Open Source .NET Blog

DESIGN DOCUMENT

34
Cylosoft
Selim Mohamed
PJ DeBisschop, Cole Mulligan, Sebastian Kazun, Thien Nguyen, Carter Cahill
sdmay20-34@iastate.edu
sdmay20-34.sd.ece.iastate.edu

Revised: 4/25 v4

Executive Summary

Engineering Standards and Development Practices

Standards:

- Software life cycle process (ISO/IEC 12207)
- Software assurance (ISO/IEC/IEEE 15026)
- Software review and audits (IEEE 1028)

Practices:

- Testing
- Refactoring
- Code reviewing
- Code simplicity

Summary of Requirements

Technologies:

- .NET 4.8 - a Windows based framework developed by Microsoft, utilized for building web-based applications and more
- MSSQL - Microsoft SQL, relational database system language developed by Microsoft
- Bootstrap 4.3 - a free and open-source CSS framework directed at responsive, mobile-first front-end web development
- CKEditor 4 - a rich text editor which enables writing content directly inside of web pages or online applications
- Microsoft Identity - a developer platform that allows developers to build applications that sign in all Microsoft identities and get tokens to call Microsoft APIs

Functionalities and use cases:

- Users (Roles & Rights) - every user role has different rights and any user cannot do what they are not authorized to
- CRUD Blog Posts - create, read, update, and delete posts
- Comments (Approve / Deny) - approving or denying a comment made by a user
- Categories (Filter) - categorize and filter posts by tags and view by most recent
- Site settings - settings for the blog such as changing themes
- Unit testing - the testing of individual units/components of the project

Applicable Courses from Iowa State University Curriculum

- COM S 309 Software Development Practices
- COM S 363 Intro to Database Management
- COM S 319 Construction of User Interfaces
- SE 329 Software Project Management

New Skills/Knowledge acquired that was not taught in courses

- .NET 4.8 Framework
- Microsoft SQL
- C#
- Bootstrap 4.3
- Microsoft Identity Framework/Owin Authentication
- Model/View/Controller Architecture

Table of Contents

1 Introduction 4

1.1 Acknowledgement 4

1.2 Problem and Project Statement 4

1.3 Operational Environment 4

1.4 Requirements 4

1.5 Intended Users and Uses 4

1.6 Assumptions and Limitations 5

1.7 Expected End Product and Deliverables 5

2. Specifications and Analysis 5

2.1 Proposed Design 5

2.2 Design Analysis 6

2.3 Development Process 6

2.4 Design Plan 6

3. Statement of Work 6

3.1 Previous Work And Literature 6

3.2 Technology Considerations 7

3.3 Task Decomposition 7

3.4 Possible Risks And Risk Management 7

3.5 Project Proposed Milestones and Evaluation Criteria 7

3.6 Project Tracking Procedures 7

3.7 Expected Results and Validation 7

4. Project Timeline, Estimated Resources, and Challenges 8

4.1 Project Timeline 8

4.2 Feasibility Assessment 8

4.3 Personnel Effort Requirements 8

4.4 Other Resource Requirements 8

4.5 Financial Requirements 9

5. Testing and Implementation 9

5.1 Interface Specifications 9

5.2 Hardware and software 9

5.3 Functional Testing 9
5.4 Non-Functional Testing 9
5.5 Process 10
5.6 Results 10
6. Closing Material 10
6.1 Conclusion 10
6.2 References 10
6.3 Appendices 10

List of figures/tables/symbols/definitions

Definitions

UI:	User Interface
CRUD:	Create/Read/Update/Delete
MVC:	Model-View-Controller

Figure 1: User Flow Diagram
Figure 2: System Block Diagram
Figure 3: Project Timeline with Subtasks and Dependencies
Figure 4: Project Timeline Gantt Chart
Figure 5: Testing Flow Diagram
Figure 6: Database Testing Diagram

1 Introduction

1.1 Acknowledgement

Thank you to our client Cylosoft, a national web development, ecommerce, and web hosting company, for their technical and design support throughout the project. They have given us a lot of freedom when it comes to the overall design and implementation of the Open Source .NET Blog, but are always available for any questions we may have regarding the features required. Cylosoft setup and gave us access to an SQL database so we could begin our implementation phase.

We would also like to thank Mohamed Selim, our Iowa State Advisor for this project, for helping us through the development and planning process.

1.2 Problem and Project Statement

General Problem Statement

In the current state of the world there are a limited amount of open source .NET blogs for users to take advantage of. Those that exist are outdated and fall behind today's technology. Because of this Cylosoft would like their own blog platform which include additional features and allow them more control.

