCollection services, IConfiguration configuration, IHostingEnvironment env)
80  {
81      if (env.IsDevelopment())
82      {
83          services.AddScoped<IElasticClient>(sp =>
84          {
85              var settings = GetElasticSearchConnectionSettings(configuration, env);
86              return new ElasticClient(settings);
87          });
88
89          services.AddScoped<ISearchProvider, ElasticSearchProvider>();
90      }
91      else
92      {
93          services.AddScoped<ISearchServiceClient>(sp =>
94          {
95              var searchServiceName = configuration["Search:Azure:Configurations:Name"];
96              var adminApiKey = configuration["Search:Azure:Configurations:Key"];
97
98              return new SearchServiceClient(searchServiceName, new SearchCredentials(adminApiKey));
99          });
100
101         services.AddScoped<ISearchProvider, AzureSearchProvider>();
102     }
103
```

Figure 22 - Demonstration of how dependencies can be easily swapped as a result of implementing dependency inversion principle and using dependency injection



```
2 references | Uchenna Okafor, 18 days ago | 1 author, 17 changes
16  public class SearchService : ISearchService
17  {
18      private readonly ISearchProvider _searchProvider;
19      private readonly IParsingService _parsingService;
20      private readonly IUnitOfWork _unitOfWork;
21
22      private readonly IThirdPartyAnswerFinder _thirdPartyAnswerFinder;
23
        0 references | Uchenna Okafor, 27 days ago | 1 author, 1 change | 0 exceptions
24      public SearchService(ISearchProvider searchProvider, IParsingService parsingService,
25          IThirdPartyAnswerFinder thirdPartyAnswerFinder, IUnitOfWork unitOfWork)
26      {
27          _searchProvider = searchProvider;
28          _parsingService = parsingService;
29          _thirdPartyAnswerFinder = thirdPartyAnswerFinder;
30          _unitOfWork = unitOfWork;
31      }
```

Figure 23 – Code snippet of the SearchService class

The sample code in Figure 22 uses different search engines based on the environment the web app is hosted on, on local development environment it would use Elastic Search, albeit, in production, it will use Azure Search. This is an example of what is possible with the new implementation, which would have previously been impossible with the previous implementation as it was tightly coupled to Elastic Search as discussed in chapter 5.3.1.

6.3.2 Extendible error parser

As mentioned in sections 5.3.2 and 5.3.2.1, the redesigned error parser framework would make it easier to add support for new modules. As proof of the extendibility of the refactored design, ASP.NET Core error pages are now supported in the new artefact.

The diagram in Figure 24 shows the type dependency diagram of the new error parser framework implementation. The code snippet in Figure 25 shows the `ErrorParsingService` class which is injected an array of `ErrorParser` modules in the constructor at runtime. The error parser modules are then registered at the composition root of the application as shown on lines 51 and 52 of Figure 26.

This means more error parser modules can be added without modifying the code in the `ErrorParsingService`. With this approach, adding support for a new web framework, for example, Laravel would be to inherit the `ErrorParser` base class, override its methods and register the concrete implementation as a dependency in the composition root like in Figure 26.

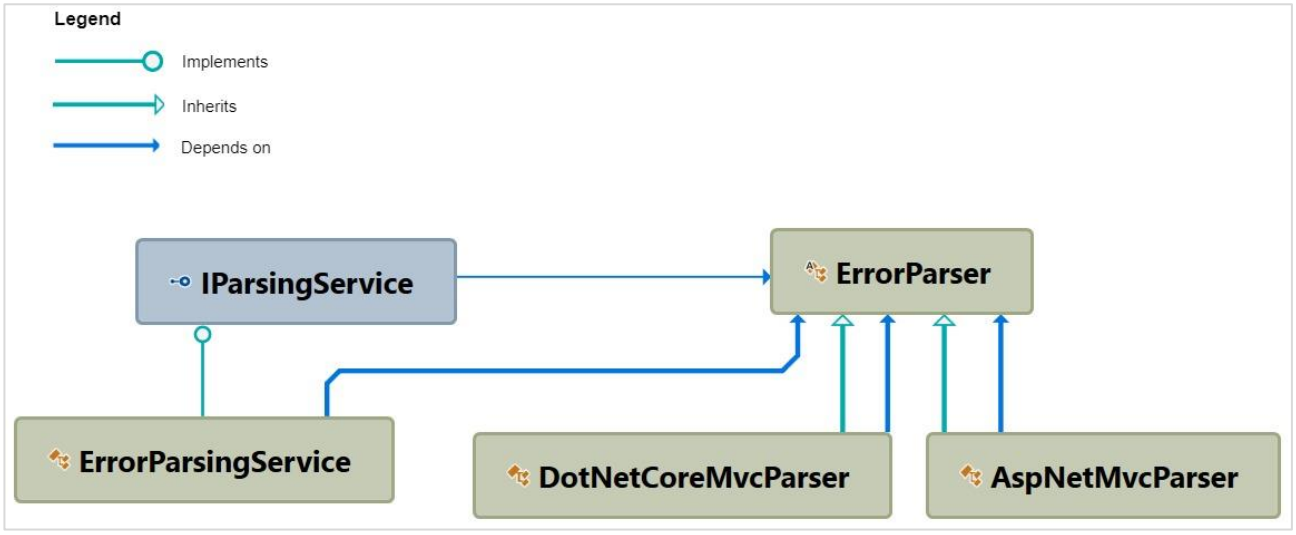
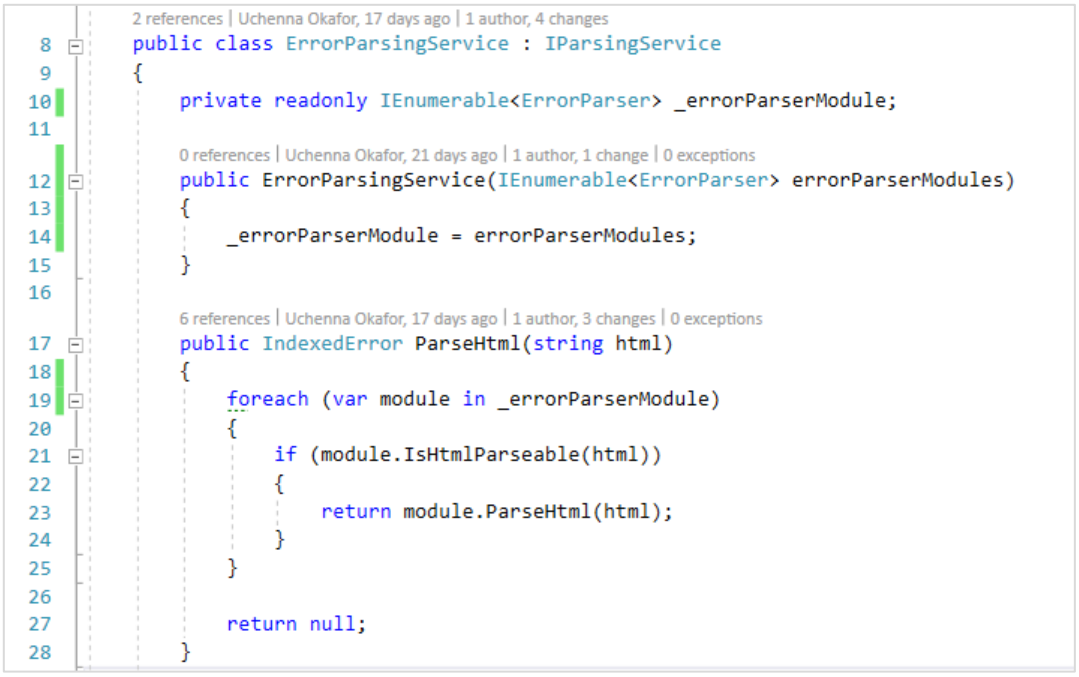


Figure 24 - Type dependency diagram of the ErrorParser (generated using Visual Studio 2017)



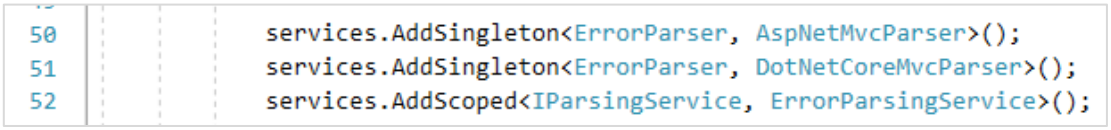
```
2 references | Uchenna Okafor, 17 days ago | 1 author, 4 changes
public class ErrorParsingService : IParsingService
{
    private readonly IEnumerable<ErrorParser> _errorParserModule;

    0 references | Uchenna Okafor, 21 days ago | 1 author, 1 change | 0 exceptions
    public ErrorParsingService(IEnumerable<ErrorParser> errorParserModules)
    {
        _errorParserModule = errorParserModules;
    }

    6 references | Uchenna Okafor, 17 days ago | 1 author, 3 changes | 0 exceptions
    public IndexedError ParseHtml(string html)
    {
        foreach (var module in _errorParserModule)
        {
            if (module.IsHtmlParseable(html))
            {
                return module.ParseHtml(html);
            }
        }

        return null;
    }
```

Figure 25 - Error Parser service class



```
50          services.AddSingleton<ErrorParser, AspNetMvcParser>();
51          services.AddSingleton<ErrorParser, DotNetCoreMvcParser>();
52          services.AddScoped<IParsingService, ErrorParsingService>();
```

Figure 26 - Injection of parsing related dependencies

6.3.3 Authentication

A full authentication system was implemented as mentioned in section 5.4.3 and the functional requirements. Implementing authentication required leveraging ASP.NET Core's inbuilt Identity services, which was modified to fit the requirements of the application.

Open Authorization (OAuth) support was added to provide more authentication options for users. The code snippet in Figure 27 shows the configuration of each OAuth provider, while Figure 28 shows the backend logic used to render the external login buttons. Finally, Figure 29 and Figure 30 show the rendered HTML of the login and register pages with the social login buttons.

```
.AddMicrosoftAccount(options =>
{
    options.ClientId = configuration["Authentication:Microsoft:ApplicationId"];
    options.ClientSecret = configuration["Authentication:Microsoft:Password"];
})
.AddGitHub(options =>
{
    options.ClientId = configuration["Authentication:Github:ClientId"];
    options.ClientSecret = configuration["Authentication:Github:ClientSecret"];

    options.Scope.Add("user:email");
    options.Scope.Add("read:user");

    options.ClaimActions.MapJsonKey(ClaimTypes.Name, "name");
    options.ClaimActions.MapJsonKey("urn:github:login", "login");
    options.ClaimActions.MapJsonKey("urn:github:avatar", "avatar_url");
});
```

Figure 27 - Code snippet of how support for external OAuth providers are configured

_ExternalLoginButtons.cshtml

```
@model Ladebug.Web.ViewModels.ExternalLoginButtons

@if ((Model.ExternalLogins?.Count ?? 0) > 0)
{
    <div class="external-logins mt-3">
        <p class="lead text-center">@Model.HeaderText</p>
        <hr />
        <form asp-page="./ExternalLogin" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <div class="row justify-content-center">
                <p>
                    @foreach (var provider in Model.ExternalLogins)
                    {
                        <button type="submit" class="btn btn-light btn-lg oauth-provider-icon justify-content-center ml-2"
                                name="provider" value="@provider.Name" title="@string.Format(Model.FormattedTitleText, provider.DisplayName)">
                            <img src="~/images/@(provider.Name.ToLower()).svg" alt="@(provider.Name) Logo" height=28 width=28 />
                        </button>
                    }
                </p>
            </div>
        </form>
    </div>
}
```

Figure 28 - The HTML for rendering the social buttons for each configured OAuth provider

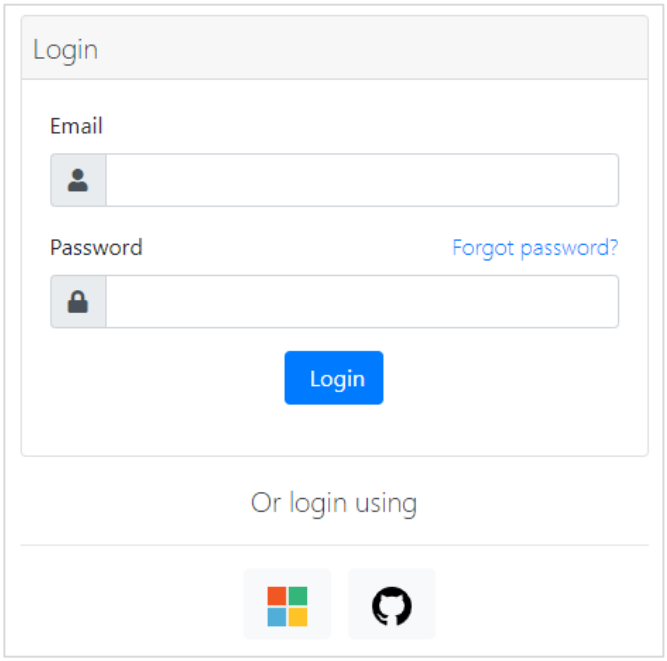


Figure 29 - Login page which includes social buttons for the external OAuth providers configured

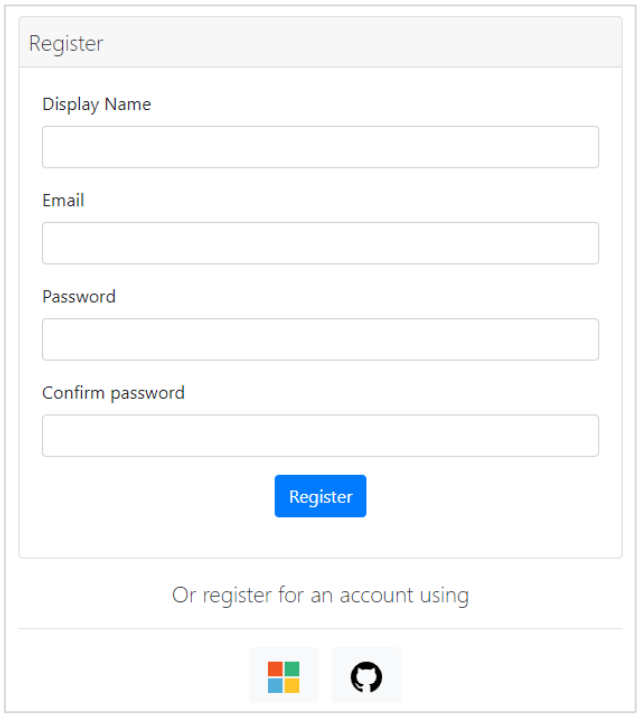


Figure 30 - Register page which including social buttons for the external OAuth providers configured

6.4 SECURITY

6.4.1 Resource Authorization

A requirement of the new system was for the implementation of private teams. For users to have read or write access to a team and its contents they must fulfil a requirement which requires them to be a member of the team. This authorization policy was enforced at the routing level of the application as shown in Figure 31, and upon failure to meet the requirement, the user is presented with an access denied page as shown in Figure 32 (Full page view can be found in Appendix K).

```
0 references | Uchenna Okafor, 19 days ago | 1 author, 3 changes | 0 exceptions
protected override async Task HandleRequirementAsync(AuthorizationHandlerContext context
{
    var user = await _userManager.GetUserAsync(context.User);

    var path = _httpContextAccessor.HttpContext.Request.Path;
    var teamAlias = path.Value.Split('/')[2];

    if (_unitOfWork.Teams.IsUserATeamMember(user, teamAlias))
    {
        context.Succeed(requirement);
    }
    else
    {
        context.Fail();
    }
}
```

Figure 31 - Authorization requirement for accessing a private team resource

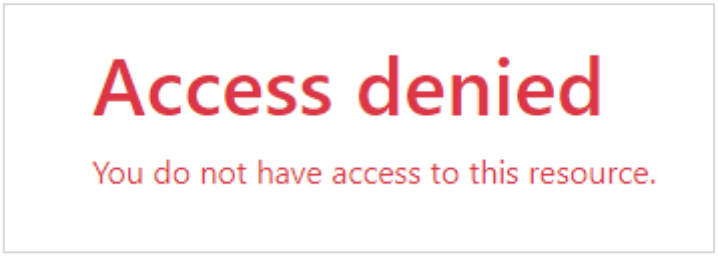


Figure 32 - Access denied page

6.4.2 Sandboxed iframes

The HTML document of the error page users have posted is stored for browsing purposes (see Figure 34), therefore it needs to be securely displayed otherwise it would be vulnerable to persistent cross-site scripting attacks (OWASP Foundation, 2017). This security risk was mitigated by applying a sandbox attribute to the iframe element that rendered the raw HTML as shown in Figure 33. The sandbox attribute prevents scripts from being executed within the context of the iframe element (w3schools, n.d.), therefore fixing a potential cross-site scripting vulnerability.

