

Community Board

Final Report

Team 4:

Brittany

Brandon

George

Edgar

Final Report	1
Project Documentation Version History	3
Introduction	4
Goals and Objectives	4
Functional requirements	4
Non-Functional requirements	4
Inverse requirements	4
Statement of Scope	5
Software Context	5
Major Constraints	5
Usage Scenario	5
User Profiles	5
Use Cases	6
Use Case Diagram Considerations	9
Software Interface Description	10
External Machine Interfaces	10
External System Interfaces	10
Human Interface	10
Behavioral Model and Description	10
Events	10
States	10

Sequence Diagrams	11
Restrictions, Limitations, and Constraints	14
Configuration Management	14
User Project Resources	14
Project Estimates	15
Project Resources	17
Risk Management	17
Risk Table	17
Project Schedule	18
Project Tasks and Timeline Chart	18
Staff Organization	18
Team Structure	18
Management Reporting and Organization	18
Data Design	19
Internal Software Data Structures	19
API Driven Architecture (Globally)	19
Temporary Data Structure	19
Repository Design Pattern	19
Dependency Injection Design Pattern	20
Database Description	20
Entity Relationship Diagram	20
Architectural and Component-Level Design	22
Architecture Diagram	22
MVC Design Pattern	22
User Interface Design	22
Description of the User Interface	22
Screen Images	23
Home Page	23
Register & Login Page/Form	23
Contact/Report Page	24
Create/Update Announcement Page	25
Manage Announcements	26
Objects and Actions	26
Interface Design Rules	27
Components Available	27
Restrictions, Limitations, and Constraints	27
Testing Issues	28
Classes of Tests (See Testing Types for further details)	28
Expected Software Response	28

Performance Bounds	28
Identification of Critical Components	28
Packaging and Installation Issues	28
Testing Types	29
White Box Testing	29
Integration Testing	29
API Testing with Postman	30
API Endpoints:	31
Sample API Valid Response:	32
Sample API Invalid Response (Unauthorized):	32
User/UI Testing	33
Testing Chart	33
Potential Enhancements	37
Refactoring	37
Lessons Learned	37
Appendix	37
References/Supplementary Information	38
Source Code	38
Production Environment	38
Website	38
API (Showing a few endpoints):	38

Project Documentation Version History

Version Number	Date	Revision Author	Description
1.0	2/22/2021	Everyone	Initial Documentation
1.1	2/28/2021	Edgar	New Database Design & Added Reference

Introduction

Goals and Objectives

Our goal is to develop an announcement website for people to offer their services such as sales, tutoring, job offering, school-related events. Users will be able to contact the person of the announcement and report inappropriate behavior.

Functional requirements

- The user will register an account and log in. If login is invalid, an error message will display.
- Logged in users can create, update, and delete their announcements.
- Users will be able to view and search announcements made.
- If no announcements are found given the user's criteria, a "Not Found" message will be displayed.
- Users will be able to contact the person of the post via filling an email form. If the user is not logged in, the system will redirect them to the register page.
- Anyone will be able to report inappropriate announcements and will notify administrators.

Non-Functional requirements

- Privacy protection (not sharing private information).
- Quick response time when searching/loading posts.
- Error messages/handling, no crashes of server.
- Users must have an internet connection.

Inverse requirements

- Website doesn't have its own chat system. Customers must fill the contact form and communicate through email.
- Website does not keep track of announcement logs. Once a post has been deleted or been marked as "closed," it will disappear forever.
- Website will not have admin controls to see reports. When an announcement is reported, it will be saved in the database and notify the administrators via email for further action.
- Administrators will review and remove or clear the reported announcement.

Statement of Scope

Upon opening the website, the user will see the home page in which recent announcements will display. Users will either search for announcements of their interest or register an account in order to be able to post announcements in the board. Once registered, the user will have access to the posting form and be able to publish. On the other hand, any customer

interested in any announcement service may click the “contact” button. If not registered, the system will ask the customer to make an account in order to access the contact form. After filling the contact form, an email will be sent to the author of the announcement and the communication goes on from there. Once the customer and author come to an agreement, the author will be able to mark the announcement as “Closed,” which will be deleted after some time.

Software Context

Due to the COVID-19 pandemic, bulletin boards are not in use anymore. Schools closed, people are at home, and school events/clubs are not as known. Very few events are sent through the school emails which not many people check. For those reasons, our team has decided to make these drastic changes alive again.

The purpose of this website is to provide our community a virtual bulletin board (similar to what OCC had in the walls before the pandemic). This is highly useful for those looking to provide their service such as tutoring, those looking for a job, club, or event to join/apply, or even people who might be graduating and want to sell their books or school supplies for a cheaper price.

Major Constraints

- Internet access required for the whole system.
- Database access.
- REST API calls using HTTP protocol.

Usage Scenario

User Profiles

User creates an account with name, username, email, and password. Users post as many announcements of services, or events on the message board. Customers can browse the board for items of their interest, then create an account with their information to email a message to the user. Anyone can report inappropriate announcements.

Use Cases

Home Page

Actor	Role	Description
User/Customer	Poster/Visitor	Views/Searches announcements already posted
Program (API)	Data Communication	Handles HTTP

		request/response
Database	Data Storage	Extracts announcements to view

User Case Interaction

Predecessor: None. This is the landing page. Everyone has access to this page regardless of having an account or not.

Successor: Register Page.

Register Page

Actor	Role	Description
User/Customer	Poster/Visitor	Enters credential information
Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Registers user credentials

User Case Interaction

Predecessor: Home page.

Successor: Login Page.

Log in

Actor	Role	Description
User/Customer	Poster/Visitor	Enters credential information
Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Extracts data for authentication

User Case Interaction

Predecessor: Register Page. The user must register an account first.

Successor: Any other use case.

Create Announcement

Actor	Role	Description
User	Poster	Creates announcement using form
Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Saves and stores announcement

User Case Interaction

Predecessor: Login. The user must be logged in.

Successor: Any other use case.

Update Announcement

Actor	Role	Description
User	Poster	Updates the announcement.
Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Updates the announcement

User Case Interaction

Predecessor: Login & Create Announcement. The user must be logged in and have created an announcement.

Successor: Any other use case.

Delete Announcement

Actor	Role	Description
User	Poster	Deletes the announcement

Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Deletes the announcement

User Case Interaction

Predecessor: Login & Create Announcement. The user must be logged in and have created an announcement.