Upon a quick survey we have come to realize that our competition currently utilizes outdated .NET or .NET CORE Frameworks, has no Microsoft Identity implementation, has no MVC architecture, and offers paid themes for customization.

General Solution Approach

In response to our competition, our project addresses their issues by utilizing current techniques and practices as well as the current .NET Framework, a Microsoft Identity implementation, an MVC architecture, and free themes for customization. On top of all that our project is also open-source.

1.3 Operational Environment

The .NET server and database will be hosted through Microsoft's Azure cloud service.

1.4 Requirements

Functional Requirements

Normal User Area for viewing content

- A blog visitor shall be able to sort posts by most recent
- A blog visitor shall be able to filter posts by tags
- A blog visitor shall be able to filter posts by categories
- A blog visitor shall be able to view approved comments on posts
- A blog visitor shall be able to sign up/log in for an account
- A blog visitor shall be able to comment on posts

- A blog visitor shall be able to search for specific posts

Admin Area for controlling content

- An admin shall be able to Create, Read, Update, and Delete posts
- An admin shall be able to view new comments and approve them for others to see
- An admin shall be able to add posts to certain categories for users to filter by
- An admin shall be able to edit user roles
- An admin shall be able to change the theme

Non-Functional Requirements

- The software shall be accessible at any time
- The software shall properly handle errors behind the scenes out of the users view
- The software shall display content properly based on device screen size
- The software shall utilize Microsoft Identity for security
- The software shall follow the MVC architecture to ensure that future modifications are done with speed and ease
- If a blog visitor is not signed in, then the blog shall not allow them access to unauthorized portions of the application
- The user interface shall be visually appealing
- The product shall appear simple to use

1.5 Intended Users and Uses

Our intention is to build an Open Source .NET blogging platform that allows Cylosoft to keep their customers informed about their activities, while also being flexible enough for other individuals or businesses to use as their own blog.

1.6 Assumptions and Limitations

Assumptions

- The end product will be open-source.
- The end product will be able to support multiple users at a time.
- The end product will be viewable on various media device screen sizes.

Limitations

- The length of time to produce the end product will not surpass May 2020.
- Enough database space to hold thousands of blog posts
- The cost of the product cannot exceed $150/month

1.7 Expected End Product and Deliverables

- Open Source .NET Blog Platform with source code
- Documentation
    - Design document

- Database design
- Architecture design

Expected Delivery date: May 1, 2020

2. Specifications and Analysis

2.1 Proposed Design

Our project is a .NET web application with the roles of admin, editor, and blog visitors. Using Microsoft Identity, an admin will login and have access to functionality that will allow them to manage the content of their blog and their blog's settings. The blog visitors go to the blog homepage where they can view the blog's content and make comments on the posts. All data is stored in an Azure MSSQL database.

