Software Project Report
Express Yourself

Iyanuoluwa Akinyemi - aiaa790@uregina.ca
Melanie MacDonald - macdon6m@uregina.ca
CS 476

Table of Contents

1. Problem Definition

  We are in an age where the internet is a prevalent part of everyone's lives, in all age groups. One thing the internet does for us is connect us to other users all around the world. Many sites exist to facilitate these connections, using many different media to communicate. With this comes the desire to express ourselves to others; to stand out and share our thoughts with any who care to receive them. This is the value of pages like blogs, which provide the blog owner a place to express themselves to the world, with the ability to use text, photos, or videos for expression, comment threads to have conversations, and to have complete control over the content on their page. Blogs are valued by all age groups, from children to the elderly, and should be easily accessed and used by all ages with any level of technical experience.

  The goal of this project is to create a website that hosts any number of blogs that are created by users who have signed up to the site. These blogs offer the users their own page as an outlet for their free expression in a space that is completely personalized to their specifications. The user can create blog posts, upload various media such as photos and videos, and change the appearance of their site in terms of overall placement of objects on the page, wallpaper, color schemes, and text styling. The blogs can be viewed by any user, whether they have signed up to the website or not, and authenticated users can comment or react to posts and comments. The site will provide this as well as give users the ability to browse through other user's blogs and view certain statistics that allow curious blog hosts to see the popularity of their space in comparison to others.

  The motivation of this site is to create an outlet for free expression that is easy enough to use and understand that people of all age groups can use it, including children, in a safe setting content can be deleted by the blog owner if necessary. There are many services that allow people to express themselves online, but few are so personalized or accessible for all age groups, and many are complicated and require a lot of maintenance and time on the part of the user to update their content. This site will hopefully simplify this process with easy to use interfaces where users can easily drag-and-drop elements to arrange their webpage, and edit various aspects of those elements for detailed customization.

2. Economic Feasibility Study

   The type of blog-hosting website we are creating is not a new concept, but is intended to be an improvement of existing solutions. There are multiple sites on the web that provide an outlet for expression, some actually offering an individualized place for their customer, and some mainly offering threads with a fixed format in which people can get together and talk. Of these different sites, there are certain features that we find to be successfully used by one site, but lacking by another. As such, we have provided a list of sites that most accurately display all the features that we wish to include in our web app, as well as those that we wish to improve, and our final product will be composed of all the desired requirements pooled from each of those sites. The examples of existing solutions that will be briefly compared against the benefits of our application are Facebook (https://www.facebook.com/), Tumblr (https://www.tumblr.com/), and Weebly (https://www.weebly.com/ca).

   Facebook is not a blogging platform, however it does have aspects of communication and a personalized space that somewhat resembles what we are trying to accomplish. The Facebook "wall" belongs to a specific user, and on it they can set a cover photo, display photo, information about themselves, as well as create posts. These posts can contain text, photos, videos, or articles, and on these posts people can comment, reply, and react to one another. Our blog seeks to have a similar system for commenting and reacting. However we will provide more personalization in terms of the appearance of a person's space, such that they can change all sorts of aspects of the layout and overall appearance of the page to their liking, whereas the Facebook wall is largely static. Additionally, Facebook requires authentication in order to view another's wall, whereas our website will not require an account to view another's blog.

   Another website that more closely resembles the app we are trying to create is Tumblr. Tumblr is a blog-hosting site that provides users with a way to customize the appearance of their space, and use it to display various media such as text and images. Users can like and comment on each other's posts, as well as browse all blogs on the site. Our blog-hosting site will provide a personalized space similar to this, with a similar system to view and search through all blogs. However, we will improve on the liking and commenting system, as for Tumblr posts they are integrated into one feed, which can be confusing and cluttered. We will provide more options to

choose from for reacting than just "liking" a post, and will separate the comments in threads and reactions into separate entities for easy viewing.

The last example is Weebly, which provides users with domain names for which to build their own websites based on a variety of templates and themes as starting points. These sites are highly customizable, having a "drag and drop" mechanism to add elements into the page, with the ability to upload media, comment, and reply to content. We intend to use a "drag and drop" method of customization similar to this, with a basic template as a starting point. However, unlike Weebly we will provide a space for the user that is free of cost, with an easy way to browse all hosted content. Additionally, Weebly requires that the user navigate away to view comments. Instead, we will show these on the same page where they can be easily viewed without having to leave the page

A summary of these sites and the improvements to be made in our app are given below.

Name	Relevant Features	Improvements by Our App
Facebook	- individualized Facebook "wall" - ability to post different media such as text, video, and images - ability to comment, reply, and react to content - personalization of cover photo and display picture visible on wall	- more personalization and freedom in terms of the layout of the page - more personalization for text style and backgrounds - authentication not required to view content
Tumblr	- customization of appearance of webpage - ability to display various media such as text, and images - ability to like posts - ability to comment on posts - can browse all blogs	- more clear commenting and replying system - ability to perform more reactions than just "like"
Weebly	- build a website with a professional appearance based off of a variety of templates and themes - ability to drag and drop elements onto the page, highly customizable appearance of webpage in terms of background images and text style - ability to upload images and videos - ability to comment and reply to content	- provide services free of cost - provide easy way to browse all hosted blogs - be able to view comments and reactions without having to leave the page

3. Software Requirements Specification

a) Implemented Functional Requirements List
<u>For any user:</u>
- Signup - a user can sign up for the site, which grants them the privilege to create a blog, comment on a blog, or react on a blog
- View Blog List - a user can see a comprehensive list of all the blogs hosted by the site, which is searchable
- View Blog - a user can view any blog to see all the content the owner has specified
- View Statistics - a user can see the statistics of a certain blog, as well as a table of all blog statistics for easy comparison
- Create Blog - an authenticated user can create a blog with a certain title, display picture, and description that appears on the "Blog List" page

<u>For user: Viewer:</u>
- Comment - authenticated users can leave a comment on any thread, which will display their text along with their username to identify them
- React - authenticated users can choose from a series of 9 reactions, which will aggregate underneath the element (post/image/video) being reacted to. There is only one reaction allowed per user per element; previous reactions for that user are replaced.

<u>For user: Owner:</u>
- Add post - blog owner can add a text post to their blog, consisting of a title and content, in which the title and content can be changed, as well as the font color and background color of the element
- Add image - blog owner can add an image to their blog, which can be linked either from the collective image library, or using a url leading directly to the image
- Add video - blog owner can add a video to their blog, which can be linked either from the collective image library, or using a url from YouTube
- Add thread - owner can choose to append a thread to any element, in which other users can post comments
- Upload image - owner can upload an image to the collective image library from their own computer
- Upload video - owner can upload a video to the collective video library from their own computer, within reason in terms of size
- Resize element - any post/image/video can be resized by dragging its lower left corner, but is limited to certain bounds
- Drag element - any post/image/video can be moved around the webpage by clicking anywhere but the edges and dragging, but is limited to certain bounds
- Delete element - any image, post, video, or thread can be deleted. Any comments or reactions attached will also be deleted.
- Save changes - any changes to a blog can be saved, so they are made persistent for any who wish to view
- Modify blog appearance - owner can change the font of the entire blog, and the background color

b) Use Case Diagrams - External Roles

Use case for a Blog owner

Actor: Blog Owner

They are authenticated users and it is the front end of the webpage

Responsibilities:

Create and manage the editorial of the blog

Editing, approving and publishing content

Manage the blog page by moderating, reacting and responding to comments

Manage subscriptions (not sure)

Rebuild the site (altering the appearance of the blog)

Capabilities:

☐   create and edit a blog. The blog owner can edit the appearance of the blog.

☐   can view any blog posted on the webpage.

☐   can upload media(images, videos, links) to the blog.

☐   can add text posts and comment to the blog

☐   can react to any post and comments
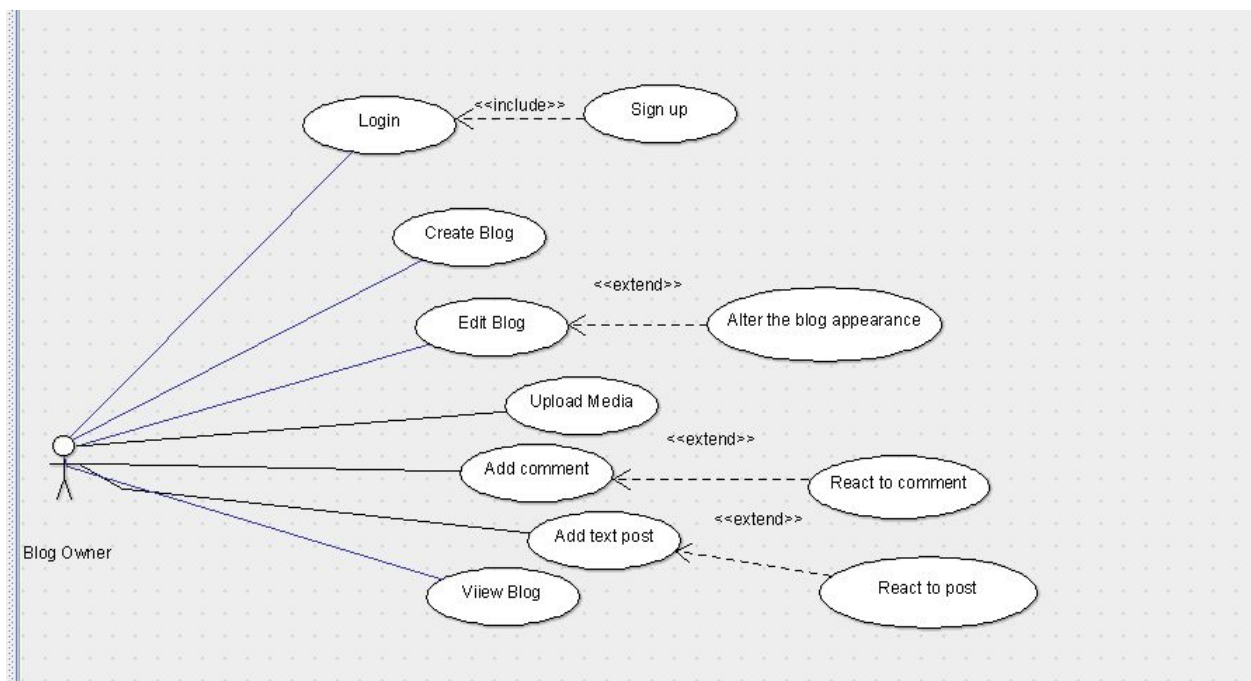
Use-case for Viewer user

Actor: Viewer user (they can both unauthorized[guest] or authorized users)

They are unauthenticated user and they are both the front end of the project

Responsibilities:

☐ can read comments, posts and write comments.

☐ can view a different blog.

☐ can upload media

Capabilities:

☐ can upload media(images, videos, links) to the blog.