Successor: Any other use case.

Contact/Email Announcement Author

Actor	Role	Description
Customer	Visitor	Fills contact form and sends a message to the author.
Email API	Email Communication	Notifies post author about customer message regarding the announcement.

User Case Interaction

Predecessor: Login. The user must be logged in in order to contact the author.

Successor: Any other use case.

Report Announcement

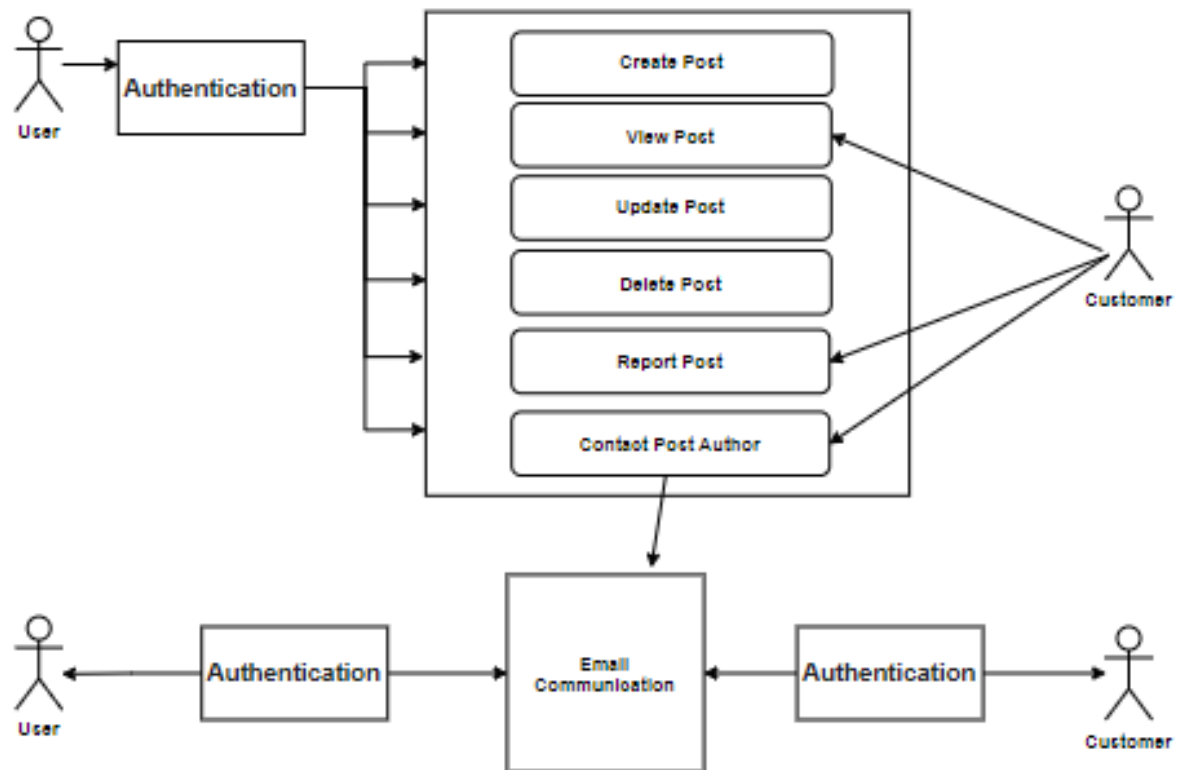
Actor	Role	Description
User/Customer	Poster/Visitor	Reports announcement
Program (API)	Data Communication	Handles HTTP request/response
Database	Data Storage	Saves report information
Email API	Email Communication	Notifies administrators about the report.

User Case Interaction

Predecessor: Home Page. The user must be on the home page and may not be registered. Anyone can report.

Successor: Any other use case.

Use Case Diagram Considerations



Software Interface Description

External Machine Interfaces

This website does not plan to communicate with any external machine.

External System Interfaces

- The frontend (what users can see) will be connected to the backend system through REST API (Application Programming Interface) requests and responses using the HTTP Protocol.
- Email API. Will be used to send email.

Human Interface

The human interface will allow:

- Users to create and modify announcements as desired.
- Customers to contact the author of the post completing the email form.

- Any person to report a misleading post.

Behavioral Model and Description

Events

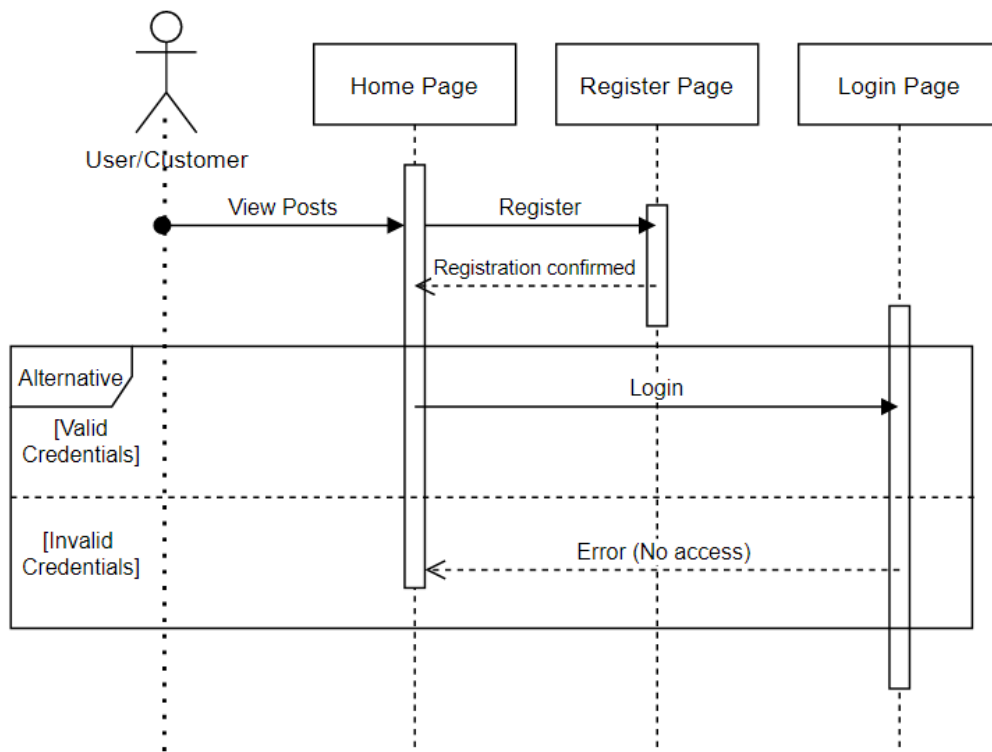
- Landing Page (Home Page).
 - Announcements are loaded
- Register Page.
- Login Page.
- Create Announcement Page.
- Update/Delete Announcement.
- Contact Author.
- Report Announcement.