2.2 Design Analysis

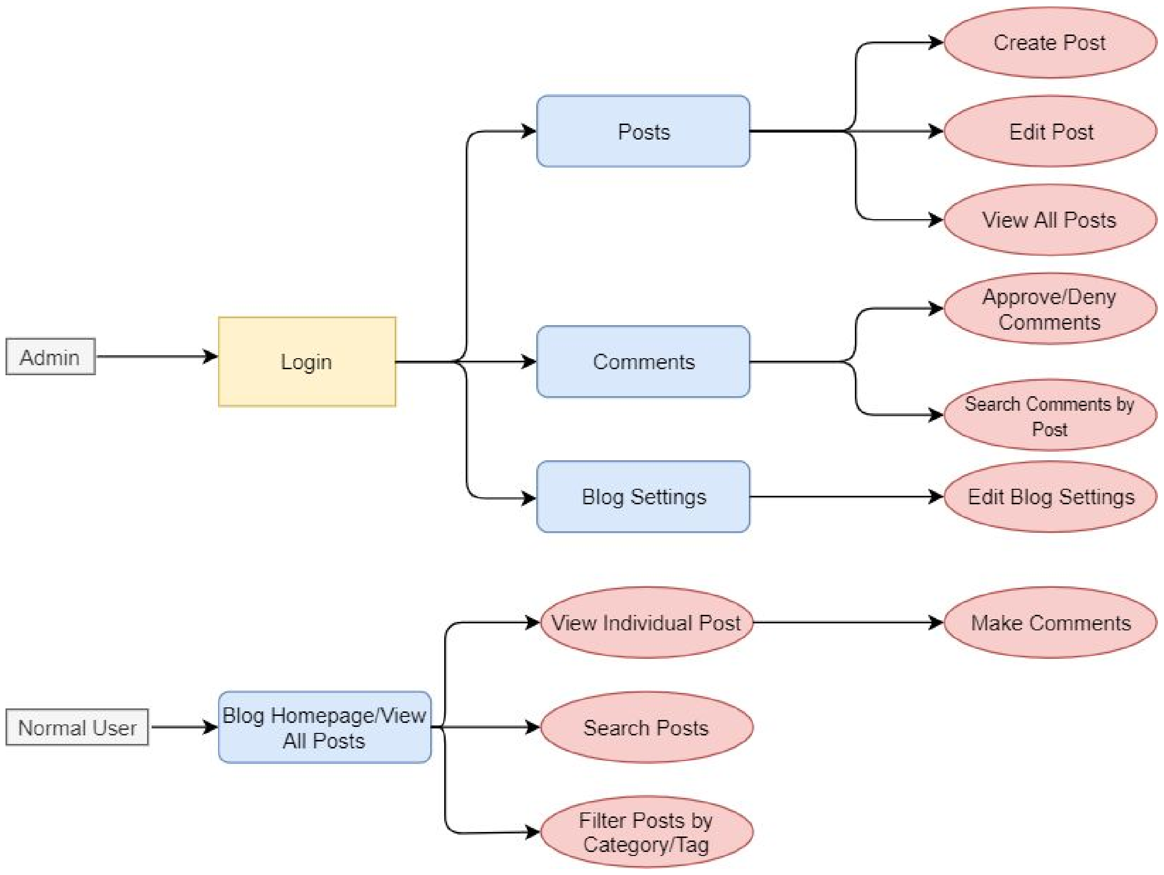


Figure 1

So far, we have began mocking up our user processes and researching .NET development. We plan to begin work with the User Flow Diagram from Figure 2 in mind. These are most of the requirements for our users to be able to operate their blogs and where in the website they can do each of these functions.

Our admin users start by logging in and are placed in the admin area. From here, they can select from the menu to work in the Posts section, the Comments section, or the Blog Settings section. These all have several actions that admins will be able to perform and the main actions are shown in the red ovals. If a user is not an admin, they will simply start on the homepage and be able to view all of the posts for a blog. They can click on a specific post to view the whole post and then make comments at the bottom of the screen. Otherwise, they can search for specific posts or show posts either assigned to a category or containing a tag.

2.3 Development Process

Our team will be following the Agile Development Process. We believe that adhering to the 12 principles will help immensely, allowing us to work flexibly and keep us on track. We will be using Trello as our agile board to help us assign tasks to the team. This will allow us to have a visual indicator on our progress for our current sprint and the effort of each teammate. Separating tasks into sprints allows us to more easily plan our work and make it manageable. We plan to release our first prototype by the end of the Fall 2019 semester in December. From there we will work incrementally to further develop features and refine our project. We will end with a final product in early May.