```
<div class="embed-responsive embed-responsive-16by9">
    <iframe scrolling="no" class="embed-responsive-item" src="@Url.Action("ViewRaw", "Exceptions", new {id = Model.Id, slug = Model.Slug })" sandbox></iframe>
</div>
```

Figure 33 - HTML code snippet of the sandboxed iframe element

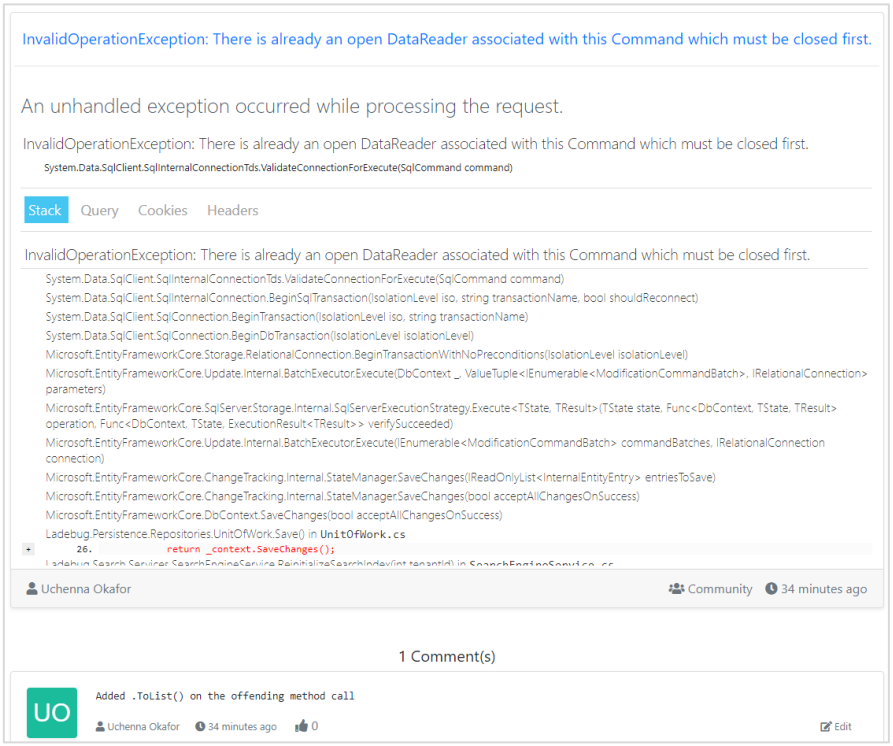


Figure 34 - Viewing error posts in the web app

6.5 Friendly URLs

Link sharing on the old Ladebug provided a poor user experience because the shared URLs were semantically incorrect (see Figure 35) and did not contain any information that hinted to the content of the URL (Moz, n.d.). The addition of semantically friendly URLs (see Figure 36) is both human and machine readable which makes it easier to share and to be found by search engines (Swati et al. 2013).

Discovery through search results will be necessary for establishing and maintaining a strong web presence in the future lifecycle of the product. Therefore a friendly URL that could help improve SEO rankings is a benefit. Developers should be able to search an error message into a search engine and have a public Ladebug post as a search result.

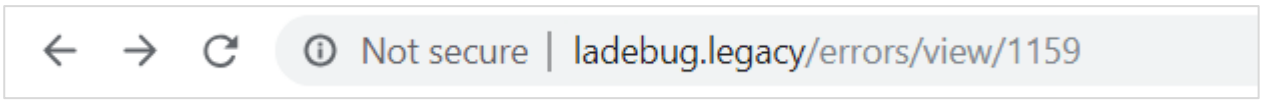


Figure 35 – Semantically incorrect URL from the old web app

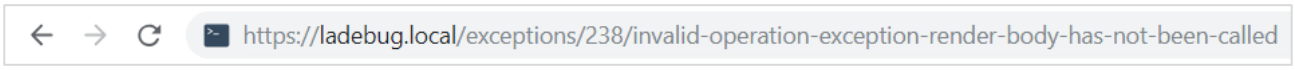


Figure 36 – Semantically correct URL from the new web app

6.6 Cloud Deployment

The final iteration of the web app and database were successfully deployed to Azure.
An attempt was made to use the official Elastic Search SaaS platform (Elastic, n.d.) to host the Elastic Search instance, but it was limited to a 2-week free trial which was not ideal. Bonsai, another Elastic Search SaaS platform offers a free plan (Bonsai, n.d.) that was suitable for the requirements of the project at the current time, and therefore was the chosen cloud provider for the Elastic Search engine.

Finally, the new chrome extension was updated over the already existing extension in the Google Chrome store. Therefore, existing Chrome extension users at Redweb were automatically updated to the latest version. The deployed web app can be accessed at https://ladebug.net. Evidence of the deployments can be found in Appendix M.

6.7 Development Tools and Technologies

Other than the core development tools like Visual Studio 2017 IDE and MSSQL Server 2017, there were a few other tools vital to the development of the artefact.

6.7.1 ReSharper

ReSharper, a plugin for the Visual Studio IDE (ReSharper, n.d.) is a productivity tool that performs live code analysis, detects code smells, redundancies, runtime errors and suggests possible improvements right as the user is typing (Christakis and Bird, n.d.). ReSharper was an essential tool used when refactoring code, it passively enforced coding conventions and has allowed for mass refactoring of class namespaces and code optimizations. Therefore, resulting in high-quality clean code.

6.7.2 Version Control (Git)

Git was primarily used as a version control tool. It helped in refactoring code, and tracking changes in the event of a bug were introduced. It made it easy to work on an isolated version of the codebase, using branches to ensure that functionality is only integrated when it has been adequately implemented without any critical bugs. Frequent commits were made to ensure that progress was saved to the cloud and if a bug was introduced, the origin could be traced and quickly resolved.

6.7.3 GitHub

GitHub was used as part of the development workflow as a project management tool. GitHub has a Kanban type board as shown in Figure 37 which was populated with tasks based on the technical and user requirements of the project. Using GitHub as a project management tool allowed for the progress of tasks to be efficiently tracked, which resulted in the better time management of the project. Finally, as part of the success criteria of the project was to have the code reviewed and evaluated, using a version control system like Git, and hosting the project on GitHub would provide easy access to the code.

6.7.3.1 Bug Tracking

Introducing changes to code inevitably introduces bugs. The number of bugs is an essential metric in assessing the quality of code (Hovemeyer and Pugh, 2004). The implementation phase involved refactoring and redesigning the existing codebase, which required an efficient way to track bugs and their states to ensure that they are eventually fixed. GitHub's issues tracking feature was leveraged to track bugs (see Figure 38).

6.7.4 Additional Libraries and tools

See Appendix N for the full list of tools and libraries used.

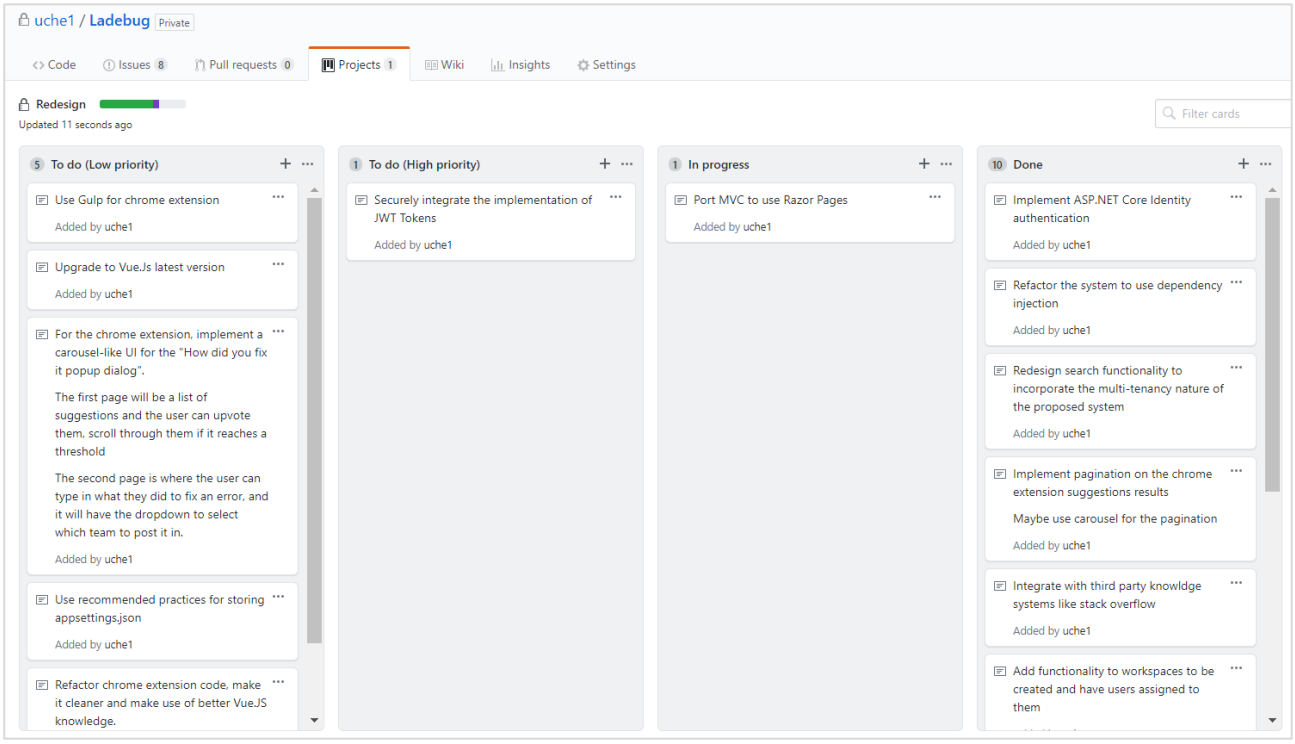


Figure 37 – Final project board for Ladebug in GitHub

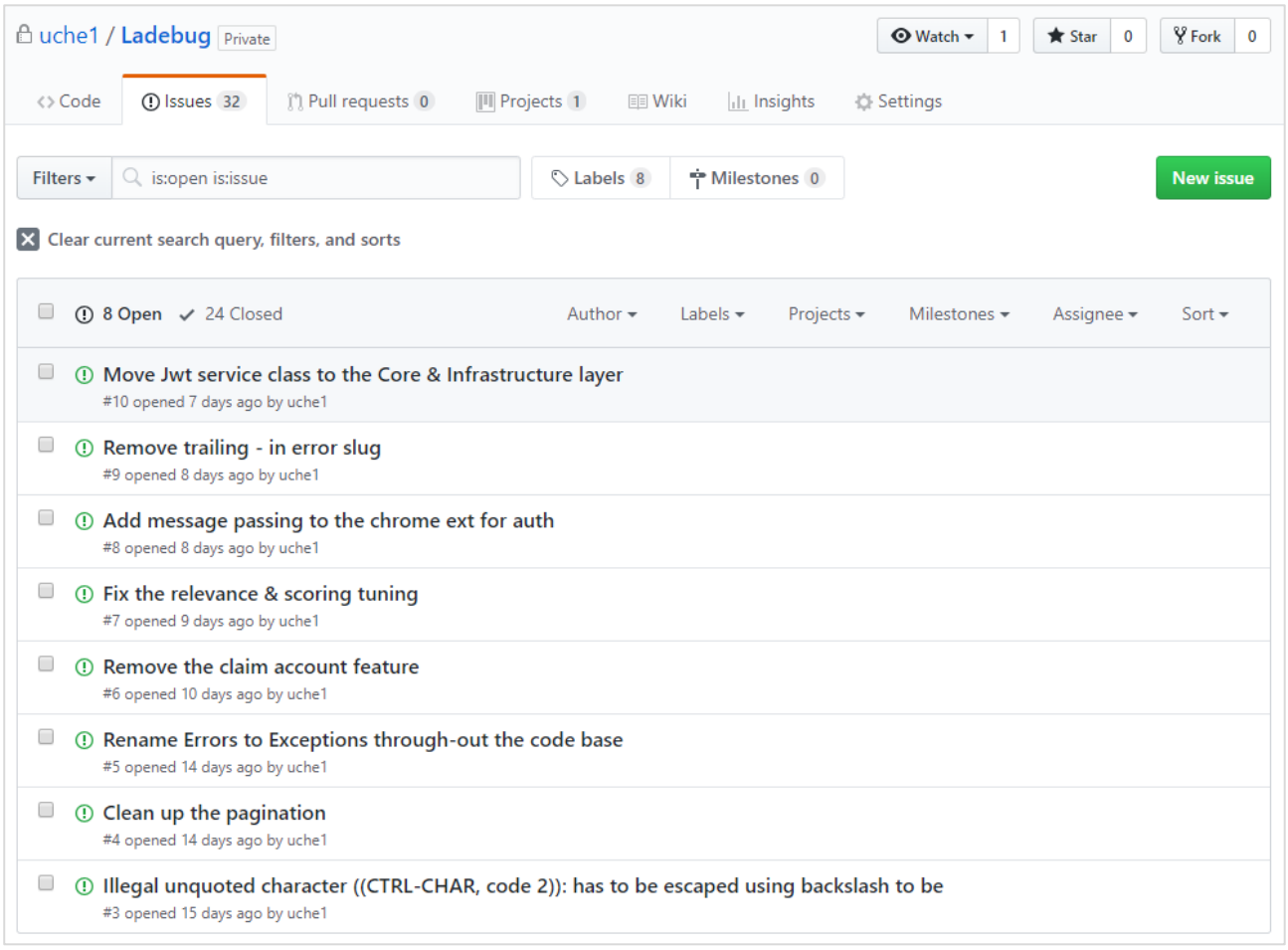


Figure 38 - Ladebug's GitHub issues

6.8 Unforeseen Issues

6.8.1 Data Migration

Prior to deployment to the cloud, a last-minute requirement from the client was to have the data in the existing system to be migrated to the new system so it could be used to beta test the system. This was a problem as the database schema of the two systems were significantly different, so the data could not be migrated with a simple backup and restore procedure.

The solution was to modify the existing system to export data in a JSON format, then a custom script was written in the new system to translate the exported JSON data into usable data. This was then used to restore all existing data into the new system.

6.9 Implementation Limitations

6.9.1 Lack of support for Drupal

Support for the Drupal web framework was a requirement from the client because there is a team that uses Drupal at Redweb. During the beta testing phase, the Drupal team could have used the system and given feedback, however, due to lack of experience with Drupal, system support for Drupal was not implemented. On the contrary, a positive to building an extensible system is that adding support for Drupal in the future will relatively straightforward.

6.9.2 Lack of unit tests

Unit tests were planned to be written and executed in the Ladebug.Tests layer, however, due to time constraints, unit tests were not added. However, the layer remains as it will be useful for when unit tests are written in the future.

7 EVALUATION

This chapter details the several methods used to evaluate the code quality of the produced artefact in comparison to the existing system. In addition, this chapter will also detail the methods used to evaluate the quality of the new system and compare it to other existing systems.

7.1 MEASURING ARTEFACT'S CODE QUALITY

According to Kannangaral (2013), the factors that affect software quality can be classified into two groups; directly measurable metrics (e.g. lines of code, class coupling), and indirectly measurable metrics (factors such as readability, understandability and so forth). Therefore, both classification methodologies will be used to measure and evaluate the code quality of the artefact.

7.1.1 Code Metrics

The goal of this research was to refactor an existing system that was fragile, rigid and immobile to be more maintainable, extendible and flexible. Visual Studio 2017, the IDE used to develop the artefact has a software measurement tool that assesses software quality by calculating code metrics measurements. The code metrics measurements can be seen and described in Table 3.

Name	**Description**	**What does this mean for the code?**
Maintainability Index	Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. (Higher values are better) [Green = good, amber = moderate, red = bad]	Maintainability Extendibility Flexibility
Cyclomatic Complexity	Measures the structural complexity of the code. (Lower values are better)	Maintainability, Extendibility
Depth of Inheritance	Indicates the number of class definitions that extend to the root of the class hierarchy. (Lower values are better)	Maintainability
Class Coupling	Measures unique references to a class. (Lower values are better)	Extendibility Maintainability
Lines of Code	An approximation to the lines of code. (Lower values are better)	Maintainability

Table 3 - Visual Studio Code Metrics Measurements (Robertson et al. 2018)

7.1.1.1 Comparing code quality

Code metrics reports of the old artefact and the new artefact can be seen in Figure 39 and Figure 40. In comparison, the old artefact was a monolith, whereas the new artefact is a multi-layered multi-project application, which was refactored and redesigned from the ground up.

Therefore, a 1-1 code comparison using code metrics to measure the change in code quality of the two codebases would not be accurate as both systems are radically different. Nonetheless, the code metrics report is still relevant as it provides valuable insight on code quality for both systems.

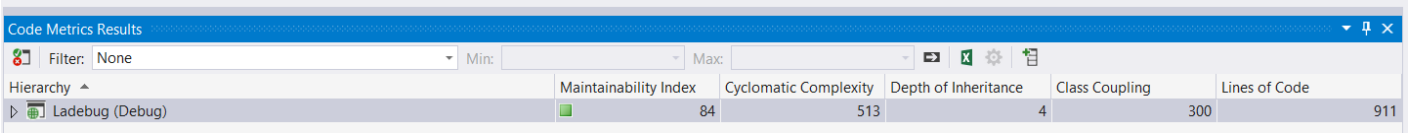


Figure 39 - Code metrics result for the existing system (generated using Visual Studio 2017)

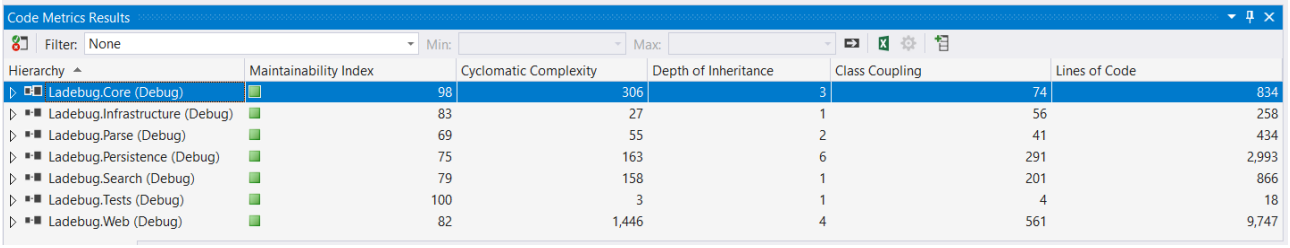


Figure 40 – Code metrics result for the artefact (generated using Visual Studio 2017)

According to the code metrics report, all layers in the new artefact is maintainable as the maintainability index value falls within the range deemed as “good”. Although the metrics report for the existing artefact indicates that the code is maintainable, it is important to acknowledge the old artefact has 911 lines of code, whereas the new artefact has over 13,00 lines of code. Static code metric algorithms use static analysis for their calculations, and low lines of code heavily influence a higher maintainability index. Evidence of this can be seen in Figure 40 where the Ladebug.Tests layer has a maintainability index of 100, because it has 18 lines of code, resulting in the other metric measurements being low.

Therefore, other methods of code analysis are essential as static code metrics algorithms do not provide in-depth analysis of code, as well as humans reviewers. Especially as the two codebases are radically different in structure, a 1-1 comparison cannot be made.

7.1.2 Code Review and Evaluation

In addition to code metrics, critical code review and evaluation were conducted and collected using a survey form. The selected software developers from Redweb were given access to the codebase of both systems on GitHub and were asked to answer the questions on the form twice, one for old codebase and the new codebase. The questions on the form consisted of open-ended questions for the code review and closed-ended questions which were useful for creating an analysis of survey submissions.

7.1.2.1 Survey results

The feedback gathered from the code review of the new artefact were overall positive; it commended the consistency of the coding conventions used, proper use of dependency injection and many other programming techniques. The code review of the old artefact was commended for its consistency of coding conventions but critiqued on being tightly coupled, being monolithic and difficult to maintain and extend. The full survey responses from the code review and evaluation can be found in Appendix Q.

The second aspect of the survey was closed-ended feedback on the effectiveness of the implementation relative to the themes of the dissertation. The survey asked for expert opinions on how maintainable, flexible and extendible the old and new systems code were.

An analysis of the closed-ended questions for both systems can be found in Figure 41. According to the results, there has been a massive improvement in the maintainability, extensibility and flexibility of the new codebase in comparison to the old codebase. The survey result seen in Table 4 shows a 25% increase in maintainability, a 150% increase in extendibility and 150% increase in flexibility from the old system to the new system. In conclusion, the objective of refactoring the existing codebase, to produce a more maintainable, extendible and flexible system has been successfully achieved.

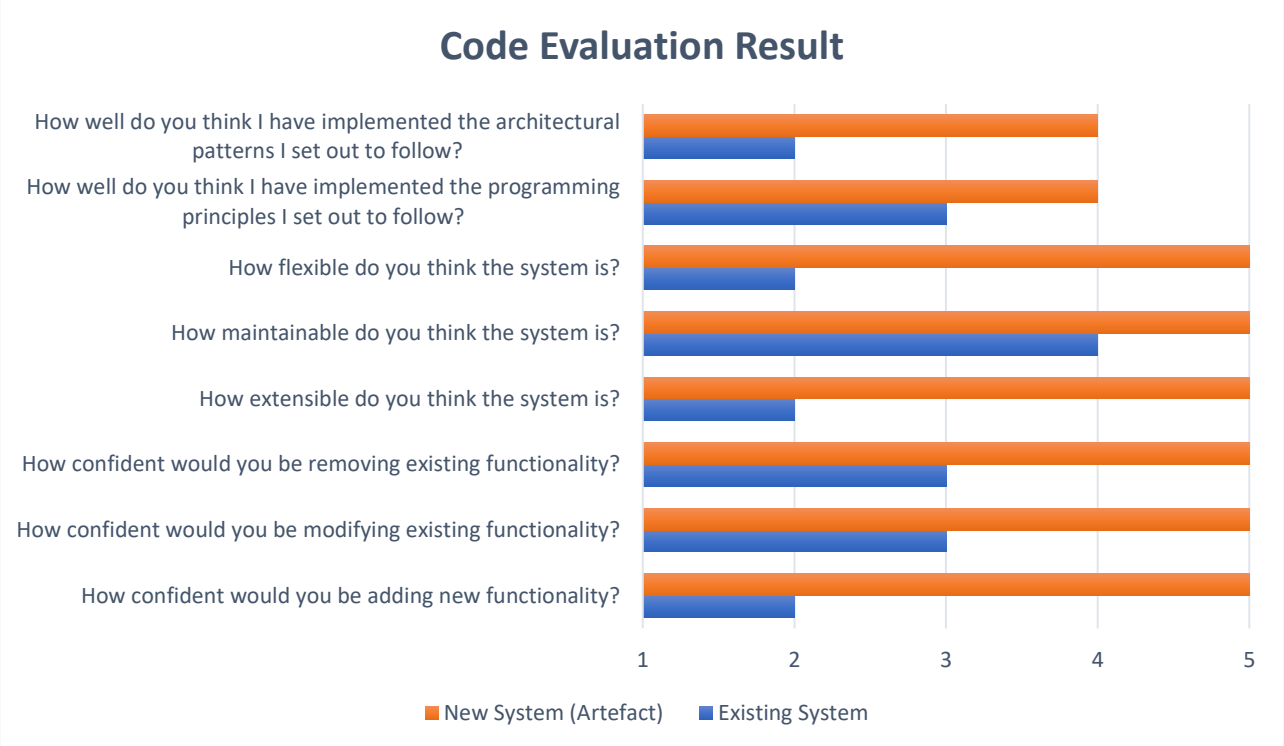


Figure 41 - Bar graph of results from code evaluation

Question	Percentage Increase from the old system to the new system
How well do you think I have implemented the architectural patterns I set out to follow?	100%
How well do you think I have implemented the programming principles I set out to follow?	33.33%
How flexible do you think the system is?	150%
How maintainable do you think the system is?	25%
How extensible do you think the system is?	150%
How confident would you be removing existing functionality?	66.66%
How confident would you be modifying existing functionality?	66.66%
How confident would you be adding new functionality?	150%

Table 4 – Analysis of code evaluation results in Figure 41

7.2 Artefact Testing and Evaluation

The final artefact was deployed to the cloud as mentioned in section 6.6. The broader accessibility of the artefact allowed for it to be beta testers by developers from Redweb. As the existing database data could be restored to the new system as mentioned in section 6.8.1, existing users could continue using the system as usual, including their existing account to best test the new system.

7.2.1 Closed Beta Testers Feedback

The existing users of Ladebug from Redweb were selected as the group to beta test the new system. The beta test phase lasted a week, commencing from the 24/04/2019 to 1/5/2019, whereby users were instructed to use the system as usual. At the end of the week, the users were to complete a survey feedback form (see Appendix P for the collected user feedback). The results of the beta feedback were all around positive, and the feedback received included suggestions on what could be improved and what was done well.

In addition to the beta testers not reporting any bugs, during the beta test week, there were also no bugs or errors reported from Azure's application insights panel (evidence can be found in Appendix O). This is a good as it is a sign of robust code, which is an indication of high-quality code.

At the end of the beta test, the system was deemed very stable as no errors occurred during use and therefore the new system was continued to be used and has completed replaced the old system, which has been deprovisioned.

7.2.1.1 Security Feedback

Figure 54 in Appendix P shows feedback from beta testers highlighting the potential security implications of Ladebug. HTML error pages can sometimes contain sensitive information that a user may not want to be public. Therefore, looking into the security implications and best practices for handling sensitive data would be necessary before public release of the product.

7.2.2 Requirements and success criteria

The project has met all criterion in the success criteria and has met 20/21 of the system requirements (Evidence can be found in Appendix V).

7.2.3 Client Feedback

During the final meeting with the client, positive feedback and praise were given to the artefact produced. The client was pleased with the outcome of the artefact, particularly with the integration of StackOverflow, the future extensibility of the system and migration of the data from the old system to the new. Evidence of client feedback can be found in Appendix R.

7.3 COMPARISON TO EXISTING Q&A PLATFORMS

A comparison of the artefact to existing Q&A platforms can be found in Table 5. The comparison revealed that the artefact has many similarities with StackOverflow; in comparison Quora does not share the same similarities due to it being a general Q&A platform. StackOverflow and the artefact have the similarity of both being targeted at developers and therefore have functionality that meets the requirements of developers.

With future development and expansion, the artefact could rival StackOverflow. Ladebug currently offers a service that can automatically search and find solutions to an error, which results in increased productivity, as developers do not have to manually search for answers to their error message.

Criteria	Artefact	StackOverflow	Quora
Can it search through multiple external services for an answer to a question?	Y	N	N
Is there high precision in matching error pages?	Y	N	N
Is there a browser plugin that automatically searches an error page and returns a list of answers?	Y	N	N
Can users have private instances/teams?	Y	Y	N
Can users post type of question?	N	Y	Y
Is the website targeted at developers?	Y	Y	N
Criterions fulfilled	5/6	3/6	1/6

Table 5 - Comparison of the artefact to existing Q&A platforms

7.4 CONCLUSION

In conclusion, most of the requirements and objectives of the project and artefact has been met. The code analysis provided insight to the codebase, meanwhile the code review and evaluation helped to provide more in-depth insight to the effectiveness of the implementation of the new system.

8 CONCLUSIONS

This project set out to refactor the code of an existing system that consisted of bad software characteristics such as fragility, immobility, rigidity and in doing so, improve the lack of maintainability, extensibility and flexibility associated with the bad design. Additionally, implement new functionality to meet the requirements of the client.

8.1 WHAT WENT WELL

The programming disciplines and architectural patterns followed, enabled a successful artefact to be built. The integration of StackOverflow and the possibility to extend the project was very successful. Additionally, the feedback from the client, beta testers and code review/evaluation were all valuable as it helped meet the success criteria of the project and evaluate the validity of the implementation. Finally, the deployment to the cloud was very successful as it increased the accessibility to the once locally accessible system.

8.2 WHAT COULD HAVE BEEN IMPROVED

The communication process with the client during the beta testing phase could have been more concise, as many emails were sent which may have led to some confusion during the beta testing phase. The researcher also feels like the project may have been a little bit overambitious, given a chance to redo the project, the scope of the research theme and implementation could have been reduced to allow a narrower focus. Regardless, the project was still a success and a good learning experience.

8.3 CONCLUSION

The overall outcome of the artefact has surpassed the researcher’s expectations, and it has solved most problems that the existing system had. An important takeaway for this project is that the groundwork for future development has been laid, therefore assuring the low costs in maintenance of the products lifecycle.

8.4 FUTURE WORK

8.4.1 Performance Optimizations

There are many aspects of the artefact's web app that could be optimized to be more performant. The priority for future work would be first to optimize the speed it takes for backend API to retrieve answers because it is vital for suggestions to be displayed to the user as soon as they are shown an error page.

8.4.1.1 Cloud infrastructure optimization

An advantage of deploying to the cloud, the app service could be scaled up (vertical) on demand to achieve better overall performance. Aside from searching and parsing, the web app is not a very demanding website. Therefore, scaling up the entire app service to increase the speed of parsing and searching would incur higher costs than is necessary.

A potential solution would be to use Azure functions. Azure functions is a serverless compute service that executes scripts or pieces of code in response to an event (Microsoft, n.d.). This cloud technology can be used to delegate the parsing and searching functionality, therefore allowing the web app to run on a smaller scaled app service while allowing the Azure function to run on a higher scaled app service. The result is a cheaper and more effective performance optimizations.
The need for Azure functions will be more noticeable when Ladebug scales and supports for error parser frameworks and third-party search services.

8.4.1.2 Code optimization

Concurrency in programming is a technique that can be used to enhance the performance optimizations of the artefact in the future. Currently, the system searches for answers using the internal search engine and by leveraging third-party services like StackOverflow. As the system scales in the future, the searching functionality would move from being synchronous to asynchronous. Concurrency will be achieved using multi-threading to do the searching in parallel to achieve faster response times.

8.4.2 Browser Extension Expansion

The chrome extension of the artefact is limited to browsers that are based on the Chromium open source project. Future expansion would be to support more widely used browsers. According to the bar chart in Figure 42, Firefox is the second most popular browser in the world, so expansion to Firefox would mean broader coverage of the most popular desktop-based browsers.

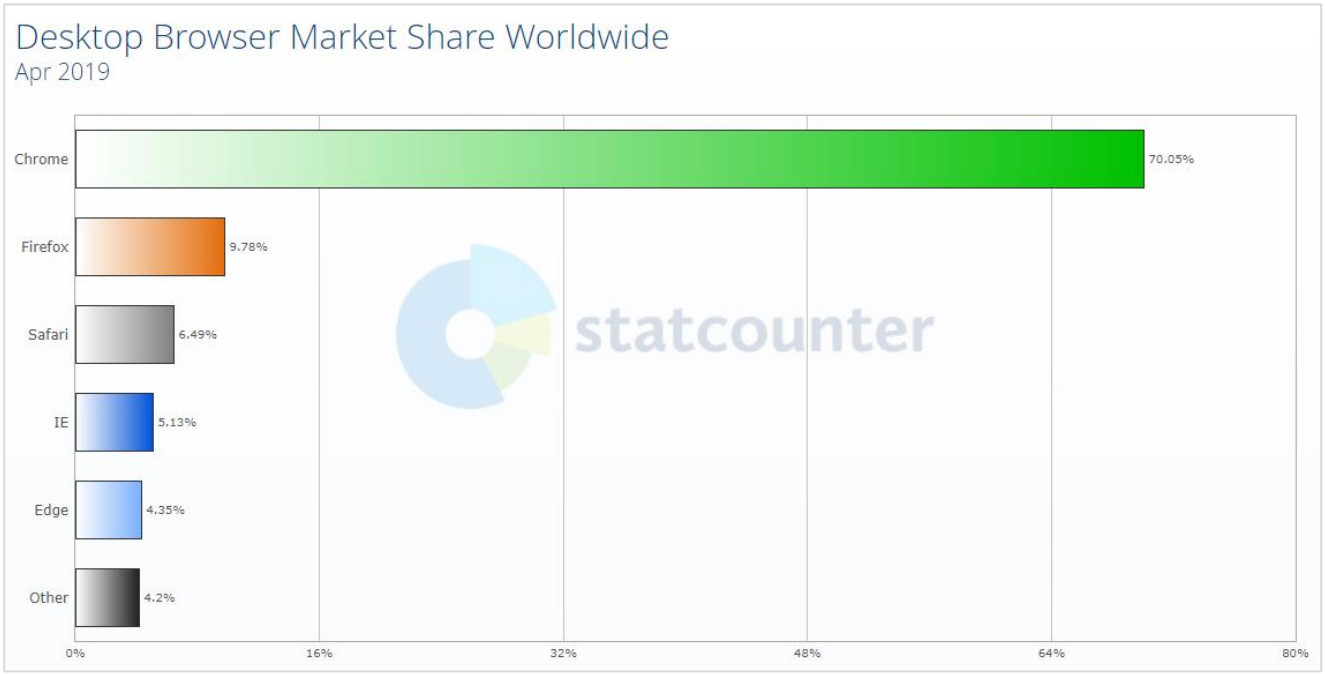


Figure 42 - Desktop web browser market share for April 2019 (StatCounter, 2019)

Research on porting chrome extensions reveals that it is relatively easy to port Chrome or Opera extension to Firefox without much changes as they all use the same WebExtension APIs (Dwomoh et al. 2019) (Callahan, 2015). Mozilla, the company that owns Firefox, provides a website (https://www.extensiontest.com/) for checking the compatibility of Chrome or Opera extensions. Further research revealed that the chrome extension of the artefact is fully compatible with Firefox without requiring any changes and ran as intended when tested on Firefox. However, the CSS and some minor functionality did not work as intended, so future work will be required to resolve the incompatibilities.

The research is positive as it has revealed that it would be straight forward to add support for Opera and Firefox. Evidence of research findings and tests into porting the extension can be found in Appendix U.

Word count: 9748

9 REFERENCES

Agarwal, R. & Umphress, D., 2008. *Extreme Programming for a Single Person Team.* Alabama, s.n.
Beck, K., 1999. Embracing change with extreme programming. *Computer*, October, pp. 70-77.
Bitglass, 2018. *Cloud Adoption Report 2018,* s.l.: s.n.
Bonsai, n.d. *Grow with Bonsai.* [Online]
Available at: https://bonsai.io/pricing
[Accessed 05 05 2019].
Callahan, D., 2015. *Porting Chrome Extensions to Firefox with WebExtensions.* [Online]
Available at: https://hacks.mozilla.org/2015/10/porting-chrome-extensions-to-firefox-with-webextensions/
[Accessed 05 05 2019].
Carr, M. & Verner, J., 1997. *Prototyping and Software Development Approaches,* s.l.: s.n.
Centers for Medicare & Medicaid Services (CMS), 2005. *Select A Development Approach.* s.l.:s.n.
Christakis, M. & Bird, C., n.d. *What Developers Want and Need from Program Analysis.* [Online]
Available at: https://www.microsoft.com/en-us/research/uploads/prod/2016/07/What-Developers-Want-and-Need-from-Program-Analysis-An-Empirical-Study.pdf
[Accessed 28 04 2019].
Clegg, D. & Barker, R., 1994. *Case Method Fast-Track: A RAD Approach,* s.l.: Addison-Wesley.
Dooley, J., 2011. *Object-Oriented Design Principles. In: Software Development and Professional Practice..* CA: Apress, Berkeley.
Dwomoh, G., Truong, A., Petcu, A. & McMillan, K., 2019. *Porting a Google Chrome extension.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Porting_a_Google_Chrome_extension
[Accessed 1 05 2019].
Elastic, n.d. *Elasticsearch-Powered SaaS Offerings.* [Online]
Available at: https://www.elastic.co/cloud/
[Accessed 05 05 2019].
Fronczak, S., 2019. *Software Development Life Cycle: Making Sense of the Different Methodologies.* [Online]
Available at: https://www.plutora.com/wp-content/uploads/2019/02/3_prototyping_model-1024x565.png
[Accessed 2019 03 10].
Hovemeyer, D. & Pugh, W., 2004. Finding Bugs is Easy. *SIGPLAN Notices*.
Jeffrey, P., 2008. *The Onion Architecture.* [Online]
Available at: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/
[Accessed 15 03 2019].
Kambalyal, C., 2010. *3-Tier Architecture.* [Online]
Available at: http://channukambalyal.tripod.com/NTierArchitecture.pdf
[Accessed 15 04 2019].
Kannangaral, S. H., 2013. *Measuring The Impact Of Refactoring On Code quality Improvement Using Internal Measures.* Sri Lanka, s.n.
Lee, T., Lee, J. B. & Peter, H., 2013. A Study of Different Coding Styles Affecting Code Readability. *International Journal of Software Engineering and Its Applications,* 7(5), pp. 413-422.
Liskov, B., 1988. Data Abstraction and Hierarchy. 23(5), pp. 17-34.
Martin, R. C., 2008. *Clean Code: A Handbook of Agile Software Craftsmanship.* s.l.:Prentice Hall.
Martin, R. C., 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* s.l.:Prentice Hall.
Martin, R. C. & Martin, M., 2002. *Agile Principles, Patterns, and Practices in C#.* s.l.:Pearson Prentice Hall.
Microsoft, 2014. *What's New in ASP.NET MVC 5.2.* [Online]
Available at: https://docs.microsoft.com/en-us/aspnet/mvc/overview/releases/whats-new-in-aspnet-mvc-52
[Accessed 20 04 2019].

Microsoft, 2019. *Introduction to ASP.NET Core.* [Online]
Available at: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.2
[Accessed 20 04 2019].
Microsoft, n.d. *Azure Functions Documentation.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/azure-functions/
[Accessed 01 05 2019].
Moz, n.d. *URLs.* [Online]
Available at: https://moz.com/learn/seo/url
[Accessed 05 05 2019].
Mullins, S., 2018. *Solving the debugging dilemma: why we need integration (part 1).* [Online]
Available at: https://www.redweb.com/insights/solving-the-debugging-dilemma-why-we-need-integration-part-1/
[Accessed 20 04 2019].
Okafor, U., 2018. *Solving the debugging dilemma: Ladebug to the rescue (part 2).* [Online]
Available at: https://www.redweb.com/insights/solving-the-debugging-dilemma-ladebug-to-the-rescue-part-2/
[Accessed 20 04 2019].
OWASP Foundation, 2017. *OWASP Top 10 Application Security Risks - 2017.* [Online]
Available at: https://www.owasp.org/index.php/Top_10-2017_Top_10
[Accessed 20 04 2019].
Pal, T., 2018. *Understanding Onion Architecture.* [Online]
Available at: https://www.codeguru.com/csharp/csharp/cs_misc/designtechniques/understanding-onion-architecture.html
[Accessed 20 04 2019].
Paul, B.-D., C, C., H, M. & D, T., 1999. Rapid application development (RAD): An empirical review. *European Journal of Information Systems,* p. 211–223.
Razina, E. & Janzen, D., 2007. *Effects of Dependency Injection on Maintainability.* s.l., Acta Press.
ReSharper, n.d. *ReSharper - The Visual Studio Extension for .NET Developers.* [Online]
Available at: https://www.jetbrains.com/resharper/
[Accessed 28 04 2019].
Robertson, C. et al., 2018. *Code metrics values.* [Online]
Available at: https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2017
[Accessed 28 05 2019].
Schenker, G., 2009. *The Dependency Inversion Principle.* [Online]
Available at: https://lostechies.com/gabrielschenker/2009/01/30/the-dependency-inversion-principle/
[Accessed 08 03 2019].
Seeman, M., 2011. *Dependency Injection in .NET.* s.l.:Wiley India Pvt. Limited.
Smith, S., 2019. *Architecting Modern Web Apps with ASP.NET Core 2 and Azure.* 2.2 ed. Redmond, Washington: DevDiv, .NET and Visual Studio product teams.
StatCounter, 2019. *Desktop Browser Market Share Worldwide.* [Online]
Available at: http://gs.statcounter.com/browser-market-share/desktop/worldwide/#monthly-201904-201904-bar
[Accessed 01 05 2019].
Swati, P., B.V, P. & Ajay, P., 2013. Search Engine Optimization: A Study. *Research Journal of Computer and Information Technology Sciences,* 1(1), pp. 10-13.
w3schools, n.d. *HTML <iframe> sandbox Attribute.* [Online]
Available at: https://www.w3schools.com/tags/att_iframe_sandbox.asp
[Accessed 20 04 2019].
Wells, D., 2001. *Planning/Feedback Loops.* [Online]
Available at: http://www.extremeprogramming.org/map/loops.html
[Accessed 10 03 2019].
Wenzel, M. et al., 2019. *Architectural principles.* [Online]
Available at: https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-

architecture/architectural-principles
[Accessed 20 04 2019].
Wikipedia, n.d. *Multitier architecture.* [Online]
Available at: https://en.wikipedia.org/wiki/Multitier_architecture
[Accessed 10 03 2019].

APPENDIX A - PROJECT PROPOSAL

Undergraduate Project Proposal Form

Please refer to the **Project Handbook Section 4** when completing this form

Degree Title: Computing	**Student's Name:** Uchenna Okafor
	Supervisor's Name: Gernot Liebchen
	Project Title/Area: Providing software as a service

Section 1: Project Overview

1.1 Problem definition - use <u>one sentence</u> to summarise the problem:

Migrating a locally designed product to a cloud-based platform and offer the product as a software service.

1.2 Project description - briefly explain your project:

During my placement year at Redweb, I built an internal product called Ladebug (pronounced Ladybug) and it was very successful, however, it was only used by one team within Redweb's local network and Redweb would like a broader rollout. Redweb offers onsite services and they would want their externally based contractors to be able to use Ladebug even though they may not have access to their local network. They would like for Ladebug to be provided as a service to other companies, and to the community as a free to use service.

My task would be to migrate the local installation of the app to a cloud-based platform and turn Ladebug into a software as a service. This means I will be integrating the current code, database and search engine to work on the cloud, integrate third party services, implement new features and expanding the product with extendibility in mind, because I am to produce a minimal viable product that can later be expanded if the project goes well.

I will also evaluate the security implications and scalability of the project as a cloud service.

1.3 Background - please provide brief background information, e.g., client:

Product

Ladebug [1] is a debugging tool/system that aims to make debugging errors easier for developers. Ladebug is designed on a web-based platform with a companion chrome extension app. The chrome extension detects if your web application threw an error/exception page and displays potential solutions on how to fix your error. These solutions are submitted by other developers who's had the same or similar error and been able to find a fix, therefore Ladebug allows developers to spend less

time debugging errors and more time on being productive. Ladebug is also being used by one team within the company, and now Redweb would like for Ladebug to have a broader rollout.

Client

- Redweb is a digital agency that provides creative, strategic and technical development of screen-based products and services to other companies. Redweb offers services such as digital marketing, content marketing, DevOps & cloud hosting, brand, creativity & design, digital transformations and web development.
- I worked for Redweb for 13 months as a placement student as a web developer
- I built Ladebug as a tool to help with debugging, however it was only used within one team inside the local network
- They want Ladebug to be provided as a service to other companies, and to the community as a free to use service.

1.4 Aims and objectives – what are the aims and objectives of your project?

Aim

Migrate the current system from the local environment to a globally accessible cloud-based service. This includes, refactor the code and search indexes to a working cloud-based instance and ensuring it can be used both by the community and privately by other companies.

Objective

- Expand current system to facilitate the new requirements with scalability and extendibility in mind.
- Allow external companies/teams to have their own private workspace, and for others to be able to use a community version, all on the same app service, not separate websites.
- Deal with the security and scalability implications of the system
- Integrate with Stack Overflow to provide better suggestions to users
- Improve the code of the existing system

Section 2: Artefact

2.1 What is the artefact that you intend to produce?

I will be building a software-based solution, which will be a cloud service implementation of the current existing system. It will be a minimal viable product.

2.2 How is your artefact actionable (i.e., routes to exploitation in the technology domain)?

The existing system is already being used, so it is already actionable, the new proposed solution will be an extension of the current system.

My project can be used by anyone or organization who does web development in ASP.NET and would like to make use of a system that could help them be more productive and save time during development.

Section 3: Evaluation

3.1 How are you going to evaluate your work?

I will evaluate my work/artefact by testing it in environment it was built for. I will have my client test it amongst the various teams within their company. Since the project is to allow community and private use, I will like to have interested developers test the community workspace and feedback.

3.2 Why is this project honours worthy?

The project is based on topics of the advanced development unit. It's about building a cloud-based service, integrating cloud services, and critically evaluating the practises and technologies used.

3.3 How does this project relate to your degree title outcomes?

The outcomes of my degree title are to investigate a computing problem by building a system and evaluate the practise/technologies used in the investigation. My project is about solving a real-world problem, which is to migrate a local system to a cloud platform, as a cloud service. I will then critically evaluate the practises and technologies used.

3.4 How does your project meet the BCS Undergraduate Project Requirements?

The project tackles a substantial computing problem-solving project, that meets a real need in the software development industry. My project focuses on the build of a software artefact, on a cloud-based platform, whereby the software artefact is provided as a cloud service, which is built using relevant web technologies (HTML, ASP.NET, Search Engines) and integration of third-party cloud and non-cloud services.

3.5 What are the risks in this project and how are you going to manage them?

Risk	Likelihood	Impact	Effect on project	Risk reduction actions
Hard drive failure	L	H	I will lose all project code and important documents	Make sure I use version control systems. For example, using OneDrive to save documents. Also, use GitHub for code. This means if it fails on my computer, then a copy will be available on the cloud service.
Not having enough funding for cloud resources	M	H	I may not be able to achieve the aims and objectives set	Just make sure that I won't be using up more resources than I have allocated. Agree with my client on how much financial help I can get to help with cloud services I will be using.
Scope creep	M	H	I may not have enough time to finish the project within the given time	Ensure that the project objectives are well defined and controlled. I will also make sure that I only do the things in the specification and not get carried away with adding new features.

Technical integration risk	L	M	The product may not work as defined in the project specifications	Do enough research in the initial planning stage to ensure that the proposed systems/technologies being used can be integrated

Section 4: References

4.1 Please provide references if you have used any.

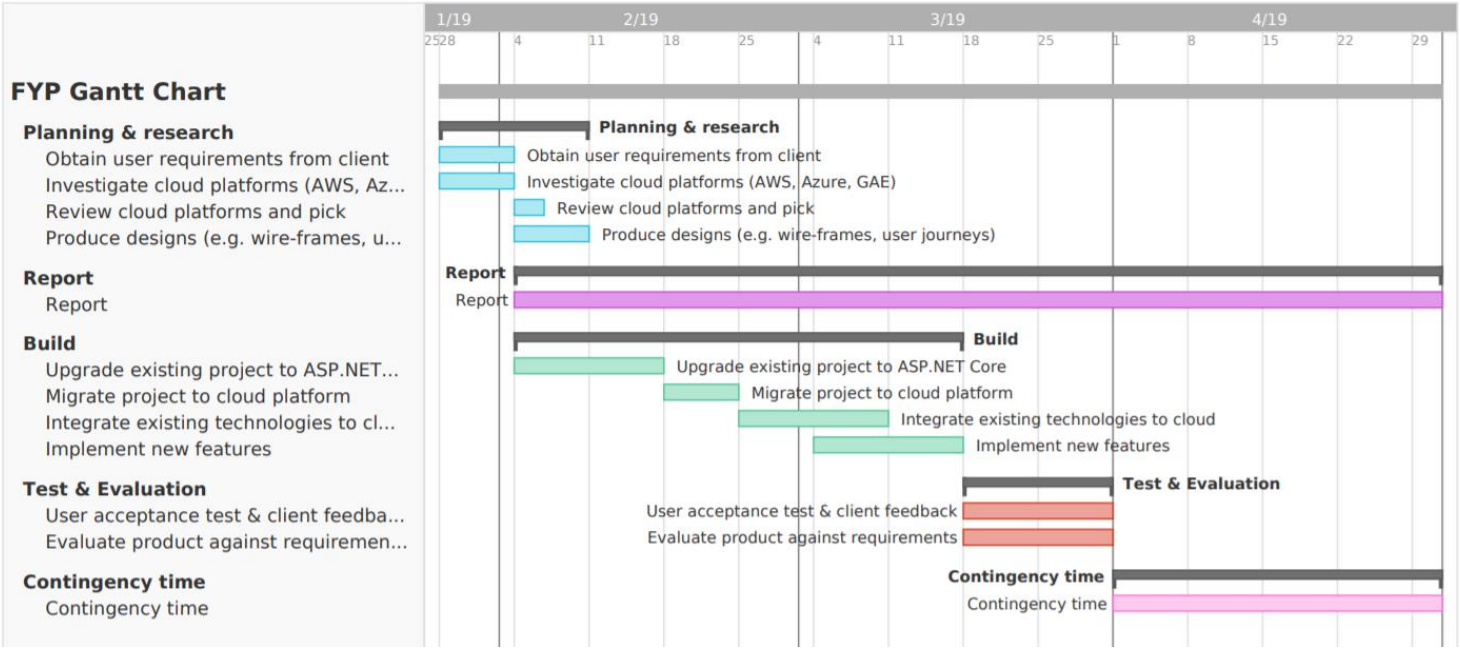
1] - Uchenna Okafor, 2018. *Solving the debugging dilemma: Ladebug to the rescue (part 2)*. 21 June 2018. Available from: https://www.redweb.com/agency/blog/solving-the-debugging-dilemma-ladebug-to-the-rescue-part-2 [Accessed 2 October 2018].

Section 5: Ethics (please delete as appropriate)

5.1 Have you submitted the ethics checklist to your supervisor? **No**

5.2 Has the checklist been approved by your supervisor? **No**

Section 6: Proposed Plan (please attach your Gantt chart below)



APPENDIX B - ETHICS FORM

Bournemouth University

Research Ethics Checklist

About Your Checklist	
Reference Id	22603
Date Created	12/10/2018 10:53:45
Status	Approved
Date Approved	01/05/2019 11:15:28
Date Submitted	27/02/2019 21:24:41

Researcher Details	
Name	Uchenna Okafor
Faculty	Faculty of Science & Technology
Status	Undergraduate (BA, BSc)
Course	BSc (Hons) Computing
Have you received external funding to support this research project?	No

Project Details	
Title	Software as a service
Start Date of Project	07/01/2018
End Date of Project	31/05/2019
Proposed Start Date of Data Collection	17/03/2019
Supervisor	Gernot Liebchen
Approver	Gernot Liebchen
Summary - no more than 500 words (including detail on background methodology, sample, outcomes, etc.)	

Summary - no more than 500 words (including detail on background methodology, sample, outcomes, etc.)

During my placement year at Redweb, I built an internal product called Ladebug (pronounced Ladybug) and it was very successful, however, it was only used by one team within Redweb's local network and Redweb would like a broader rollout. Redweb offers onsite services and they would want their externally based contractors to be able to use Ladebug even though they may not have access to the local Redweb network. They would also like for other internal teams within the company to be able to use Ladebug. In addition, Redweb would like Ladebug to be accessible by other companies and developers outside of Redweb as a service.

My task would be to port the local installation of the app to a cloud-based platform and turn Ladebug into software as a service. This means in my project I will be integrating the current code, database and search engine to work on the cloud, and expanding the product with scalability in mind so that it can be used by other companies and individuals and integrate other third-party services to enhance the effectiveness of the product.

There will also be a focus on scalability and the security implications of having a cloud app that.

None of the questions apply to my study

I am confirming that my proposed project does not:

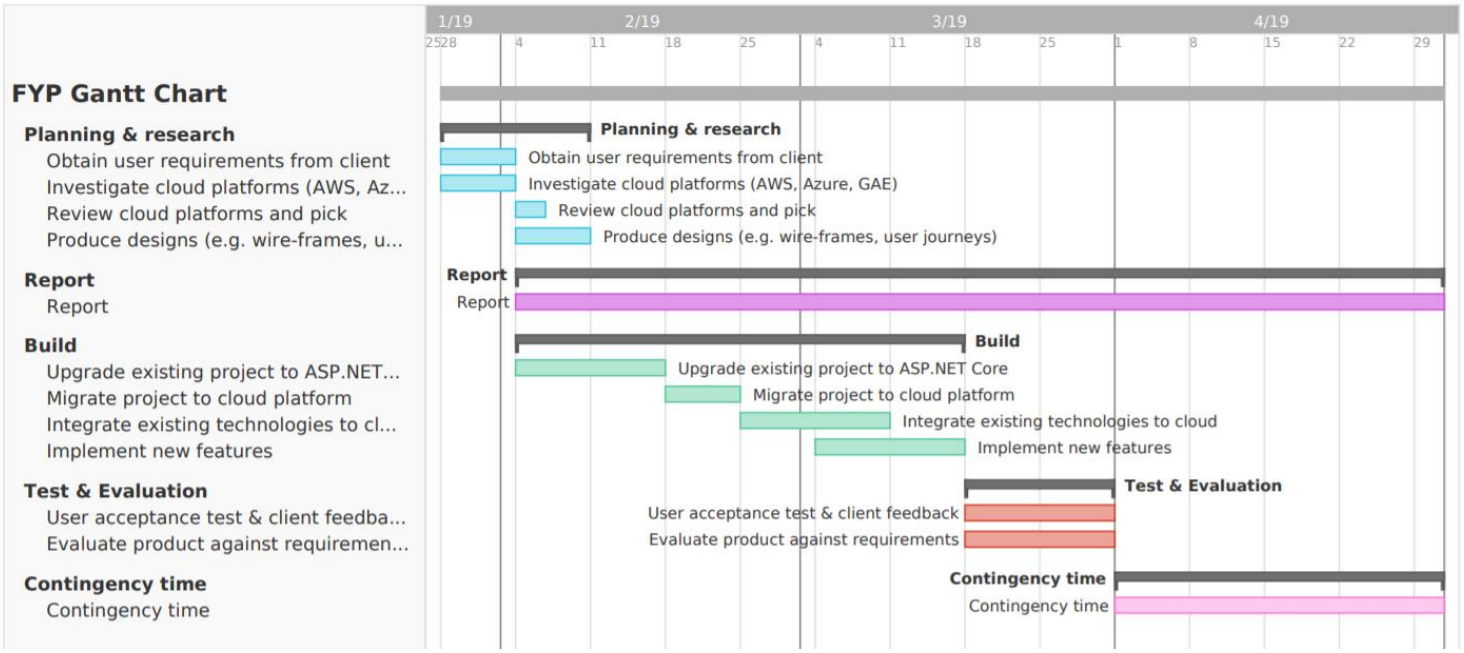
- Involve human participants
- Involve the use of human tissue
- Involve medical research requiring NHS ethical / REC Approval
- Involve the use of animals (or tissues/fluids derived from animals)
- Involve access to identifiable personal data for living individuals not already in the public domain
- Involve increased danger of physical or psychological harm for researcher(s) or subject(s)
- Raise any ethical issues associated with the use of genetically modified organisms

On this basis, my proposed project does not require a formal ethics review.

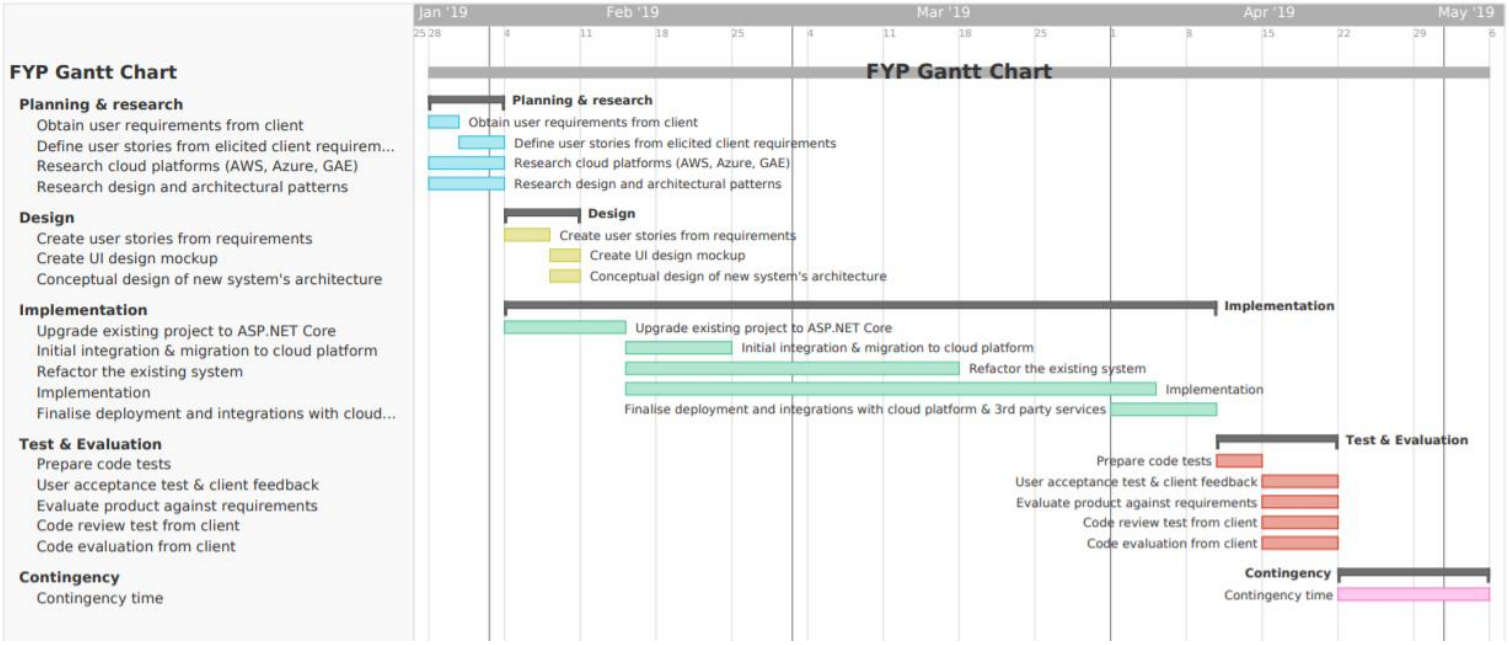
If any changes to the project involve any of the criteria above, I undertake to resubmit the project for formal ethical approval.

APPENDIX C - GANTT CHART PLANNING

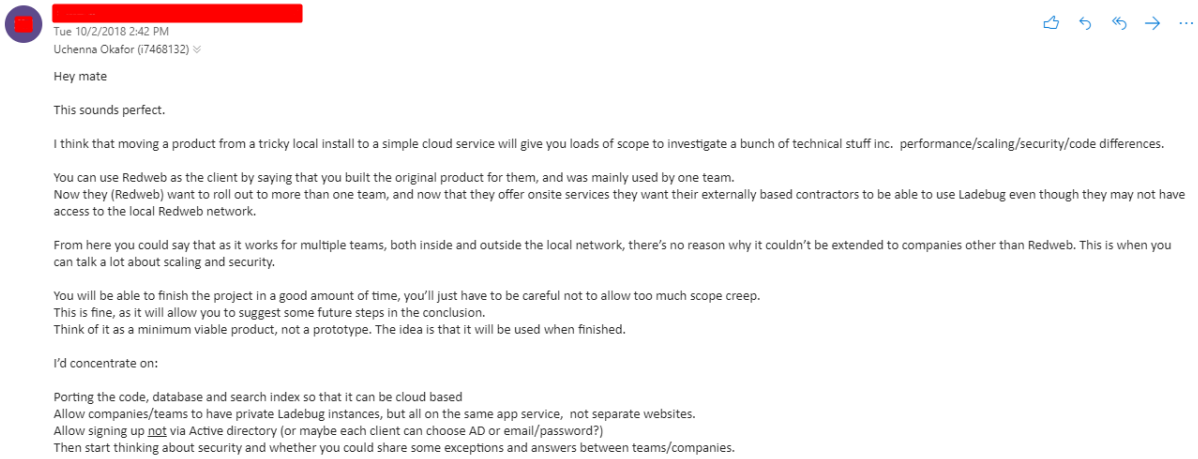
Initial Gantt Chart



Final Gantt Chart



APPENDIX D - CORE REQUIREMENTS FROM CLIENT



[redacted]
Tue 10/2/2018 2:42 PM
Uchenna Okafor (i7468132)

Hey mate

This sounds perfect.

I think that moving a product from a tricky local install to a simple cloud service will give you loads of scope to investigate a bunch of technical stuff inc. performance/scaling/security/code differences.

You can use Redweb as the client by saying that you built the original product for them, and was mainly used by one team.
Now they (Redweb) want to roll out to more than one team, and now that they offer onsite services they want their externally based contractors to be able to use Ladebug even though they may not have access to the local Redweb network.

From here you could say that as it works for multiple teams, both inside and outside the local network, there's no reason why it couldn't be extended to companies other than Redweb. This is when you can talk a lot about scaling and security.

You will be able to finish the project in a good amount of time, you'll just have to be careful not to allow too much scope creep.
This is fine, as it will allow you to suggest some future steps in the conclusion.
Think of it as a minimum viable product, not a prototype. The idea is that it will be used when finished.

I'd concentrate on:

Porting the code, database and search index so that it can be cloud based
Allow companies/teams to have private Ladebug instances, but all on the same app service, not separate websites.
Allow signing up not via Active directory (or maybe each client can choose AD or email/password?)
Then start thinking about security and whether you could share some exceptions and answers between teams/companies.

APPENDIX E - REQUIREMENTS AND PRIORITIES

Functional user requirements

ID	Category	Description	Rationale	MoSCoW Priority
F1	Authentication	Users should be able to register and log in with a local account	Better accessibility	M
F2	Authentication	Users should be able to register and log in with an external login provider using Open Authorization (OAuth)	Better accessibility and user experience	M
F3	Teams	User should be able to join a team	Client requirement	M
F4	Teams	Users should be able to see a list of all the teams they're a member of	Client requirement	M
F5	Teams	Users should be restricted to accessing or modifying content in teams they're not a member of.	Client requirement	M
F6	Answers	Users should be able to edit answers they've contributed	Client requirement	S
F7	Answers	Users should be able to delete answers they've contributed	Client requirement	C
F8	Answers	Users should be able to delete error posts they've made	Client requirement	C
F9	Chrome Extension	The user should be able to submit contributions to teams they belong to using the chrome extension.	Client requirement	M
F10	Chrome Extension	Implement a modularized component that allows for errors to be parsed on the client side	Client requirement	M
F11	Chrome Extension	The chrome extension should suggest answers from the teams the user is in and from StackOverflow	Client requirement	M

Non-functional user requirements

ID	Category	Description	Rationale	MoSCoW Priority
NF1	User Experience	The web app and Chrome extension should be robust	Provides good user experience	S
NF2	User Experience	The web app and Chrome extension should always provide user feedback when an error has occurred	Provides good user experience	S
NF3	User Experience	The system should be easy to use	Provides good user experience	M

Technical requirements

ID	Category	Description	Rationale	MoSCoW Priority
T1	System	Upgrade from ASP.NET MVC to ASP.NET Core MVC	Newer and supports the goals of the dissertation	S
T2	System	Integrate answer finding process with StackOverflow	To enhance the usefulness of the system	S
T3	Front End	The web app should be responsive	So, it can work on other devices like mobile and tablets	S
T4	Cloud	Deploy artefact to the cloud	Global accessibility	S
T5	System	The system should be built using SOLID design principles	Extensibility, maintainability and flexibility	M
T6	System	The system should be built using architectural principles	Extensibility and maintainability	M
T7	System	The system should be built to be maintainable, extendible and flexible	Maintainability	M

APPENDIX F - CONCEPTUAL DIAGRAMS OF LADEBUG

Figure 43 – The typical ASP.NET MVC error page a user would encounter (Mullins, 2018)

Figure 44 – The chrome extension displays potential solutions to the users based on their error page (Mullins, 2018)

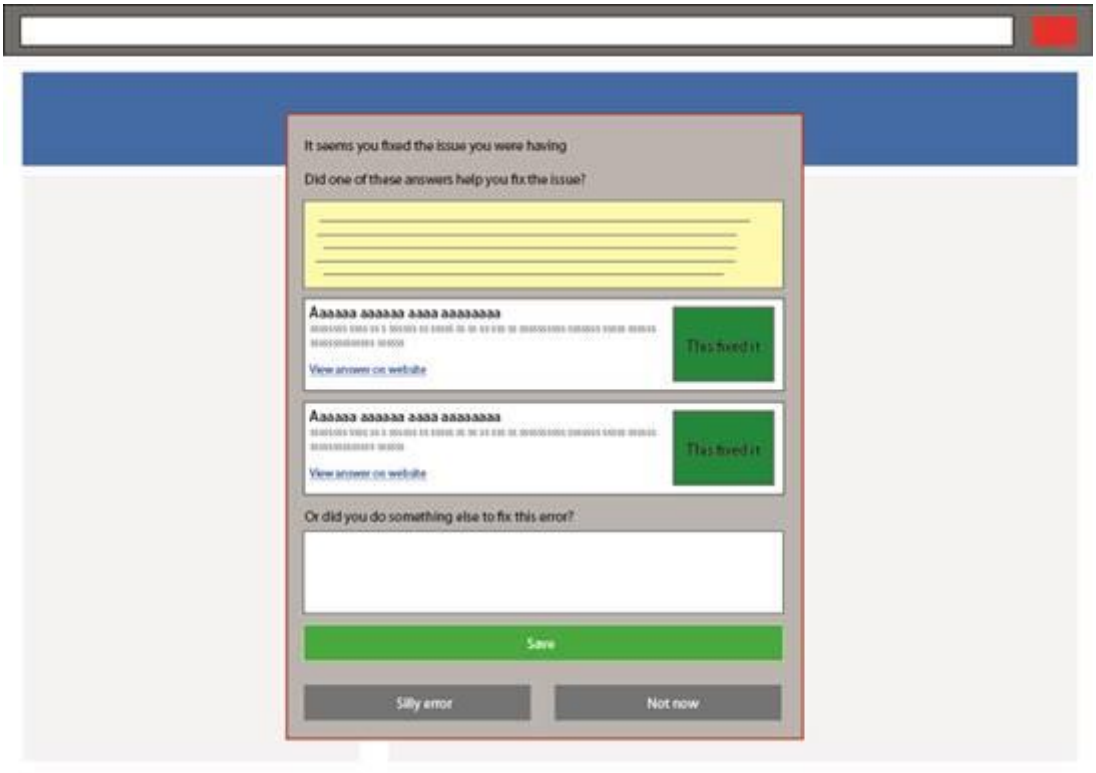


Figure 45 - An example of how the "How did you fix it" dialog would look and feel (Mullins, 2018)

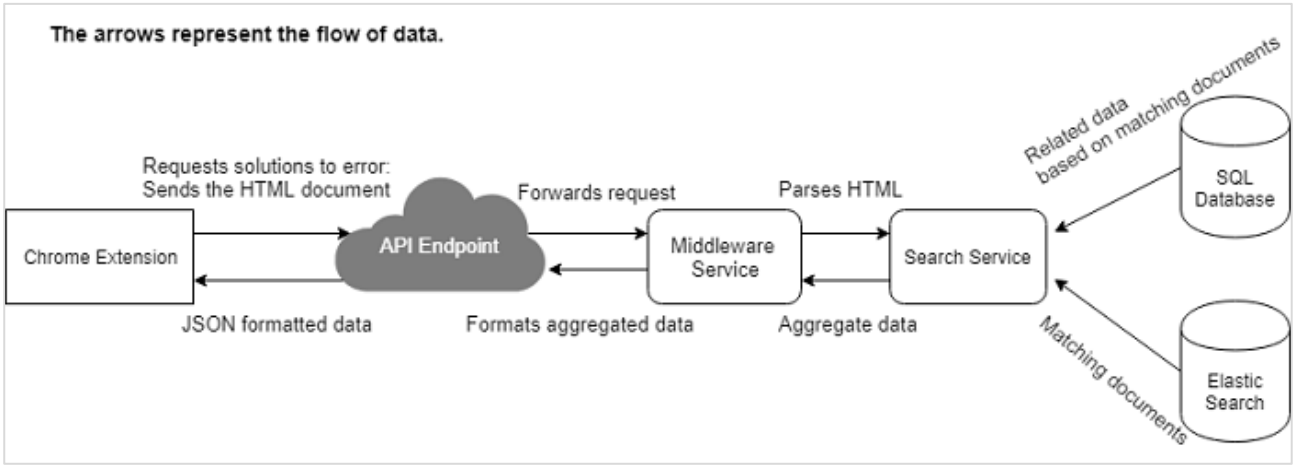


Figure 46 - An overview diagram of how the existing system works (Okafor, 2018)

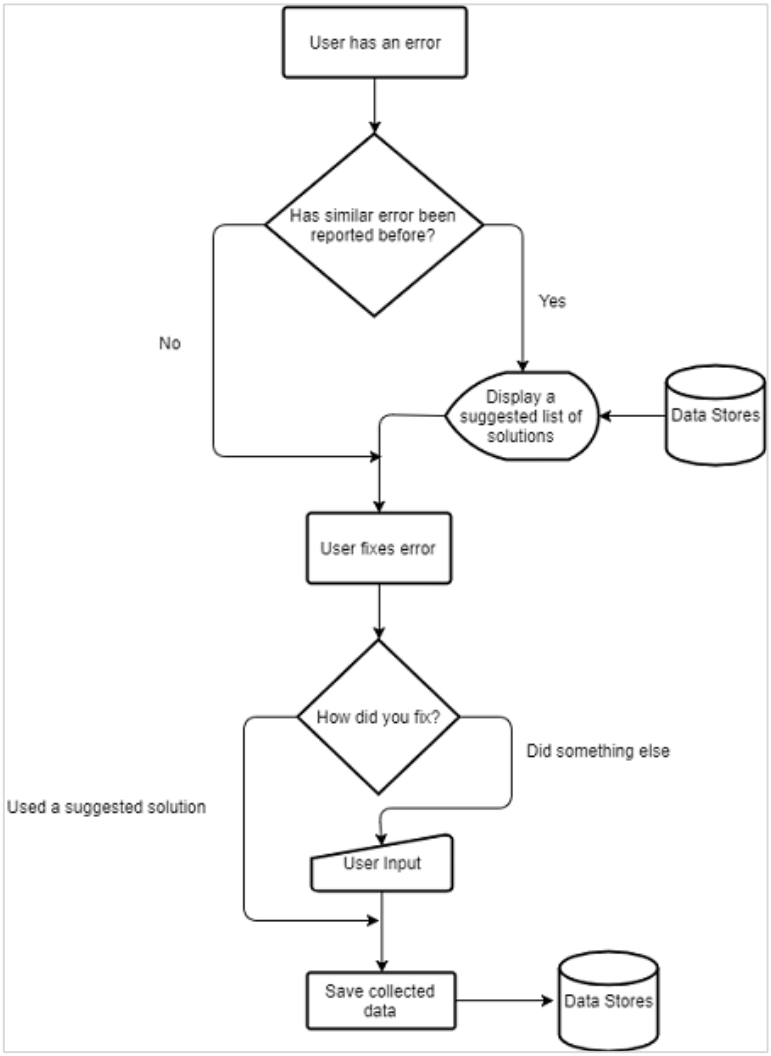


Figure 47 - Flow diagram of how Ladebug works (Okafor, 2018)

APPENDIX G - ERROR PARSING IN THE EXISTING SYSTEM

Server Error in '/' Application.

Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.GenericPage.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: Umbraco.Web.Mvc.ModelBindingException: Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.GenericPage.

Source Error:

```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[ModelBindingException: Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.GenericPage.]
   Umbraco.Web.Mvc.RenderModelBinder.ThrowModelBindingException(Boolean sourceContent, Boolean modelContent, Type sourceType, Type modelType) +586
   Umbraco.Web.Mvc.RenderModelBinder.BindModel(Object source, Type modelType, CultureInfo culture) +553
   Umbraco.Web.Mvc.UmbracoViewPage`1.SetViewData(ViewDataDictionary viewData) +201
   System.Web.Mvc.RazorView.RenderView(ViewContext viewContext, TextWriter writer, Object instance) +137
   Umbraco.Core.Profiling.ProfilingView.Render(ViewContext viewContext, TextWriter writer) +194
   System.Web.Mvc.ViewResultBase.ExecuteResult(ControllerContext context) +375
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultFilterRecursive(IList`1 filters, Int32 filterIndex, ResultExecutingContext preContext, ControllerContext controllerContext, Ac
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultFilterRecursive(IList`1 filters, Int32 filterIndex, ResultExecutingContext preContext, ControllerContext controllerContext, Ac
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultFilterRecursive(IList`1 filters, Int32 filterIndex, ResultExecutingContext preContext, ControllerContext controllerContext, Ac
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultFilterRecursive(IList`1 filters, Int32 filterIndex, ResultExecutingContext preContext, ControllerContext controllerContext, Ac
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultFilterRecursive(IList`1 filters, Int32 filterIndex, ResultExecutingContext preContext, ControllerContext controllerContext, Ac
   System.Web.Mvc.ControllerActionInvoker.InvokeActionResultWithFilters(ControllerContext controllerContext, IList`1 filters, ActionResult actionResult) +81
   System.Web.Mvc.Async.<>c__DisplayClass21.<BeginInvokeAction>b__1e(IAsyncResult asyncResult) +188
   System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeAction(IAsyncResult asyncResult) +38
   System.Web.Mvc.Controller.<BeginExecuteCore>b__1d(IAsyncResult asyncResult, ExecuteCoreState innerState) +29
   System.Web.Mvc.Async.WrappedAsyncVoid`1.CallEndDelegate(IAsyncResult asyncResult) +73
   System.Web.Mvc.Controller.EndExecuteCore(IAsyncResult asyncResult) +52
   System.Web.Mvc.Async.WrappedAsyncVoid`1.CallEndDelegate(IAsyncResult asyncResult) +39
   System.Web.Mvc.Controller.EndExecute(IAsyncResult asyncResult) +38
   System.Web.Mvc.MvcHandler.<BeginProcessRequest>b__5(IAsyncResult asyncResult, ProcessRequestState innerState) +43
   System.Web.Mvc.Async.WrappedAsyncVoid`1.CallEndDelegate(IAsyncResult asyncResult) +73
   System.Web.Mvc.MvcHandler.EndProcessRequest(IAsyncResult asyncResult) +38
   System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute() +602
   System.Web.HttpApplication.ExecuteStepImpl(IExecutionStep step) +195
   System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +128
```

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.7.3282.0

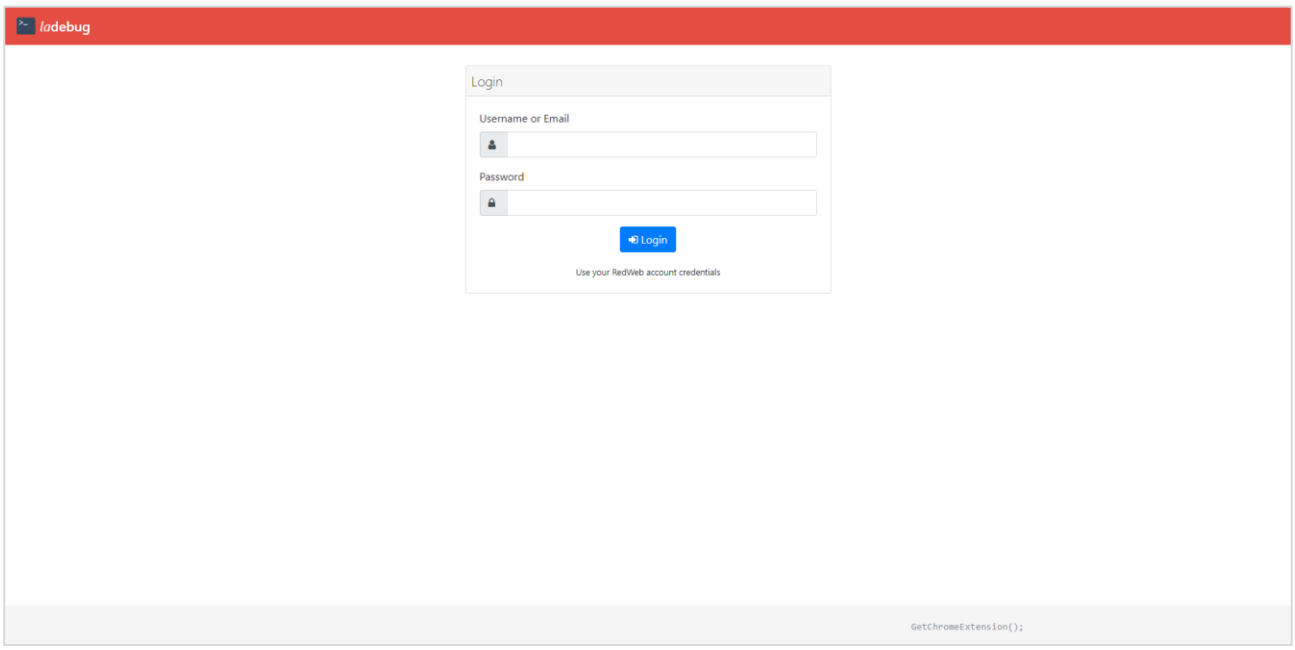
Figure 48 - An ASP.NET MVC yellow page (Error page)

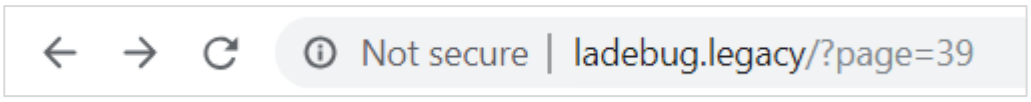