States

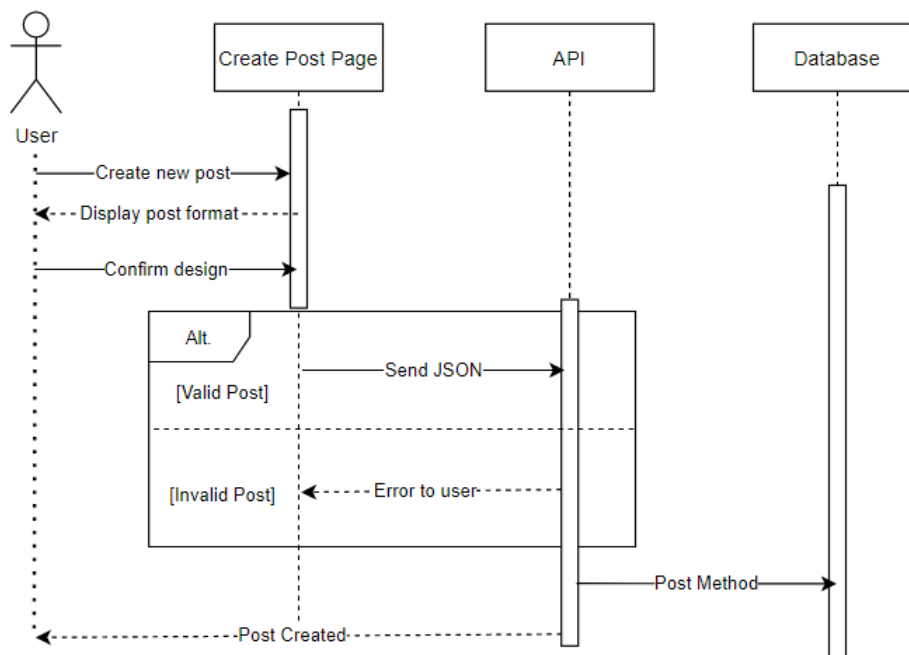
- User Authentication.
- Reported Announcement (Flagged).
- Updated/Deleted Announcement.