☐ can add text posts and comment to the blog

☐ can react to any post and comments can block any form of contents
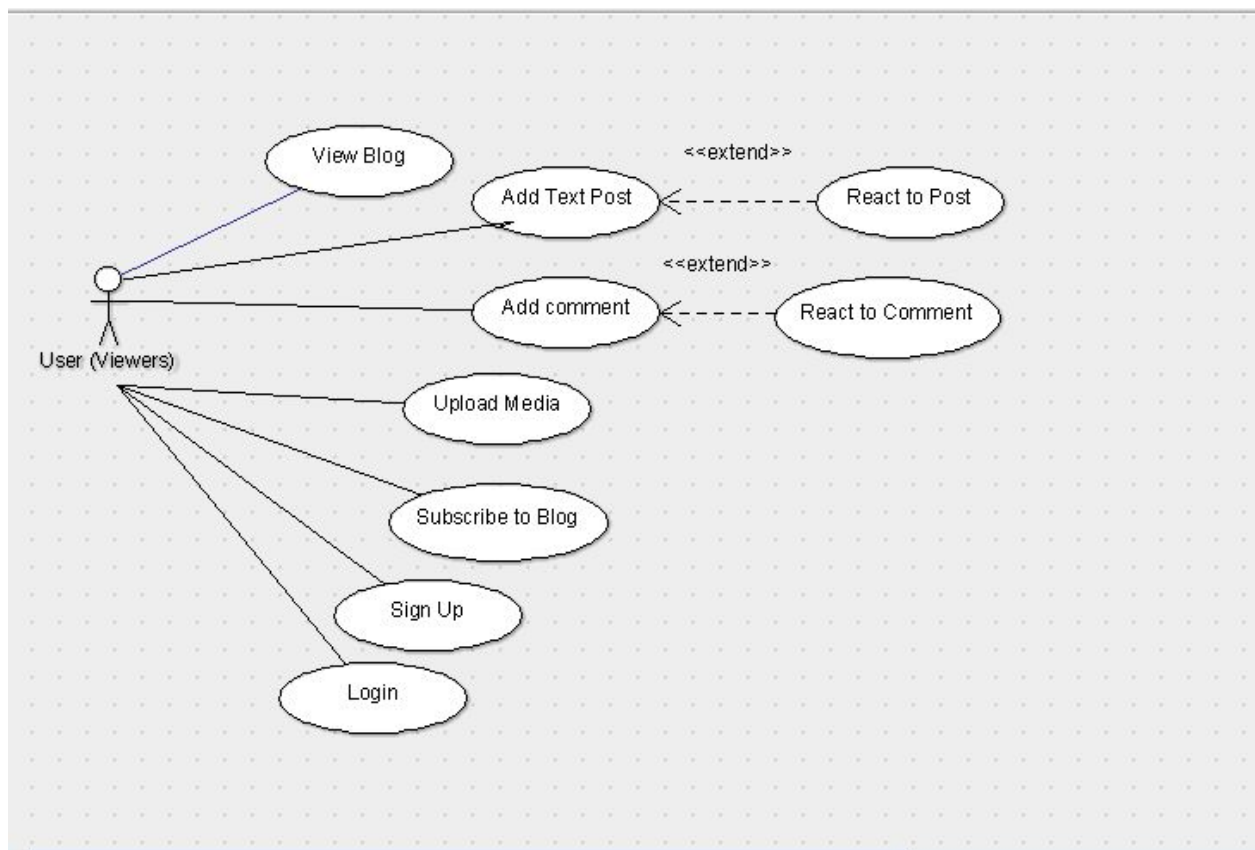
Action: "Edit Blog appearance"

Actor: Blog owner

Entry condition: the user must be authenticated and logged in.
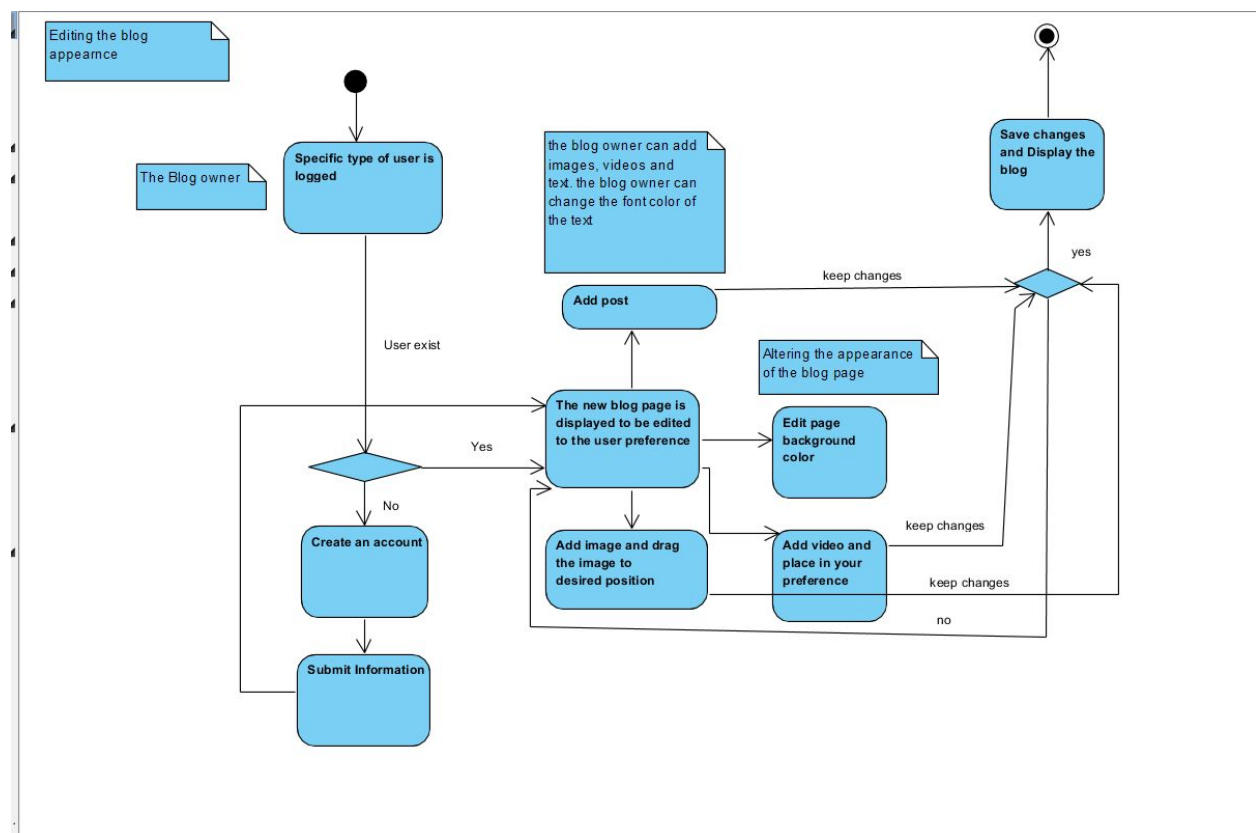
Primary Scenario:

the user must create an account and log in to the webpage. the user must then create a blog page on the webpage. The user can change the background picture, add or mute ambient music. the user can drag and drop the content to their preference.

Flow of events:

- ▢   Create a blog page.
- ▢   Altering the appearance of the blog.
- ▢   adding a background picture from the library or uploading a background image.
- ▢   Placing content on the blog

Post- Conditions: The blog is created, ready for other users to view it and the user becomes a blog owner.

Exit condition: Log out

Action: View mode - "Edit The Blog"

Actors: Blog owner, Guest (unauthenticated users)

Entry conditions: an authenticated user must log in and an unauthenticated user can use the webpage without logging in.
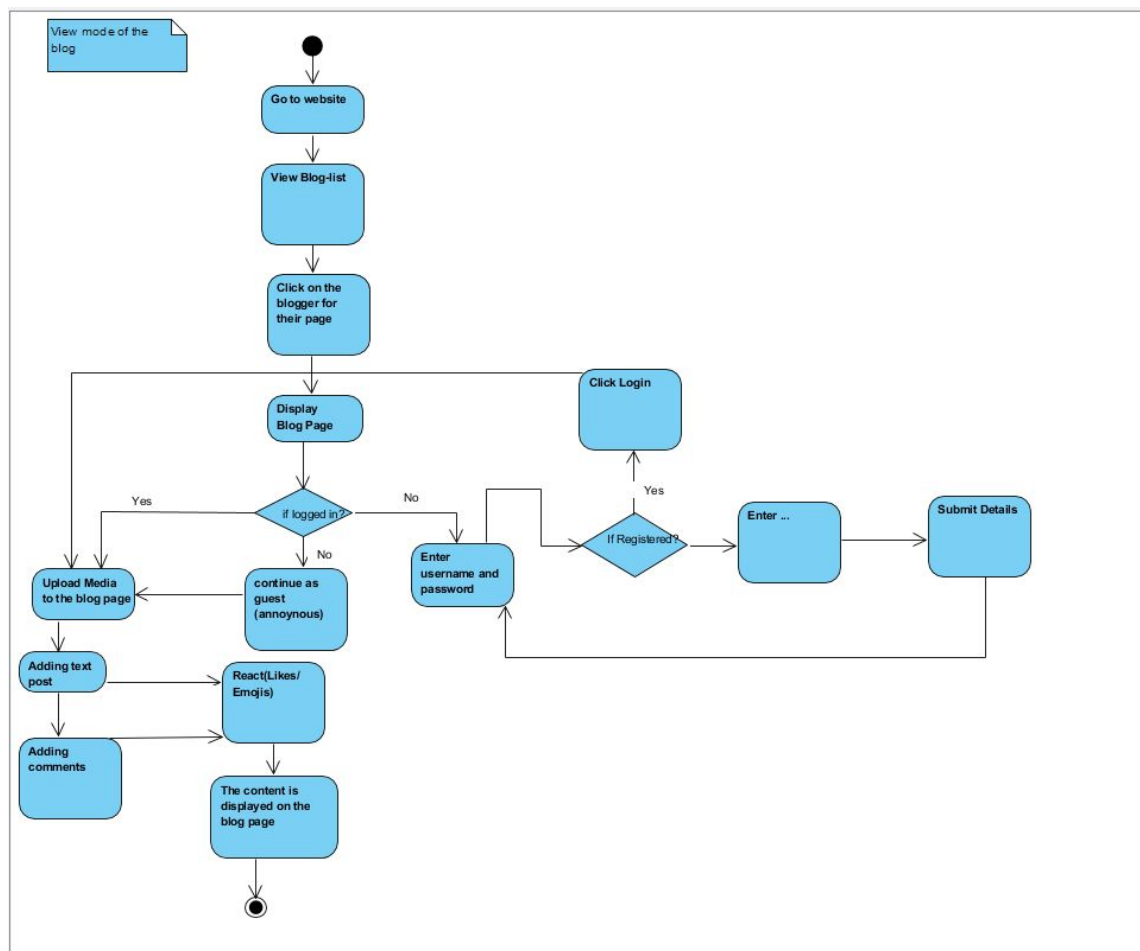
Primary Scenario:

A user must either login or visit a blog on the web page, the user must choose the "upload media" button to able to direct to the upload media page. To add text post and comments, the user must click on add comment or post icon and they will be directed to the page with a text box that allow the user the user to add text in different fonts, colours etc..

Flows of events:

- ☐ Upload media (pictures, videos, links)
- ☐ adding text post to a blog
- ☐ Adding comments
- ☐ Reacting to both posts and comments.

Post-conditions: The content (media, posts, comments) uploaded is displayed on the blog page. The content blocked by the blog owner is blocked from other user's sight.

Exit condition: authenticated user can log out of the blog

Action: "View Statistic"

Actor: Viewers

Entry condition: Both authenticated unauthenticated user [Viewers] can use the webpage without logging in.
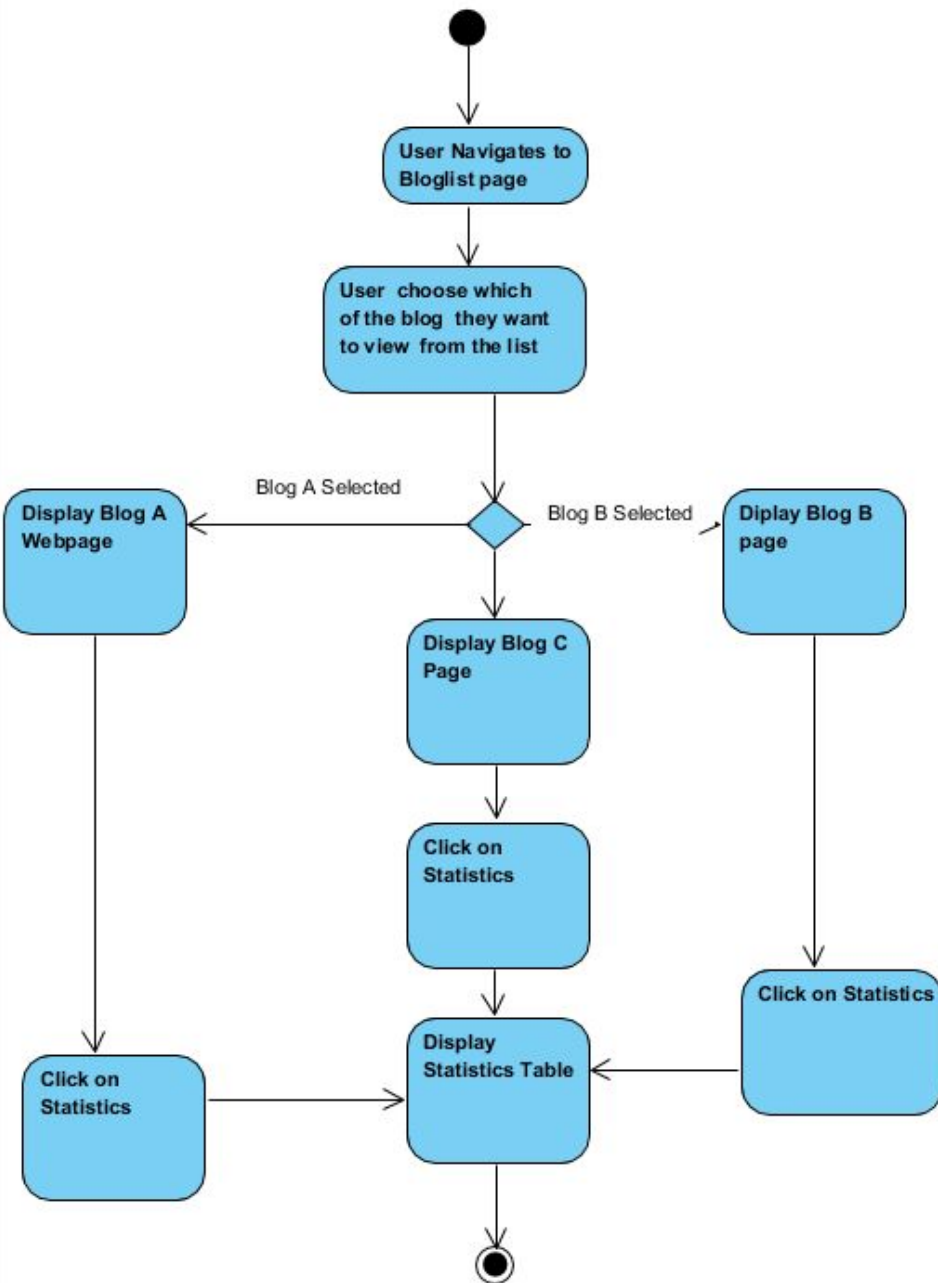
Primary Scenario: User must either login or visit a blog as a guest on the web page, the user must choose the "view blog list " button to able to direct to the page that contains information about different bloggers. Then the user views the blog and at the bottom of the blog webpage, the user must click on "Statistics" and they will be directed to the page that contains the number of comments, commenter and reaction of the blog page.