```
{
   "Title":" Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.GenericPage. ",
   "Subtitle":"Umbraco.Web.Mvc.ModelBindingException: Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.We
   "Type":3,
   "Properties":{
      "Description":"An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the
      "Exception Details":"Umbraco.Web.Mvc.ModelBindingException: Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content typ
      "Exception Type":"Umbraco.Web.Mvc.ModelBindingException",
      "Exception Message":" Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.C
      "Source Error":"\n\nAn unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of t
      "Stack Trace":"
[ModelBindingException: Cannot bind source content type Umbraco.Web.PublishedContentModels.HomePage to model content type Umbraco.Web.PublishedContentModels.Gener
<>c__DisplayClass21.<BeginInvokeAction>b__1e(IAsyncResult asyncResult) +188\n   System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeAction(IAsyncResult asy
<BeginExecuteCore>b__1d(IAsyncResult asyncResult, ExecuteCoreState innerState) +29\n   System.Web.Mvc.Async.WrappedAsyncVoid`1.CallEndDelegate(IAsyncResult asyncR
<BeginProcessRequest>b__5(IAsyncResult asyncResult, ProcessRequestState innerState) +43\n   System.Web.Mvc.Async.WrappedAsyncVoid`1.CallEndDelegate(IAsyncResult a
   }
}
```

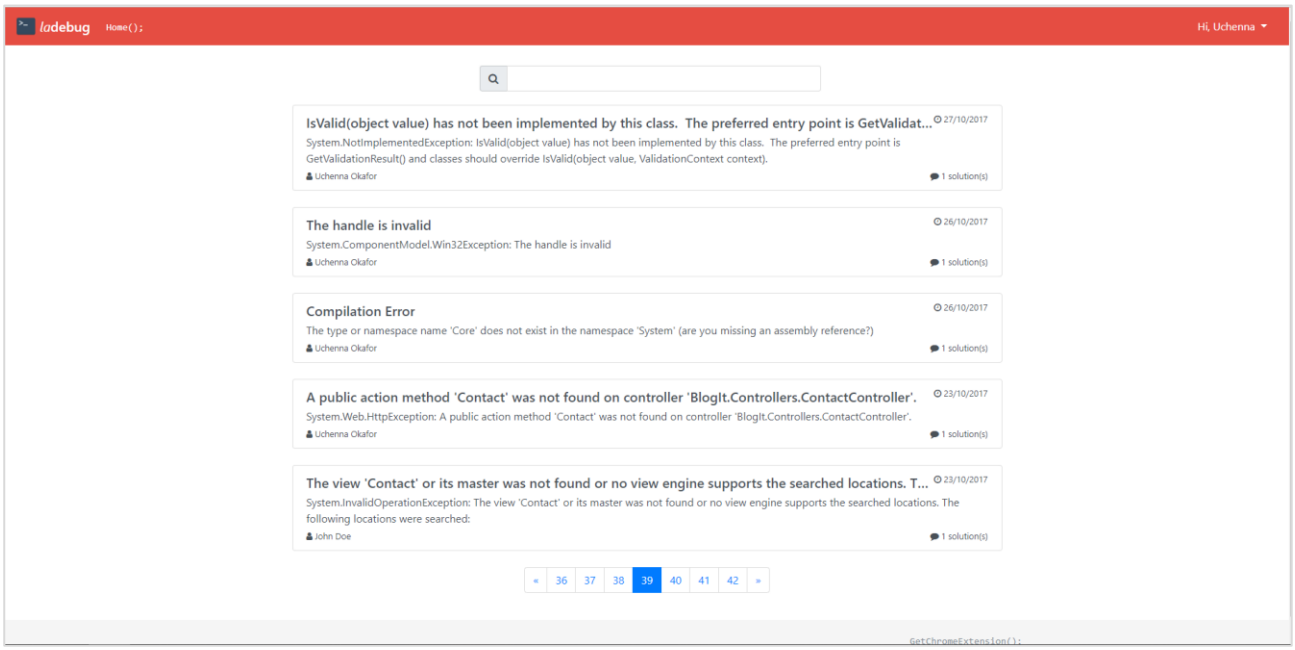

Figure 49 – A JSON object representation of the ASP.NET error page in Figure 48 after it has been parsed by the error parser

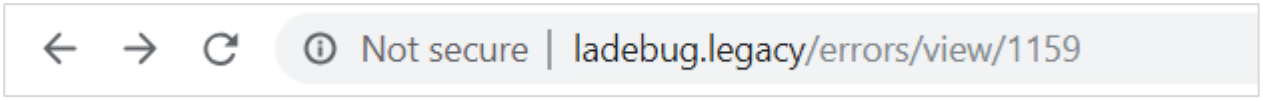
APPENDIX H - SCREENSHOTS OF THE EXISTING SYSTEM

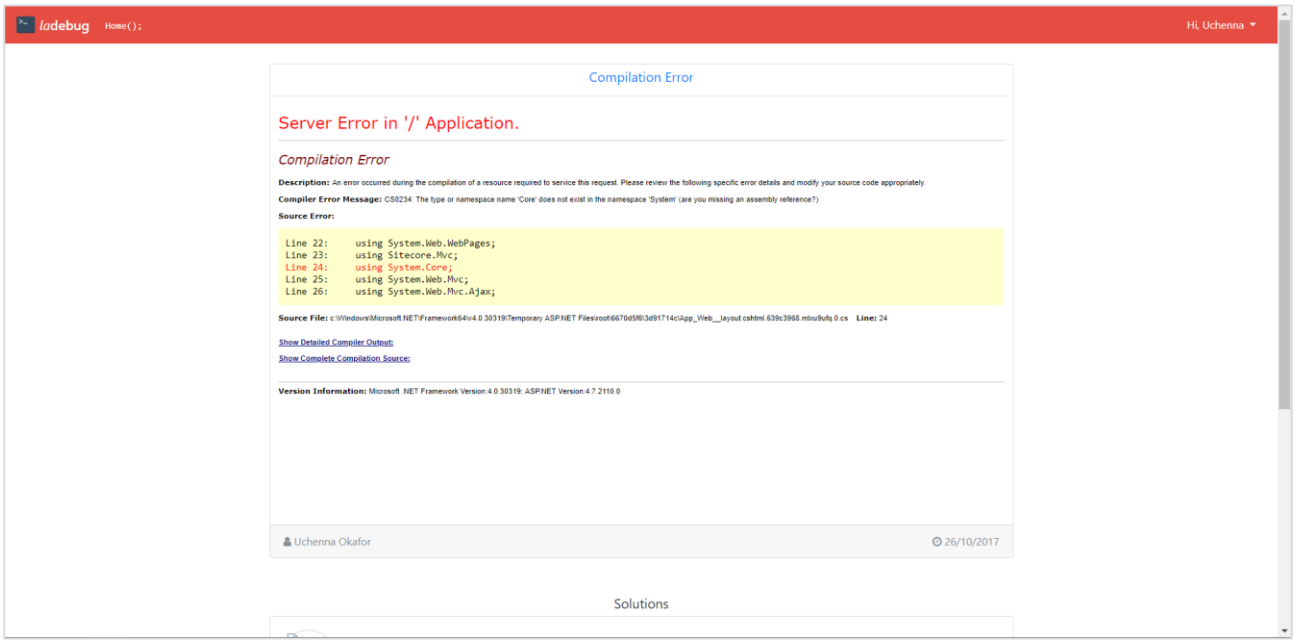
Web App

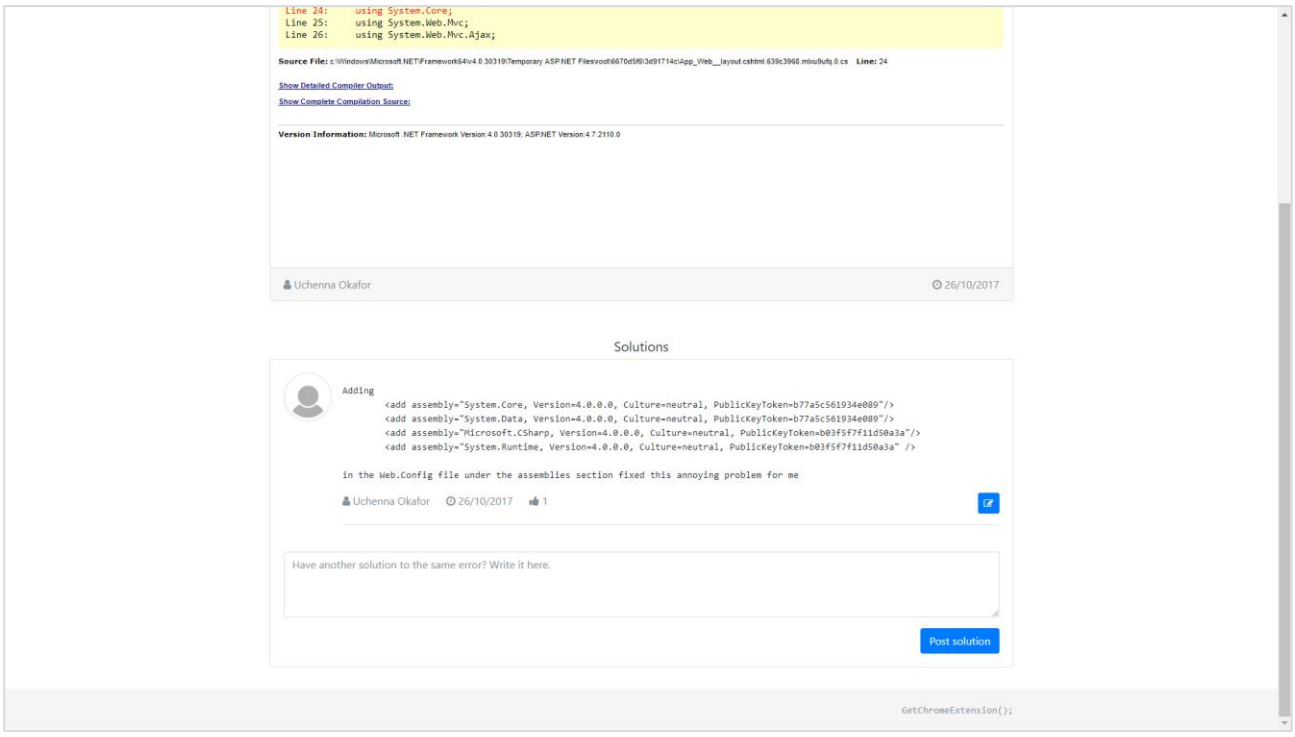


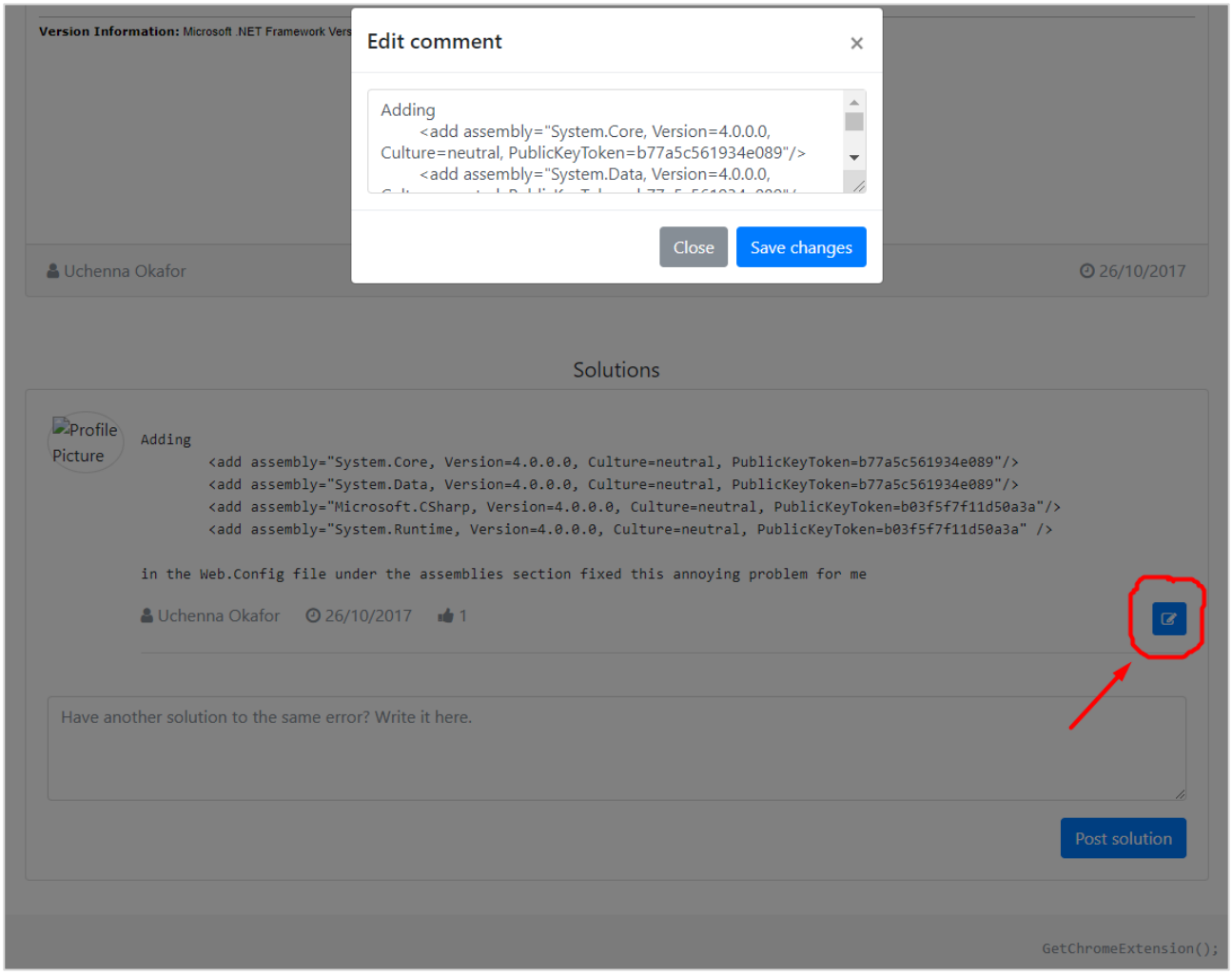


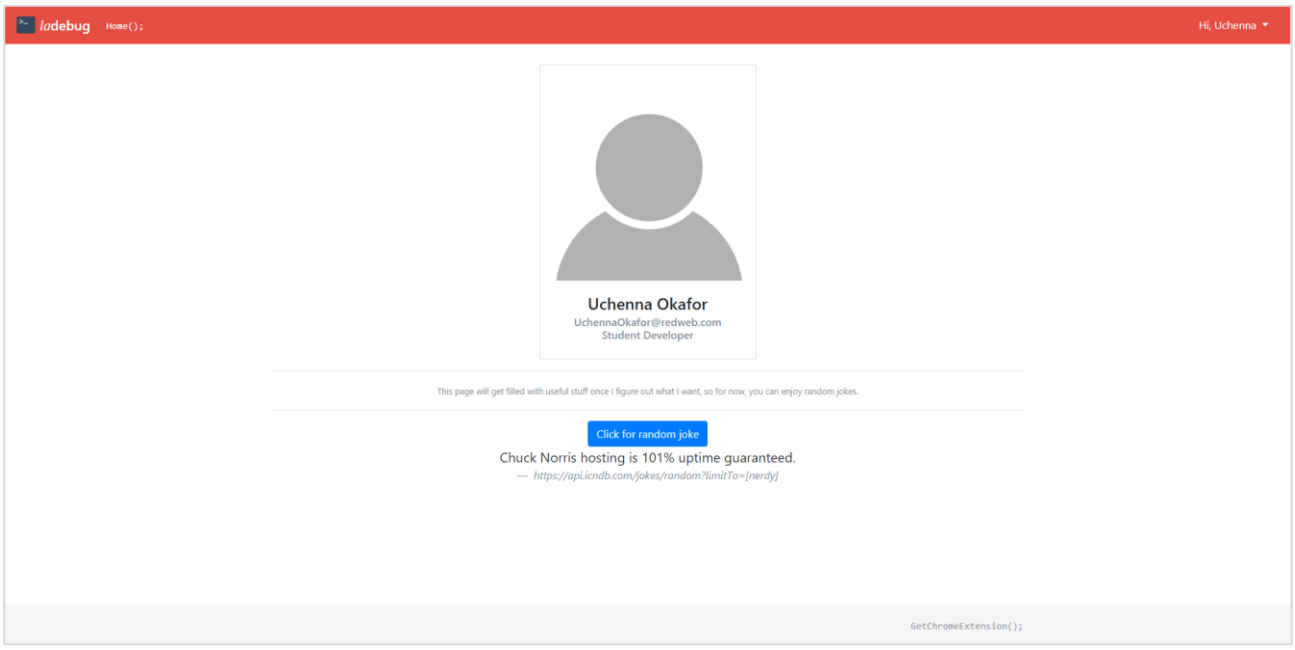


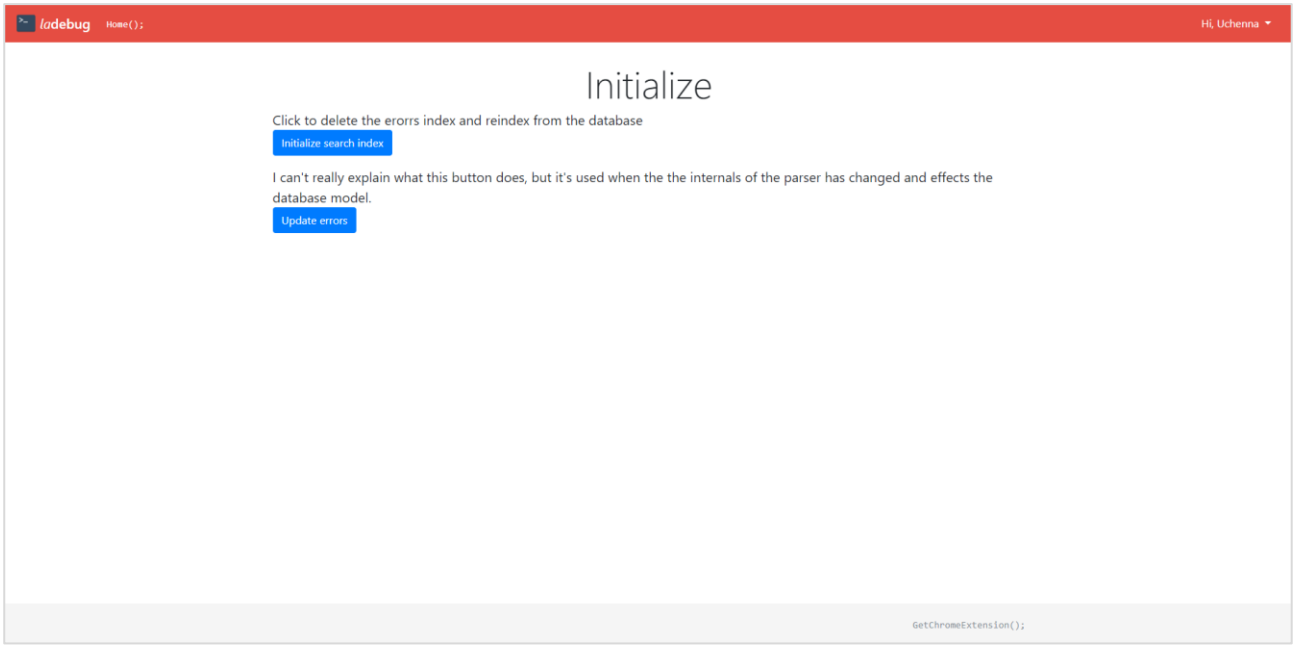

Not secure | ladebug.legacy/errors/view/1159

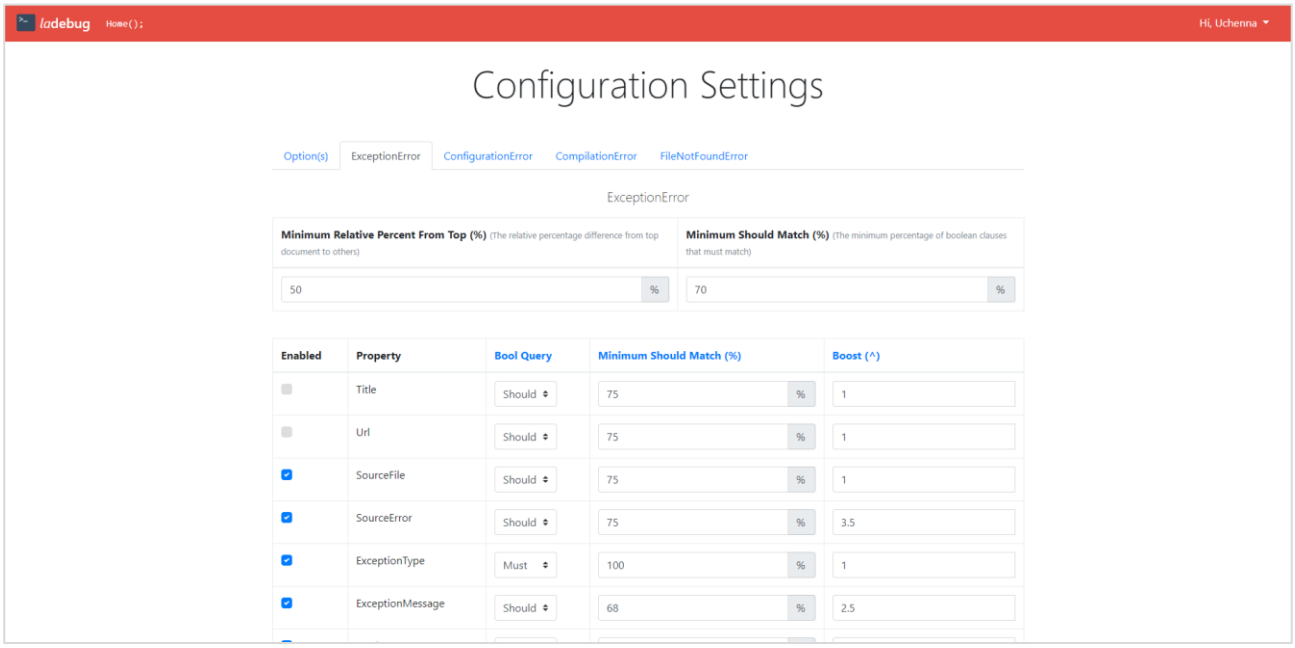

ladebug
Home();
Hi, Uchenna
Compilation Error
Server Error in '/' Application.
Compilation Error
Description: An error occurred during the compilation of a resource required to service this request. Please review the following specific error details and modify your source code appropriately.
Compiler Error Message: CS0234: The type or namespace name 'Core' does not exist in the namespace 'System' (are you missing an assembly reference?)
Source Error:
Line 22: using System.Web.WebPages;
Line 23: using Sitecore.Mvc;
Line 24: using System.Core;
Line 25: using System.Web.Mvc;
Line 26: using System.Web.Mvc.Ajax;
Source File: c:\Windows\Microsoft.NET\Framework64\v4.0.30319\Temporary ASP.NET Files\root\6670d5f6\3d91714c\App_Web__layout.cshtml.639c3968.mbiu9ufq.0.cs Line: 24
Show Detailed Compiler Output:
Show Complete Compilation Source:
Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.7.2110.0
Uchenna Okafor
26/10/2017
Solutions


Line 24: using System.Core;
Line 25: using System.Web.Mvc;
Line 26: using System.Web.Mvc.Ajax;
Source File: c:\Windows\Microsoft.NET\Framework64\v4.0.30319\Temporary ASP.NET Files\root\6670d5f6\3d91714c\App_Web__layout.cshtml.639c3968.mbiu9ufq.0.cs Line: 24
Show Detailed Compiler Output:
Show Complete Compilation Source:
Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.7.2110.0
Uchenna Okafor
26/10/2017
Solutions
Adding
<add assembly="System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
<add assembly="System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
<add assembly="Microsoft.CSharp, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
<add assembly="System.Runtime, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
in the Web.Config file under the assemblies section fixed this annoying problem for me
Uchenna Okafor
26/10/2017
1
Have another solution to the same error? Write it here.
Post solution
GetChromeExtension();

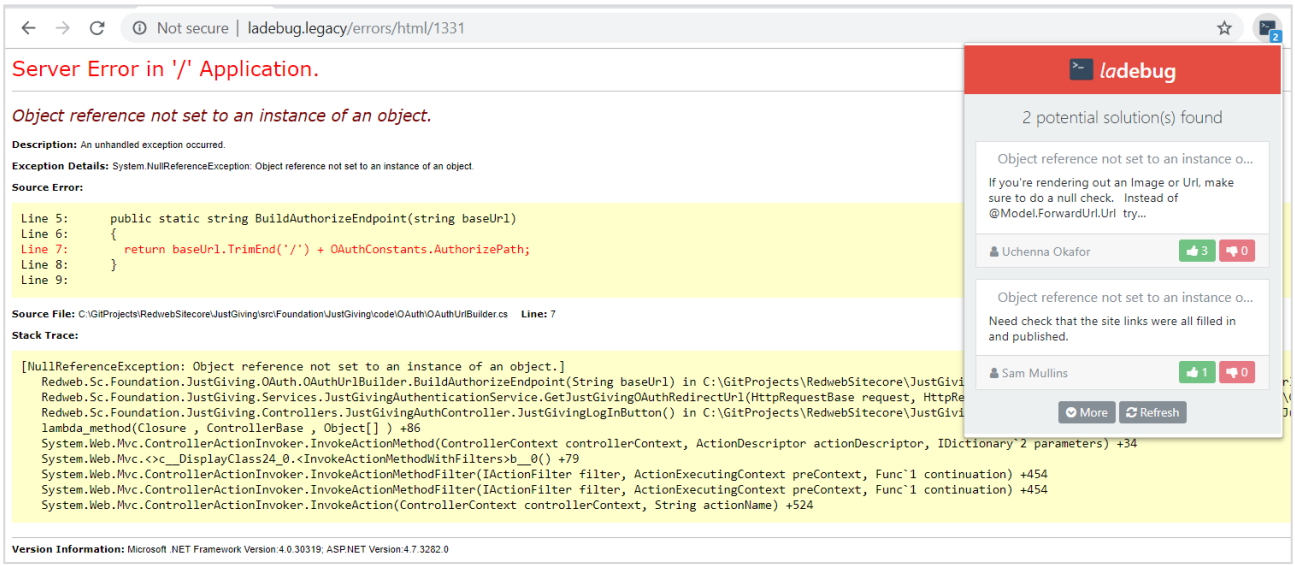

Version Information: Microsoft .NET Framework Vers
Edit comment
Adding
<add assembly="System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
<add assembly="System.Data, Version=4.0.0.0,
Close
Save changes
Uchenna Okafor
26/10/2017
Solutions
Profile Picture
Adding
<add assembly="System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
<add assembly="System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
<add assembly="Microsoft.CSharp, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
<add assembly="System.Runtime, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
in the Web.Config file under the assemblies section fixed this annoying problem for me
Uchenna Okafor
26/10/2017
1
Have another solution to the same error? Write it here.
Post solution
GetChromeExtension();

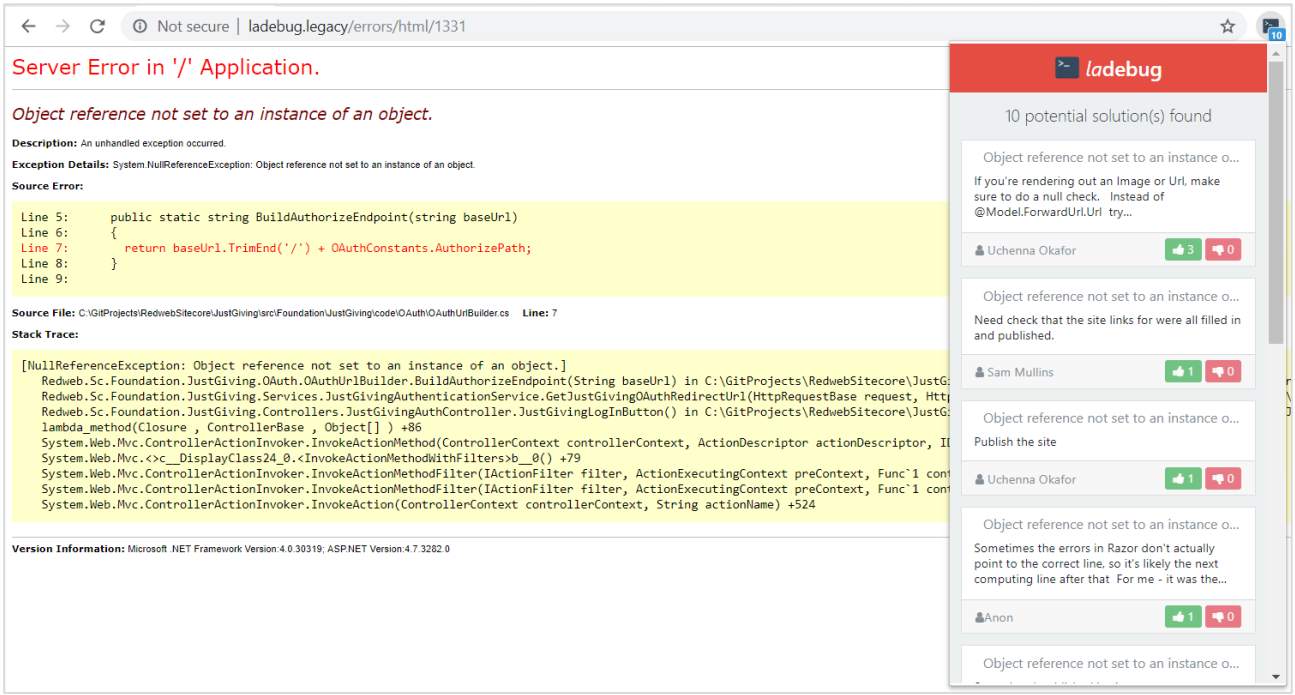

ladebug
Home();
Hi, Uchenna
Uchenna Okafor
UchennaOkafor@redweb.com
Student Developer
This page will get filled with useful stuff once I figure out what I want, so for now, you can enjoy random jokes.
Click for random joke
Chuck Norris hosting is 101% uptime guaranteed.
— https://api.icndb.com/jokes/random?limitTo=[nerdy]
GetChromeExtension();

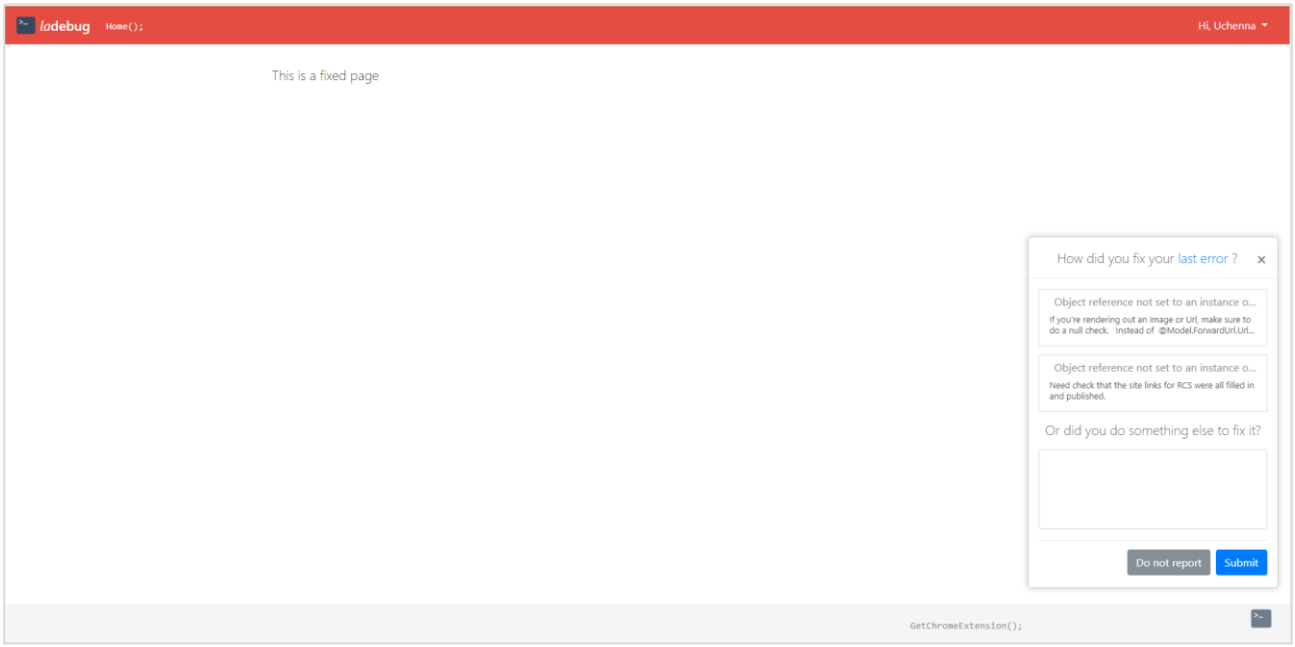

ladebug
Home();
Hi, Uchenna
Initialize
Click to delete the erorrs index and reindex from the database
Initialize search index
I can't really explain what this button does, but it's used when the the internals of the parser has changed and effects the database model.
Update errors
GetChromeExtension();


ladebug
Home();
Hi, Uchenna
Configuration Settings
Option(s)
ExceptionError
ConfigurationError
CompilationError
FileNotFoundError
ExceptionError
Minimum Relative Percent From Top (%) (The relative percentage difference from top document to others)
Minimum Should Match (%) (The minimum percentage of boolean clauses that must match)
50
%
70
%
Enabled
Property
Bool Query
Minimum Should Match (%)
Boost (^)
Title
Should
75
%
1
Url
Should
75
%
1
SourceFile
Should
75
%
1
SourceError
Should
75
%
3.5
ExceptionType
Must
100
%
1
ExceptionMessage
Should
68
%
2.5

Chrome Extension






ladebug
Home();
Hi, Uchenna
This is a fixed page
How did you fix your last error ?
Object reference not set to an instance o...
If you're rendering out an Image or Url, make sure to do a null check. Instead of @Model.ForwardUrl.Url...
Object reference not set to an instance o...
Need check that the site links for RCS were all filled in and published.
Or did you do something else to fix it?
Do not report
Submit
GetChromeExtension();

APPENDIX I - EXISTING AND PROPOSED ERD DIAGRAMS

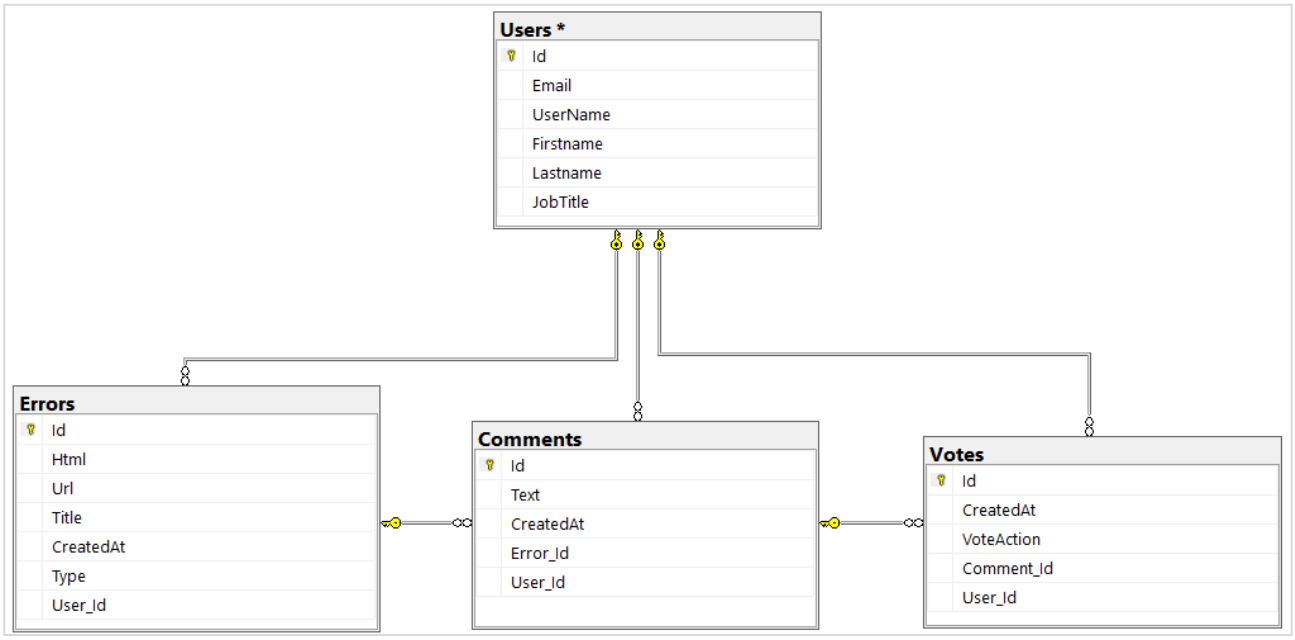


Figure 50 - ERD of the existing system

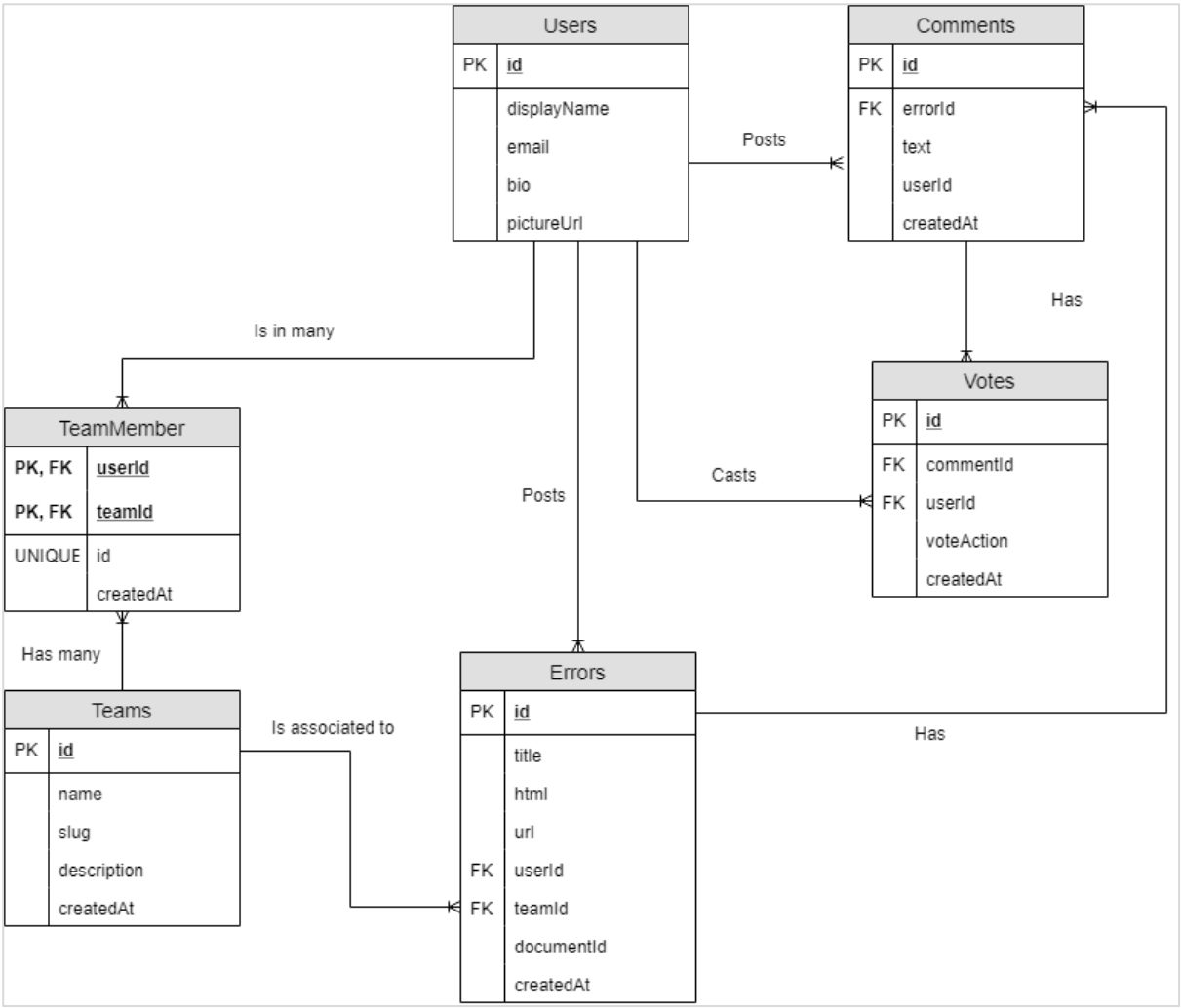
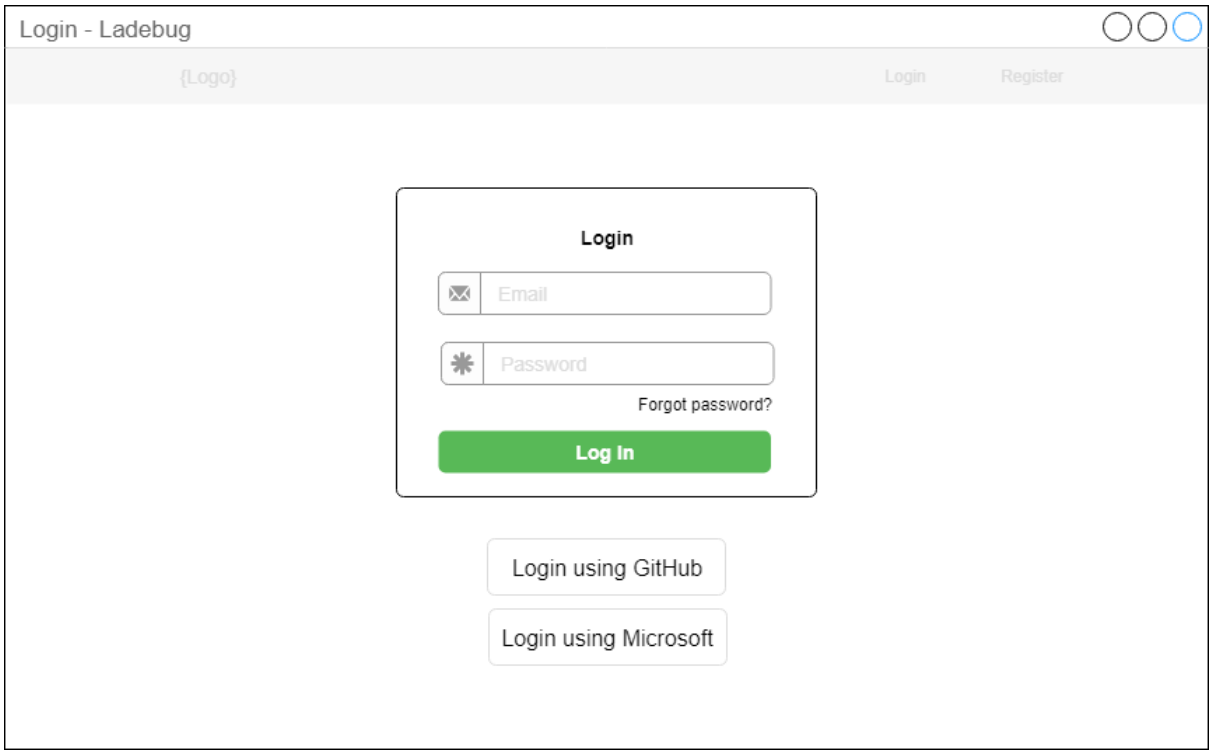
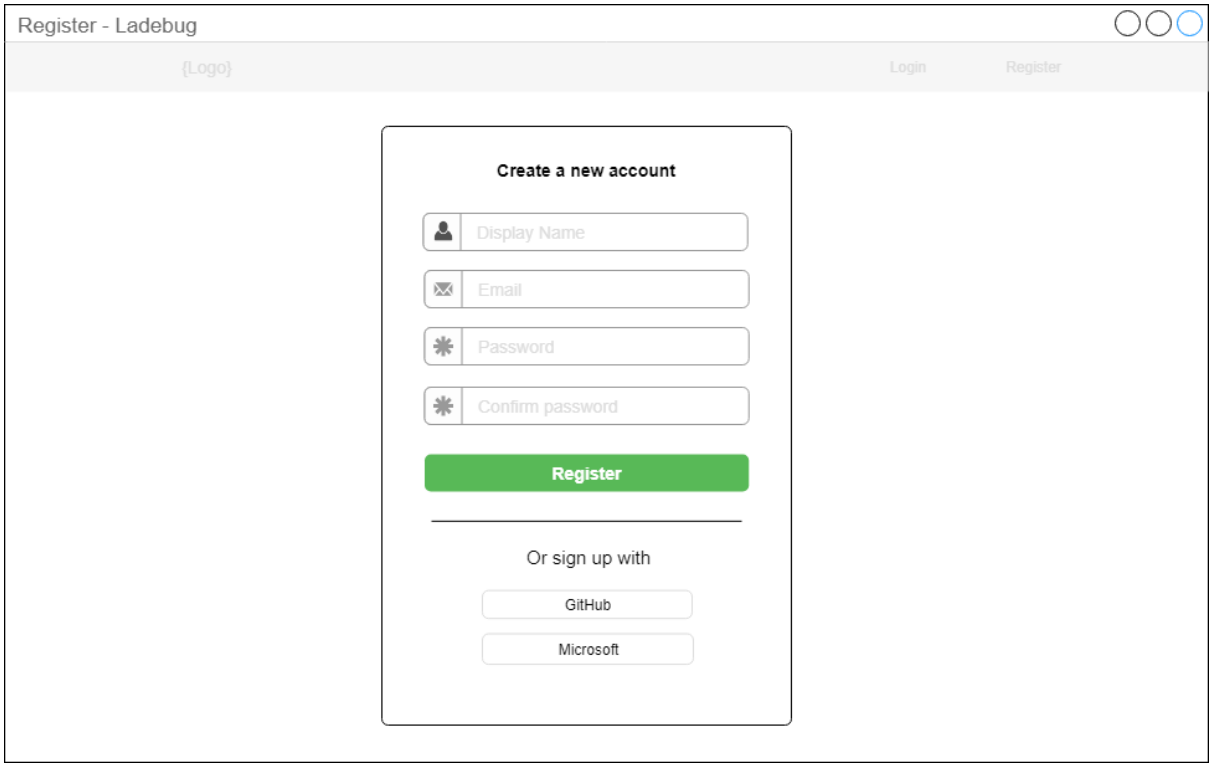


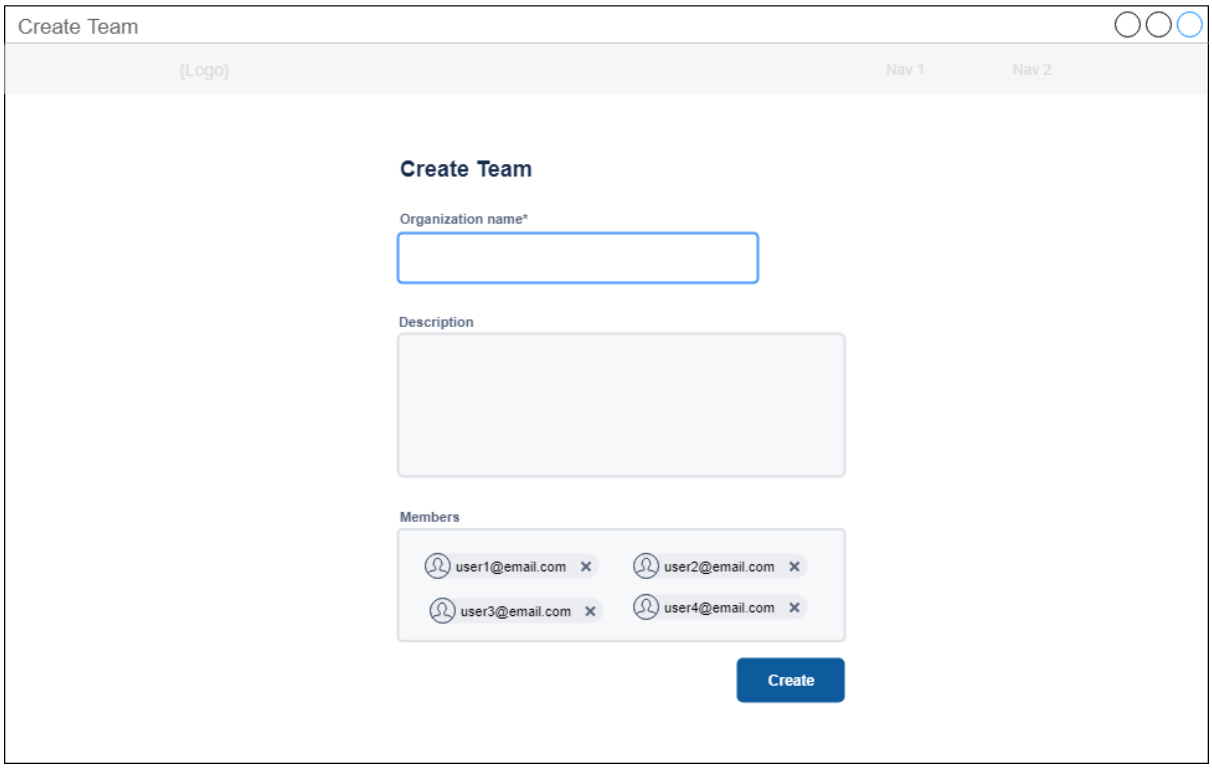
Figure 51 - ERD of the new proposed system

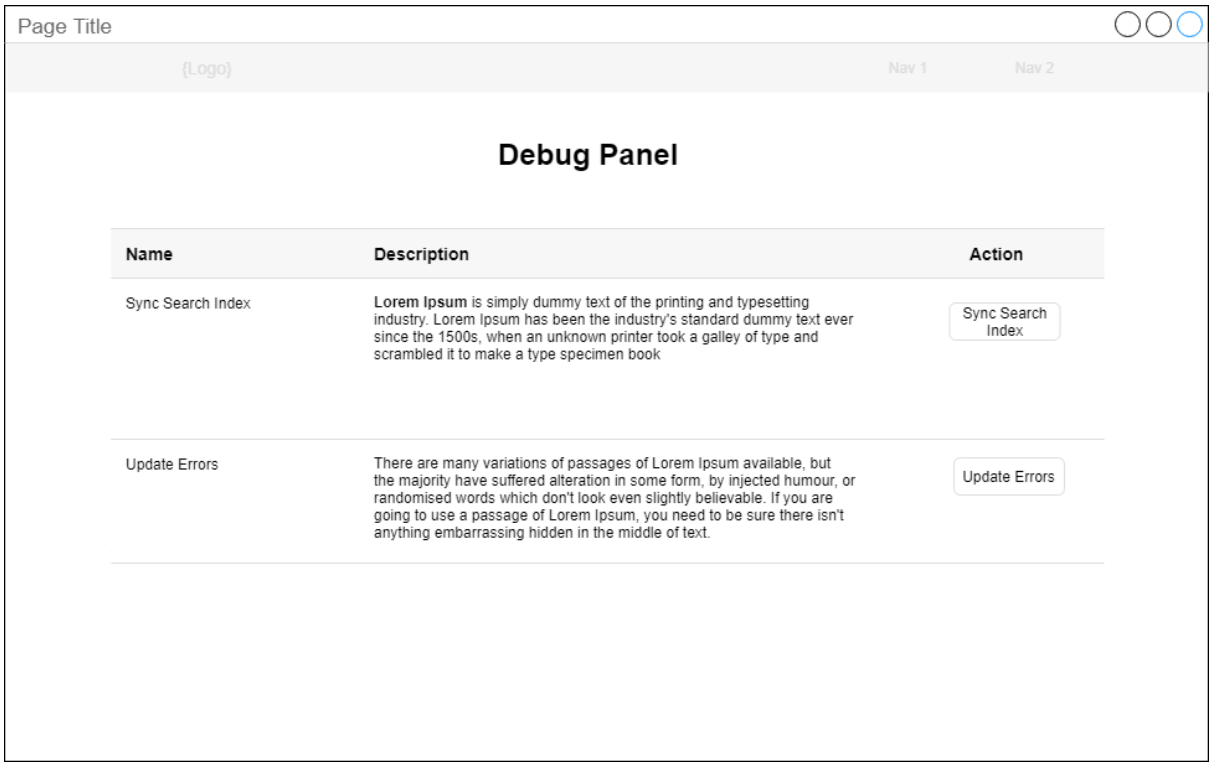
APPENDIX J - LOW FIDELITY DESIGNS OF ARTEFACT

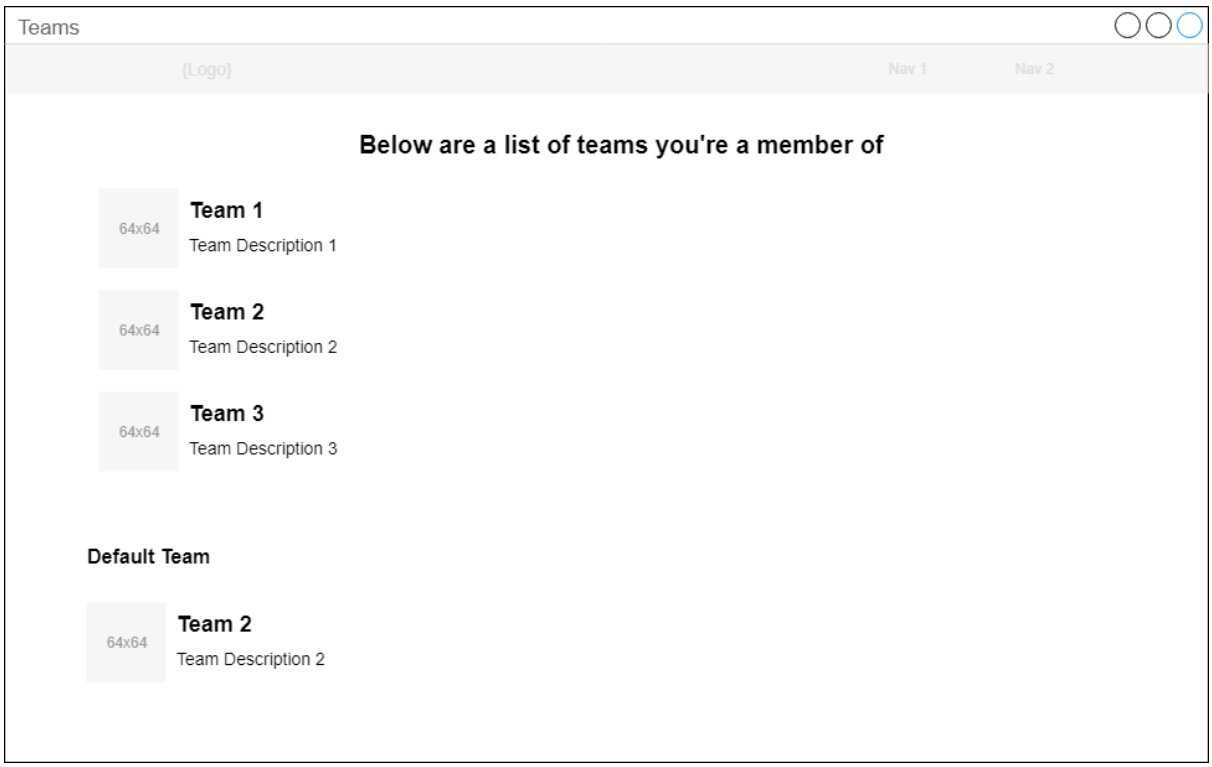
Web App





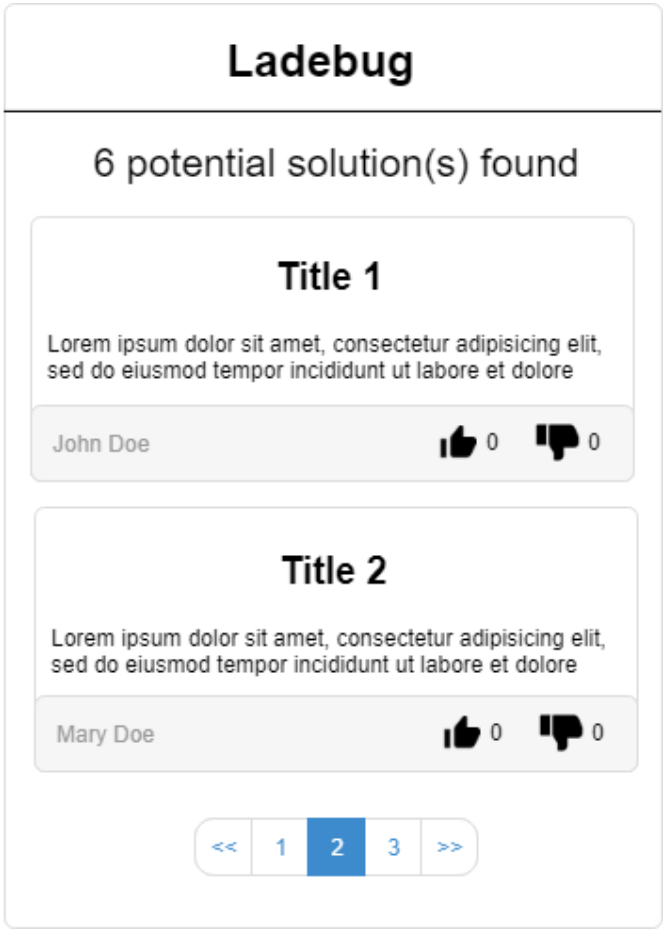

Create Team
{Logo}
Nav 1
Nav 2
Create Team
Organization name*
Description
Members
user1@email.com
user2@email.com
user3@email.com
user4@email.com
Create


Page Title
{Logo}
Nav 1
Nav 2
Debug Panel
Name
Description
Action
Sync Search Index
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book
Sync Search Index
Update Errors
There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text.
Update Errors

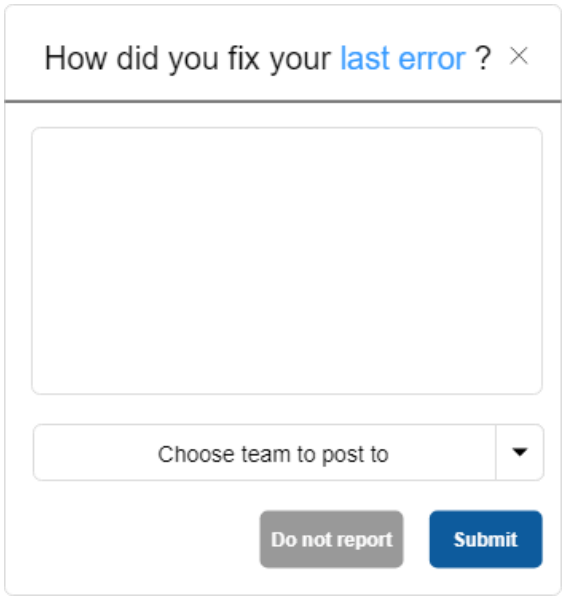


Chrome extension

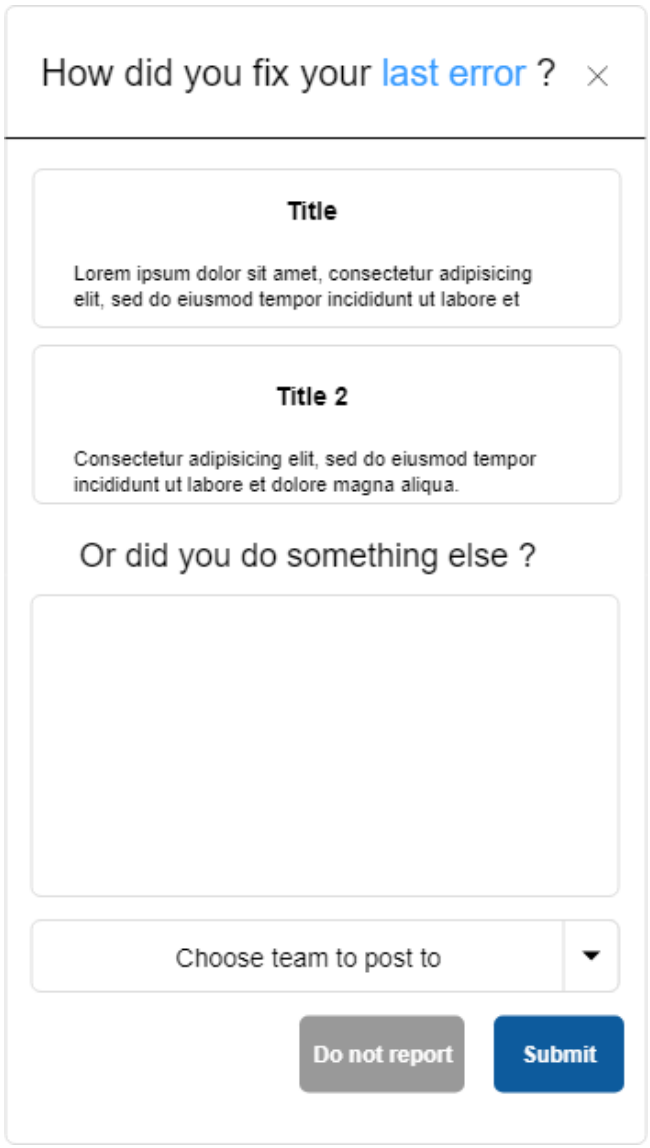
The suggestions dialog



The “how did you fix it?” dialog.

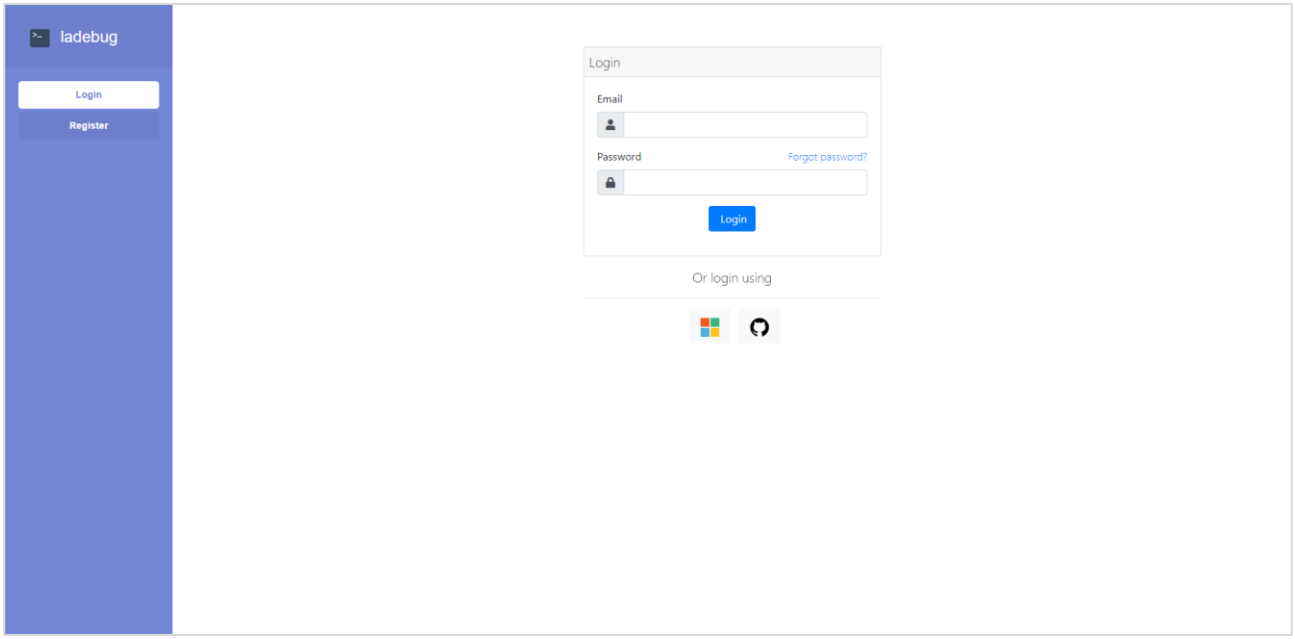


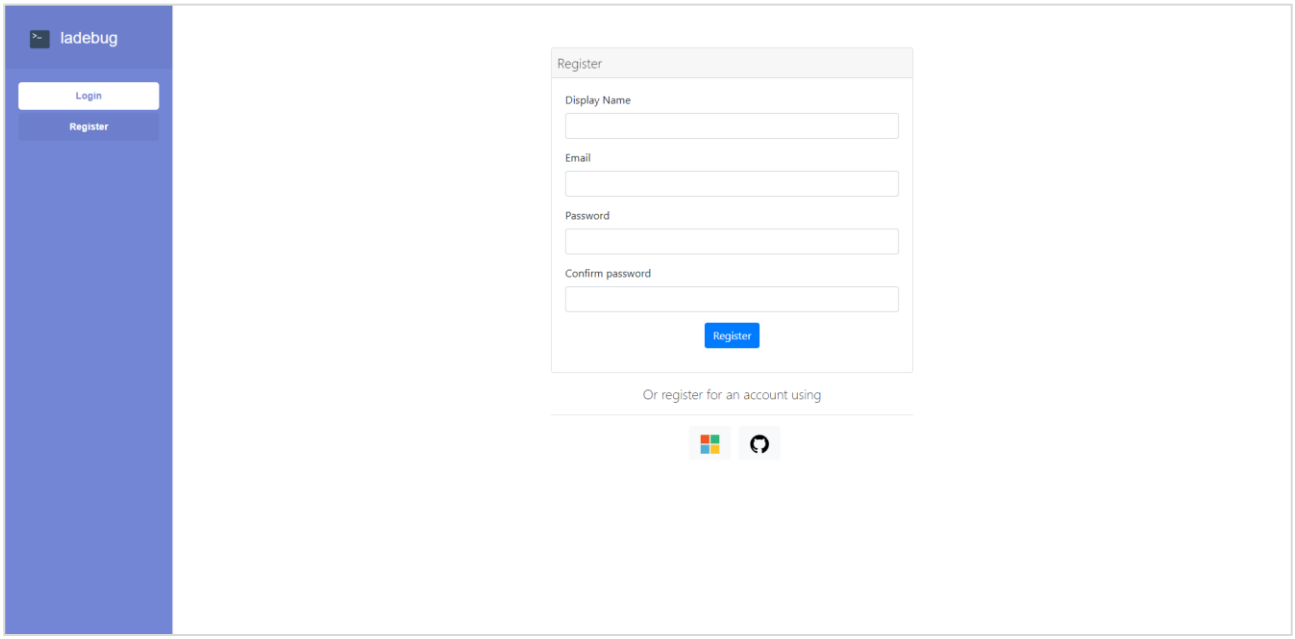
The advanced “how did you fix it?” dialog.