Sequence Diagrams

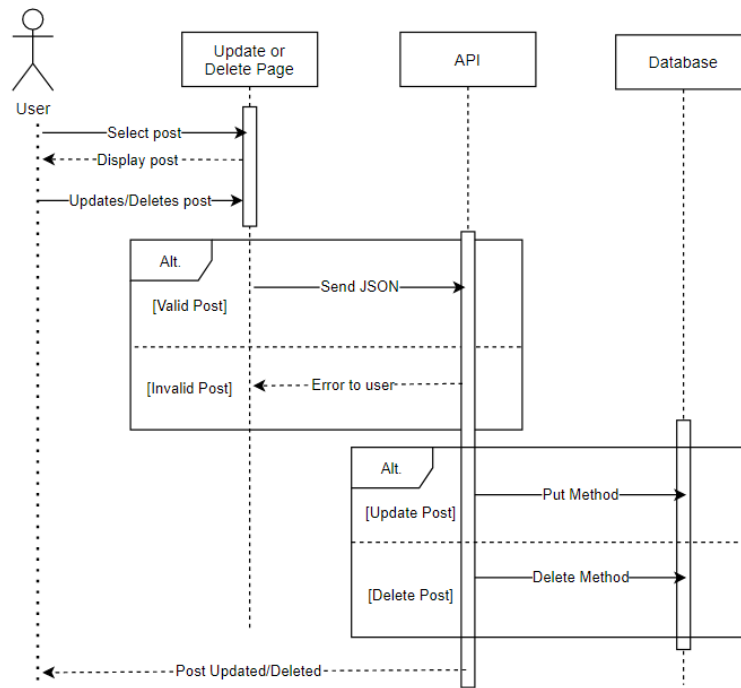
Home Page & Registration/Login



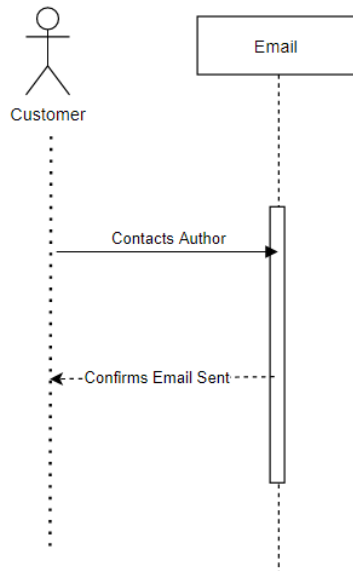
Create Announcement



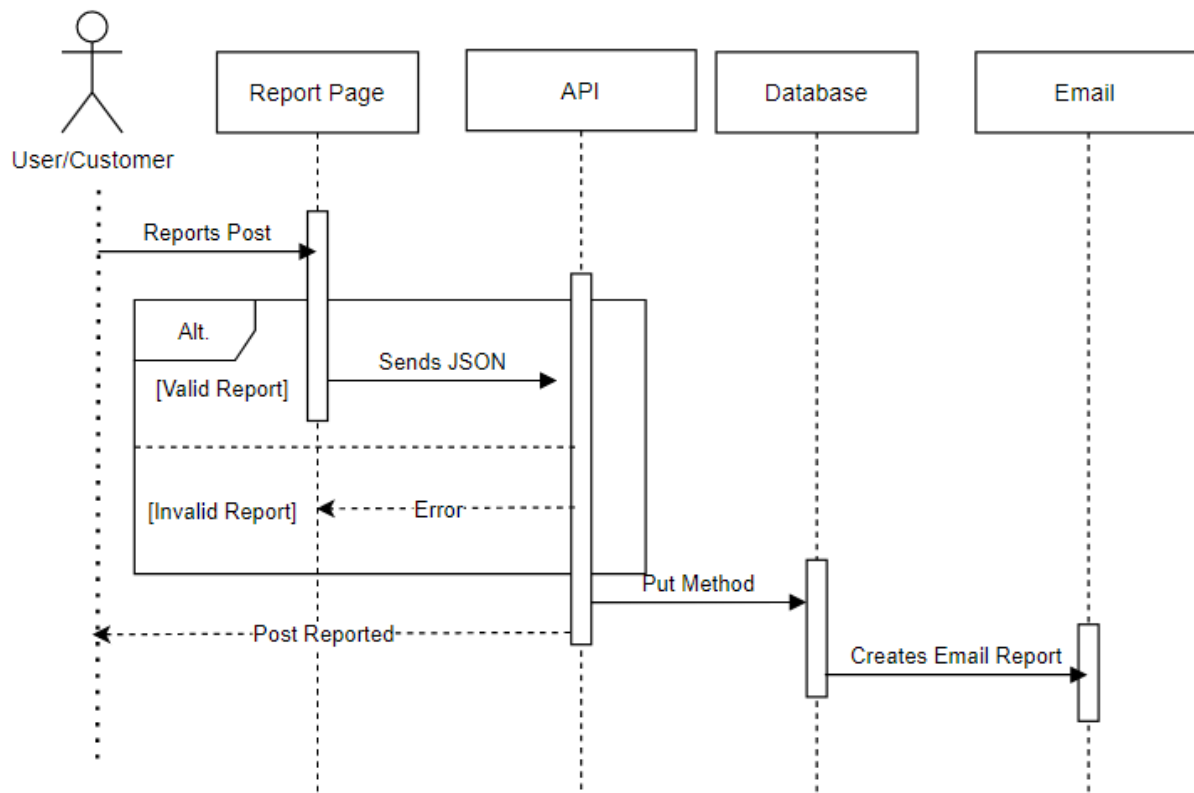
Update/Delete Announcement



Email Announcement Author



Report Announcement



Restrictions, Limitations, and Constraints

- Website load time may depend on the user's internet speed
- Web browsers limited to: Edge, Safari, Chrome, Firefox, Opera.
- Database Access Performance: Entity Framework can be slower when accessing DB.
- Server hosting plans may need to be high to support better performance when accessing the database with Entity Framework in production. This project is made by students with not much experience. Therefore, the server will be optimized to the best of our ability.

Configuration Management

To collaborate in our team, we are using:

- Google docs:
 - Reports for SRS/SPMP/Final Report.
 - Team notes for later use.
- Github/Git:
 - Frontend design & implementation code.
 - Backend design & implementation code.
- Zoom & Discord:

- o Communication.
 - o Project planning.
- Draw.io:
 - o Designing:
 - UML diagrams.
 - ERD diagram.
 - Use-case & Sequence diagrams.

User Project Resources

Minimal Hardware Resources

Development:

- Processor: x86 or x64.
- RAM: 1 GB (minimum), 4GB or higher (recommended)
- Hard disk: 2GB (minimum), 4GB or higher (recommended).

User:

- Any 32-bit or 64-bit Operating System.

Minimal Software Resources

Development:

- Visual Studio
- VS Code (Optional)
- .NET Core 3.1
- ASP.NET Core 3.1
- SQL Server Management Studio
- Postman

User:

- Any web browser (Internet Explorer NOT recommended)

Project Estimates

Information Domain Values							
Measurement Parameter	Count		Simple	Average	Complex		Total
Number of user inputs	20	X	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	=	80.00
Number of user outputs	3	X	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	=	15.00
Number of user inquiries	6	X	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	=	24.00
Number of files	0	X	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	=	.0
Number of external interfaces	2	X	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	=	14.00
Count=Total							133.00

Count Total

Complexity Weighting Factors
 // heading of the second table Rate each factor on a scale of 0 to 5:
 (0 = No influence, 1 = Incidental, 2 = Moderate, 3 = Average, 4 = Significant, 5 = Essential):

Question	0	1	2	3	4	5
1. Does the system require reliable backup and recovery?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2. Are data communications required?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
3. Are there distributed processing functions?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4. Is performance critical?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
5. Will the system run in an existing, heavily utilized operational environment?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6. Does the system require on-line data entry?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8. Are the master file updated on-line?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9. Are the inputs, outputs, files, or inquiries complex?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10. Is the internal processing complex?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
11. Is the code designed to be reusable?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
12. Are conversion and installation included in the design?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
13. Is the system designed for multiple installations in different organizations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
14. Is the application designed to facilitate change and ease of use by the user?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Total						
22.00						

Show Total of weighting Factor

The Function Points is: Show Function Points 115.71

Programming Language	LOC/FP (average)	Select
Assembly Language	320	<input type="radio"/>
C	128	<input type="radio"/>
COBOL	105	<input type="radio"/>
Fortran	105	<input type="radio"/>
Pascal	90	<input type="radio"/>
Ada	70	<input type="radio"/>
Object-Oriented Languages	30	<input checked="" type="radio"/>
Fourth Generation Languages (4GLs)	20	<input type="radio"/>
Code Generators	15	<input type="radio"/>
Spreadsheets	6	<input type="radio"/>
Graphical Languages (icons)	4	<input type="radio"/>

LOC/FP:

Software Project	a _b	b _b	c _b	d _b	Select
Organic	2.4	1.05	2.5	0.38	<input checked="" type="radio"/>
Semi-detached	3.0	1.12	2.5	0.35	<input type="radio"/>
Embedded	3.6	1.20	2.5	0.32	<input type="radio"/>

Effort (E) = a_b(KLOC)^{b_b} = Duration (D) = c_b(E)^{d_b} =

Project Resources

People

- Brittany
- Brandon
- George
- Edgar

Hardware

- Laptops, Desktops

Software

- C# with ASP.NET Razor
- SQL Server Management Studio
- Postman
- Draw.io
- Visual Studio
- Visual Studio Code
- Google Docs
- Github/Git
- Discord

Risk Management

Risk Table

Risk Name	Impact	Mitigation	Contingency Plan
Absenteeism (Sick, COVID)	Medium	If there is time, let the team member complete when available.	Other team members may assist.
Difficulty Using unfamiliar software/tools/ technologies	Medium	Read documentation, take a small crash course, trial and error, practice.	Use familiar software.
Project Changes	Medium	Only necessary changes are the ones to be allowed.	Project documents and design would have to be changed and redesigned.
Time Concerns	Low	Team will have to work on tasks faster, which implies more hours dedicated.	Consider a revaluation of tasks' length or size.
Errors/Exceptions/ Crashes	Medium	Error handling while coding.	Debug or redesign code structure.