Flows of Events: View Statistics table

Post Conditions: The content in the table is displayed on the webpage.

Exit Condition: Users can go back to the previous page (the blog page) or they can log out of the blog.

User Navigates to
Bloglist page

User choose which
of the blog they want
to view from the list

Blog A Selected

Display Blog A
Webpage

Blog B Selected

Diplay Blog B
page

Display Blog C
Page

Click on
Statistics

Click on Statistics

Click on
Statistics

Display
Statistics Table

Action: Adding Comment

Actor: Blog owner, Viewers(both registered user and guest user)

Entry condition: Blog owner must be logged in, Viewers can be both registered user or unregistered user(guest)

Primary Scenario: Blog owner must have an account and log in to the webpage. The user can view blogs on the webpage. Blog owner can comment on a post or image and they can reply a comment with another comment.   The viewer user can view a blog and can comment on the image, posts on the webpage.

Flow of Events:

☐   Add a text post

☐   Add a comment/ blog owner can reply a comment with a comment.

Post-conditions: The comment added is displayed on the blog page.

Exit Condition: The users can either log out or continue to view blogs on the webpage.
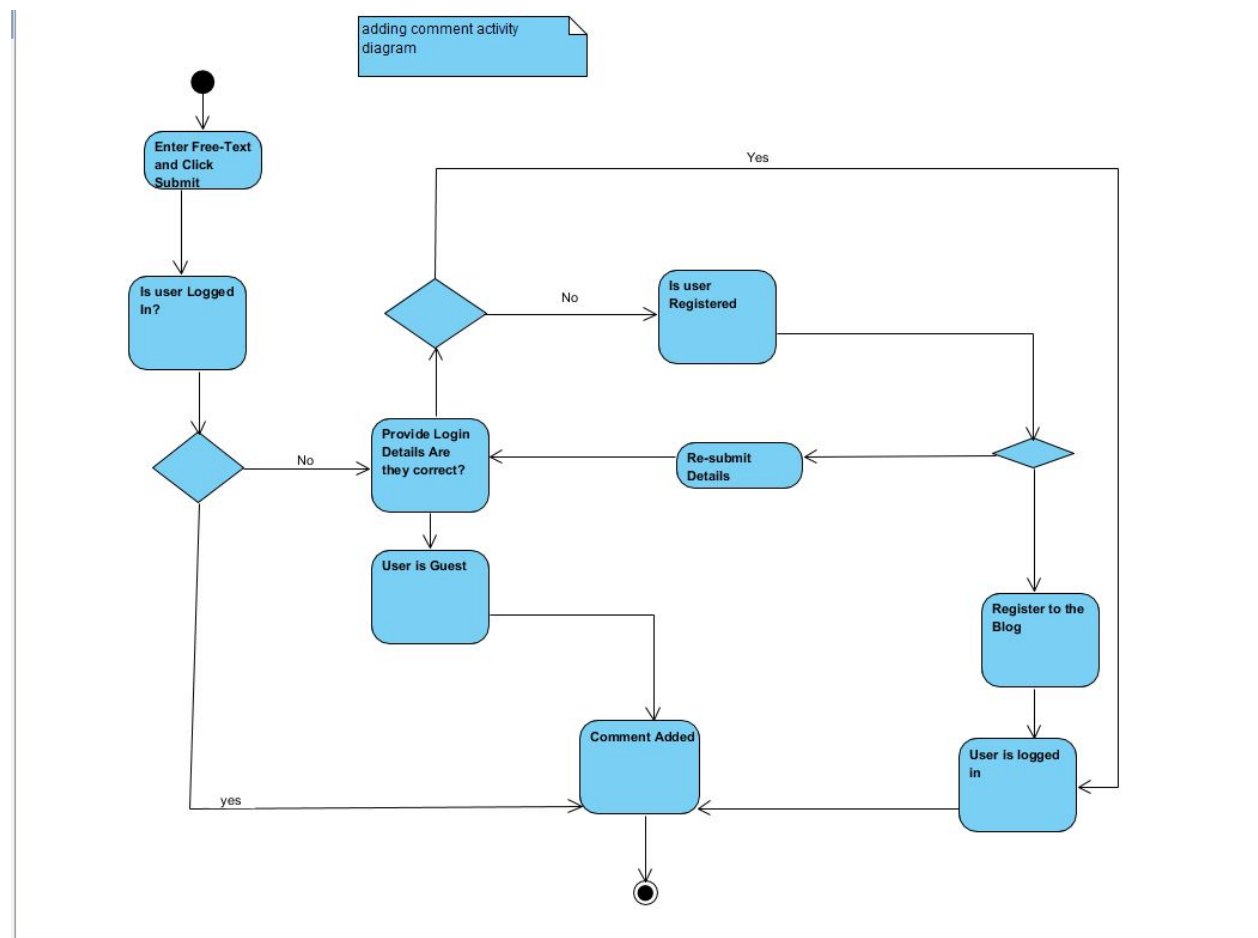
c) Software Qualities

Correctness:
The completeness and consistency of the information entered by the users. The right media (images, videos) is uploaded properly. The website should detect when there are broken media on the blog. Correctness is because we want to have a website that reliably displays a variety of persistent information across many users. Also, the functional requirements are captured through the correctness quality. The program should be able to successfully log the blog owner into the website. If the user does not have an account, the software should be able to create a successful account and it should allow the user to use certain functions of the website if they decide to not create an account. The application should recognize the placement of objects (how the blog was designed) on the blog website. The website should recognize when posts, comments, and reactions are added to the blog and identify who posted the comment. It should be able to correctly log the user out of the site. Statistics reported on the site should also be correct.

Time-Efficient:
The built-in command should execute at the right time which allows the users to use little effort to navigate the website (i.e the website should be execution efficient). The media should load very fast so that the users will be able to view it. Videos, images and other content additions or modification should be done within a good duration. The website should be able to store every media uploaded to the blog on the database (i.e the website should be storage-efficient)

Robustness: The program should be able to handle incorrect entry of a password, username during login and clearly show the user that. The application should be able to deal with invalid data being entered in the sign-up page. The user should be able to access blogs without logging in, while preventing people who aren't the owner from modifying its contents. The application should be able to handle special characters in the input for the text boxes for posts without causing an error in storage. The application should able to handle a large number of people accessing the webpage without causing impaired access for other users.

User-friendliness:
This simple website blog allows the user to quickly and efficiently create and edit their own blog. The tools used for altering the design of the blog''s appearance is easy to use and understand without explanation or tutorial. Inappropriate content (i.e. rude comments) can be removed at ease by the blog owners through deleting and recreating threads.

4. Design Specification Document

a) Logical Software Architecture

N-layer architecture (4 layer architecture):
This consist of the presentation layer, business logic and the database layer. The business logic layer is divided into two component and object oriented software architecture will implemented. The N-layer architecture provides maintainability of the program because of how each layers are independent of each other(isolation), changes can be made in one layer of the architecture and it does not impact other components in another layers. N-layer architecture provides high cohesion and low coupling which makes the program to be good to use and maintainable. N-layer architecture allows for the ability to reuse the functionality of each layer. N-layer architecture provides encapsulation for data, it provides information on the data types and implementation during designing of this program.
Object oriented architecture style is used in the business logic layer, the divided into two: the presentation layer transfer all the data to and store it, the second part convert the data transfer (requested by the client) from the presentation to the format compatible to the form in the database and transition it to the database layer.

b) Design Patterns

Factory Design Pattern
This design pattern is used to help create multiple related objects by reusing code between them based on those similarities. This is helpful for us in our application because we are populating a webpage with multiple types of elements, namely a post, image, video, thread, or reaction. For each of these elements, certain attributes are shared between them. Namely, each element needs to have a function to buildHTML, such that the HTML that will be built on the web-page for each element will be formed. The implementation for this function differs for each of these elements, so they each override buildHTML in the Factory. Each element also has to have its size and coordinates on the webpage defined, and the implementation to determine these values is exactly the same for each, so it makes sense to have this implemented in the Factory while each child will inherit and reuse the same function. Each element also has custom settings as well, that set them apart, and these functions are implemented separately. Having this Factory means code reuse and clarity when reading and understanding the code.

Factory Design Pattern Diagram
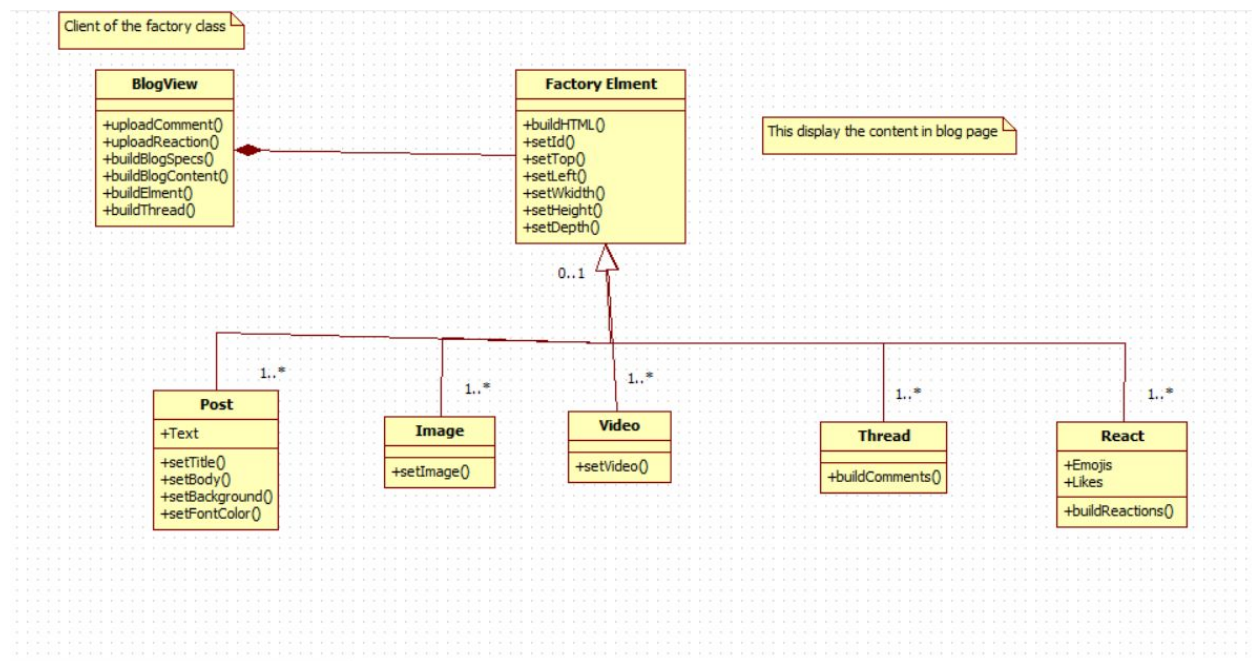
Factory Class:

```
class Element:
    def __init__(self, row):
    self.id = self.setId(row)
    self.top = self.setTop(row)
    self.left = self.setLeft(row)
    self.width = self.setWidth(row)
    self.height = self.setHeight(row)
    self.depth = self.setDepth(row)

    #abstract function, each child will override
    def buildHtml(self):
    pass
    def setId(self, row):
    return row[1]
    def setTop(self, row):
    return row[2]
    def setLeft(self, row):
    return row[3]
    def setWidth(self, row):
    return row[4]
    def setHeight(self, row):
    return row[5]
    def setDepth(self, row):
    return row[6]
```

Observer Design Pattern:

Observer design pattern allows multiple observer objects or views to connect with the subject for a change of notification. when the state of each of the concrete subject changes, it notifies the observer and the views are updated by the observer. And the subject keeps track of all the concrete observer. Observer pattern allows one-to-many dependency between the observer (objects) so that when one observer (object) changes state, all its subject (dependents) are notified and automatically updated. Observer pattern makes the program loosely coupled and thereby increasing the reusability of the code.