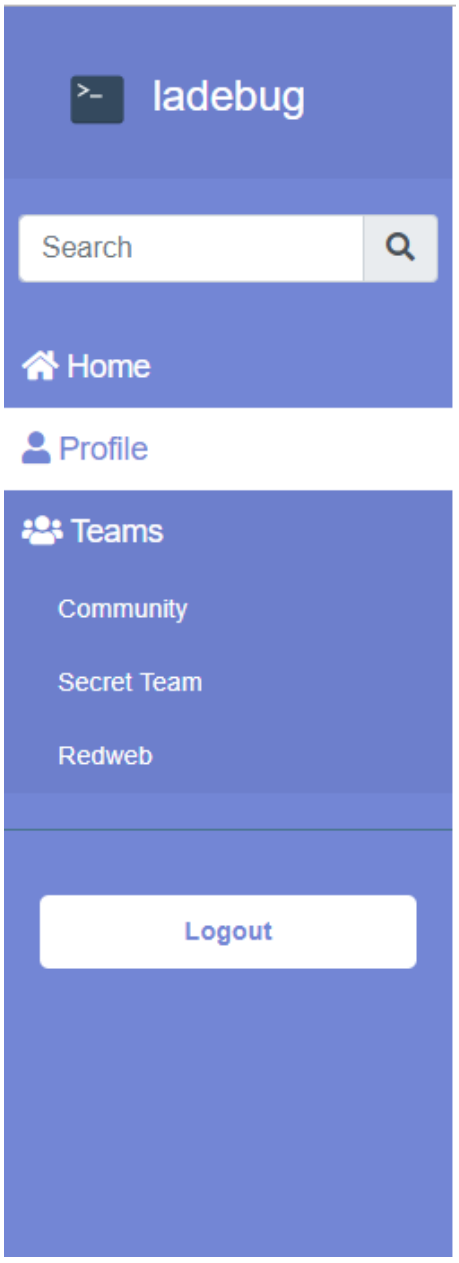


APPENDIX K - ARTEFACT SCREENSHOTS

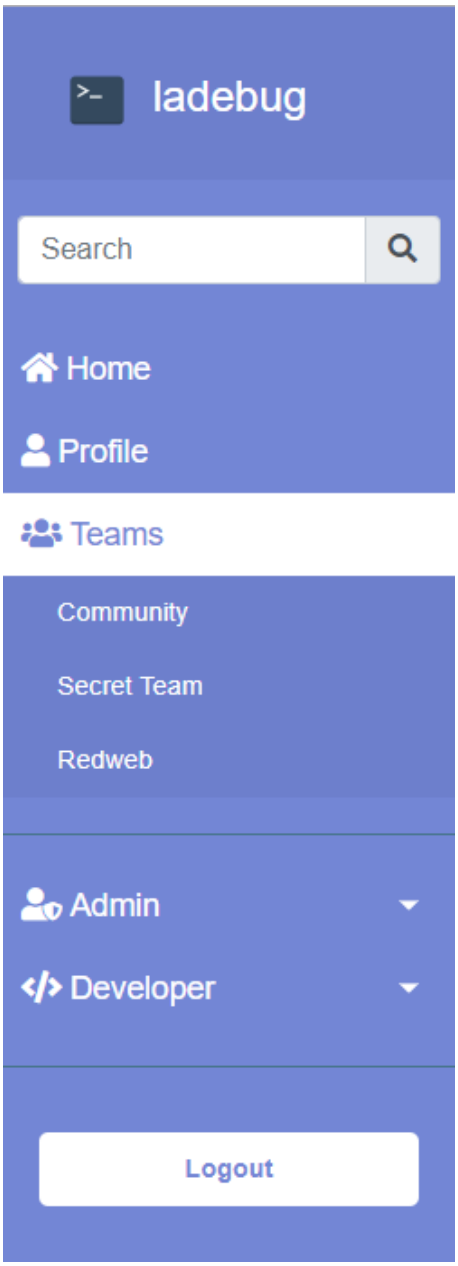
Web App



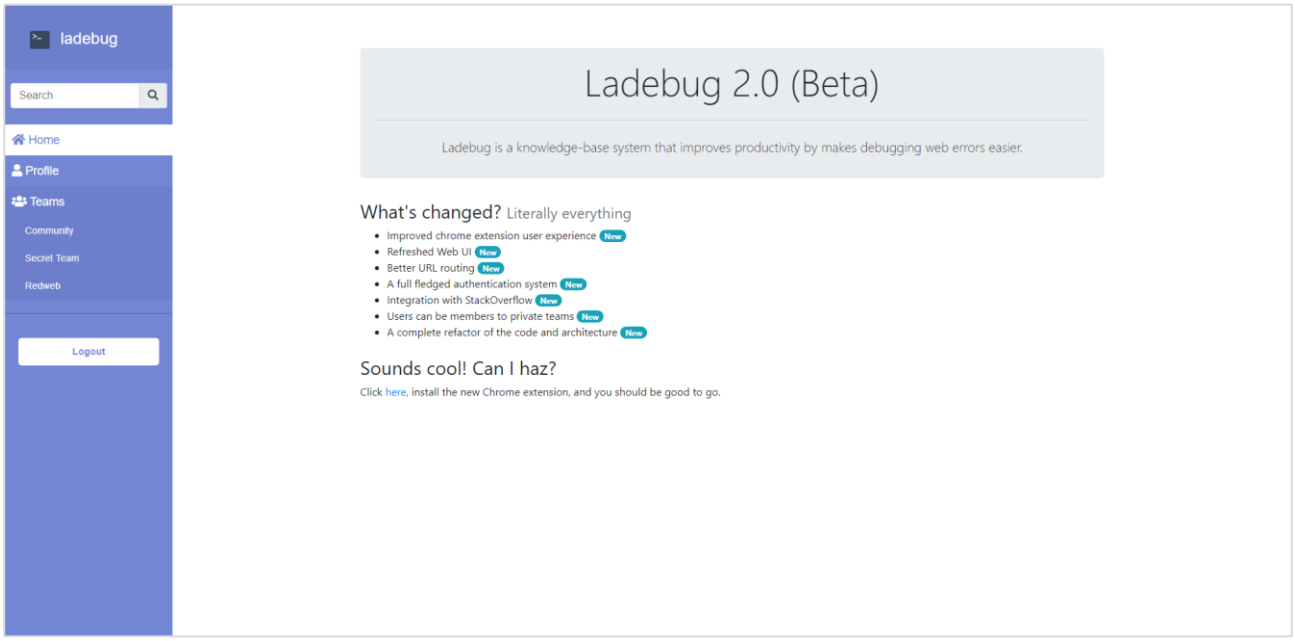


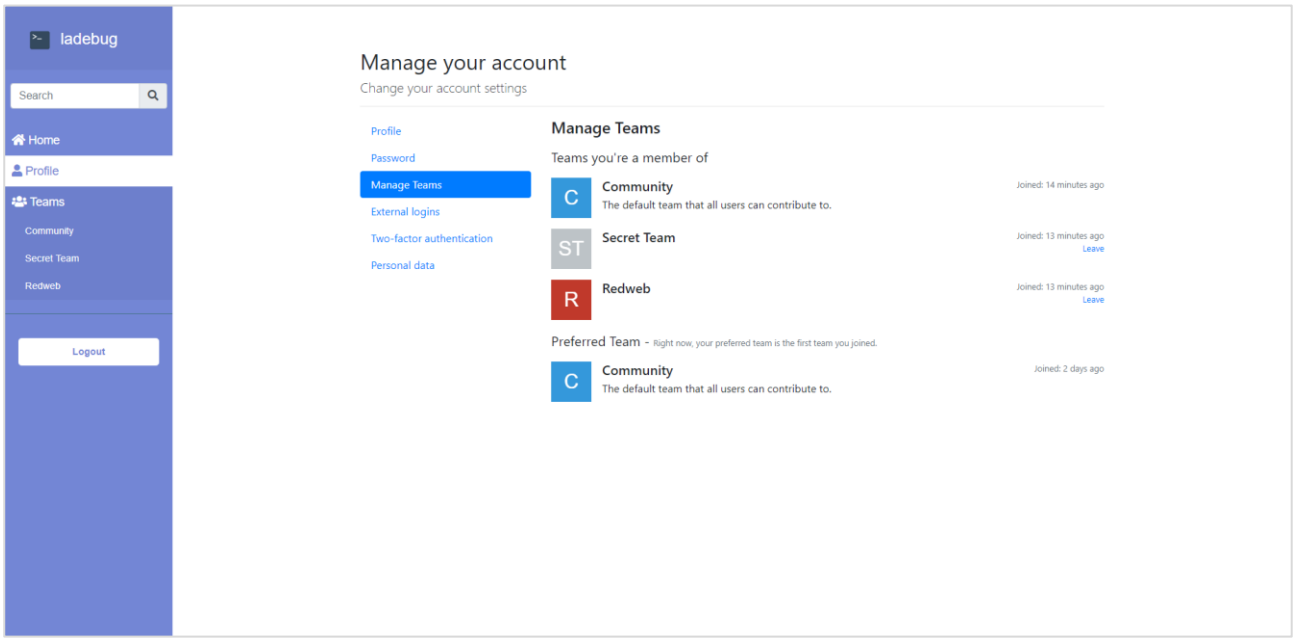


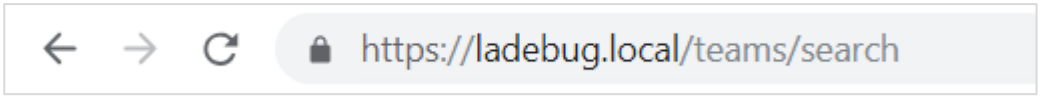
The navbar for normal users

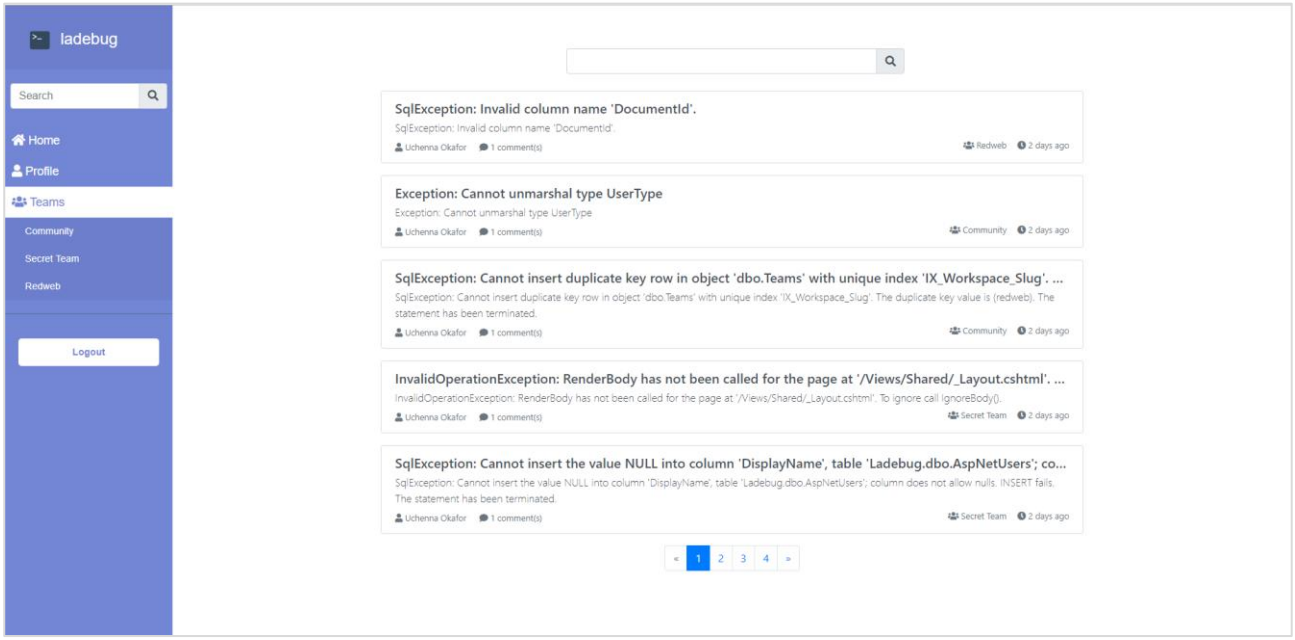


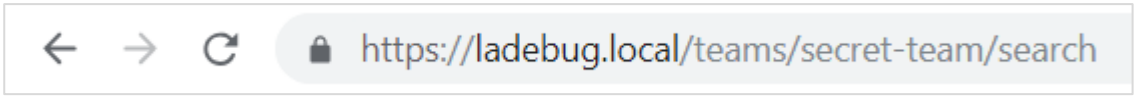
The navbar for admins

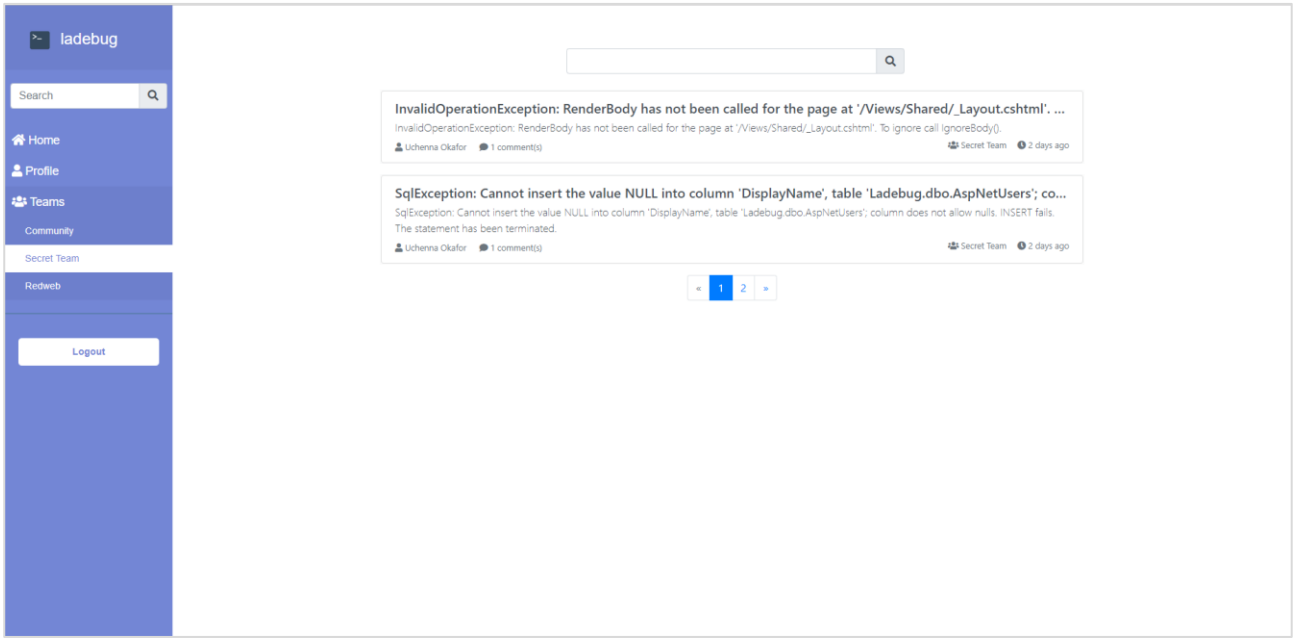

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
Ladebug 2.0 (Beta)
Ladebug is a knowledge-base system that improves productivity by makes debugging web errors easier.
What's changed? Literally everything
Improved chrome extension user experience New
Refreshed Web UI New
Better URL routing New
A full fledged authentication system New
Integration with StackOverflow New
Users can be members to private teams New
A complete refactor of the code and architecture New
Sounds cool! Can I haz?
Click here, install the new Chrome extension, and you should be good to go.

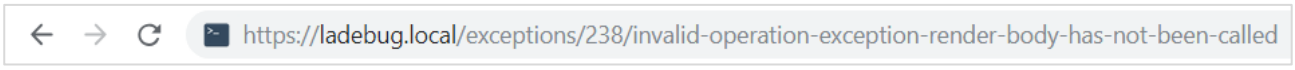

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
Manage your account
Change your account settings
Profile
Password
Manage Teams
External logins
Two-factor authentication
Personal data
Manage Teams
Teams you're a member of
C
Community
The default team that all users can contribute to.
Joined: 14 minutes ago
ST
Secret Team
Joined: 13 minutes ago
Leave
R
Redweb
Joined: 13 minutes ago
Leave
Preferred Team - Right now, your preferred team is the first team you joined.
C
Community
The default team that all users can contribute to.
Joined: 2 days ago

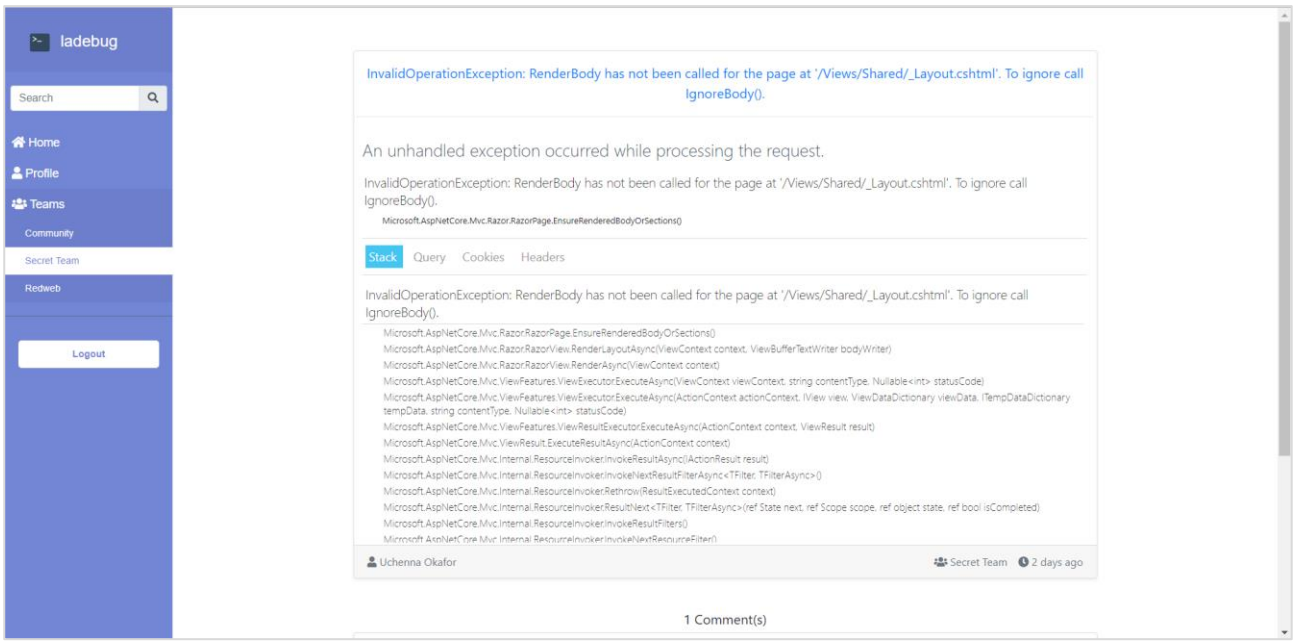

https://ladebug.local/teams/search

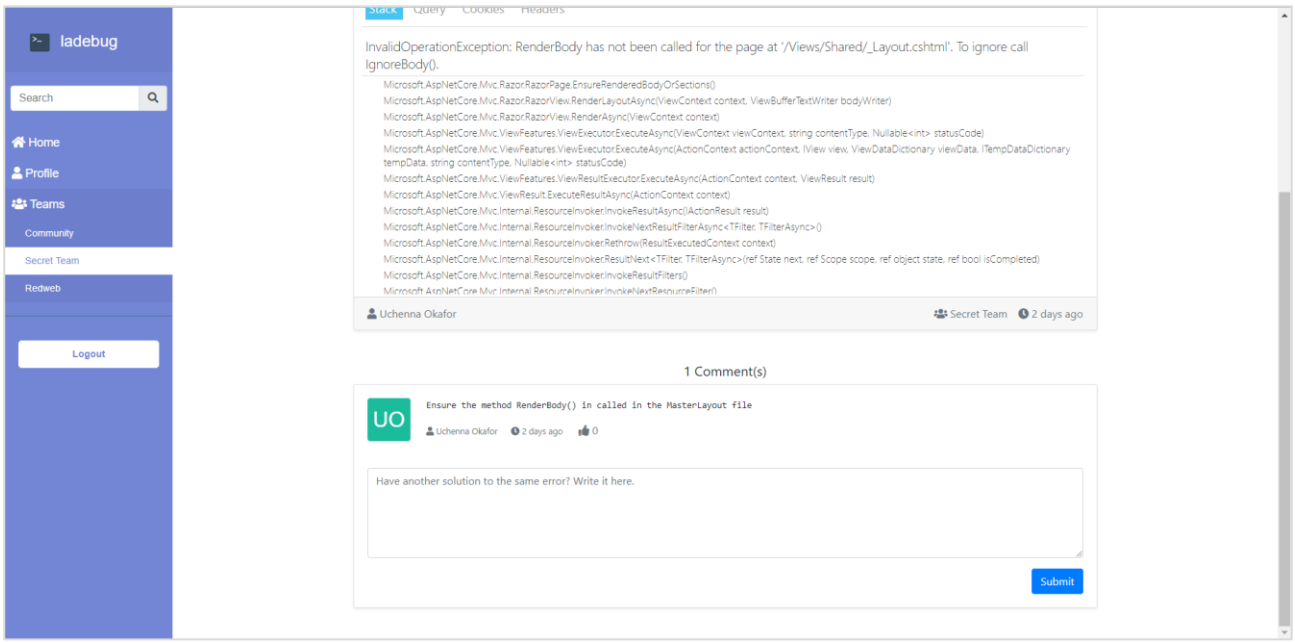

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
SqlException: Invalid column name 'DocumentId'.
SqlException: Invalid column name 'DocumentId'.
Uchenna Okafor
1 comment(s)
Redweb
2 days ago
Exception: Cannot unmarshal type UserType
Exception: Cannot unmarshal type UserType
Uchenna Okafor
1 comment(s)
Community
2 days ago
SqlException: Cannot insert duplicate key row in object 'dbo.Teams' with unique index 'IX_Workspace_Slug'. ...
SqlException: Cannot insert duplicate key row in object 'dbo.Teams' with unique index 'IX_Workspace_Slug'. The duplicate key value is (redweb). The statement has been terminated.
Uchenna Okafor
1 comment(s)
Community
2 days ago
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. ...
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
Uchenna Okafor
1 comment(s)
Secret Team
2 days ago
SqlException: Cannot insert the value NULL into column 'DisplayName', table 'Ladebug.dbo.AspNetUsers'; co...
SqlException: Cannot insert the value NULL into column 'DisplayName', table 'Ladebug.dbo.AspNetUsers'; column does not allow nulls. INSERT fails.
The statement has been terminated.
Uchenna Okafor
1 comment(s)
Secret Team
2 days ago
« 1 2 3 4 »

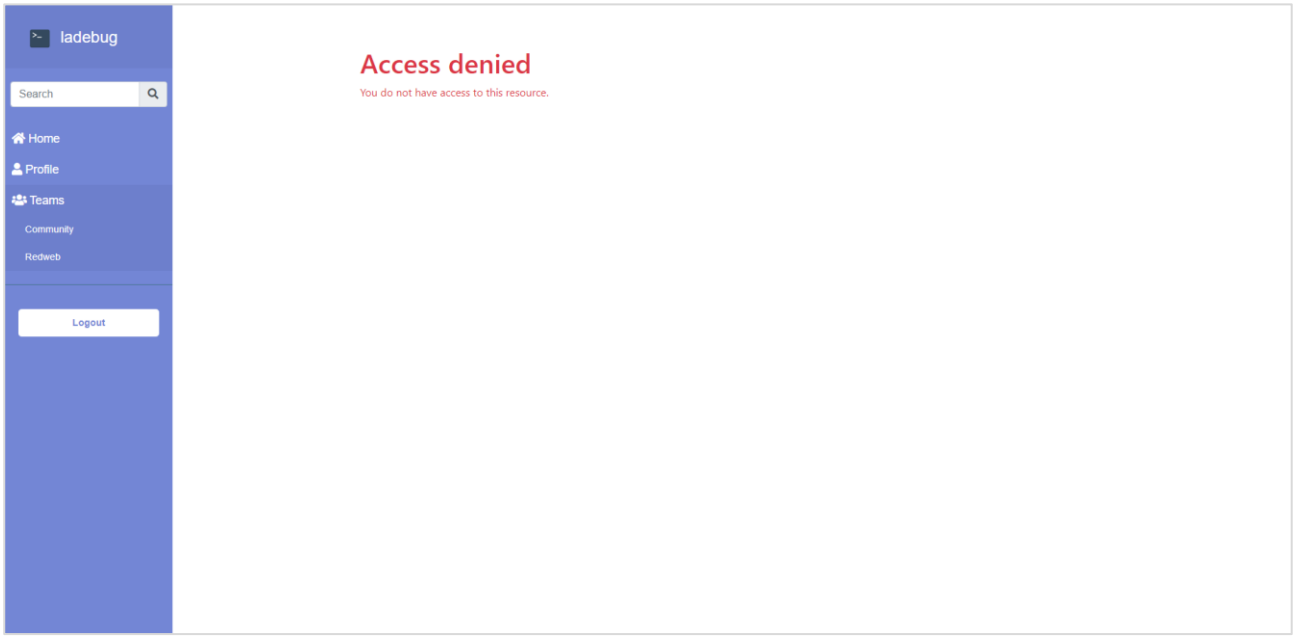

https://ladebug.local/teams/secret-team/search

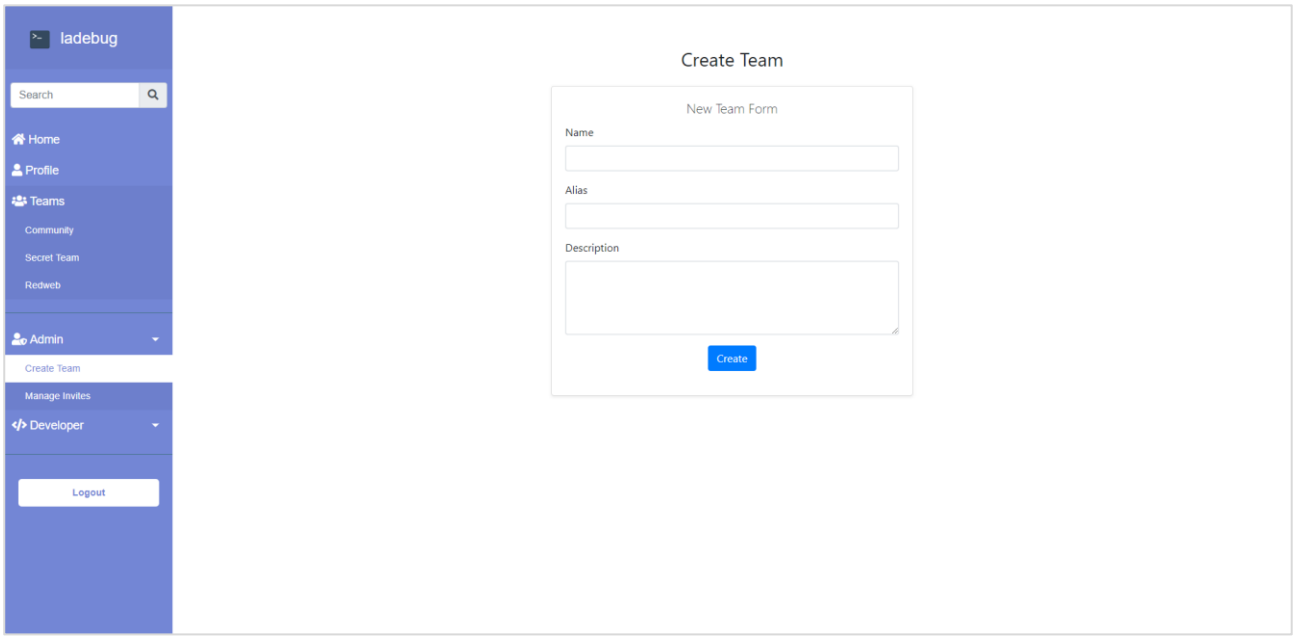

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. ...
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
Uchenna Okafor
1 comment(s)
Secret Team
2 days ago
SqlException: Cannot insert the value NULL into column 'DisplayName', table 'Ladebug.dbo.AspNetUsers'; co...
SqlException: Cannot insert the value NULL into column 'DisplayName', table 'Ladebug.dbo.AspNetUsers'; column does not allow nulls. INSERT fails.
The statement has been terminated.
Uchenna Okafor
1 comment(s)
Secret Team
2 days ago
« 1 2 »

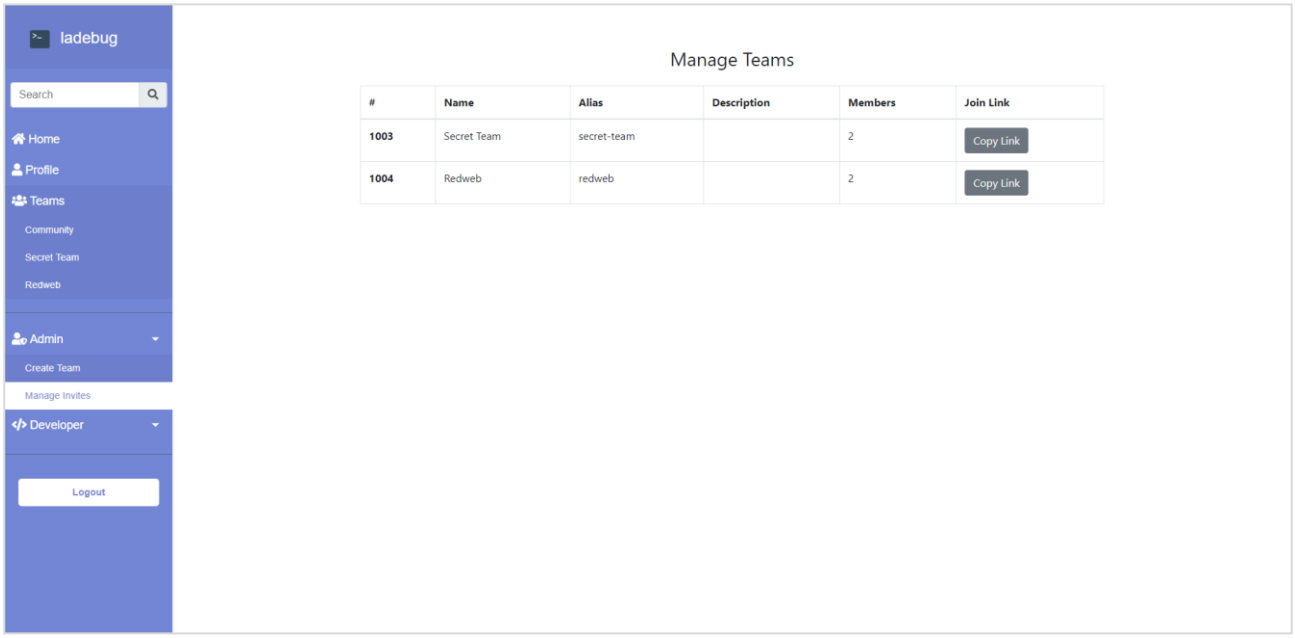

https://ladebug.local/exceptions/238/invalid-operation-exception-render-body-has-not-been-called

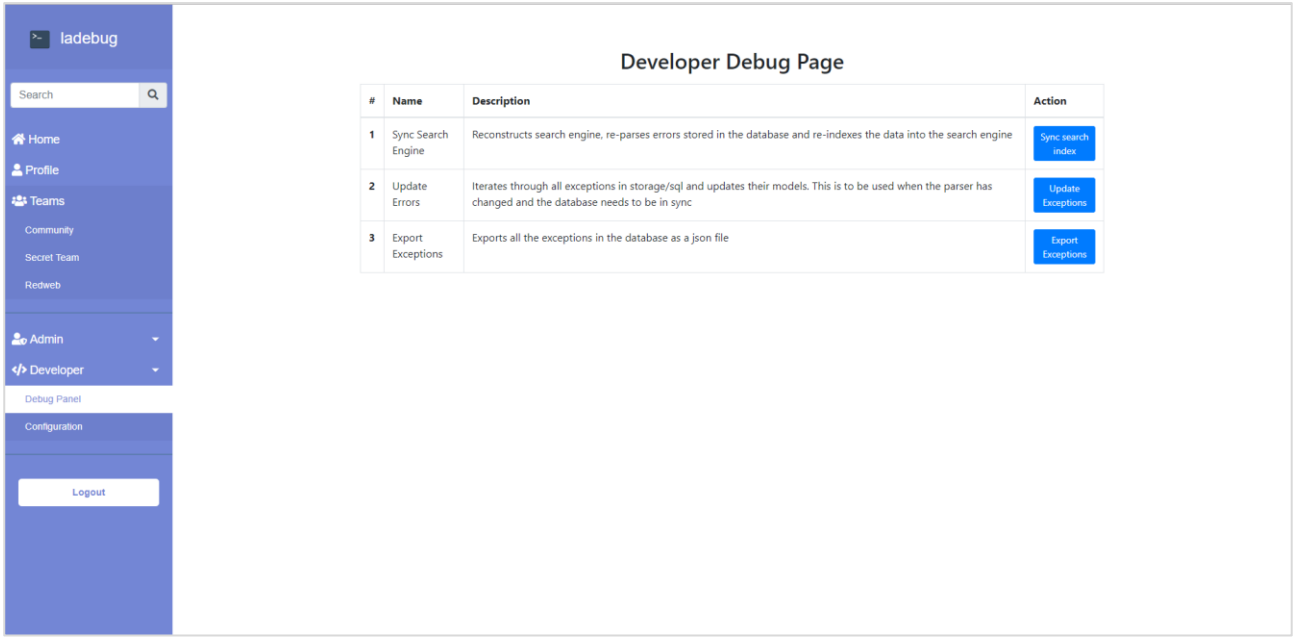

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
An unhandled exception occurred while processing the request.
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
Microsoft.AspNetCore.Mvc.Razor.RazorPage.EnsureRenderedBodyOrSections()
Stack
Query
Cookies
Headers
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
Microsoft.AspNetCore.Mvc.Razor.RazorPage.EnsureRenderedBodyOrSections()
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderLayoutAsync(ViewContext context, ViewBufferTextWriter bodyWriter)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderAsync(ViewContext context)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ActionContext actionContext, IView view, ViewDataDictionary viewData, ITempDataDictionary tempData, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewResultExecutor.ExecuteAsync(ActionContext context, ViewResult result)
Microsoft.AspNetCore.Mvc.ViewResult.ExecuteResultAsync(ActionContext context)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeResultAsync(IActionResult result)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeNextResultFilterAsync<TFilter, TFilterAsync>()
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.Rethrow(ResultExecutedContext context)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.ResultNext<TFilter, TFilterAsync>(ref State next, ref Scope scope, ref object state, ref bool isCompleted)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeResultFilters()
Uchenna Okafor
Secret Team
2 days ago
1 Comment(s)

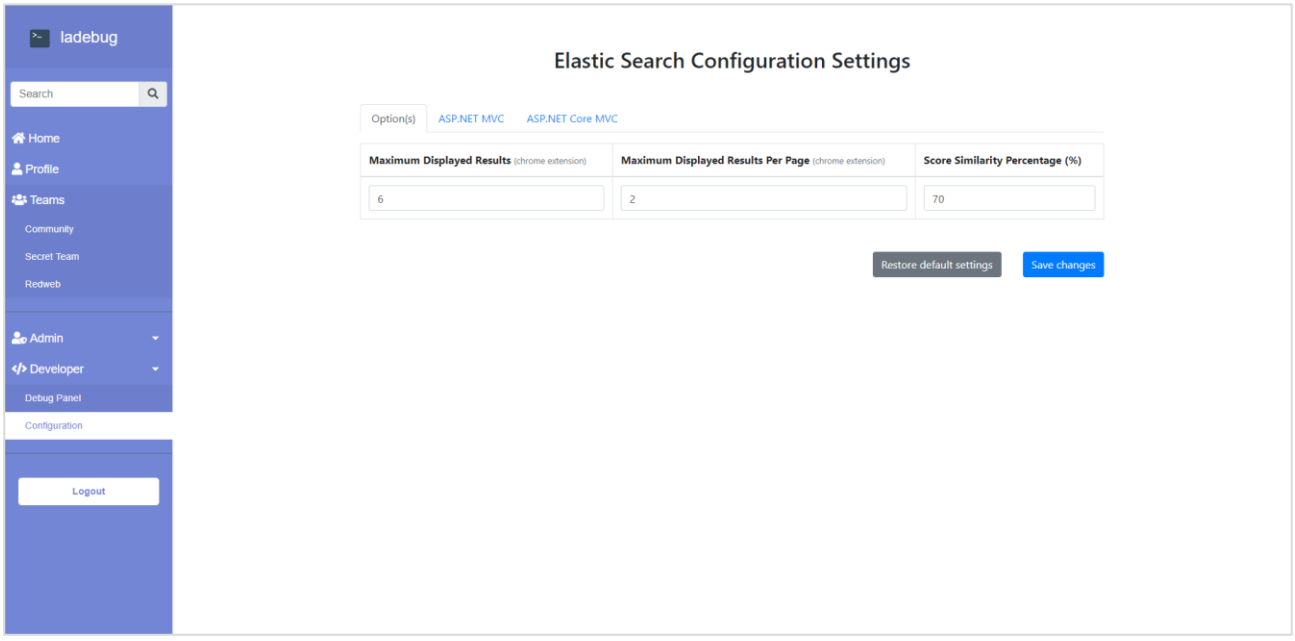

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Logout
InvalidOperationException: RenderBody has not been called for the page at '/Views/Shared/_Layout.cshtml'. To ignore call IgnoreBody().
Microsoft.AspNetCore.Mvc.Razor.RazorPage.EnsureRenderedBodyOrSections()
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderLayoutAsync(ViewContext context, ViewBufferTextWriter bodyWriter)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderAsync(ViewContext context)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ActionContext actionContext, IView view, ViewDataDictionary viewData, ITempDataDictionary tempData, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewResultExecutor.ExecuteAsync(ActionContext context, ViewResult result)
Microsoft.AspNetCore.Mvc.ViewResult.ExecuteResultAsync(ActionContext context)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeResultAsync(IActionResult result)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeNextResultFilterAsync<TFilter, TFilterAsync>()
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.Rethrow(ResultExecutedContext context)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.ResultNext<TFilter, TFilterAsync>(ref State next, ref Scope scope, ref object state, ref bool isCompleted)
Microsoft.AspNetCore.Mvc.Internal.ResourceInvoker.InvokeResultFilters()
Uchenna Okafor
Secret Team
2 days ago
1 Comment(s)
UO
Ensure the method RenderBody() in called in the MasterLayout file
Uchenna Okafor
2 days ago
0
Have another solution to the same error? Write it here.
Submit

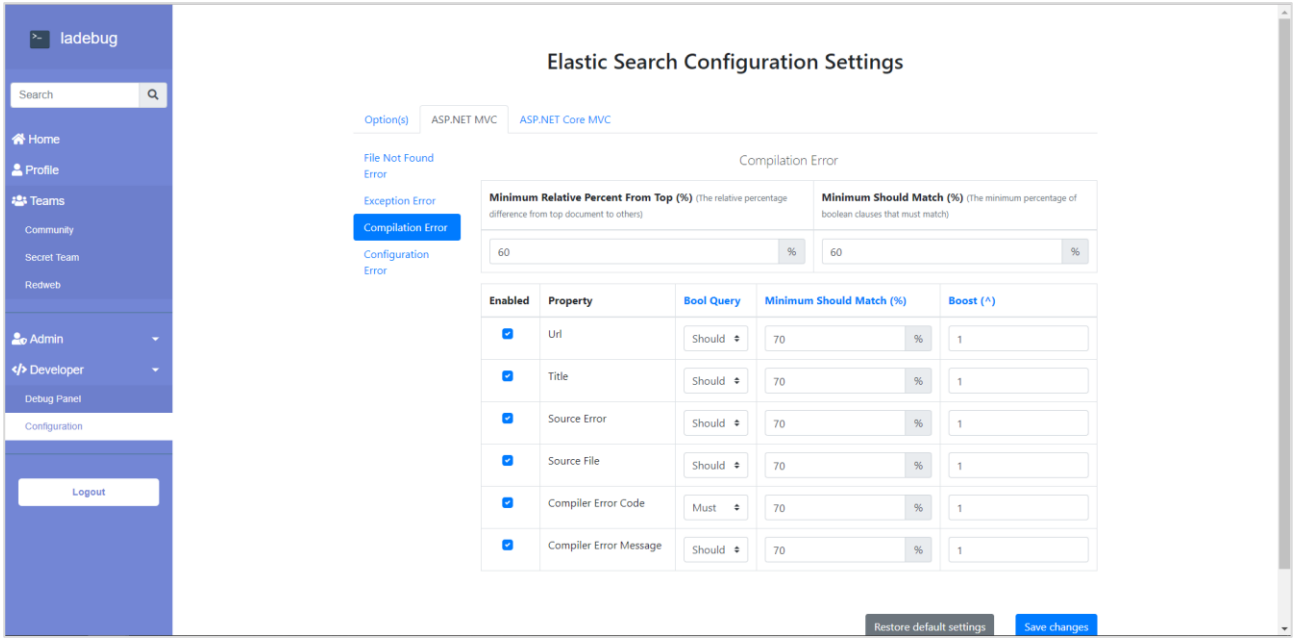

ladebug
Search
Home
Profile
Teams
Community
Redweb
Logout
Access denied
You do not have access to this resource.


ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Admin
Create Team
Manage Invites
Developer
Logout
Create Team
New Team Form
Name
Alias
Description
Create

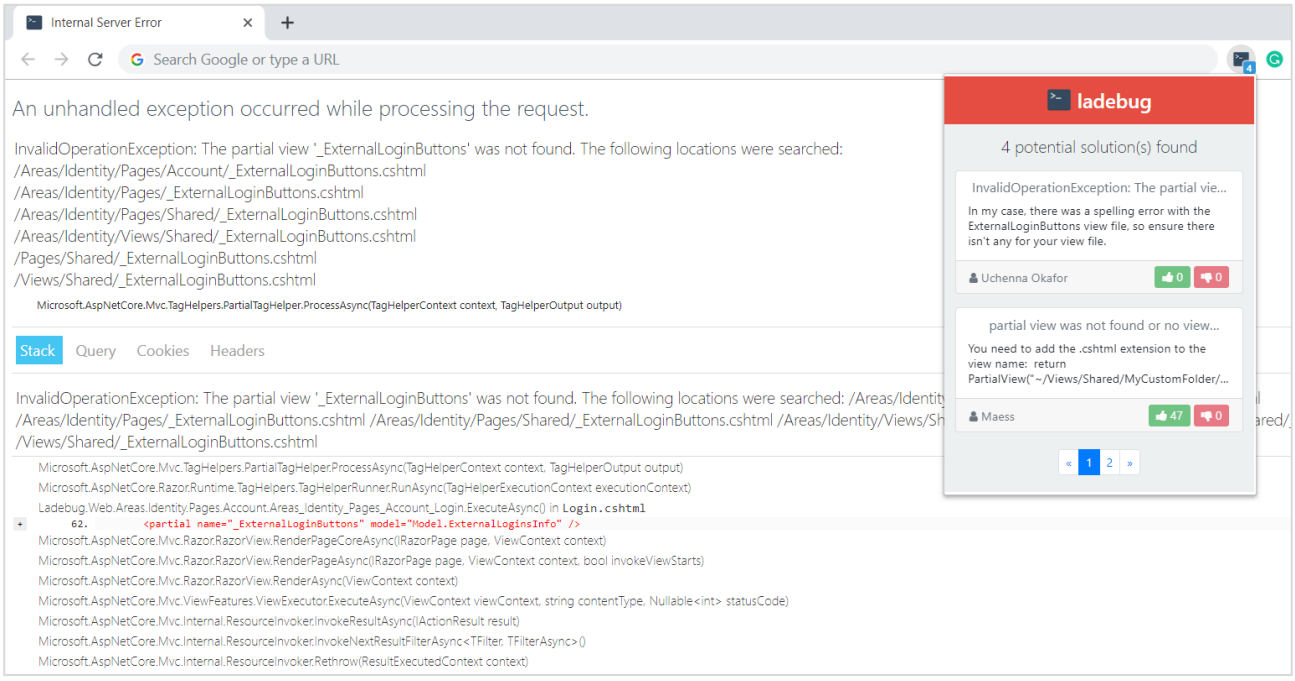

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Admin
Create Team
Manage Invites
Developer
Logout
Manage Teams
Name Alias Description Members Join Link
1003 Secret Team secret-team 2 Copy Link
1004 Redweb redweb 2 Copy Link

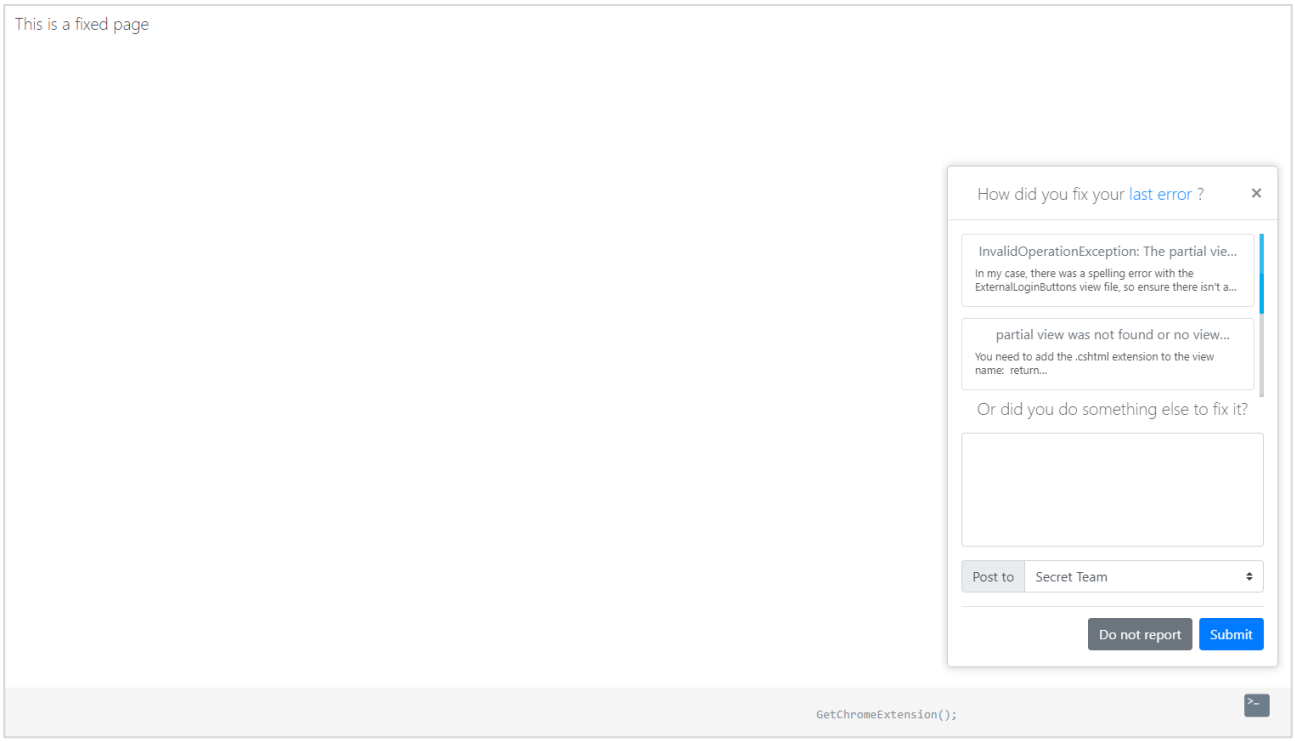

ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Admin
Developer
Debug Panel
Configuration
Logout
Developer Debug Page
Name Description Action