2.4 Design Plan

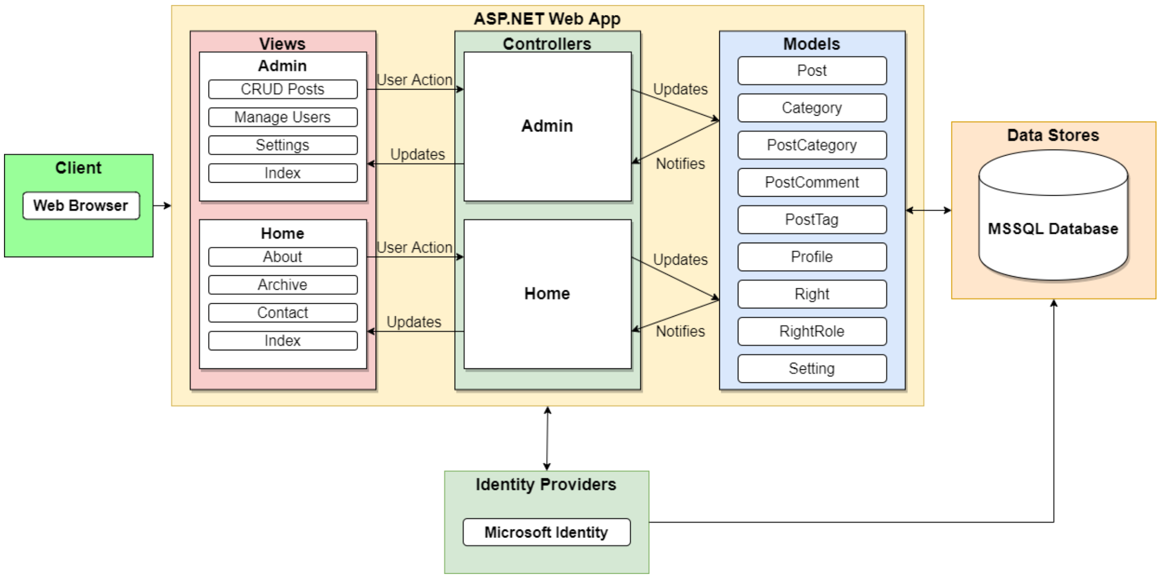


Figure 2

This is our first conceptual block diagram for how our Model View Controller system will function. You can start in either the Admin Area View or Normal User View, but both follow the same flow. Starting from the Admin Area View you will have options like CRUD Posts, a functionality allowing you to Create, Update, Read, or Delete any and all posts on the blog. After deciding what action you would like to perform the VIEW sends a user action request to the controller module. Inside this main module there are two main controllers, one for the admin and one for the home. Since we are doing CRUD Posts from the view we will also use the corresponding controller. This controller then updates the model, the model being our representation of the data contained inside the database. The model, after receiving the update and changing whatever needs to be changed, will then notify the controller which in turn will then update the corresponding view.

3. Statement of Work

3.1 Previous Work And Literature

BlogEngine.NET is the product that Cylosoft had us look at to get an idea of what we were building. This product is an open source blog builder with very similar features as to those required for our project. They offer a fairly easy setup and good design with clear methods for building the blog, with post management, comment management, user management, and a blog settings page. They also offer $10 theme packages for users to pick from.

Our project plans to offer a more attractive UI than similar products as well as incorporate up-to-date libraries. A point Cylosoft made was that current products aren’t actively maintained so as technology advances the current solutions just aren’t up to date.

3.2 Technology Considerations

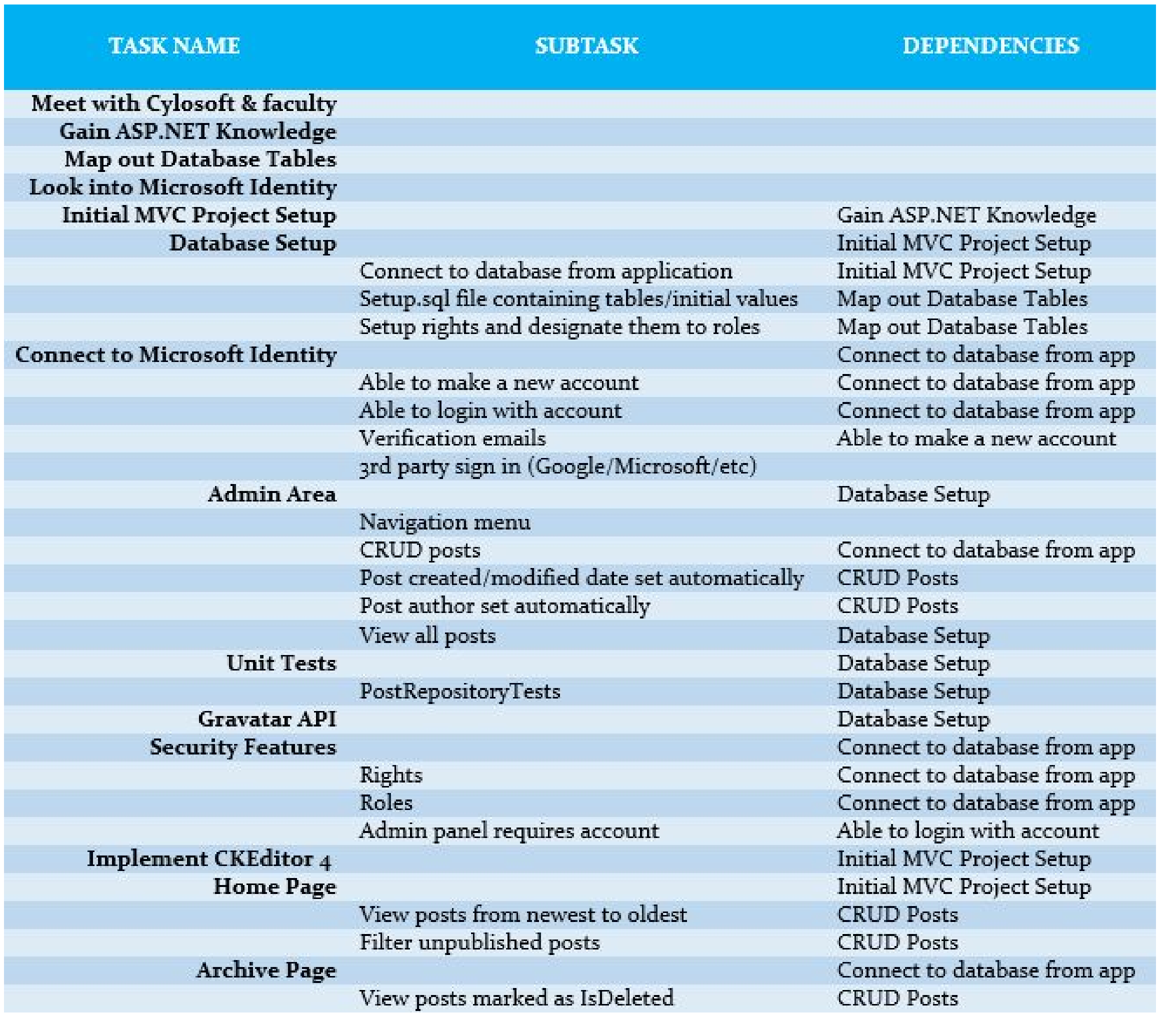
Our most prominent technology that we are developing with is .NET, which is a popular option with good amounts of documentation making it easy for our team to pick up quickly to begin work on the project. It also allows for simple server setup with many hosting services and integrates well with the other technologies in the project. C# will be used for handling a majority of our backend systems as it handles database transactions easily and has many examples performing similar features compared to our own solution.