Project Schedule

Project Tasks and Timeline Chart

Task ID	Task	Completion Date	Team Member(s)	Predecessor	Successor
A	SRS/SPMP	8 Feb 2021	Everyone		ALL
B	Prototype Presentation	15 Feb 2021	Brandon, Edgar	A	C
C	SDD	22 Feb 2021	Brittany, Edgar	A	D
D	Backend & Frontend	5 April 2021	Edgar	A	E
E	Testing	15 April 2021	Everyone	D	F
F	Final Report Guide	26 April 2021	Brandon, George	E	G
G	Final Presentation	26 April 2021	Everyone	ALL	

Staff Organization

Team Structure

Brittany	Database Design (ERD), Diagrams, Tester
Brandon	Documentation, Reports, Presentation, Tester
George	Diagrams, Documentation, Reports, Tester
Edgar	FullStack Programming (C#, ASP.NET Core Razor Pages, API design/testing, database implementation, HTML, CSS, JavaScript)

Management Reporting and Organization

Tasks are divided among team members. However, we will be contributing to other tasks as necessary (as team members gain experience in coding, they may contribute to the source code as well). We will be keeping in contact and meeting with each other to discuss progress in zoom and discord.

Data Design

Internal Software Data Structures

Program will create objects (models) for database entries, DTOs (Data Transfer Objects) for sending data to the user/client through our main API. Arrays & lists may be used for storing data collections from API calls.

API Driven Architecture (Globally)

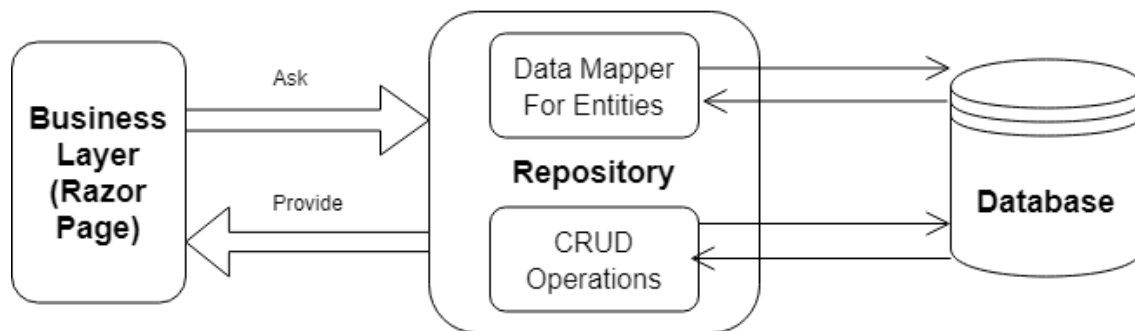
REST API will be used throughout the whole application to perform CRUD (Create, Read, Update, Delete) operations.

Temporary Data Structure

A temporary file may be used to store user announcement reports for administrators to check.

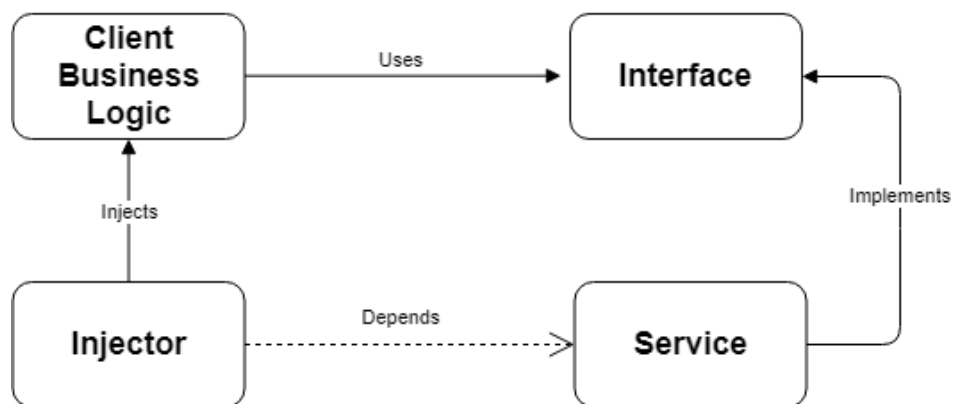
Repository Design Pattern

Repository Pattern



Dependency Injection Design Pattern

Dependency Injection Pattern



Database Description

Database contains information about a user (name, username, password), announcements (name, posted date, user, description, etc.), and report history.

Entity Relationship Diagram

(Next Page)

dbo. AspNetRoles

Id	int IDENTITY	PK
Name	nvarchar(256)	NULL
NormalizedName	nvarchar(256)	NULL AK
ConcurrencyStamp	nvarchar(max)	NULL

dbo. AspNetUserRoles

UserId	int	PK FK
RoleId	int	PK FK

dbo. __EFMigrationsHistory

MigrationId	nvarchar(150)	PK
ProductVersion	nvarchar(32)	

dbo. AspNetRoleClaims

Id	int IDENTITY	PK
RoleId	int	FK
ClaimType	nvarchar(max)	NULL
ClaimValue	nvarchar(max)	NULL

dbo. User

Id	int IDENTITY	PK
UserName	nvarchar(256)	
NormalizedUserName	nvarchar(256)	NULL AK
Email	nvarchar(256)	
NormalizedEmail	nvarchar(256)	NULL
EmailConfirmed	bit	
PasswordHash	nvarchar(max)	
SecurityStamp	nvarchar(max)	NULL
ConcurrencyStamp	nvarchar(max)	NULL
PhoneNumber	nvarchar(max)	NULL
PhoneNumberConfirmed	bit	NULL
TwoFactorEnabled	bit	
LockoutEnd	datetimeoffset(7)	NULL
LockoutEnabled	bit	
AccessFailedCount	int	
FirstName	nvarchar(500)	
LastName	nvarchar(500)	
DateRegistered	datetime2(7)	

dbo. AspNetUserLogins

LoginProvider	nvarchar(450)	PK
ProviderKey	nvarchar(450)	PK
ProviderDisplayName	nvarchar(max)	NULL
UserId	int	FK

dbo. AspNetUserClaims

Id	int IDENTITY	PK
UserId	int	FK
ClaimType	nvarchar(max)	NULL
ClaimValue	nvarchar(max)	NULL

dbo. AspNetUserTokens

UserId	int	PK FK
LoginProvider	nvarchar(450)	PK
Name	nvarchar(450)	PK
Value	nvarchar(max)	NULL

dbo. Announcements

Id	int IDENTITY	PK
Name	nvarchar(80)	
Type	nvarchar(50)	
Description	nvarchar(300)	
Image	varbinary(max)	NULL
CreatedAt	datetime2(7)	
UserId	int	FK