1 Sync Search Engine Reconstructs search engine, re-parses errors stored in the database and re-indexes the data into the search engine Sync search index
2 Update Errors Iterates through all exceptions in storage/sql and updates their models. This is to be used when the parser has changed and the database needs to be in sync Update Exceptions
3 Export Exceptions Exports all the exceptions in the database as a json file Export Exceptions


ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Admin
Developer
Debug Panel
Configuration
Logout
Elastic Search Configuration Settings
Option(s)
ASP.NET MVC
ASP.NET Core MVC
Maximum Displayed Results (chrome extension)
Maximum Displayed Results Per Page (chrome extension)
Score Similarity Percentage (%)
6
2
70
Restore default settings
Save changes


ladebug
Search
Home
Profile
Teams
Community
Secret Team
Redweb
Admin
Developer
Debug Panel
Configuration
Logout
Elastic Search Configuration Settings
Option(s)
ASP.NET MVC
ASP.NET Core MVC
File Not Found Error
Exception Error
Compilation Error
Configuration Error
Compilation Error
Minimum Relative Percent From Top (%) (The relative percentage difference from top document to others)
Minimum Should Match (%) (The minimum percentage of boolean clauses that must match)
60
60
Enabled
Property
Bool Query
Minimum Should Match (%)
Boost (^)
Url
Title
Source Error
Source File
Compiler Error Code
Compiler Error Message
Should
Must
70
1
Restore default settings
Save changes

Chrome Extension





APPENDIX L - OVERVIEW DIAGRAMS OF NEW SYSTEM

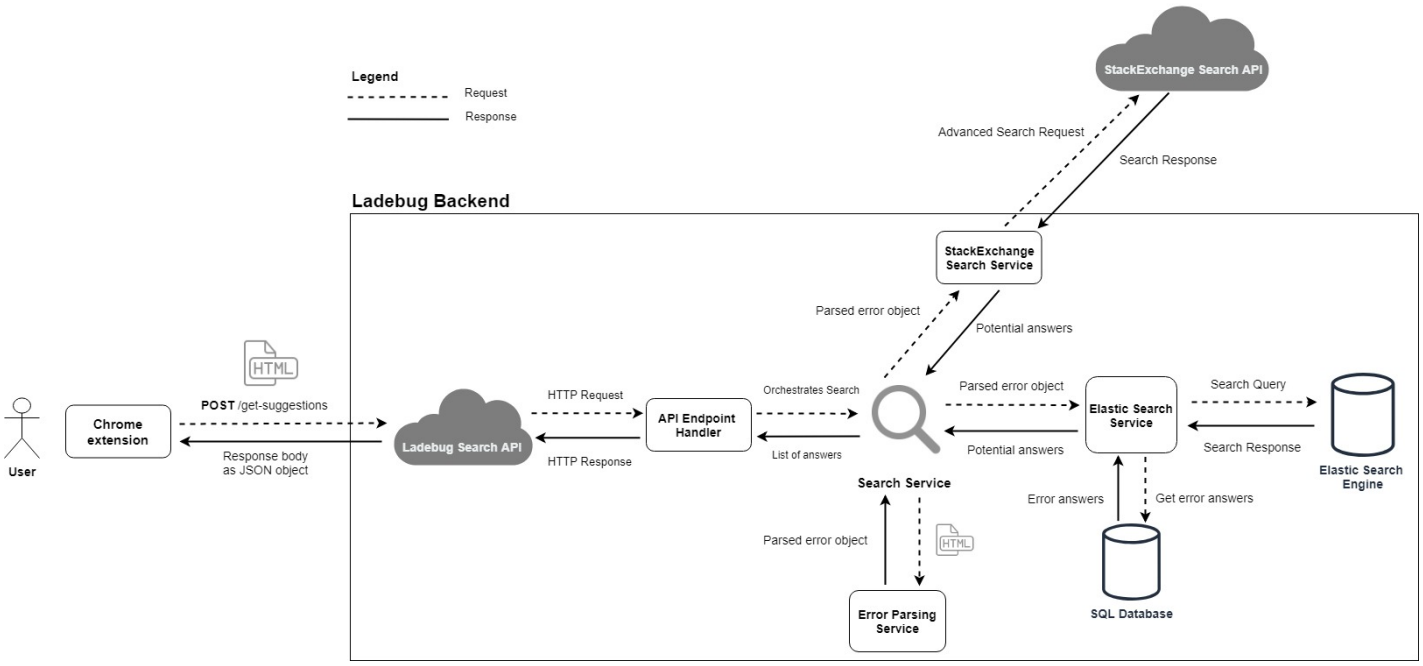


Figure 52 - An overview diagram of how the new system works

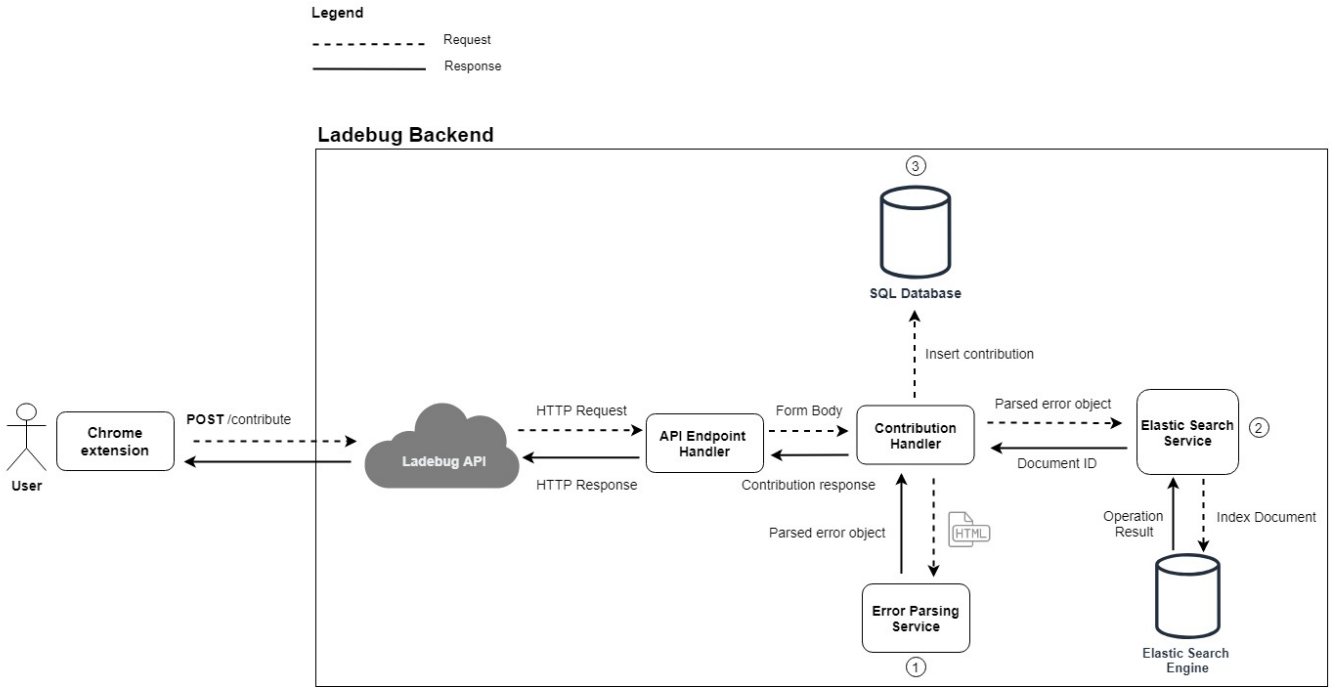
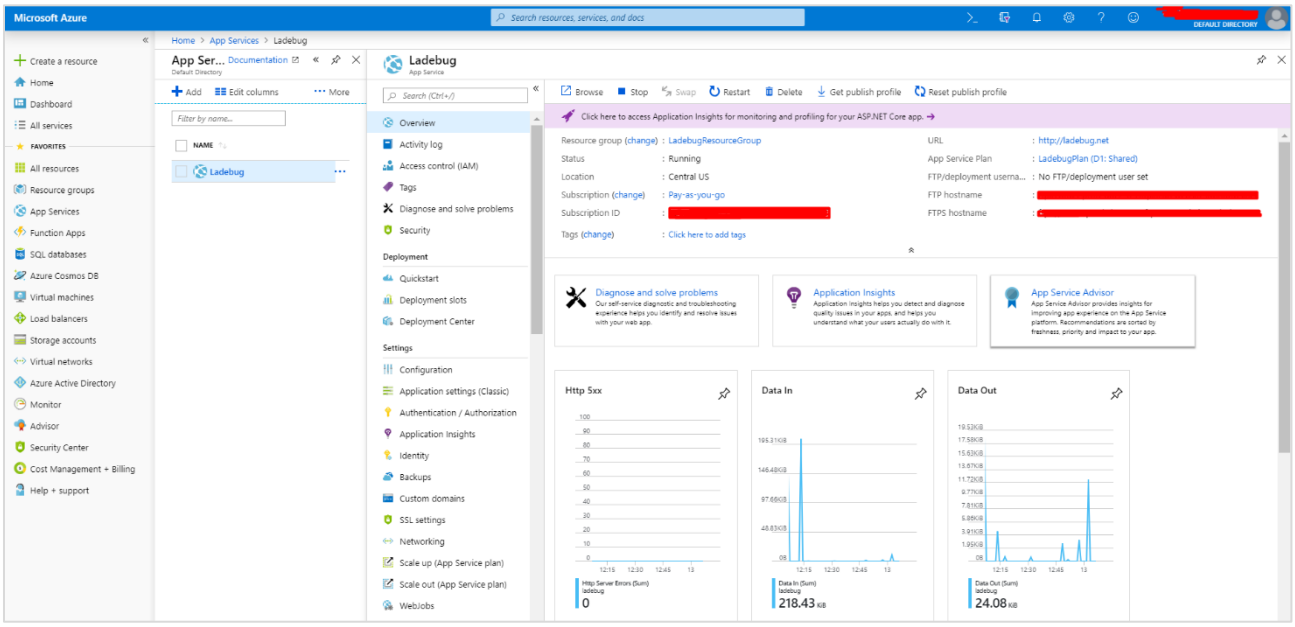
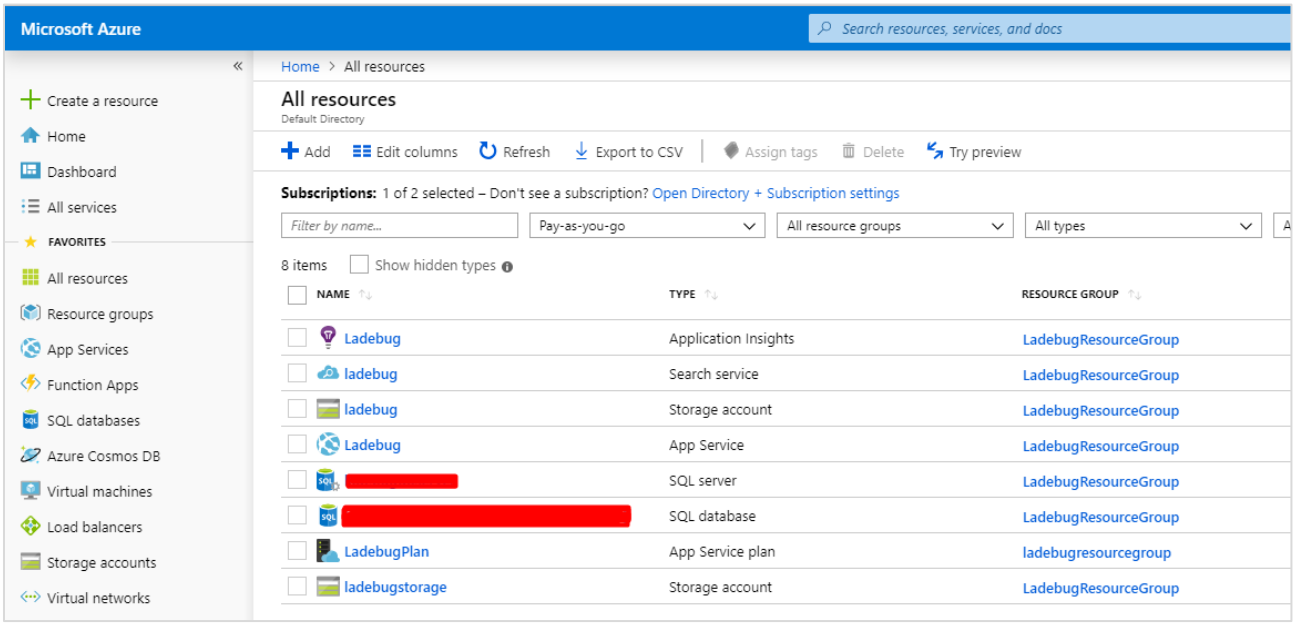


Figure 53 - An overview diagram of how contributions are created into the new system

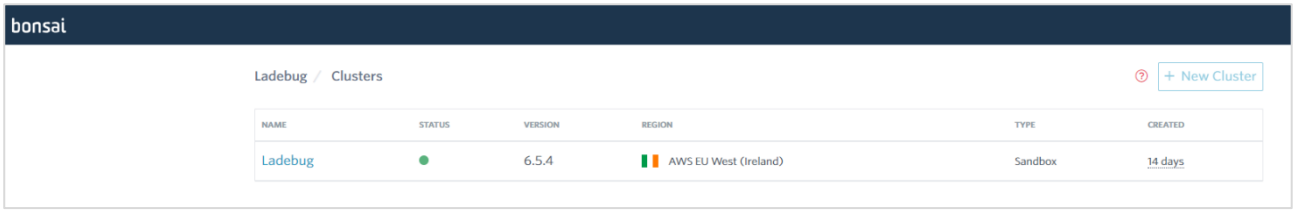
APPENDIX M - EVIDENCE OF ARTEFACT'S CLOUD DEPLOYMENT

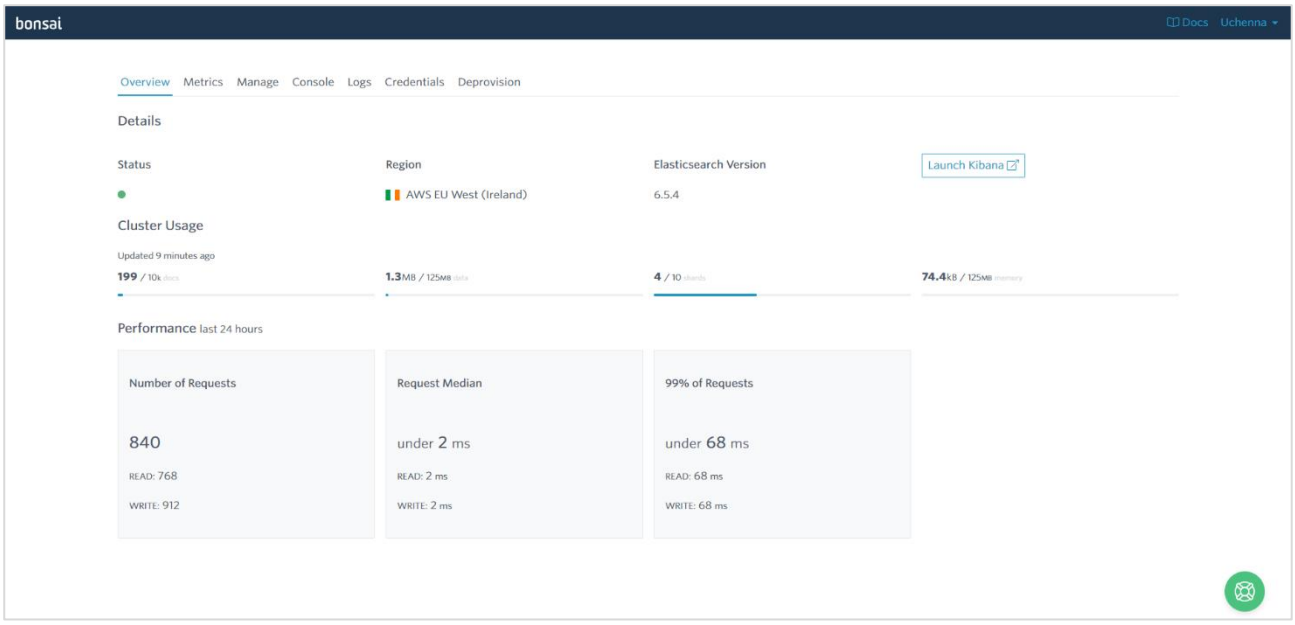
Azure

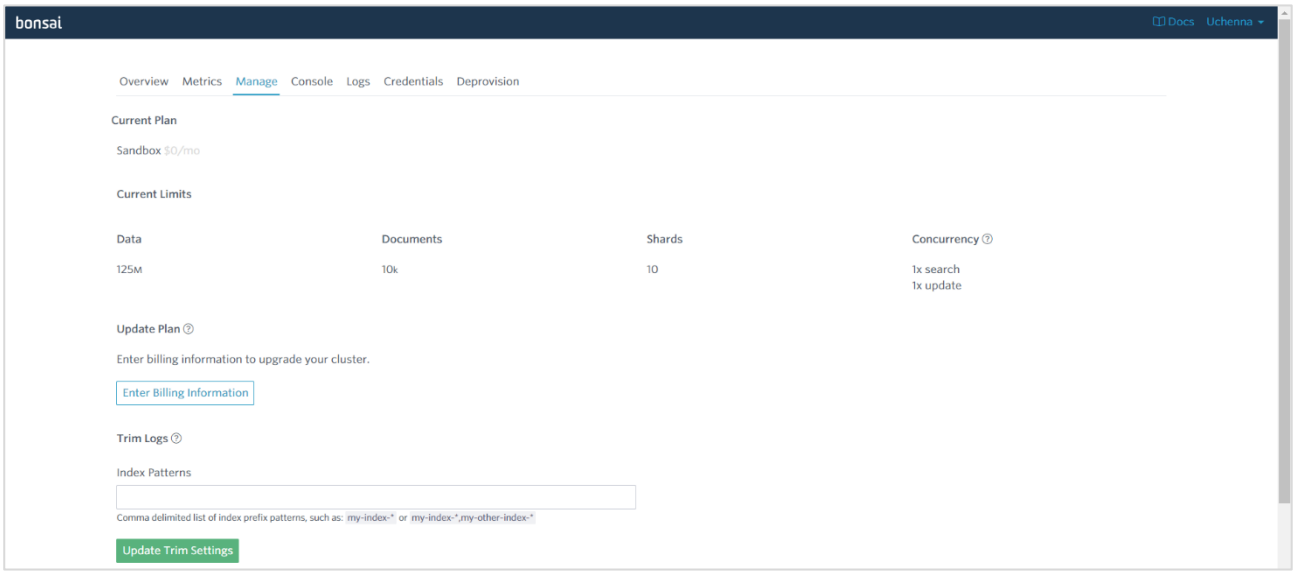




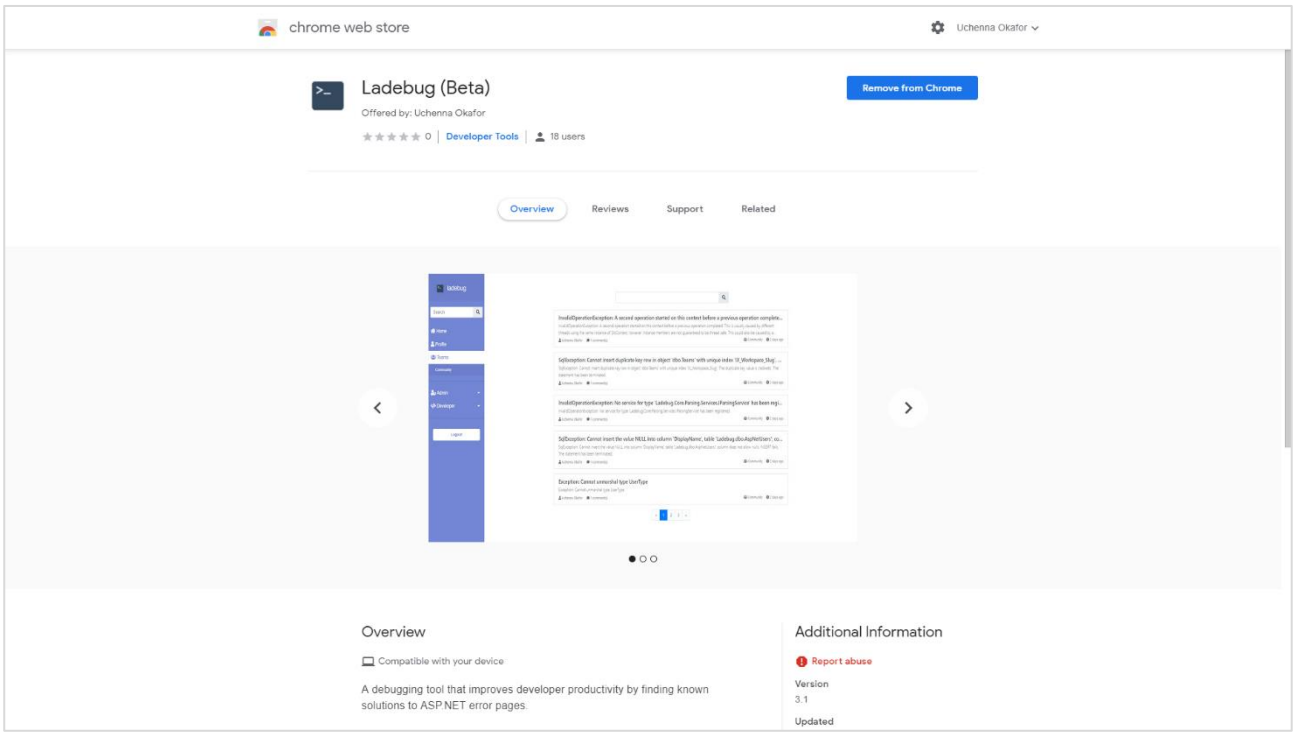
Bonsai

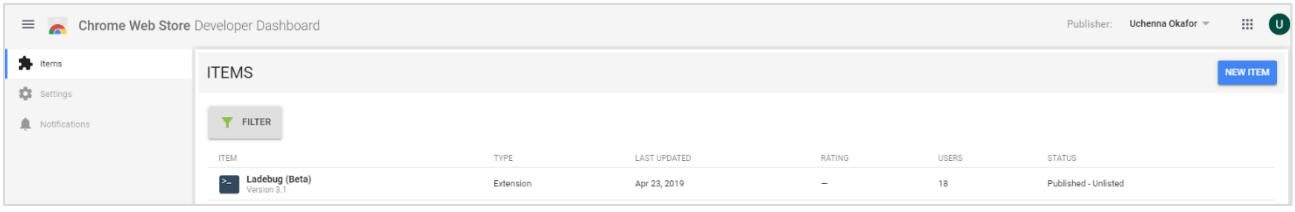

bonsai
Ladebug / Clusters
+ New Cluster
NAME
STATUS
VERSION
REGION
TYPE
CREATED
Ladebug
6.5.4
AWS EU West (Ireland)
Sandbox
14 days


bonsai
Docs
Uchenna
Overview
Metrics
Manage
Console
Logs
Credentials
Deprovision
Details
Status
Region
Elasticsearch Version
Launch Kibana
AWS EU West (Ireland)
6.5.4
Cluster Usage
Updated 9 minutes ago
199 / 10k
1.3MB / 125MB
4 / 10
74.4kB / 125MB
Performance last 24 hours
Number of Requests
840
READ: 768
WRITE: 912
Request Median
under 2 ms
READ: 2 ms
WRITE: 2 ms
99% of Requests
under 68 ms
READ: 68 ms
WRITE: 68 ms


bonsai
Docs
Uchenna
Overview
Metrics
Manage
Console
Logs
Credentials
Deprovision
Current Plan
Sandbox $0/mo
Current Limits
Data
Documents
Shards
Concurrency
125M
10k
10
1x search
1x update
Update Plan
Enter billing information to upgrade your cluster.
Enter Billing Information
Trim Logs
Index Patterns
Comma delimited list of index prefix patterns, such as: my-index-*, or my-index-*,my-other-index-*
Update Trim Settings

Chrome Store





APPENDIX N - FULL LIST OF LIBRARIES USED

Backend

Library Name	Purpose
AngleSharp	For parsing the HTML document of error pages
Humanizer	For printing objects like dates in a human readable manner
AspNet.Security.OAuth.GitHub	An extension for configuring GitHub as an OAuth provider
Microsoft.Azure.Search	Library for interacting with Azure Search
NEST	Library for interacting with Elastic Search
Newtonsoft.Json	For serialization and deserialization of JSON data
NUnit	Unit tests
Sendgrid	Mailing
Slugify.Core	For generating URL slugs
Gulp	For simplifying the build process of front-end scripts and styles

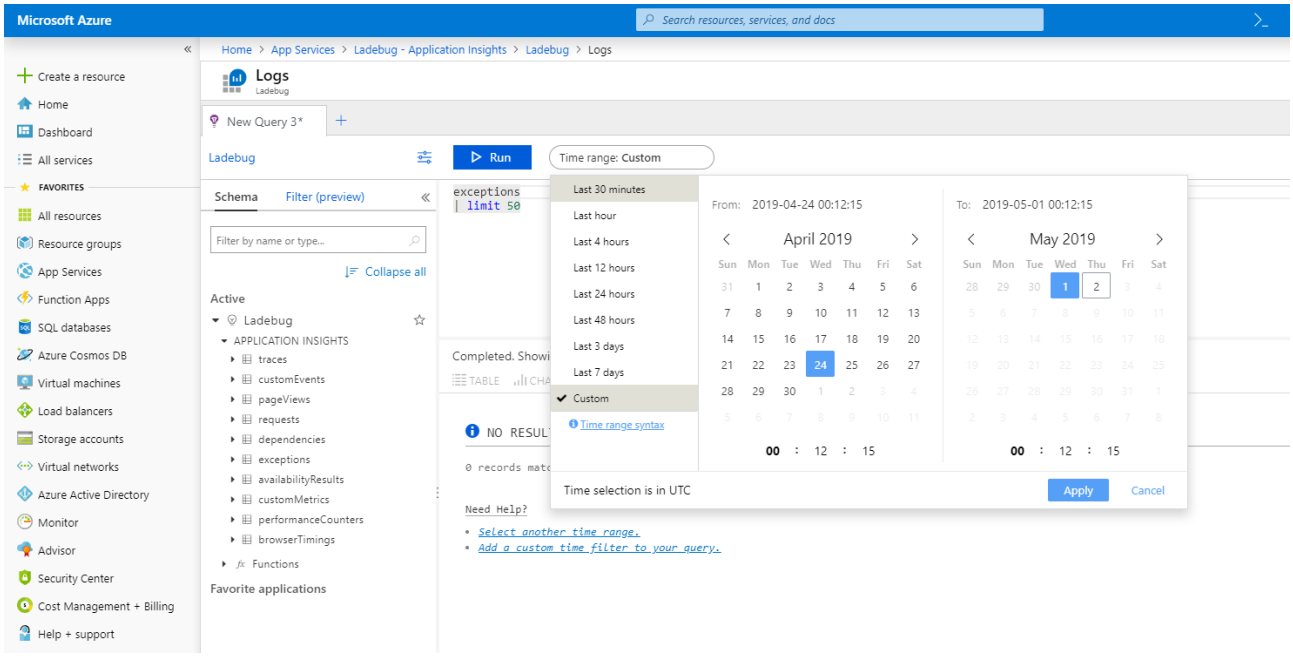
Front End

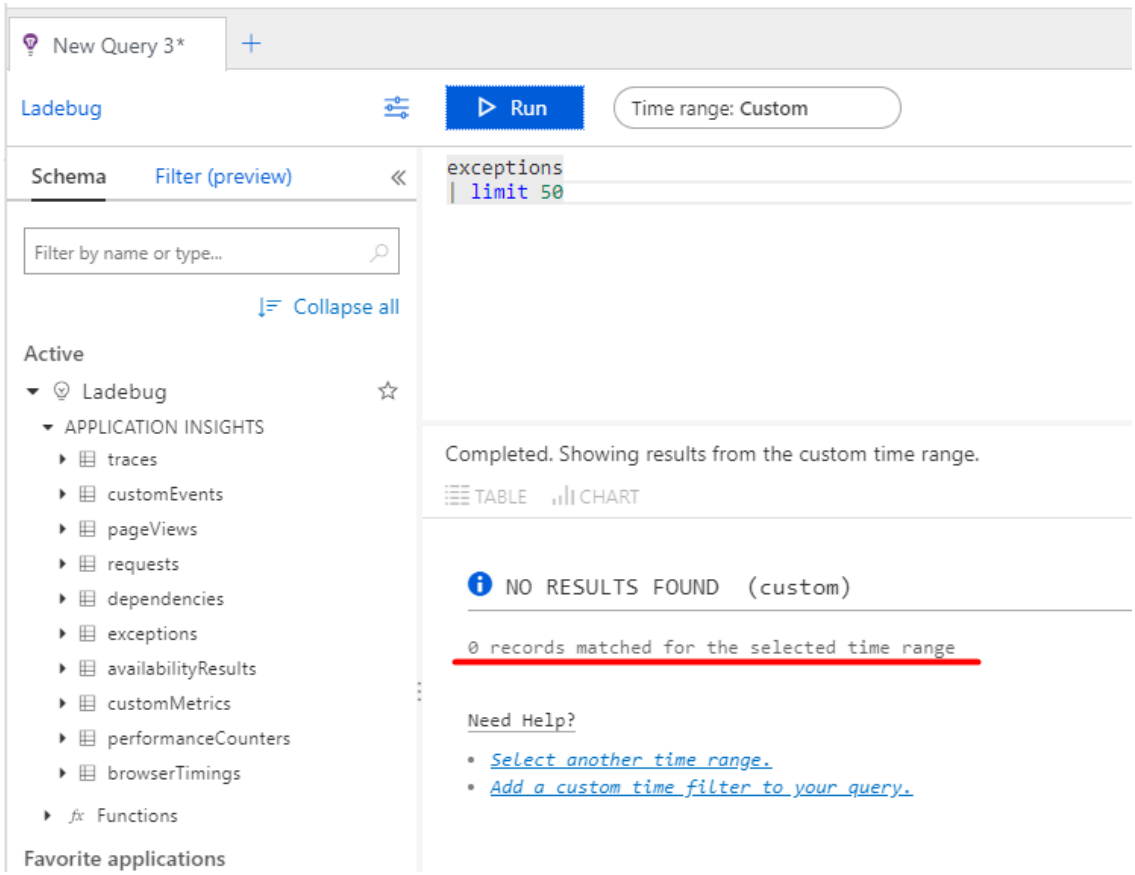
Library Name	Purpose
JQuery	For dynamic DOM manipulation
Bootstrap	For building a responsive Web UI
Vue.JS	For building dynamic web UI
Font Awesome	For element icons
Popper.js	For floating HTML elements
Code-Prettify	Displaying code snippets in a more friendly manner

Development

Tool Name	Purpose
Fiddler	This tool was essential to the development because the Ladebug chrome extension had to communicate with the backend server through an API, and as the API was also being refactored and changed, it was important to be able to inspect the HTTP traffic to be able to debug errors.
Postman	Helped with the development of the backend APIs Ladebug used

APPENDIX O - AZURE APPLICATION INSIGHTS





APPENDIX P - CLOSED BETA FEEDBACK

Feedback Survey from Tester 1

P1. Web App

* Q1 Were you able to login/register using either your Microsoft or GitHub account?

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q2 Once logged in, were you able to view all the teams you were a member of?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q3 Were you able to navigate to a team and only shown the errors posted in that specific team?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q4 Were you presented with an "Access denied" page when attempting to view an error belonging to a team you weren't a member of

- ( ) Yes
- ( ) No
- (●) I didn't try to view an error I didn't have access to
- ( ) other: enter answer

* Q5 Were you able to join a team using a join link

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q6 Were you able to create a team?

- ( ) Yes
- ( ) No
- (●) Not applicable
- ( ) other: enter answer

P2. Chrome extension

* Q7 Do you feel like the integeration of StackOverflow added value to Ladebug?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q8 Do you feel like the integeration of StackOverflow added value to your experience using Ladebug?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q9 Once you were shown the "How did you fix it dialog"(the screenshot above), were you shown a list of teams that you could post the error to?

- Yes
- No
- Not applicable
- other: enter answer

* Q10 Were you able to navigate through the pages of solutions suggested in the chrome extension popup(the screenshot above)?

- Yes
- No
- Not applicable
- other: enter answer

* Q11 Were you able to successfully post "something else you did" to fix an error to the selected team?

- Yes
- No
- Not applicable
- other: enter answer

P4. General Feedback

* Q12 How would you rate the user experience from using the new Ladebug?

Drag the slider from 1-10. 1 being the lowest, and 10 being the highest

7

Q13 Did you encounter any bugs during your testing period? If so, briefly list them below

enter answer

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Q14 Do you have any general feedback of the new Ladebug? If so, please write it in the box below

team when joining one that I am already in, just maybe show me a popup and take me to the team page

It seems odd that you have to be an admin to make a team, maybe this is intended though

P5. Developer Profile

* Q15 What is your job title?

This information is used to emphasize the validity of your feedback and be able to refer you in my dessertation whilst keeping you anonymous.

Student Developer

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Feedback Survey from Tester 2

P1. Web App

* Q1 Were you able to login/register using either your Microsoft or GitHub account?

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q2 Once logged in, were you able to view all the teams you were a member of?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q3 Were you able to navigate to a team and only shown the errors posted in that specific team?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q4 Were you presented with an "Access denied" page when attempting to view an error belonging to a team you weren't a member of