When a new user registers and becomes a "blog owner" after creating a blog, the subject(both SQL database and Blob storage) is notified. When the blog owner makes a changes on the blog view page; for example on blog view, when the user wants to get or upload image or video the blog view queries the subject for information, the blob storage is notified and the subject performs the task either by getting the media from the library for user or uploading the media to the storage and displaying the media in the blog webpage.

Observer Class:

#interface class, no implemented functions

```
class Observer:
        def __init__(self):
        self.html = self.buildContent()

        #abstract, all children override
        def buildContent(self):
        pass
```

The observer class builds the content of the blog, queries the subject for information.

Subject Class:

```
class Subject:
        def __init__(self):
        self.connection = self.connect()

        #abstract, all children override
        def connect(self):
Pass
```
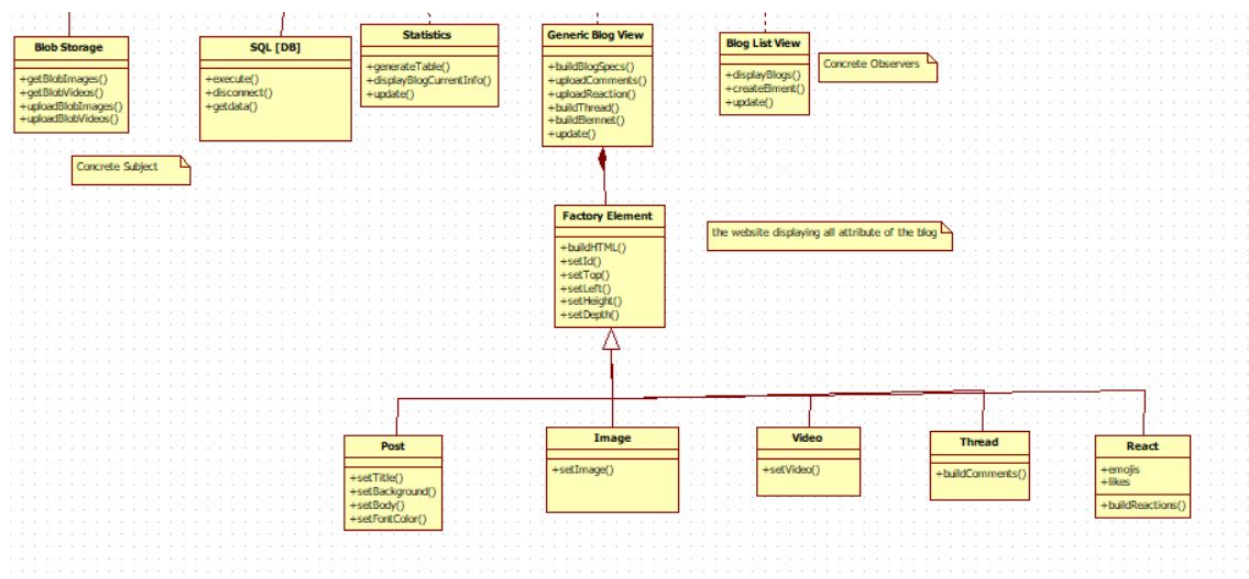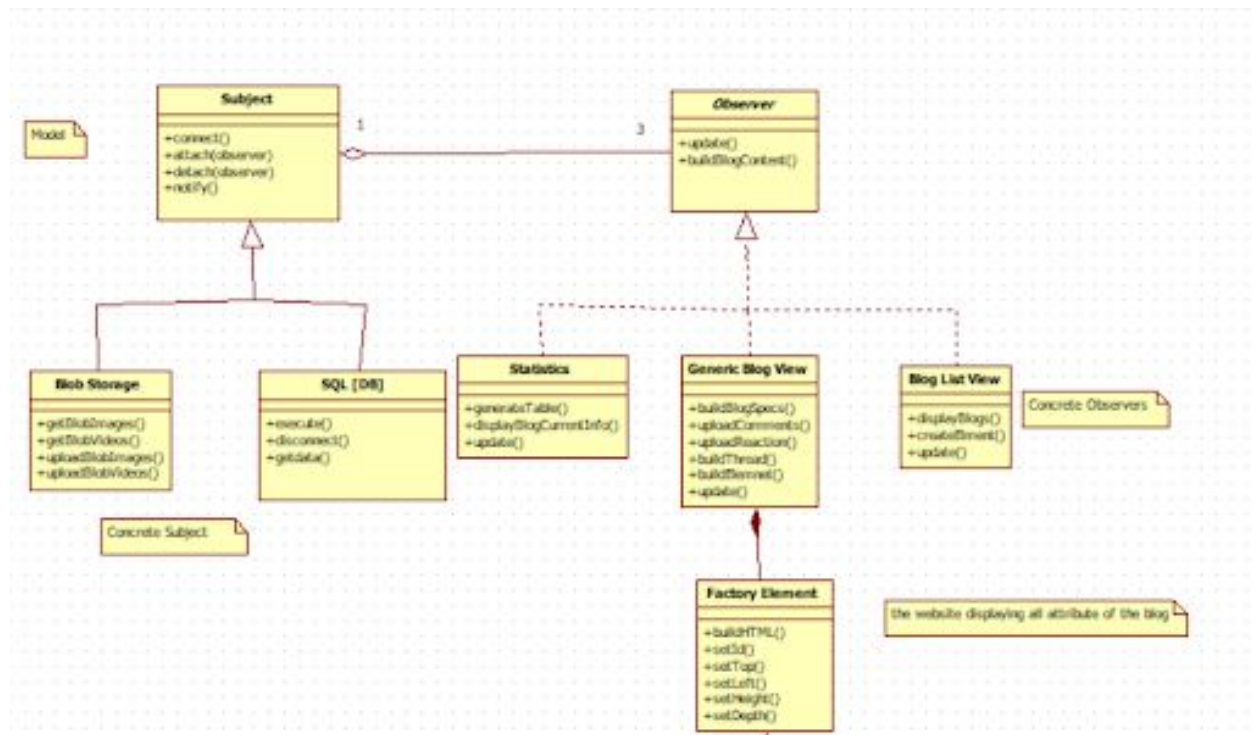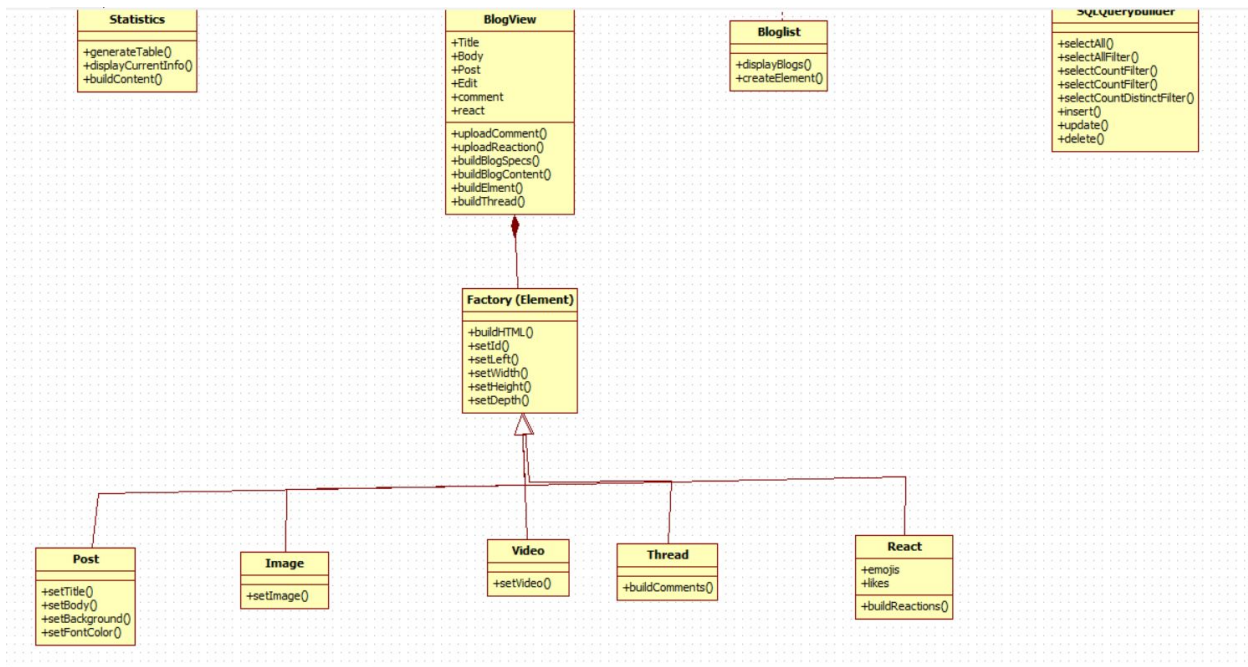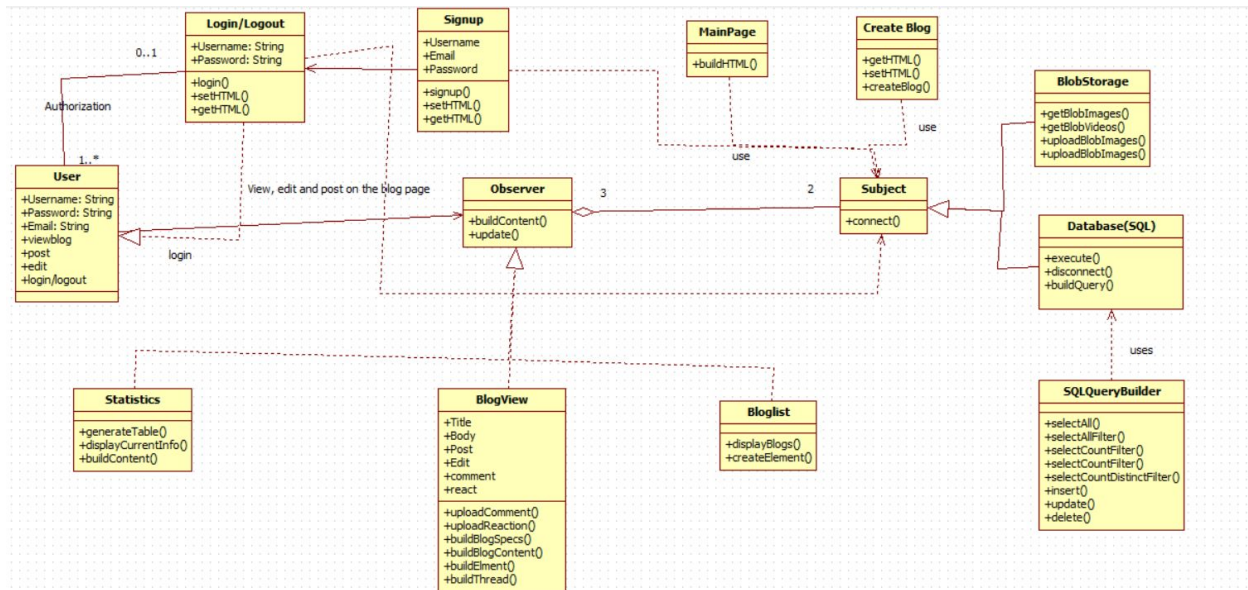
The class Subject connecting to both blob storage and the SQL database and notifies the each of the concrete observer when changes are made in each of the page.