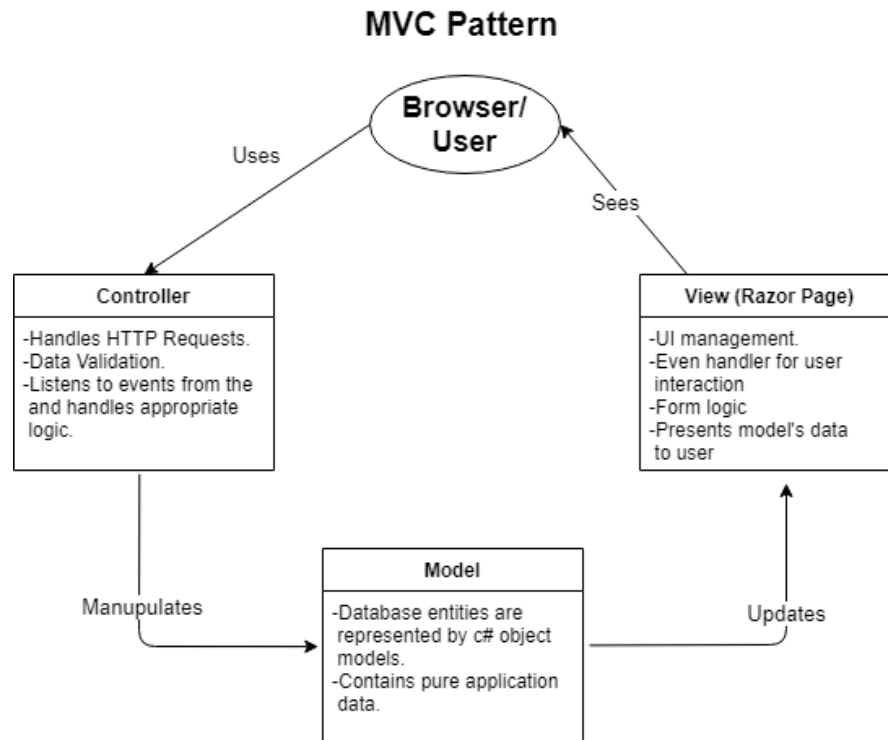
dbo. Reports

Id	int IDENTITY	PK
ReportCause	nvarchar(100)	
ReportDescription	nvarchar(300)	
ReportDate	datetime2(7)	
AnnouncementId	int	FK

Architectural and Component-Level Design

Architecture Diagram

MVC Design Pattern



User Interface Design

Description of the User Interface

The user interface is designed using HTML and CSS with Razor Pages in ASP.NET Core 3.1. CSHTML files are used for each razor page allowing to embed C# code into HTML when designing our different application pages.. Responsive design is handled by Bootstrap classes.

Main Pages:

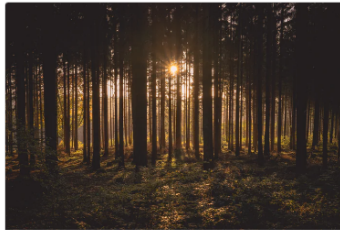
- Home Page (Announcements View)
- Register/Login Page
- Contact Page
- Report Page

- Create Announcement Page
- Manage Announcements Page (For editing/deleting)
- Update Announcement Page

Screen Images

Home Page

Welcome to the Community Board!



Announcement 1

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!

[Contact](#)


Announcement 2

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!

[Contact](#)


Announcement 3

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!

[Contact](#)


Register & Login Page/Form

Register

Please fill in this form to create an account!

☐ I accept the [Terms of Use & Privacy Policy](#)

[Register](#)

Already have an account? [Login here](#)

Login

Login to start posting or contact!

[Log In](#)

Don't have an account? [Register here](#)

Contact/Report Page

Contact (Author's Name Here)

Subject

Email Body

Send

Report Announcement

Report Reason

Report Body

Report

Create/Update Announcement Page

Create Announcement

Name

-- Announcement Type --

Picture: No file chosen

Description

Send

Update Announcement

Name

-- Announcement Type --





Picture: No file chosen

Description

Send

Manage Announcements

Manage Announcements for Specific User

Name	Description	Edit	Delete
Announcement 1	Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum simil ...		
Announcement 2	Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum simil ...		
Announcement 3	Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum simil ...		

Objects and Actions

- Home Page:
 - Announcement View: displaying data of the announcement.
 - Contact Button: in each announcement, the user will be able to contact the author.
 - Report Button: in each announcement, the user will be able to report the announcement.
- Register Page:
 - First/Last Name Input: The user will enter their name.
 - Username Input: The user will enter their username.
 - Email Input: The user will enter their email.
 - Password Input: Where the user enters their password.
 - Confirm Password Input: Users must fill this input for password validation.
 - Terms and Conditions Checkbox: User must agree with the terms to be able to register
 - Register Button: User click button to register.
 - Login here link: If the user has already an account, they might want to navigate to the login page
- Login Page:
 - Username/Email Input: The user will enter their username or email to be able to login.
 - Password Input: Where users enter their password for validation and access to their account.
 - Login Button: User click button to login.
 - Register Here Link: If the user does not have an account they might want to create one using this link.

- Contact/Report Page:
 - Subject/Report Reason Input: Input for users to define the subject of email or reason of reporting the announcement.
 - Email/Report Body Textarea: User will type the body of their text here.
 - Send/Report Button: User button to send the request to the server.
- Create/Update Announcement Page
 - Name Input: Announcement name.
 - Announcement-Type Dropdown: User will specify the type of announcement this is.
 - Upload Picture File Input: User input to select image (PNG, JPEG).
 - Description Textarea: Input for describing the announcement.
 - Post Button: Button to send the data to the server.
- Manage Announcements Page:
 - Table: This table shows the name, description, and 2 buttons.
 - Edit Button/Icon: Button to let the user edit their announcement.
 - Delete Button/Icon: Button to let the user delete their announcement.