- ( ) Yes
- ( ) No
- (●) I didn't try to view an error I didn't have access to
- ( ) other: enter answer

* Q5 Were you able to join a team using a join link

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q6 Were you able to create a team?

- ( ) Yes
- ( ) No
- (●) Not applicable
- ( ) other: enter answer

P2. Chrome extension

* Q7 Do you feel like the integeration of StackOverflow added value to Ladebug?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q8 Do you feel like the integeration of StackOverflow added value to your experience using Ladebug?

- (●) Yes
- ( ) No
- ( ) other: enter answer

* Q9 Once you were shown the "How did you fix it dialog"(the screenshot above), were you shown a list of teams that you could post the error to?

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q10 Were you able to navigate through the pages of solutions suggested in the chrome extension popup(the screenshot above)?

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

* Q11 Were you able to successfully post "something else you did" to fix an error to the selected team?

- (●) Yes
- ( ) No
- ( ) Not applicable
- ( ) other: enter answer

P4. General Feedback

* Q12 How would you rate the user experience from using the new Ladebug?

Drag the slider from 1-10. 1 being the lowest, and 10 being the highest

8

Q13 Did you encounter any bugs during your testing period? If so, briefly list them below

No

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Q14 Do you have any general feedback of the new Ladebug? If so, please write it in the box below

Have the ability to turn off community/Stack Overflow suggestions and only see team suggestions.

OPEN-ENDED QUESTIONS CODING

P5. Developer Profile

* Q15 What is your job title?

This information is used to emphasize the validity of your feedback and be able to refer you in my dessertation whilst keeping you anonymous.

Developer

Security Feedback

From: [redacted]
Sent: 23 April 2019 15:35
To: Sam Mullins <SamMullins@redweb.com>
Subject: RE: ladebug v2 beta

Hi Sam,

I apologise in advance for not having used the application, so it may well be that this in unfounded. On this sort of thing I'm interested in risks. And it's not necessarily the application, but the user.

What if the developer over shares? Puts a password, client details, connection string etc. that would inadvertently would aid an attacker? Does the application caution the user against putting in sensitive data if certain strings are entered? A yellow screen of death can appear on a badly configured live site too…

Where is the data stored, and is it isolated and encrypted? Is the development data and suggestions anonymous? Could I see that lots of Redwebbers were submitting items about Sitecore, and know that RNLI were under active development, and put two and two together?

So for Redweb, for example, the risk might be reputational rather than data related – clients could see that we're slack about renewing Sitecore licences for example, based on the blog posts!

Thanks

[redacted]

From: Sam Mullins
Sent: 23 April 2019 15:07
To: [redacted]
Subject: RE: ladebug v2 beta

Hey [redacted]

The closest you'll get will be namespaces with client names in, it shouldn't have anything more than that.
This is meant to help with the development process, it's not possible to use on the live sites as Yellow Screens of Death will never appear.

This is something we're interested in though, making sure that everything is secure when we are storing data like this.

If there are specific things you like checked, it would be really good for these to be put to Uchenna so that he can use as feedback in his dissertation (and so you can also know the answers).

Cheers
Sam

Figure 54 - Security concerns for Ladebug from Director of Operations at Redweb

APPENDIX Q - CODE EVALUATION

Evaluation of the Existing System

P1. Design & Architecture

Q1 How well do you think I have implemented the programming principles I set out to follow?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

3

Q2 How well do you think I have implemented the architectural patterns I set out to follow?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

2

* Q3 In regards to the code, what was done well?

You separated persistence into separate repositories.

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Q4 In regards to the code, what could have been improved?

You could've used constructor injection in the controllers. You could have had interfaces for your repositories so that they could be easily replaced with different implementations. Your search service was tightly coupled to elastic search and would require a few changes to replace it with another. The parser, searcher, repositories and interface/API were all in the same assembly, which is a bit monolithic.

P2. Score Matrix

* Q5 How confident would you be adding new functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

2

* Q6 How confident would you be modifying existing functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

3

* Q7 How confident would you be removing existing functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

3

* Q8 How extensible do you think the system is?

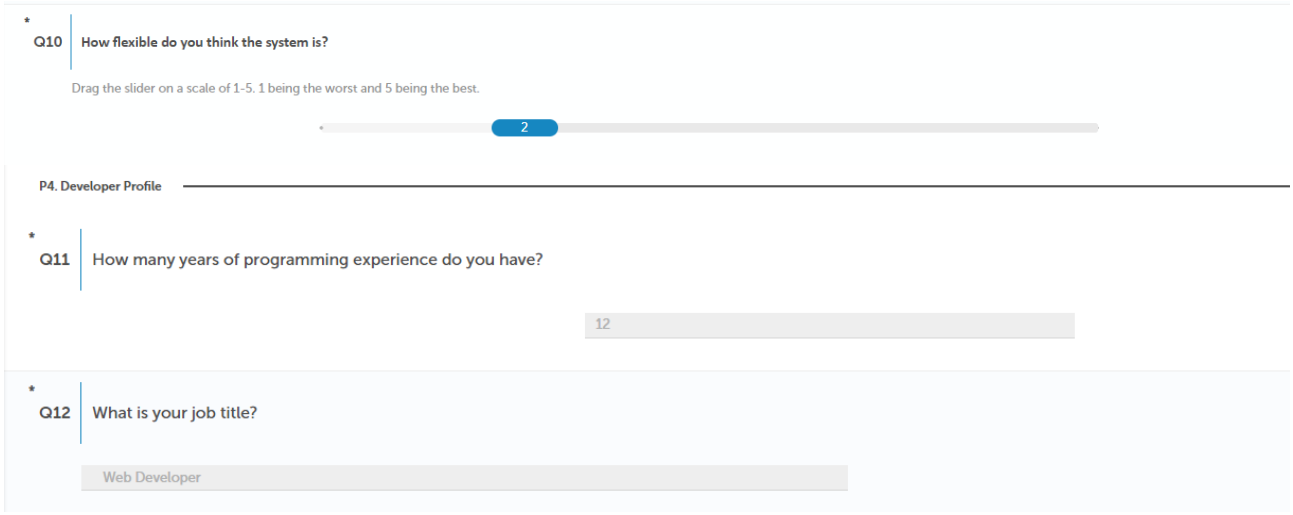
Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

2

* Q9 How maintainable do you think the system is?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

4


*
Q10
How flexible do you think the system is?
Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.
2
P4. Developer Profile
*
Q11
How many years of programming experience do you have?
12
*
Q12
What is your job title?
Web Developer

Evaluation of the New System (Artefact)

P1. Design & Architecture

Q1 How well do you think I have implemented the programming principles I set out to follow?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

4

Q2 How well do you think I have implemented the architectural patterns I set out to follow?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

4

* Q3 In regards to the code, what was done well?

Good separation of code making sure that each class does one thing well. Good use of patterns that are often used in the industry. Used the most up to date tools and techniques. Good use of DI/IOC. Good consistent code style.

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Q4 In regards to the code, what could have been improved?

Ideally the Elastic search code, Azure search code, and stack overflow code should be in different assemblies but as it is only actually one class I can see why they are not.

P2. Score Matrix

* Q5 How confident would you be adding new functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

5

* Q6 How confident would you be modifying existing functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

5

* Q7 How confident would you be removing existing functionality?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

5

* Q8 How extensible do you think the system is?

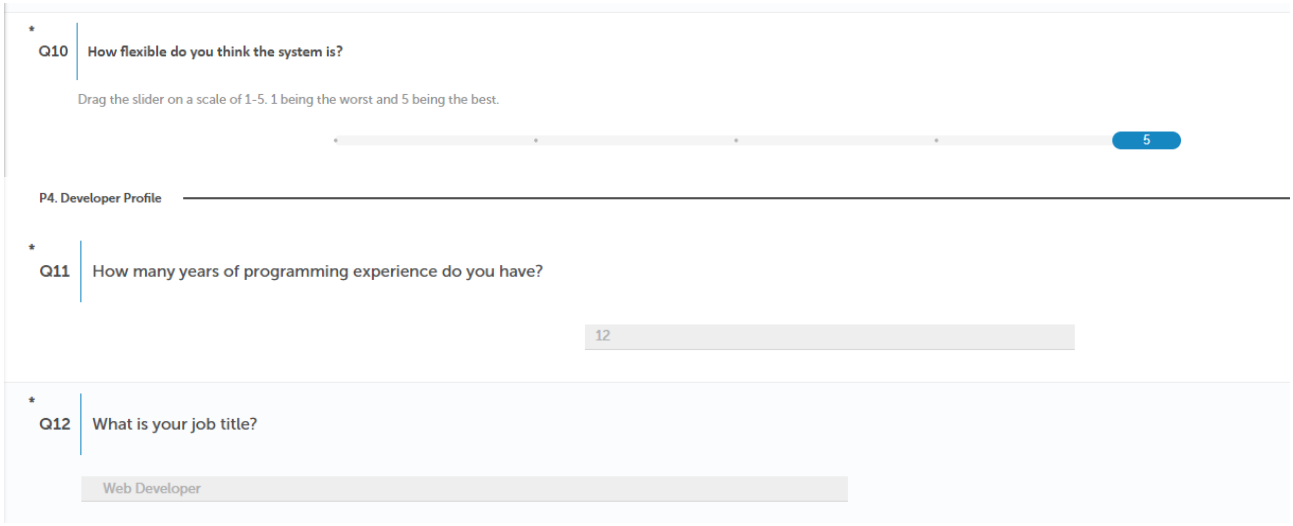
Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

5

* Q9 How maintainable do you think the system is?

Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.

5


*
Q10
How flexible do you think the system is?
Drag the slider on a scale of 1-5. 1 being the worst and 5 being the best.
5
P4. Developer Profile
*
Q11
How many years of programming experience do you have?
12
*
Q12
What is your job title?
Web Developer

APPENDIX R - CLIENT FEEDBACK

P1. Feedback

* **Q1** What do you think went well with the new Ladebug?