Our database system will run using MSSQL, another popular technology that stays in the Microsoft development ecosystem. Our account system is Microsoft Identity, which offers an easy to use way to register and login in users and use our MSSQL database to store basic account information securely. It also provides a role system, which will be utilized for allowing access to certain parts of the software.

The project will also utilize MVC for clear documentation and modular development purposes, as well as Bootstrap for responsive design.

A different design approach could be to not use MVC at all, in lieu of just using stand alone Razor pages. This results in less clear code and simply wouldn’t be as modular, which would not suffice for our project as it benefits greatly from using modular pieces for things like posts between the admin system and the normal user views.

3.3 Task Decomposition



TASK NAME	SUBTASK	DEPENDENCIES
Meet with Cylosoft & faculty		
Gain ASP.NET Knowledge		
Map out Database Tables		
Look into Microsoft Identity		
Initial MVC Project Setup		Gain ASP.NET Knowledge
Database Setup		Initial MVC Project Setup
	Connect to database from application	Initial MVC Project Setup
	Setup.sql file containing tables/initial values	Map out Database Tables
	Setup rights and designate them to roles	Map out Database Tables
Connect to Microsoft Identity		Connect to database from app
	Able to make a new account	Connect to database from app
	Able to login with account	Connect to database from app
	Verification emails	Able to make a new account
	3rd party sign in (Google/Microsoft/etc)	
Admin Area		Database Setup
	Navigation menu	
	CRUD posts	Connect to database from app
	Post created/modified date set automatically	CRUD Posts
	Post author set automatically	CRUD Posts
	View all posts	Database Setup
Unit Tests		Database Setup
	PostRepositoryTests	Database Setup
Gravatar API		Database Setup
Security Features		Connect to database from app
	Rights	Connect to database from app
	Roles	Connect to database from app
	Admin panel requires account	Able to login with account
Implement CKEditor 4		Initial MVC Project Setup
Home Page		Initial MVC Project Setup
	View posts from newest to oldest	CRUD Posts
	Filter unpublished posts	CRUD Posts
Archive Page		Connect to database from app
	View posts marked as IsDeleted	CRUD Posts

Figure 3

Figure 3 contains the task decompositions, associated subtasks, and the dependencies of those tasks for our first semester.

3.4 Possible Risks And Risk Management

Our team is still new to many of the technologies and have not done complicated development in the system so there could still be knowledge gaps to overcome. However, active development is underway and our team is actively becoming more comfortable with development using these technologies.

There is a potential risk for data loss but we will mitigate this by providing several options for backups.

Another risk is possibly not completing the project in time, however, we are following the agile development process to stay on time. This will allow us to incrementally push out features in a short period of time. If we do run out of time, we may have to alter or cut features entirely.

Another risk is miscommunicating the requirements of the project but we have mitigated this by meeting up with our Cylosoft representatives on multiple occasions to clarify project points.

One last potential risk is failure to integrate 3rd party technologies into our project, such as Gravatar. Though, we expect this to be a minor risk as much of this risk. Risks like this can be attributed to our inexperience with all these technologies, which is becoming less of a problem since we are quickly learning them.