This is full diagram of the observer pattern connected to the factory pattern
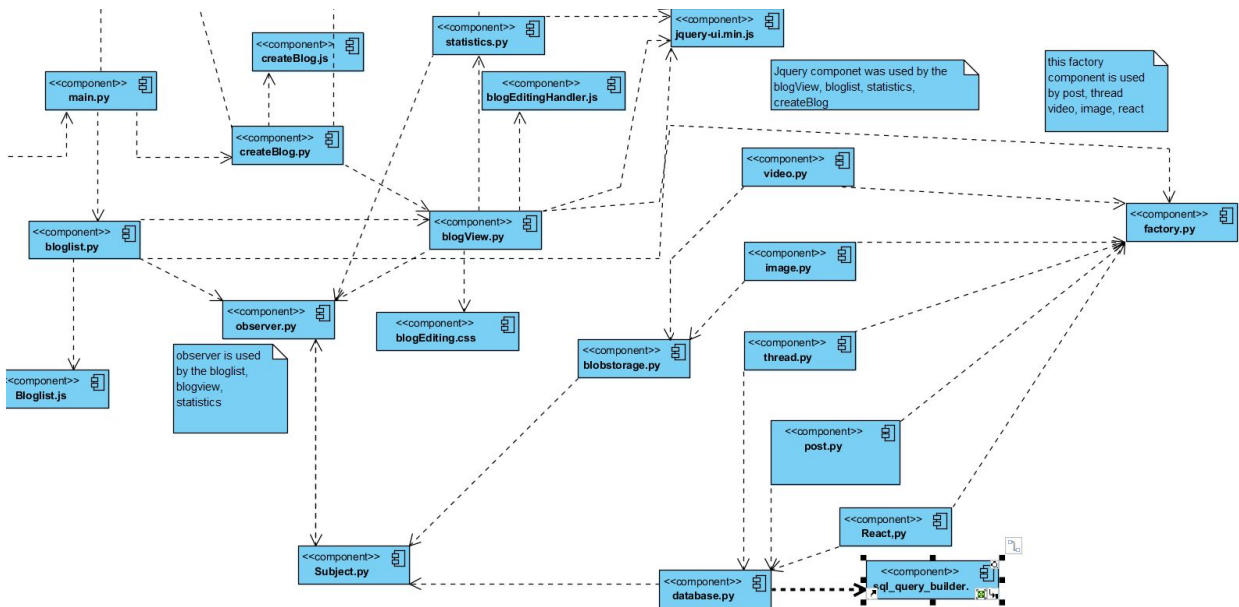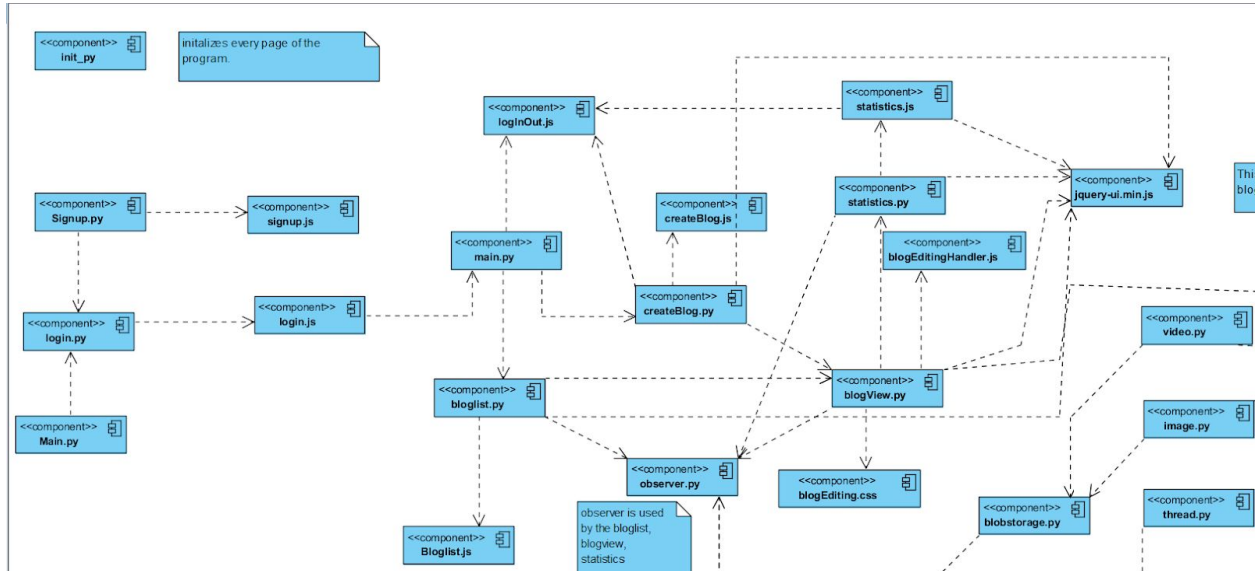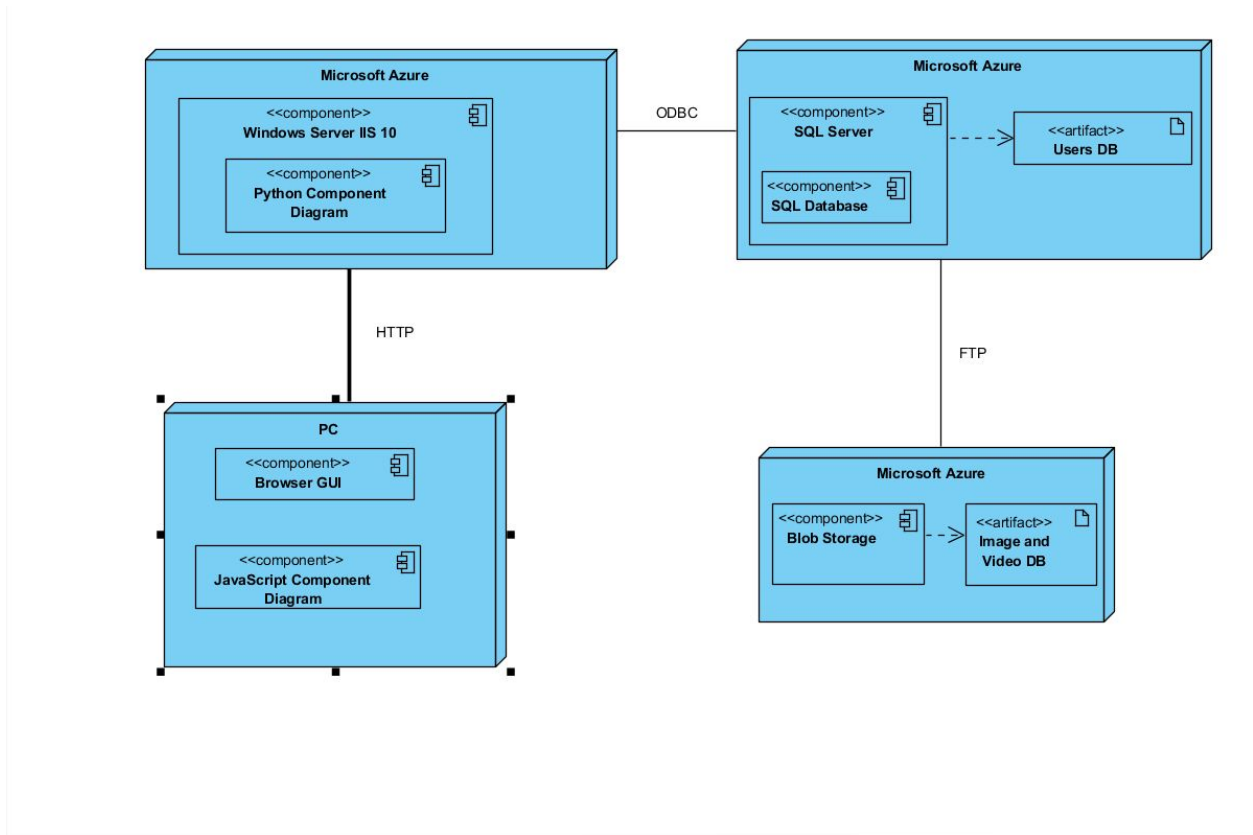
c) Class Diagrams

Login/Logout
+Username: String
+Password: String
+login()
+setHTML()
+getHTML()

0..1

Authorization

1..*

User
+Username: String
+Password: String
+Email: String
+viewblog
+post
+edit
+login/logout

login

Signup
+Username
+Email
+Password
+signup()
+setHTML()
+getHTML()

MainPage
+buildHTML()

Create Blog
+getHTML()
+setHTML()
+createBlog()

use

use

BlobStorage
+getBlobImages()
+getBlobVideos()
+uploadBlobImages()
+uploadBlobImages()

View, edit and post on the blog page

Observer
+buildContent()
+update()

3

2

Subject
+connect()

Database(SQL)
+execute()
+disconnect()
+buildQuery()

uses

Statistics
+generateTable()
+displayCurrentInfo()
+buildContent()

BlogView
+Title
+Body
+Post
+Edit
+comment
+react
+uploadComment()
+uploadReaction()
+buildBlogSpecs()
+buildBlogContent()
+buildElment()
+buildThread()

Bloglist
+displayBlogs()
+createElement()

SQLQueryBuilder
+selectAll()
+selectAllFilter()
+selectCountFilter()
+selectCountFilter()
+selectCountDistinctFilter()
+insert()
+update()
+delete()

Statistics
+generateTable()
+displayCurrentInfo()
+buildContent()

BlogView
+Title
+Body
+Post
+Edit
+comment
+react
+uploadComment()
+uploadReaction()
+buildBlogSpecs()
+buildBlogContent()
+buildElement()
+buildThread()

Bloglist
+displayBlogs()
+createElement()

SQLQueryBuilder
+selectAll()
+selectAllFilter()
+selectCountFilter()
+selectCountFilter()
+selectCountDistinctFilter()
+insert()
+update()
+delete()

Factory (Element)
+buildHTML()
+setId()
+setLeft()
+setWidth()
+setHeight()
+setDepth()

Post
+setTitle()
+setBody()
+setBackground()
+setFontColor()

Image
+setImage()

Video
+setVideo()

Thread
+buildComments()

React
+emojis
+likes
+buildReactions()

18

5. Programs
a) Component Diagram

b) Deployment Diagram

c) List of Implemented Classes and Functions

☐ BlobStorage -connect, getBlobImages,getBlobVideos,uploadBlobImages, uploadBlobVideos

☐ BlogView- uploadComment, uploadReaction, buildBlogSpecs, buildBlogContent, buildElment,buildThread

☐ Bloglist: displayBlogs, createElment

☐ CreateBlog - getHTML, setHTML, createBlog

☐ Database - execute, disconnect

☐ Factory - buildHTML, setId, setTop,setLeft, setWidth, setHeight, setDepth.

☐ Image - setImage

☐ Login - getHTML, setHTML, login(). atrribute- email and password.

☐ Post - setTitle, setBody, setBackground, setFontColor. attribute:

☐ React - buildReactions

☐ Signup: getHTML,setHTML signup. attribute: username, email, password.

☐ SQLQueryBuilder - selectAll, selectAllFilter, selectCountFilter, selectCountDistinctFilter, insert, update, delete.