Interface Design Rules

- Simple and easy to use user interface.
- Responsive design.
- Easy to navigate through pages.

Components Available

- HTML:
 - Forms (For sending requests/responses to server)
 - Inputs (Provides user interaction)
 - Buttons (Clickable control)
 - Labels (For description of components)
 - Textareas (For user input)
 - Other HTML tags (select, sections, headers, footers, navs, etc.)

Restrictions, Limitations, and Constraints

- Mobile view might be a problem. Responsiveness must be achieved when designing pages.
- REST API response time. API should have good performance.
- Web Page load speed.
- User internet's connection.

Testing Issues

Classes of Tests (See Testing Types for further details)

- White Box Testing: testing where design/implementation is known to the tester.
- Integration Testing: test components work together in group as expected.
- API/Performance Testing: test API requests/responses and database access.
- User Interface Testing: test UI usability.

Expected Software Response

The software will pass or fail the tests.

Performance Bounds

- Server performance issues are expected.
- Server crashes are definitely expected during development.
- RAM/in-memory database access performance/speed issues expected.

Identification of Critical Components

No critical components are identified as of now.

Packaging and Installation Issues

A browser is required to be able to open the webpage (any browser).

Testing Types

White Box Testing

Instead of unit testing, we used white box testing. After a component or class method was completed, it was tested and we made sure it worked. If it didn't, depending on the error, debugging was taken place in order to find the error and fix it at the moment.

Integration Testing

Individual modules of the REST API were combined and tested as a group. This mostly consisted of testing the two main Controllers (which handle HTTP requests/responses) and made sure we had a test for each possible output (such as Not Found, Unauthorized, OK, and BadRequest responses). When writing out the tests, xUnit, Fluent Assertions, and HttpClient C# libraries were used, which made it a lot easier to test. Also, we used the AAA (Arrange-Act-Assert) pattern when writing tests.

See test results in picture next page.

Test Explorer	
<div> <div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div>18</div> <div>18</div> <div>0</div> </div> <div> <div></div> <div></div> <div></div> </div> </div> <div>Search Test Explorer</div>	
Test	Duration
CommunityBoard.IntegrationTests (18)	19.4 sec
CommunityBoard.IntegrationTests (18)	19.4 sec
AnnouncementsControllerTests (12)	3.7 sec
Create_ReturnsOk_WhenCreatedSuccessfully	105 ms
Create_ReturnsUnauthorized_WhenNotLoggedIn	51 ms
Delete_ReturnsBadRequest_WhenUserDoesNotOwnIt	207 ms
Delete_ReturnsNoContent_WhenDeleted	2.5 sec
GetAll_WithAnnouncements_ReturnsList	118 ms
GetByName_ReturnsOkButEmptyResponse_WhenDoesNotExist	99 ms
GetByName_ReturnsOk_WhenExists	125 ms
GetFromUser_ReturnsOk_WhenUserIsAuthorized	132 ms
GetFromUser_ReturnsUnauthorized_WhenUserIsNotAuthorized	108 ms
Get_ReturnsAnnouncement_WhenExist	102 ms
Update_ReturnsBadRequest_WhenWrongIdPassed	96 ms
Update_ReturnsOk_WhenUpdatedSuccessfully	111 ms
ReportsControllerTests (6)	15.7 sec
Create_ReturnsOk_WhenCreated	267 ms
Delete_ReturnsForbidden_WhenUserLoggedIn	2.4 sec
Delete_ReturnsNoContent_WhenAdminLoggedIn	2.8 sec
GetAllFromAnnouncement_ReturnsOk_WhenAdminLoggedIn	3 sec
Get_ReturnsForbidden_WhenUserLoggedIn	3.6 sec
Get_ReturnsOk_WhenAdminLoggedIn	3.6 sec
Group Summary CommunityBoard.IntegrationTests Tests in group: 18 ⌚ Total Duration: 19.4 sec Outcomes <div>18 Passed</div>	

API Testing with Postman

When testing the API, we used Postman to perform all http requests such as GET, POST, PUT, DELETE. We made sure data was in JSON format and that data was being stored and retrieved from the database accordingly. We also tested and made sure user roles were giving correct http responses (i.e. users should get Forbidden or Unauthorized http status code when trying to view reports, only admins are allowed to do so).

API Endpoints:

CommunityBoard
V1
Announcements
GET GetAll
POST Create
GET Get
DEL Delete
GET GetAnnouncementsFromUser
PUT Update
GET GetByName
Identity
POST Register
POST Login
POST Refresh
GET GetUser
Reports
GET GetAll
GET Get
POST Create
DEL Delete
GET GetAllFromAnnouncement

GET

https://localhost:5001/api/v1/announcements

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
-----	-------	-------------	-----------

Body

Cookies

Headers (4)