3.5 Project Proposed Milestones and Evaluation Criteria

1. Post creation
    a. Styled text
    b. Viewable to non-admins
    c. Include images
2. User management
    a. Login/Register
    b. Role assignment
    c. Edit user settings
    d. Delete users
3. Change other site settings
    a. Blog theme
4. Admin area fully implemented

We plan to implement unit testing for our backend system as well as several acceptance tests with Cylosoft.

3.6 Project Tracking Procedures

In the beginning we were using an excel document to plan and keep track of our weekly progress. The Weekly Plan document outlines the week, the task to be done, who it is assigned to, completion percent, and shows which parts are dependent on each other. Since then, we have moved towards utilizing a Gantt chart for planning. For next semester we also plan to use a Gantt chart to plan out our project and track its progress as we go, as it provides more visual information compared to our current setup.

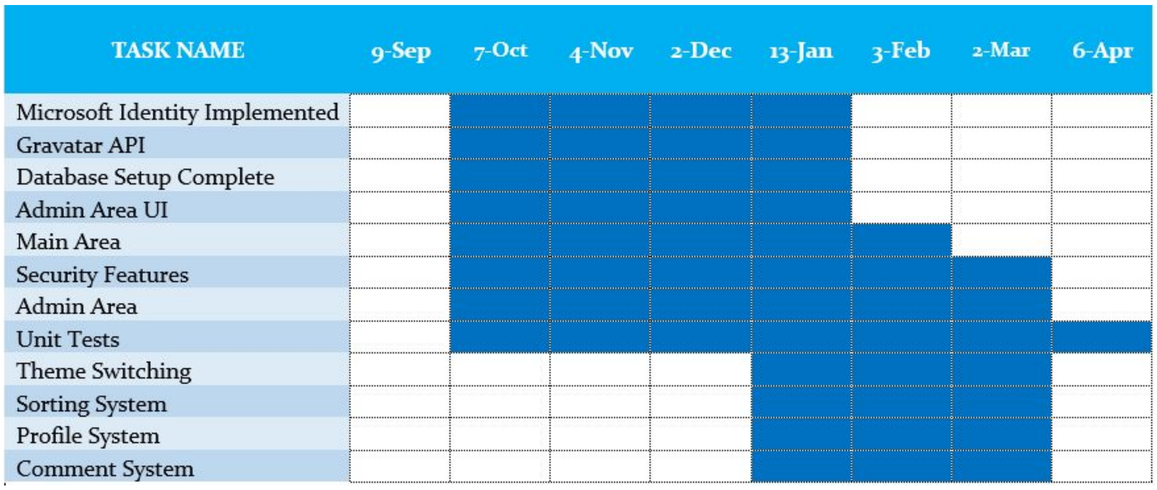
3.7 Expected Results and Validation

Our desired outcome is to have a fully functional Open Source Blog up and running by the end of May. We hope to have all the features Cylosoft required implemented and fully functioning in our application. In regards to the pages accessible to a normal user they will have the ability to view most recent posts, filter those posts by tags, and comment on the posts. In regards to the admin area the functionality will include: full management of users, roles, and rights, ability to create, read, update, and delete posts, approve or deny comments, create or delete categories and tags, and the changing of site settings. The admin area will only be accessible to users logged in and that have a role marked as Administrator or Editor. Once a user has access to the admin area only certain options will be accessible to them based on their current role.

At a High Level we will confirm our solutions through acceptance testing: having our project contacts review and provide feedback on what needs to be adjusted. We will also revisit our requirements and make sure each one has been met.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline



TASK NAME	9-Sep	7-Oct	4-Nov	2-Dec	13-Jan	3-Feb	2-Mar	6-Apr
Microsoft Identity Implemented		■	■	■	■			
Gravatar API		■	■	■	■			
Database Setup Complete		■	■	■	■			
Admin Area UI		■	■	■	■			
Main Area		■	■	■	■	■		
Security Features		■	■	■	■	■	■	
Admin Area		■	■	■	■	■	■	
Unit Tests		■	■	■	■	■	■	■
Theme Switching					■	■	■	
Sorting System					■	■	■	
Profile System					■	■	■	
Comment System					■	■	■	

Figure 4

4.2 Feasibility Assessment

The end goal of our project is to have an Open Source Blog application. It will be available for free on GitHub for anyone to download and use for themselves. Some challenges we expect to face is the implementation of all the desired features from Cylosoft and ensuring they work properly. Another potential issue is making sure this application is truly "open source" and that other potential users are able to setup and use the application fairly easily.