```
The new login works really well allowing the possibility of logging in with a different email.
Being able to separate into teams is also really useful and will stop too many results appearing.
The upgrade of the extension worked very well.
```

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

* **Q2** What could have been done better?

```
I think stack overflow answers should somehow be differentiated from ladebug answers.
```

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

* **Q3** How could the new system be improved going forward?

```
I think there should be a message making the user confirm that the data in the exception message
and in their answer is safe when posting to the community team.
```

OPEN-ENDED QUESTIONS CODING

To set answer tags, please go to account settings and then import tags.

You need to define tags to tag the answer.

Q4 Any final comments?

```
Good job!
```

APPENDIX S - FOLDER STRUCTURE OF EXISTING SYSTEM

```
+---Ladebug
|   +---App_Data
|   +---App_Start
|   +---Authentication
|   |   +---ActiveDirectory
|   |   \---OAuth
|   +---Content
|   |   +---css
|   |   \---images
|   +---Context
|   +---Controllers
|   |   \---Api
|   +---Extensions
|   +---fonts
|   |   \---font-awesome-4.7.0
|   |       +---css
|   |       +---fonts
|   |       +---less
|   |       \---scss
|   +---Migrations
|   +---Models
|   |   +---Entity
|   |   \---Enums
|   +---Parser
|   +---Properties
|   |   \---PublishProfiles
|   +---Repositories
|   |   \---Base
|   +---Scripts
|   +---Service
|   |   +---Extension
|   |   +---Models
|   |   |   +---Attributes
|   |   |   +---Base
|   |   |   \---Types
|   |   \---Settings
|   |       \---Models
|   +---Settings
|   +---Utils
|   +---ViewModels
|   \---Views
|       +---Auth
|       +---Dashboard
|       +---Errors
|       +---ErrorTests
|       +---Help
|       \---Shared
|           \---Partials
```

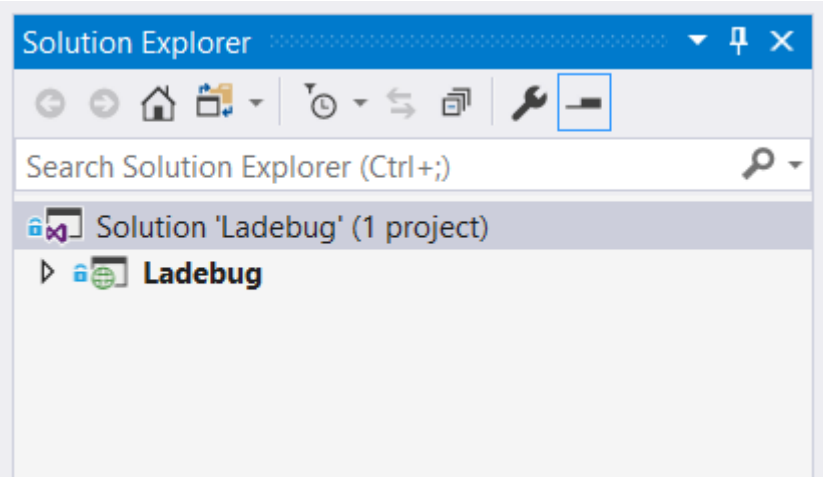


Figure 55 – Project view of the existing web app

APPENDIX T - FOLDER STRUCTURE OF ARTEFACT

```
+---Ladebug
    +---Ladebug.Core
    |   +---Models
    |   |   +---Entities
    |   |   \---Enums
    |   +---Parse
    |   |   +---Base
    |   |   +---Models
    |   |   |   \---ElasticSearch
    |   |   \---Services
    |   +---Repositories
    |   |   \---Base
    |   +---Search
    |   \---Services
    +---Ladebug.Infrastructure
    |   +---Extensions
    |   +---Factories
    |   \---Services
    |       \---Mail
    +---Ladebug.Parse
    |   +---Core
    |   |   +---Base
    |   |   +---Models
    |   |   |   \---ElasticSearch
    |   |   \---Services
    |   +---Modules
    |   +---Services
    |   \---Utils
    +---Ladebug.Persistence
    |   +---Configurations
    |   +---Context
    |   +---Extensions
    |   +---Migrations
    |   +---Repositories
    |   |   \---Base
    |   +---Seeding
    |   |   \---Migrate
    |   \---Services
    +---Ladebug.Search
    |   +---AzureSearch
    |   +---ElasticSearch
    |   +---External
    |   |   \---StackOverflow
    |   |       \---Models
    |   +---Services
    |   \---Utils
    +---Ladebug.Tests
    +---Ladebug.Web
    |   +---Areas
    |   |   \---Identity
    |   |       +---Pages
    |   |       |   \---Account
    |   |       |       \---Manage
    |   |       \---Services
    |   +---Controllers
    |   |   \---Api
    |   |       \---v1
    |   +---DependencyInjection
```

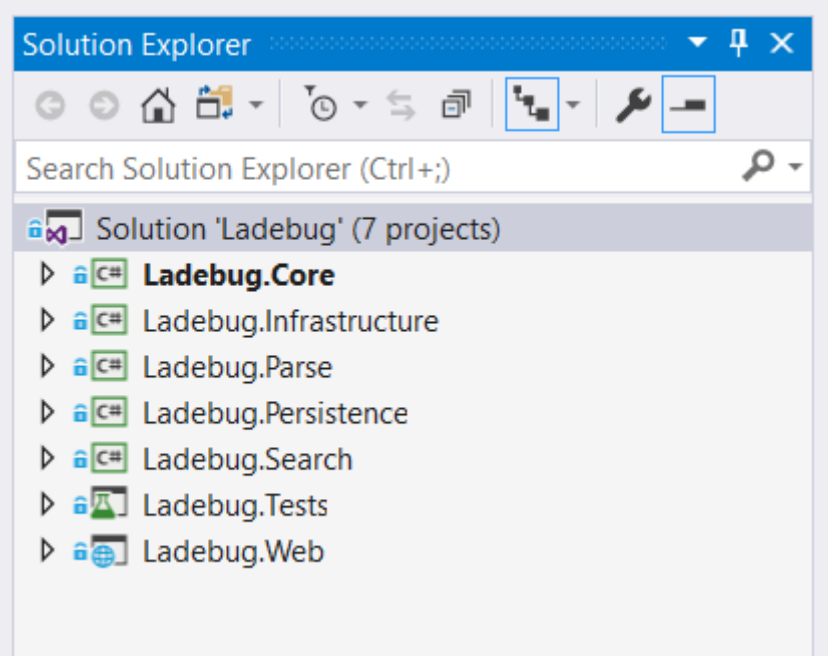
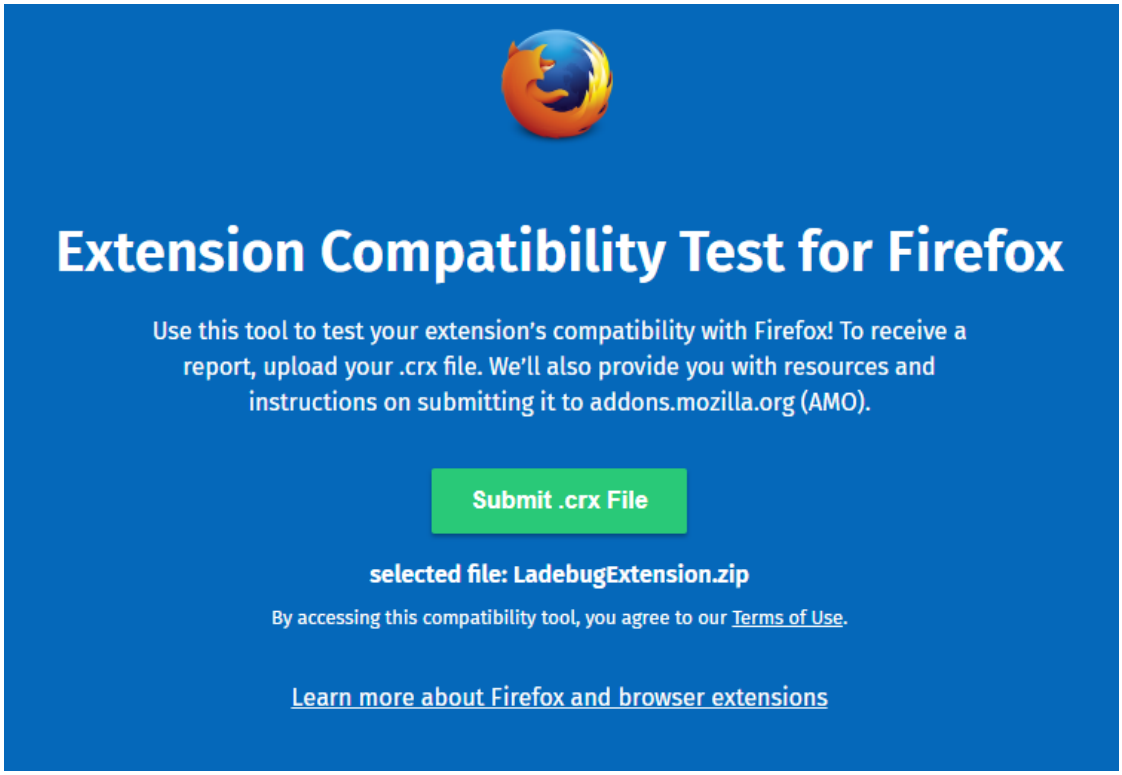
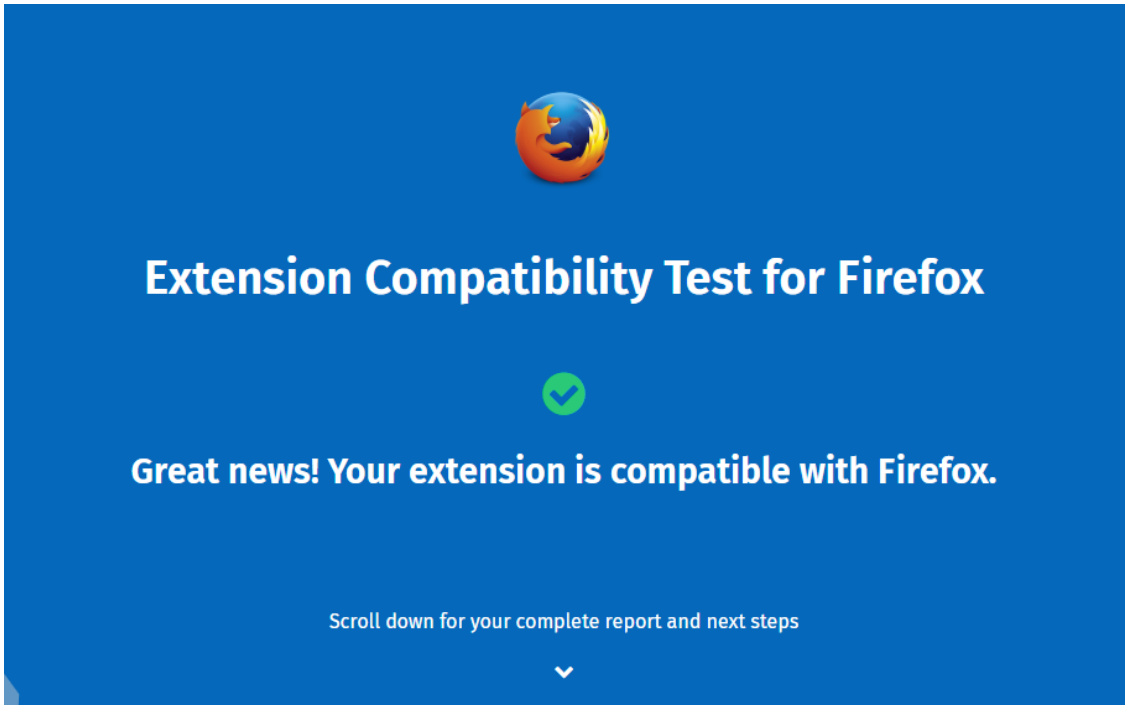


Figure 56 - Project view of the refactored web app

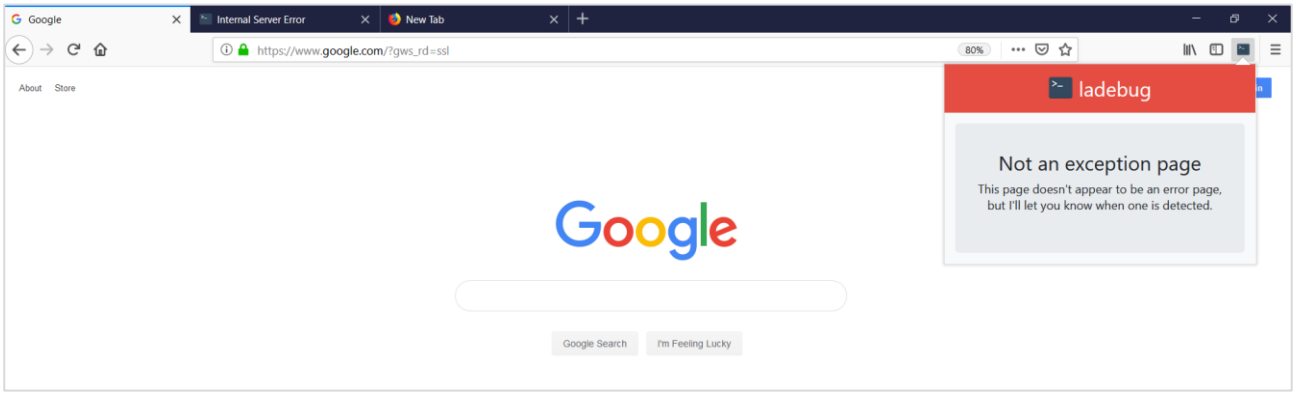
APPENDIX U - CHROME EXTENSION'S FIREFOX COMPATIBILITY

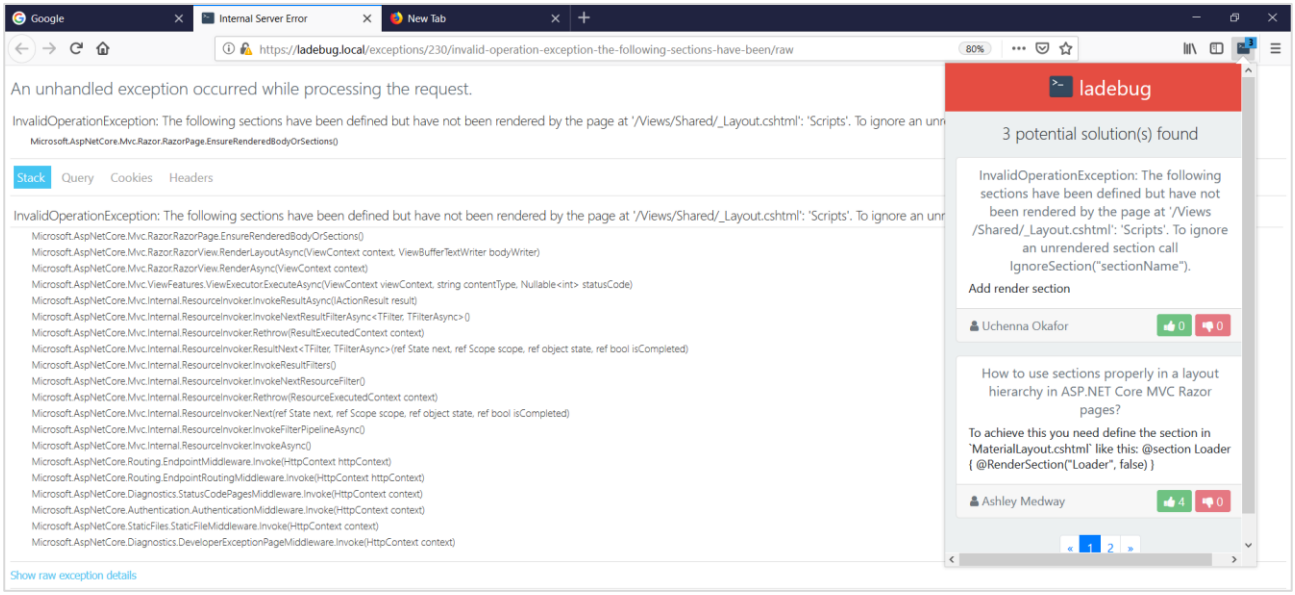
Checking the compatibly of the Chrome extension with Firefox

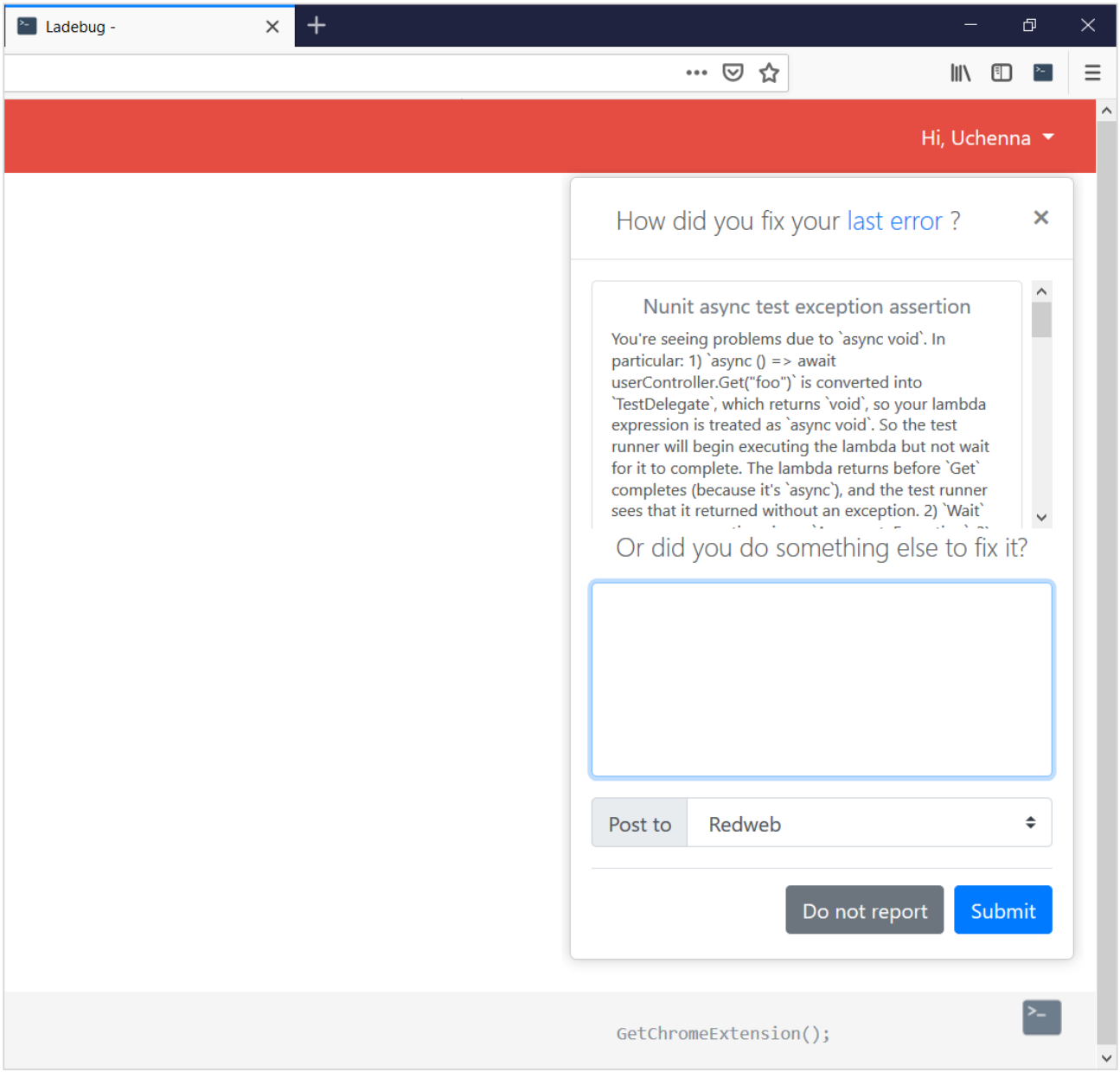




Testing Chrome extension on Firefox






Ladebug -
Hi, Uchenna
How did you fix your last error ?
Nunit async test exception assertion
You're seeing problems due to `async void`. In particular: 1) `async () => await userController.Get("foo")` is converted into `TestDelegate`, which returns `void`, so your lambda expression is treated as `async void`. So the test runner will begin executing the lambda but not wait for it to complete. The lambda returns before `Get` completes (because it's `async`), and the test runner sees that it returned without an exception. 2) `Wait`
Or did you do something else to fix it?
Post to
Redweb
Do not report
Submit
GetChromeExtension();

APPENDIX V - MEETING REQUIREMENTS AND FUFILLING SUCCESS CRITERIA

Success criteria fulfilments

ID	Description	Fulfilled?
SC1	Obtain system requirements from the client	Y
SC2	Research into the design patterns, programming practices and architectural patterns that enables building maintainable, extendible and flexibility software	Y
SC3	Migrate the existing system from a local environment to a cloud computing provider	Y
SC4	Refactor the existing system and follow industry standard practices for software development	Y
SC5	Implement the features in the system requirements	Y
SC6	Obtains and evaluate feedback from the users that will test the new system	Y
SC7	Obtain critical feedback from experienced developers who will review the code quality of the artefact and have them gauge their level of confidence in extending parts of the system without fear of breaking the entire system	Y
SC8	Receive feedback from the client on artefact and suggestions on how the system could be improved and the future direction of the product	Y

Requirements

Functional user requirements

ID	Category	Description	Requirement met?
F1	Authentication	Users should be able to register and log in with a local account	Y
F2	Authentication	Users should be able to register and log in with an external login provider using Open Authorization (OAuth)	Y
F3	Teams	User should be able to join a team	Y
F4	Teams	Users should be able to see a list of all the teams they're a member of	Y
F5	Teams	Users should be restricted to accessing or modifying content in teams they're not a member of.	Y
F6	Answers	Users should be able to edit answers they've contributed	Y
F7	Answers	Users should be able to delete answers they've contributed	Y
F8	Answers	Users should be able to delete error posts they've made	Y
F9	Chrome Extension	The user should be able to submit contributions to teams they belong to using the chrome extension.	Y
F10	Chrome Extension	Implement a modularized component that allows for errors to be parsed on the client side	Y
F11	Chrome Extension	The chrome extension should suggest answers from the teams the user is in and from StackOverflow	Y

Non-functional user requirements

ID	Category	Description	Requirement met?
NF1	User Experience	The web app and Chrome extension should be robust	Y
NF2	User Experience	The web app and Chrome extension should always provide user feedback when an error has occurred	Y
NF3	User Experience	The system should be easy to use	Y

Technical requirements

ID	Category	Description	Requirement met?
T1	System	Upgrade from ASP.NET MVC to ASP.NET Core MVC	Y
T2	System	Integrate answer finding process with StackOverflow	Y
T3	Front End	The web app should be responsive	Partially
T4	Cloud	Deploy artefact to the cloud	Y
T5	System	The system should be built using SOLID design principles	Y
T6	System	The system should be built using architectural principles	Y
T7	System	The system should be built to be maintainable, extendible and flexible	Y

APPENDIX W - PORTABLE MEDIA CONTENTS

On the USB attached you will find

Dissertation report.pdf

Artefact (Source code for both the Chrome extension and the web app)