☐ Statistics - generateTable, displayCurrentInfo, buildContent

☐ Thread - buildComments

☐ Video - setVideo

d) Screenshots of Data Tables

Users Table:

Definition:

  username varchar(15) not null,
  email varchar(30) not null,
  pwd varchar(15) not null,
  primary key (email),
  unique (email),
  unique (username)

Screenshot:

USERNAME	EMAIL	PWD
Legit	legit@legit.ca	legitpwd
Melanie	macdon6m@uregina.ca	password
Wilmurr	murr@gmail.com	password
newUser	newEmail	newPwd

Blog Table:

Definition:
        username varchar(15) not null,
        BlogName varchar(20) not null,
        imageSource varchar(100),
        descr varchar(300),
        backgroundColor varchar(20),
        font varchar(70),
        primary key (blogName),
        unique (username),
        unique (blogName)

Screenshot:

USERNAME	BLOGNAME	IMAGESOURCE	DESCR	BACKGROUNDCOLOR	FONT
newUser	Blog Demo	https://expressiveblob.blob.core.w...	This is a demonstration	rgb(255, 255, 255)	arial
Legit	I Like Chocobos	https://expressiveblob.blob.core.w...	This blog has a bunch of pictures ...	rgb(255, 255, 179)	arial
Melanie	My Space	https://expressiveblob.blob.core.w...	This is really cool I highly recomm...	rgb(255, 255, 255)	arial
test	test			rgb(0, 64, 64)	"courier new"

Posts Table:

Definition:
        blogName varchar(20) not null,
        id varchar(10),
        topVal int,
        leftVal int,
        width int,
        height int,
        depth int,
        title varchar(50),
        body varchar(1000),
        backgroundColor varchar(20),
        fontColor varchar(20),
        foreign key (blogName) references blog(blogName)

Screenshot:

BLOGNAME	ID	TOPVAL	LEFTVAL	WIDTH	HEIGHT
I Like Chocobos	post1	356	453	350	400

DEPTH	TITLE	BODY	BACKGROUNDCOLOR	FONTCOLOR	HASTHREAD
5	What is the title?	What is the content?	rgb(255, 255, 255)	rgb(33, 37, 41)	False

Images Table:

Definition:
        blogName varchar(20) not null,
        id varchar(10),
        topVal int,
        leftVal int,
        width int,
        height int,
        depth int,
        imageSource varchar(100),
        foreign key (blogName) references blog(blogName)

Screenshot:

BLOGNAME	ID	TOPVAL	LEFTVAL	WIDTH	HEIGHT
Blog Demo	image1	68	449	314	434
I Like Chocobos	image1	84	90	293	293
My Space	image1	86	59	300	300

DEPTH	IMAGESOURCE	HASTHREAD
3	https://expressiveblob.blob.core.w...	True
7	https://expressiveblob.blob.core.w...	True
1	/static/default.gif	True

Videos Table:

Definition:

    blogName varchar(20) not null,
    id varchar(10),
    topVal int,
    leftVal int,
    width int,
    height int,
    depth int,
    videoSource varchar(100),
    foreign key (blogName) references blog(blogName)

Screenshot:

BLOGNAME	ID	TOPVAL	LEFTVAL	WIDTH	HEIGHT
I Like Chocobos	video1	46	383	420	283

DEPTH	VIDEOSOURCE	HASTHREAD
4	https://expressiveblob.blob.core.w...	False

Comments Table:

Definition:

        blogName varchar(20) not null,
        attachedToId varchar(10) not null,
        username varchar(15) not null,
        comment varchar(300) not null,
        foreign key (blogName) references blog(blogName)

Screenshot:

BLOGNAME	ATTACHEDTOID	USERNAME	COMMENT
Blog Demo	image1	test2	
Blog Demo	image1	test	
My Space	image1	Melanie	this is a comment

Reactions Table:

Definition:

        blogName varchar(20) not null,
        attachedToId varchar(10) not null,
        username varchar(15) not null,
        emote varchar(35) not null,
        foreign key (blogName) references blog(blogName)

Screenshot:

BLOGNAME	ATTACHEDTOID	USERNAME	EMOTE
Blog Demo	image1	test	em-heart
Blog Demo	image1	Legit	em-smile
I Like Chocobos	image1	Legit	em-sob
Blog Demo	image1	test2	em---1

e) Link to Web-based Application

6. Technical Documentation

a) Programming Languages

Client-side: JavaScript, Jquery, HTML, CSS
Business-logic: Python with a Flask microframework
Database: SQL

b) Reused Algorithms and Programs

Listed below are the libraries used as well as the references used within the code from each library for use in the website:

Bootstrap - https://getbootstrap.com/

- `https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css`

- `//netdna.bootstrapcdn.com/bootstrap/3.0.2/css/bootstrap-theme.min.css`

- `https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js`

Jquery - https://jquery.com/

- `//ajax.googleapis.com/ajax/libs/jqueryui/1.7.1/themes/base/jquery-ui.css`

- `https://code.jquery.com/jquery-3.3.1.min.js`

w3 - https://www.w3schools.com/w3css/

- `https://www.w3schools.com/w3css/4/w3.css`

Datatables - https://datatables.net/

- `https://cdn.datatables.net/v/dt/dt-1.10.18/datatables.min.css`

- `https://cdn.datatables.net/v/dt/dt-1.10.18/datatables.min.js`

Popper - https://popper.js.org/

- `https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js`

Emoji - https://afeld.github.io/emoji-css/

- `https://afeld.github.io/emoji-css/emoji.css`

Font Awesome - https://fontawesome.com/

- `https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css`

Google Fonts - https://developers.google.com/fonts/

- https://fonts.googleapis.com/css?family=Lato

Flask - http://flask.pocoo.org/

- this was downloaded as a plugin in the web app, provided a microframework for Python

Pyodbc - https://pypi.org/project/pyodbc/

- this was downloaded as a plugin in the web app, provided communication between Python and the SQL database

c) Software Tools and Environments

Microsoft Azure

Used to host the website, as well as provide certain services such as creating and managing an SQL database, and Blob storage (for storing unstructured data, such as images and videos). Any queries to the database or Blob storage go through this service. The service also helps in integrating our Python code and deploying it to the website. Additionally, the site has an option to pull changes from a repository in Github so that any changes made to the program will cause the site to rebuild and redeploy with every change. The SQL database is used on every page except for the "Main" page, and the Blob storage is used on the "Create Blog" and "Blog View" pages to facilitate retrieving images and videos from storage, as well as uploading those images or videos to the shared space. The benefit of this tool is flexibility in terms of website deployment and development, such that any choice of language for the Business Logic layer or Database is supported, and the site can easily be controller in terms of starting/restarting the service, downloading plugins as necessary, or viewing error logs to track catastrophic failures. The access to Blob storage and the Database was well suited to our application as well, in that it is important that large quantities of both structured and unstructured data be stored and manipulated for our program.

Github

Used for version control as well as an IDE for the code. Github hosts a shared repository, in which we stored the code that the website deploys from, as well as any documentation shared between teammates, and static files such as JavaScript, CSS, or default images. The code stored in this repository can be modified, and that was the main method of doing code changes and development on the site. The coding done here is reflected in the following pages: "Create Blog", "Blog List", "Blog View", and "Statistics". The benefit of this tool was integration with the deployment of our website, such that code changes can be viewed immediately, which helps when so much of our program is reliant on visuals and inspecting the different components working together, which cannot be done when coding and testing in a separate IDE.

Brackets

Also used as an IDE for coding and testing HTML, CSS, and JavaScript without having to perform a push to Github each time to see the results. The coding done here is reflected on the "Main" page, which is the landing page, as well as the "Login" and "Signup" pages. The benefit of this was to allow visualization of the pages before they are pushed to the Github, as having multiple people developing on the Github can lead to strange behavior on the webpage as the website could be currently building and redeploying without the knowledge of one of those developers. Additionally, Brackets provides a way to run and preview the code changes quickly, without having to wait for the webpage to redeploy, and that quick reaction and visualization is helpful for testing styling and debugging JavaScript.

7. Acceptance Testing

a) Functional Testing

i) Modify blog content, changes should persist

Assumptions: User account and blog have already been created. User has navigated to "Blog View" for their own blog by selecting it from the "Blog List" page or from the landing page

Test Steps:

i) User starts off with an empty blog.
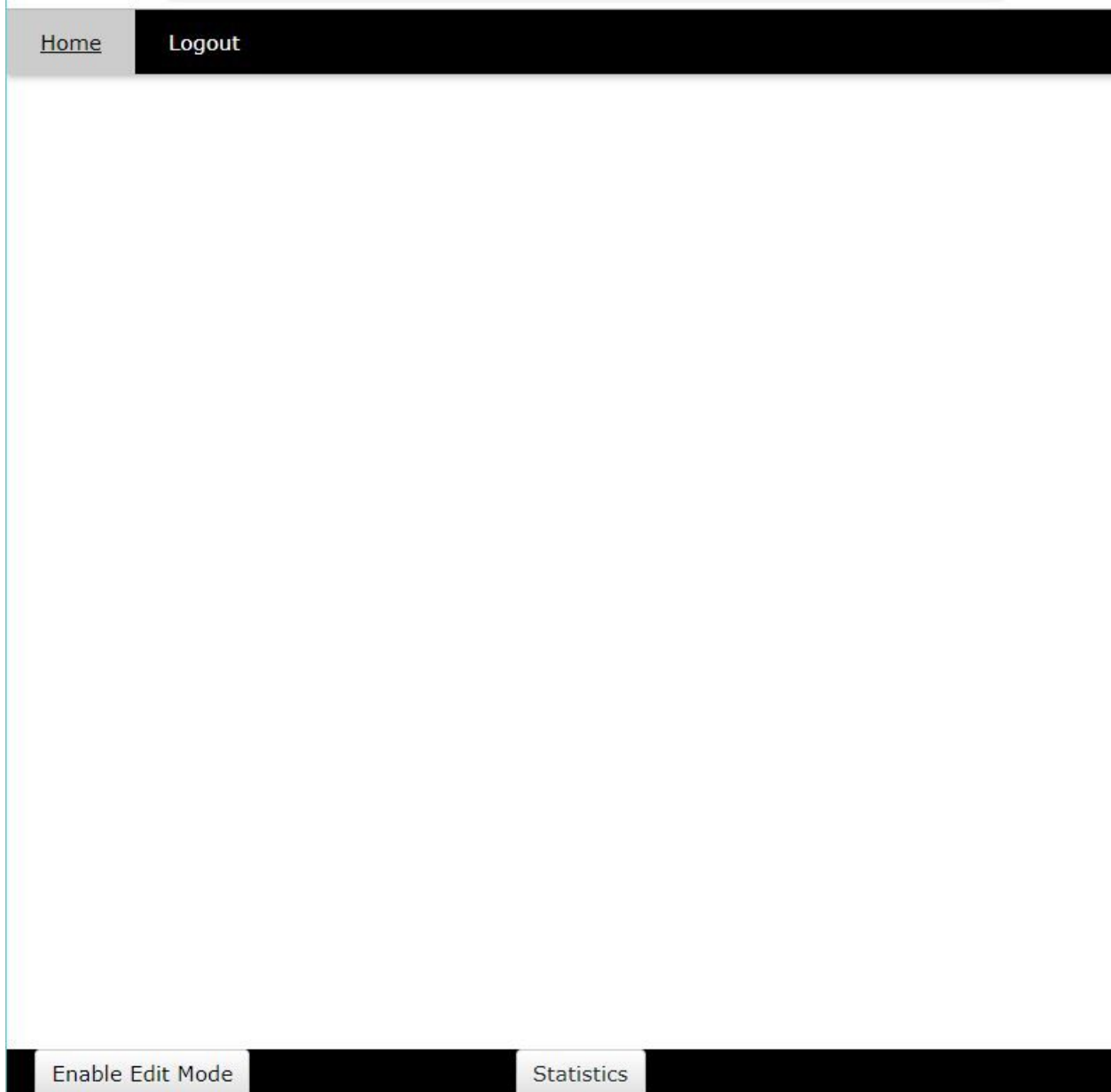
ii) User hits "Enable Edit Mode", and selects "Add post" from the toolbar at the top left of the page. User hits "Edit" and changes the content of the post. User repeats for an image and a video.
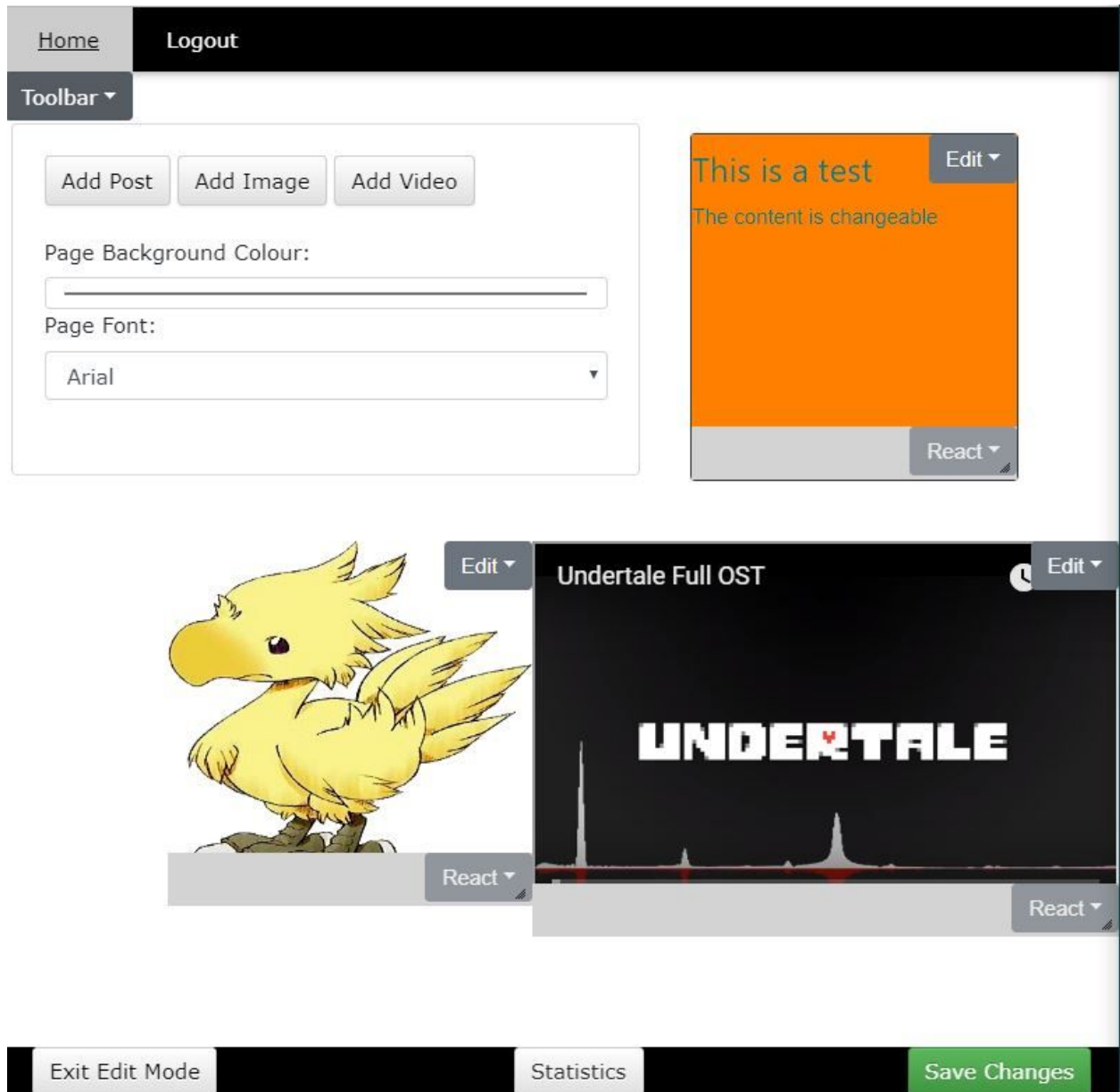
iii) User hits "Save Changes". The page automatically reloads in view mode. All content is displayed where the user left it in the state they left it.
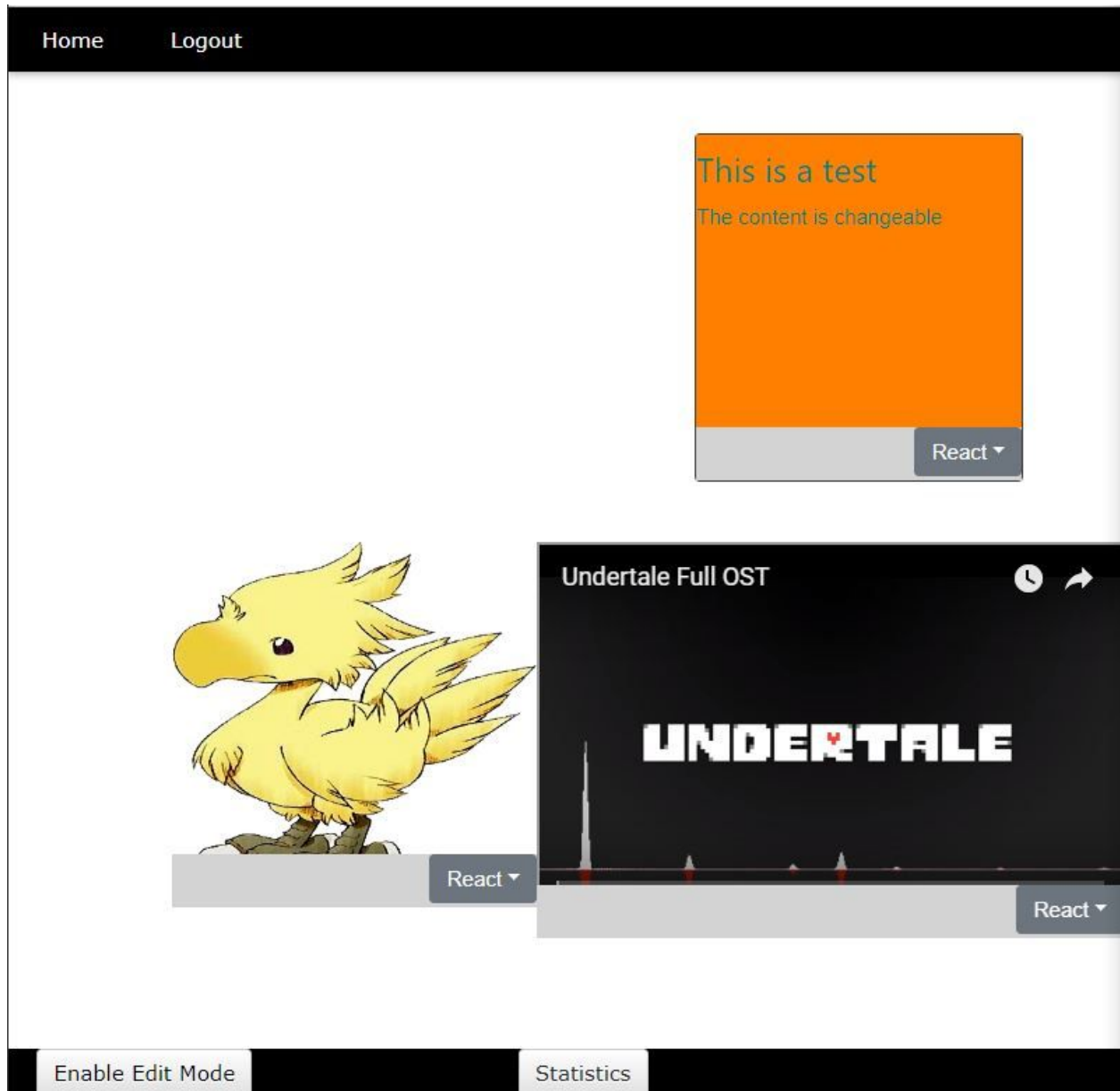
Assumptions: User account and blog have already been created, user is on the "Blog View" page for their own blog.