4.3 Personnel Effort Requirements

Task	Personnel Effort
Initial MVC Project Setup	Preliminary research into .NET MVC project structure and setup, as well as incorporating Razor pages and Microsoft Identity. Ensuring this is done properly, knowledge of the MVC architecture and how it is used will be essential to making the project.
Connection to Database	Look into the different ways of incorporating a

	database into a .NET application, such as the repository pattern. Perform an initial implementation and build out a testing project to verify it works as intended.
Design and Functionality of Admin Page	Build out a list of all the required functionality for the Admin page. Build a more detailed list of the step by step process each functionality will need in order to function properly. Create a mock up UI and begin implementation.
Testing	Test everything as it is being built. This essentially doubles everyone's work load. After implementation of a feature we will need to build out a test method to confirm our change works as intended.

4.4 Financial Requirements

A Microsoft Azure hosting platform for the database and website is expected to be $150 per month.

5. Testing and Implementation

5.1 Interface Specifications

Users will need a browser application with web access capabilities, such as Google Chrome or Firefox. Microsoft SQL will be used for database storage while C# will be used for backend and frontend controller logic, and Bootstrap will be used for User Interface design.

For testing, we will use a multitude of browser clients to test the application on our local machine.

There are no hardware interfaces that we will be using.

5.2 Hardware and software

We will be using C#'s built-in Unit Testing Framework for testing individual modules such as the controller. We will also use the unit testing framework for testing our database as well. One component of our project utilizes Microsoft Identity, in which we will mock classes for testing using the Moq framework and the xUnit.net framework.

The User Interface will be tested using Selenium.

We are not utilizing any hardware for our project.

5.3 Functional Testing

Unit Testing
Each component shall be extensively tested and compared against a predetermined test set that is

deemed as the correct specification. This means that every class and its features will be completely covered before each commit.

Integration Testing
The individual components must be put together and all be tested to ensure that they can communicate with each other, especially the database correctly. This testing focuses on ensuring the system works when adding new parts and is tested against the correct UI flow as well as comparing the expected data with the actual data in the database.

Acceptance Testing
Once our entire project is tested for correct communication, we will meet with our Cylosoft representatives who will go over and check our project and provide feedback for any necessary improvements or changes to the software. We will also test our project against our Functional and Nonfunctional requirements to make sure it is within scope.

User Testing
We will have a third party, someone who knows nothing about our project, use our software in the eyes of the end user and provide feedback. This feedback will be used to revise the necessary project components in terms of UI/UX.

The following figure shows the order of the test flow we will follow:

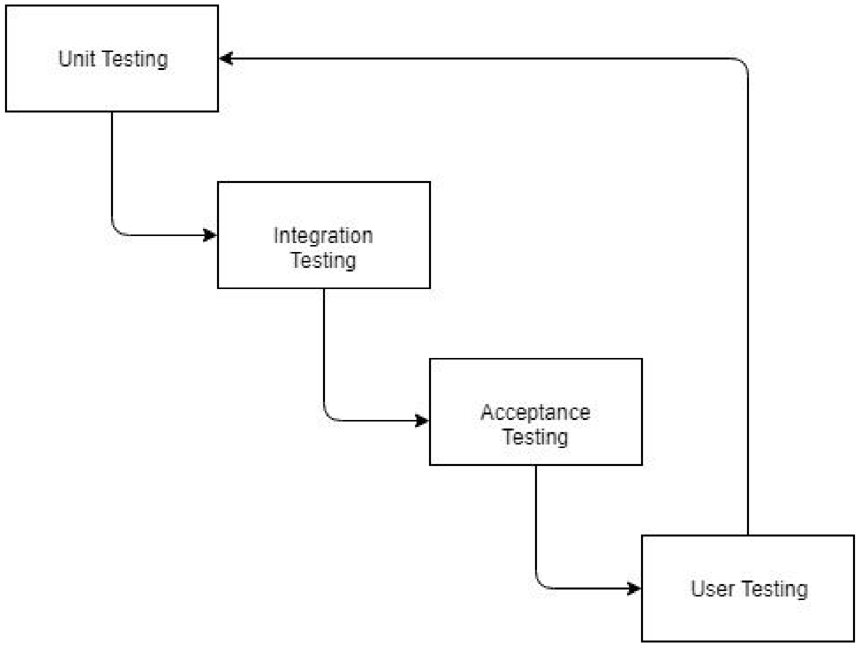


Figure 5

5.4 Non-Functional Testing

Performance Testing

The performance of database queries will be our main obstacle here. General webpage loading should not be an issue, but if certain pages have a large amount of data queries it may slow down loading time.