Test Results

Status: 200 OK Time: 214 ms Size: 1.09 MB Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    {
3      "id": 2026,
4      "name": "Test Email Here Too",
5      "type": 0,
6      "description": "Can Test the email here as well!",
7      "image": null,
8      "createdAt": "2021-04-12T20:07:00.8161503",
9      "userId": 1,
10     "user": null
11   },
12   {
13     "id": 2025,
14     "name": "Test Email Here",
15     "type": 1,
16     "description": "For Email Testing",
17     "image": null,
18     "createdAt": "2021-04-12T20:06:36.1940169",
19     "userId": 2,
20     "user": null
21   },
22   {
23     "id": 2024,
24     "name": "Test Masonry",
25     "type": 4,
26     "description": "Some description Some description Some description Some description Some description Some description Some descriptionSome descriptionSome description",
27     "image": null,
28     "createdAt": "2021-04-05T22:02:31.9930606",
29     "userId": 1,
30     "user": null
```

[illegible]

User/UI Testing

When testing the UI, we made sure all buttons, inputs, and elements worked and performed the expected result. Each one of us tested the website, and in Edgar's case, he asked out some friends and family members to test the website as well. The following chart was used for test cases.

Testing Chart

Test Case #	Description	Input	Expected Output	Actual Output	Pass/Fail
1	Login using valid user name	User name	Redirect to Homepage	Same as Expected	Pass
2	Register with first name/last name	Main characters no special characters or numbers	Redirect to Login	Same	Pass
3	Register with first name/last name	With special characters and numbers	Redirect to Login	Same	Pass
4	Register with first name/last name	Leave blank	Show "required" message	Same	Pass
5	Register with email	Email includes @website.com	Redirect to Login	Same	Pass
6	Register with email	Does not include @	Show invalid email error	Same	Pass
7	Register with email	Does not include .com .net .edu	Show invalid email error	Redirect to Login	Fail
8	Register with email	With special characters	Show error	Same	Pass
9	Register with email	Leave blank	Show "required" message	Same	Pass

10	Terms and Conditions Checkbox	Check terms and condition box	Redirect to Login	Same	Pass
11	Terms and Conditions Checkbox	Leave uncheck	Show "required" message	Same	Pass
12	Created password with less than 8 characters	Less than 8	Show error	Same	Pass
13	Created password with more than 20 characters	More than 20	Valid Password	Same	Pass
14	Created password with between 8 and 20 characters	Password between 8-20 characters	Valid Password	Same	Pass
15	Contact page Subject field	Subject under 100 characters	Valid Subject	Same	Pass
16	Contact page Subject field	Subject over 100 characters	User input blocked	Same	Pass
17	Contact page Subject field	No characters in subject line	Required Message	Same	Pass
18	Contact page email body field	Type more than 500 characters	User input blocked	Same	Pass
19	Contact page email body field	Type less than 500 characters	Valid email body	Same	Pass
20	Contact page email body field	Leave blank	Required Message	Same	Pass
21	Report announcement page Report Reason field	Under 100 characters	Valid Report	Same	Pass
22	Report announcement page Report Reason field	Over 100 characters	User input blocked	Same	Pass

23	Report announcement page Report Reason field	No characters in subject line	Required Message	Same	Pass
24	Report announcement page Report Body field	Type more than 300 characters	User input blocked	Same	Pass
25	Report announcement page Report Body field	Type less than 300 characters	Valid Report Body	Same	Pass
26	Report announcement page Report Body field	Leave blank	Required Message	Same	Pass
27	Create & Update announcement page Name field	Under 80 characters	Valid Name	Same	Pass
28	Create & Update announcement page Name field	Over 80 characters	User input blocked	Same	Pass
29	Create & Update announcement page Name field	No characters in subject line	Required Message	Same	Pass
30	Create & Update announcement - Announcement type field	Non selected field	Required Message	Same	Pass
31	Create & Update announcement - Announcement type field	Selected field	Valid Type	Same	Pass
32	Create & Update announcement - Picture field	No picture	Valid Announcement	Same	Pass
33	Create & Update announcement - Picture field	With picture	Valid Announcement	Same	Pass
34	Create & Update announcement - Announcement description field	Description of announcement less than 300 characters	Valid Announcement	Same	Pass
35	Create & Update announcement - Announcement	Description of announcement more than 300	User input blocked	Same	Pass

	description field	characters			
36	Create & Update announcement - Announcement description field	Leave blank	Required Message	Same	Pass
37	Create & Update announcement - Maximum 1080 x 1080 pixels on picture	Upload a picture with more than 1080 x 1080 pixel	Adjust picture to size	Same	Pass
38	Create & Update announcement - Minimum size 600 x 600 pixels	Upload a picture with less than 600 x 600 pixels	Adjust picture to size	Same	Pass
39	Create & Update announcement - upload multiple pictures at once	Upload multiple pictures at once	System won't let multiple	Same	Pass
40	Create & Update announcement - upload single pictures	Upload single picture	Valid picture	Same	Pass
41	Manage Page - Edit option button	Click	Redirects to Update Page	Same	Pass
42	Manage Page - Delete option button	Click	Deletes Announcement	Same	Pass
43	Upload Image Type	Non-image file	Invalid Image Error	Same	Pass
44	Upload Image Type	Image type (JPEG, PNG)	Valid Image	Same	Pass
45	Sending email - receiver should be announcement author	Test email	Should receive email	Same	Pass
46	Sending email - sender should be logged in user	Test email	Should send email	Incorrect email	Fail

Maintenance Guide

Potential Enhancements

- Chat embedded in the website.
- Report management page in website (this is available in the API, but not on the website itself).
- Announcement log storage.
- Better work on the front end, making it look prettier.
- Better security to prevent CSRF attacks (usage of refresh tokens).
- More search options when searching for an announcement.
- User account management page. Let them change their name, password, email, username, etc.
- Add options to sign in with social media or other services (like GitHub).
- Admin pages to manage users (for example: being able to ban a user).
- API Pagination, filtering, validation.

Refactoring

- Simplifying methods (similar to using divide and conquer). Making them into smaller pieces and more readable, instead of long methods.
- Razor page model classes can have cleaner code in their main methods (Which are OnGet() & OnPost()). Rather than writing the whole page function in those 2 methods, we could break them up and make them look cleaner.
- Usage of more design patterns.
- Redesign or modify parts of API code to follow a more RESTful architecture.
- Reduce code duplication in integration tests by making methods for repeated code.

Lessons Learned

- Communication is important and could have gone better.
- Time management is key, balancing time between other classes/courses.
- We could've had more meetings than we did.

Appendix

References/Supplementary Information

<http://www.rspa.com/docs/Projectplan.html>

<http://groups.umd.umich.edu/cis/course.des/cis525/js/f00/gamel/cocomo.html>

<https://www.geeksforgeeks.org/mvc-design-pattern/>

<https://stackify.com/dependency-injection/>

<https://www.c-sharpcorner.com/UploadFile/b1df45/getting-started-with-repository-pattern-using-C-Sharp/>

<https://sqldbm.com/Home/>

Source Code

<https://github.com/Ragdety/Community-Board>

Production Environment

Website

<https://communityboard.azurewebsites.net>

API (Showing a few endpoints):

<https://communityboardapi.azurewebsites.net/api/v1/announcements>

<https://communityboardapi.azurewebsites.net/api/v1/announcements/1>

<https://communityboardapi.azurewebsites.net/api/v1/announcements/name=Another>

<https://communityboardapi.azurewebsites.net/api/v1/identity/users/1>

Website above is running a trial version of Azure, might be down after a few weeks.