Test Steps:

i) User hits "Enable Edit Mode" and selects the "Page Background Colour" input from the "Toolbar" dropdown and selects a color. User selects the "Page Font" input from the "Toolbar" dropdown and selects "Georgia".
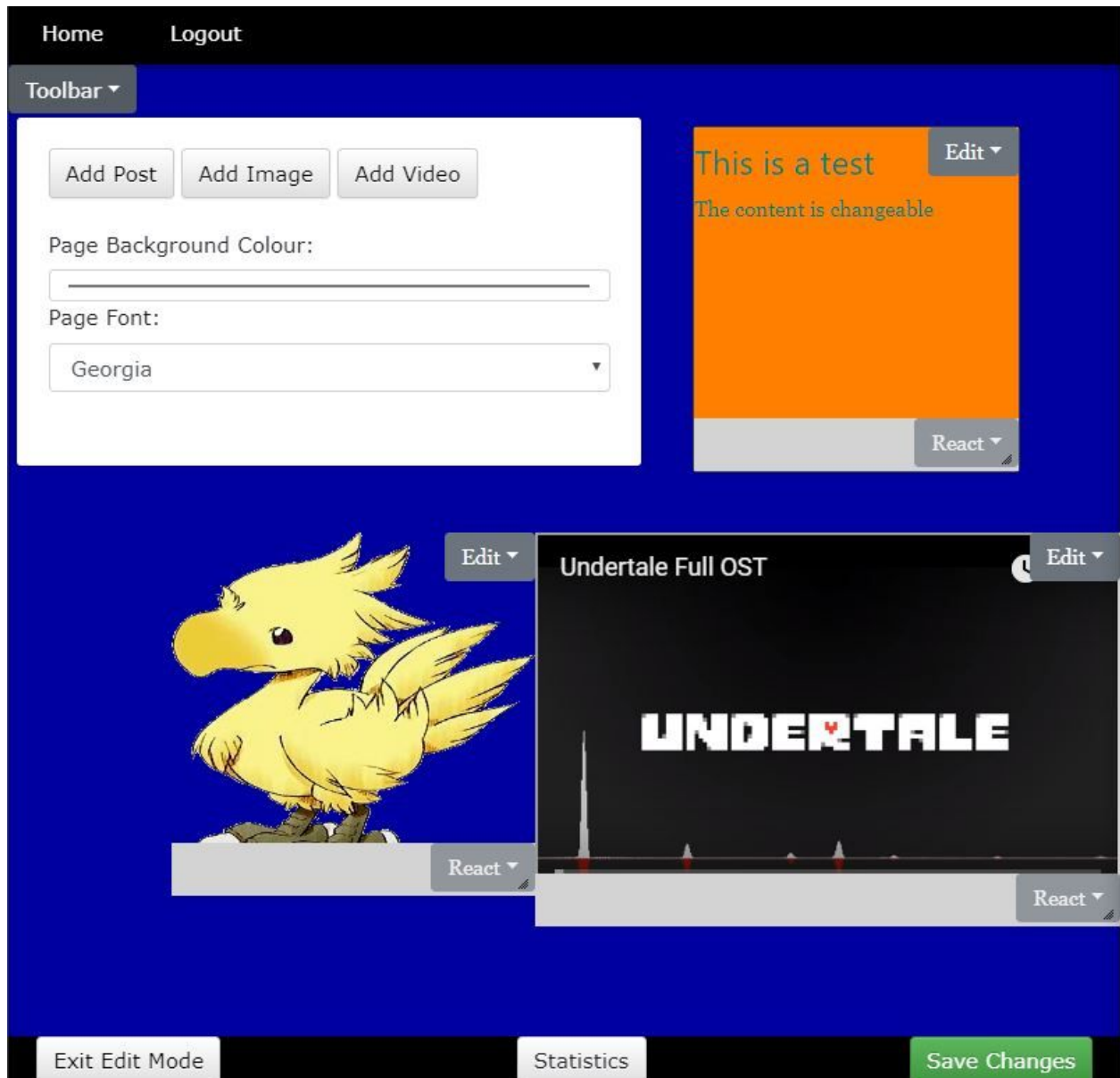
ii) User hits "Save Changes" and the page reloads automatically. The page reloads with the same style the user left it in.
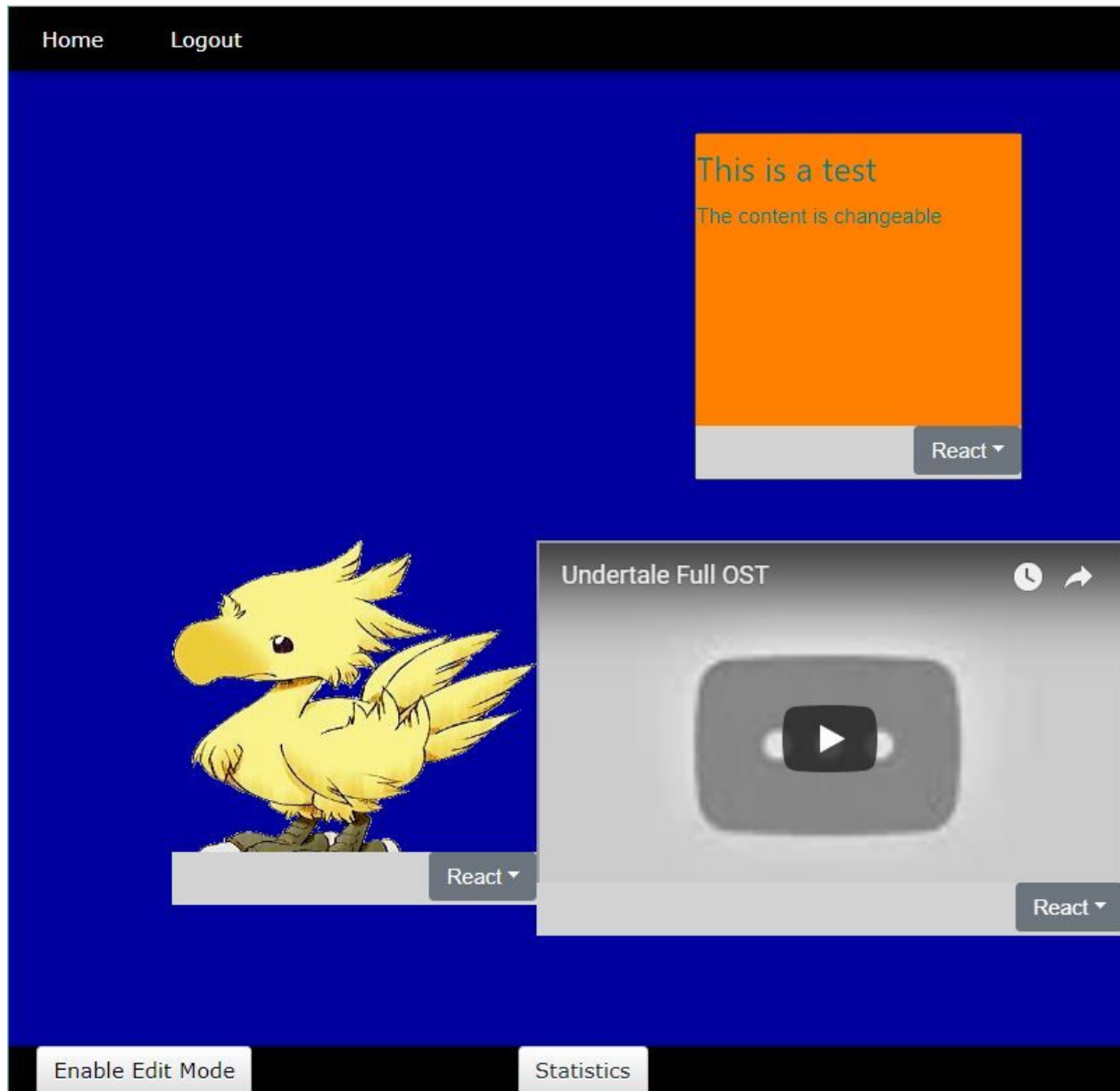
iii) Create blog produces correct listing on bloglist

Assumptions: User account has already been created, user has navigated to "Create Blog" from the landing page.