Success condition: Every webpage loads in <500ms

Security Testing

The security of our project will be tested by ensuring unauthorized users do not have access to more than they are allowed. This will be done using Microsoft Identity features, like allowing us to easily check a user's role. If a user does not have the correct role they will not be able to access or use the desired feature.

Success condition: A user without proper privileges is unable to perform the desired action.

Usability Testing

The usability of the website will ensure that the main home page is accessible and viewable from any modern device.

Success condition: When visiting the homepage the display adjusts accordingly to the device screen size and provides the same functionality as on a desktop computer.

Modifiability Testing

The modifiability of our application will be tested by ensuring that changes can be made easily.

5.5 Process

User Interface Testing

- Testing of the user interface is currently done by hand. This means if we want to test a new feature or change works as intended we must boot up the application, navigate to where the desired change was made, and test the implementation is correct.

Database Testing

- The database is currently utilizing the built in unit testing system of C#. For each table in our database there is a corresponding model in our application. Each model can then be manipulated using the repository corresponding to it.
- Each repository has a corresponding test class in our test project. This breaks up each repository into their main functions: GetAll(), Get(), Create(), Update(), and Delete(). From here we can then test that each of these individual functions work appropriately.

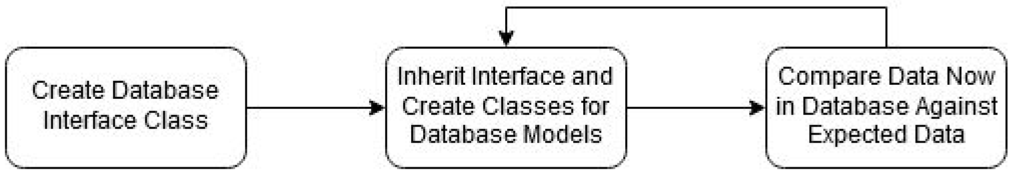

Create Database Interface Class
Inherit Interface and Create Classes for Database Models
Compare Data Now in Database Against Expected Data

Figure 6

5.6 Results

In regards to integration testing we made sure the components and database all worked together. We have verified using our testing project that the methods: Get(), GetAll(), Create(), Update(), and Delete() work as intended for posts. We utilized the Moq framework and C# Unit Tests to Unit

Test our application, which ensured that our controller logic worked as specified, mainly concerning the Admin Area of our project. Our Acceptance Test results were satisfactory as indicated by our client, Cylosoft, who reviewed our project monthly and gave feedback. Our final meeting with them indicated that our client was satisfied with the product we have put forth. Our User Testing was satisfactory concerning our features, but not for UI design. This led us to make changes to improve our UI in general until the Users found the design satisfactory.

For Non-Functional testing, we were concerned with performance, usability, and security. The performance of our database queries and page loading was deemed satisfactory. The usability of our software was deemed satisfactory as it performed as needed with differing screen sizes, which was tested by our groups varying laptop sizes. The security of our software was deemed satisfactory as we manually, and automatically using Selenium IDE, confirmed that unauthorized users would not be able to access features that were not meant for them.

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

So far, we have a basic prototype where accounts can be created and a post can be created and viewed from the homepage. We plan to add the comments system and do a lot of UI work so that users can have unique and creative blogs. Our plan is to get the admin area done early in the spring semester so that content can be created and then we can focus on the themes and the comments system. This is the best plan because we can make sure that the actual content that makes the blog work is there and we have a lot of time to make the creative elements better.

6.2 References

"BlogEngine.NET", *Github*, https://github.com/rxtur/BlogEngine.NET

6.3 Appendices

I. Operation Manual

This is the developer option. Use this if you are interested in seeing how things work or would like to implement your own functionality.

Environment:

- Visual Studio 2017 +
- .NET Framework 4.8 +
- Microsoft SQL Database through Azure

STEPS:

1. Clone repository through Visual Studio
2. Use the setup.sql inside of the Database folder to load the necessary tables into your database.
3. Open solution in Visual Studio 2017 +
4. Switch to solution view
5. Inside OpenSourceBlog open the Web.config
    1. Edit the connectionString="" to be your Azure database connection string
    2. Edit **ALL** instances of sdmay20.34@gmail.com to be your gmail (for sending confirmation emails). Add your password as well.
6. Inside OpenSourceBlog.Test open the App.config
    1. Edit the connectionString="" to be your Azure database connection string
7. Build and run solution to load website in the browser
    1. There are times where you may get an error, if you get this try building the projects individually, cleaning and/or rebuilding the solution.
8. Click login in the top right
9. Username: admin@gmail.com Password: Admin1@

II. ALTERNATIVE/INITIAL VERSIONS

We do not have any Alternative versions for this project.