Test Steps:

i) User inputs a blog name and description into the form. User selects image by hitting the radio button for "Get image from Library" and selects an image. User hits submit

Blog Name

Testing blog

Identifying Picture

○ Get image from URL ● Get image from Library

Choose image:

very_important.jpg ▼

Upload image to library:

Choose File   No file chosen

Upload Image

Preview:

Preview:

Description

This is a blog intended for acceptance testing|

ii) User navigates to "Blog List" page. Newly created blog is listed with correct image source, blog name, and description.

iv) Comments display the current username

Assumptions: User is signed in and has navigated to a blog with a thread.

Test Steps:
i) User selects the input box on a thread and enters a comment
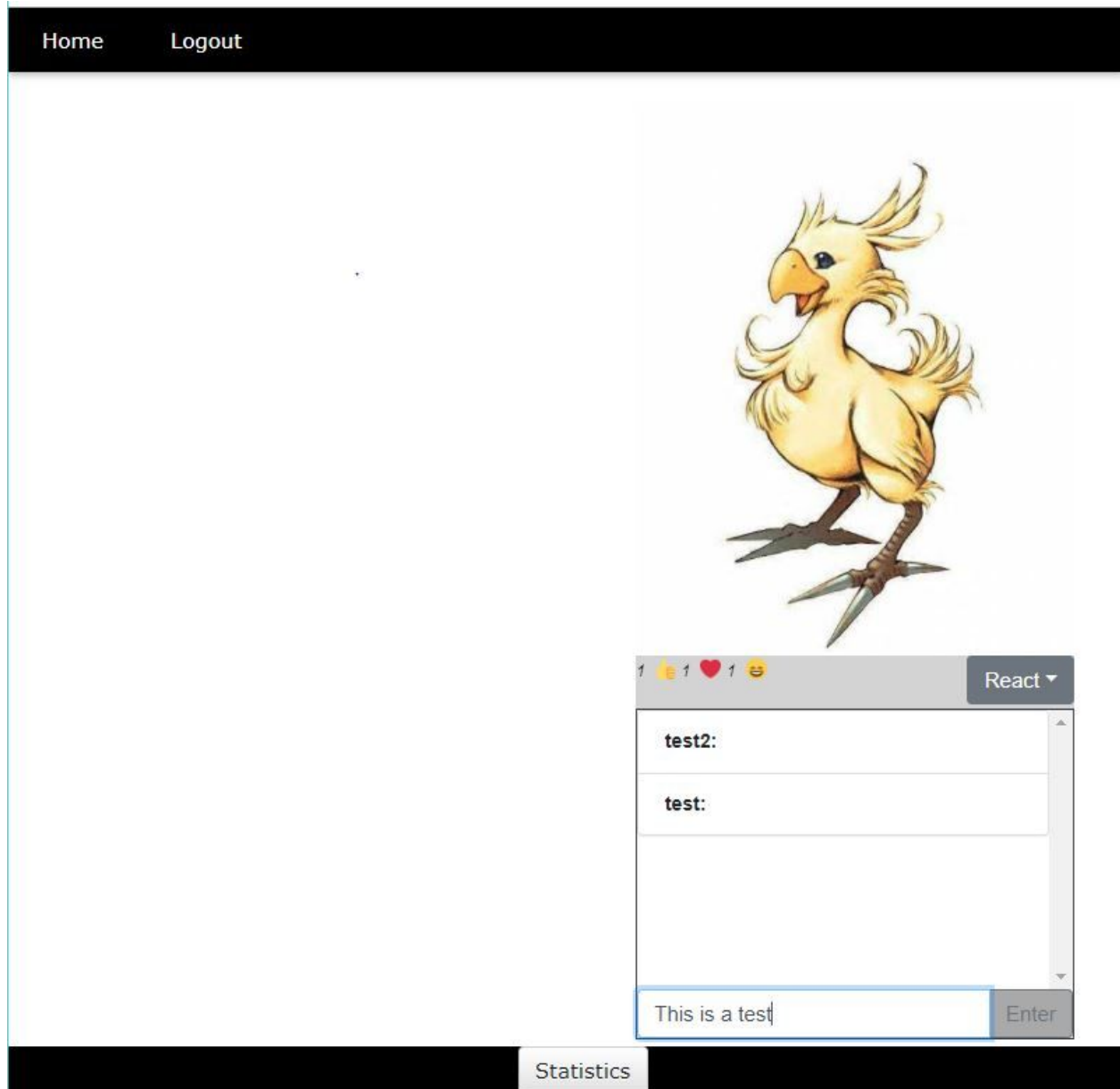
ii) User hits "Enter" on the comment box and the page reloads automatically. The comment is now displayed at the bottom of the thread box showing the correct current user.
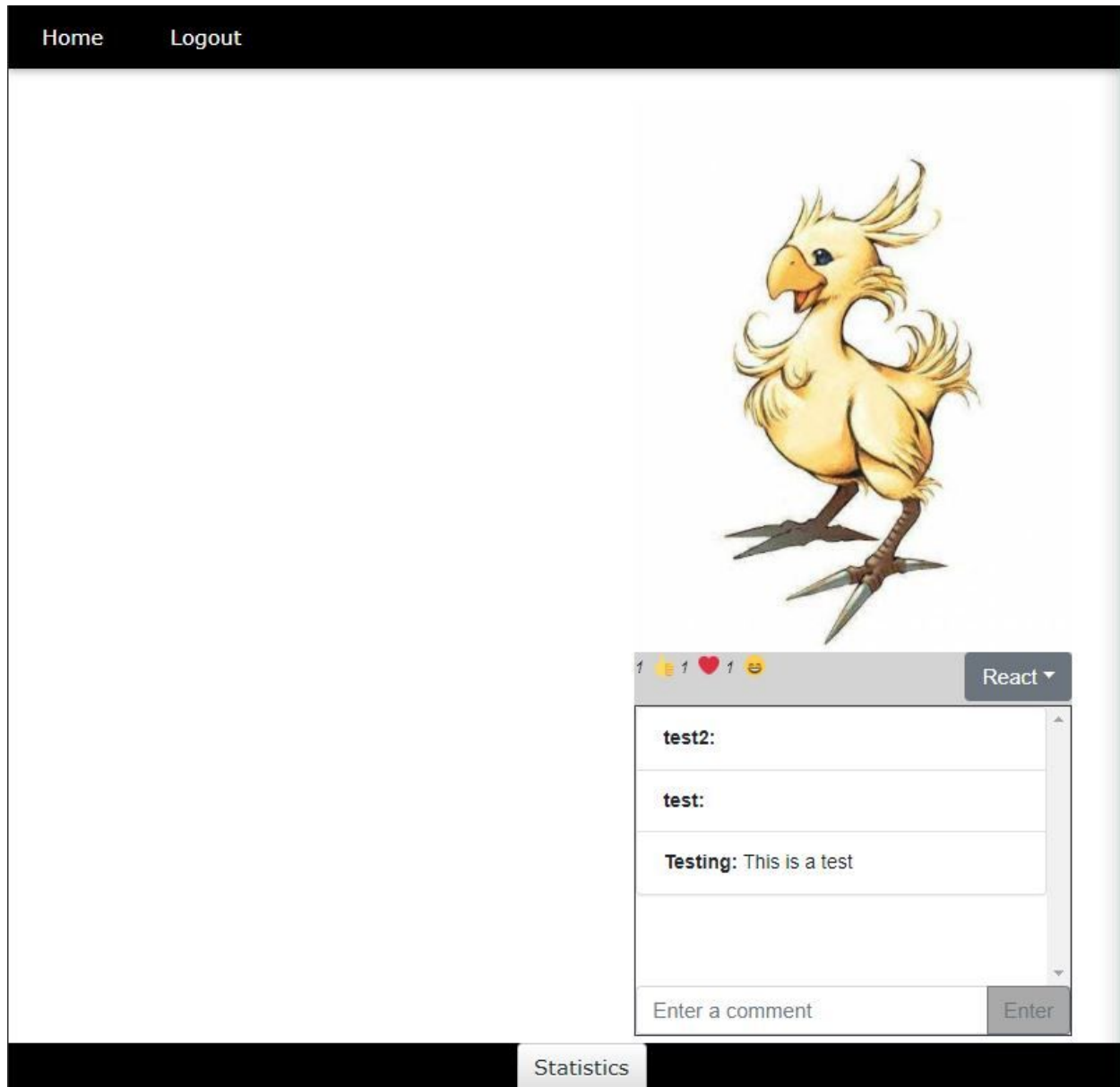
v) Statistics page shows the right number of reactions and comments

Assumptions: User is on the "Blog View" page for a blog that contains comments and reactions.

i) User verifies the number of reactions, comments, and commenters
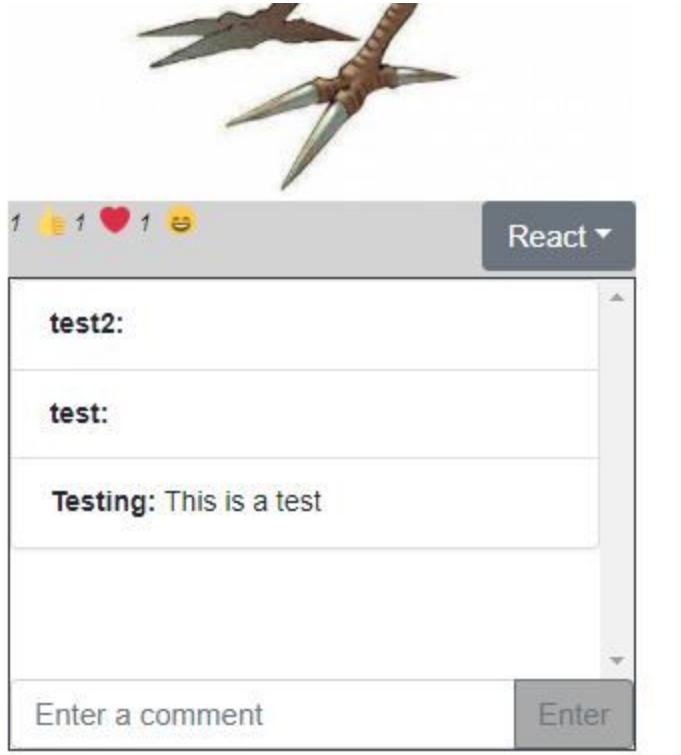
ii) User verifies that the number of comments, users commented, and reactions reported in both the top information and the table matches the blog

Home   Logout

Statistics for Blog "Blog Demo"

Blog Name: Blog Demo

Owner: newUser

Number of Comments: 3

Number of Users Commented: 3

Number of Reactions: 3

Global Statistics

Show 10 ▾ entries   Search: [   ]

Blog Name ▲	Owner ⬍	Number of Comments ⬍	Number of Commenters ⬍	Number of Reactions ⬍
Blog Demo	newUser	3	3	3
I Like Chocobos	Legit	0	0	1
My Space	Melanie	1	1	0
test	test	0	0	0
Testing blog	Testing	0	0	0

Showing 1 to 5 of 5 entries   Previous   1   Next

b) Robustness Testing

i) Blog content cannot be dragged out of bounds

Assumptions: User starts on the "Blog View" page for their own blog.

Test Steps:

i) User hits "Enable Edit Mode" and attempts to drag the post out of bounds to the right.
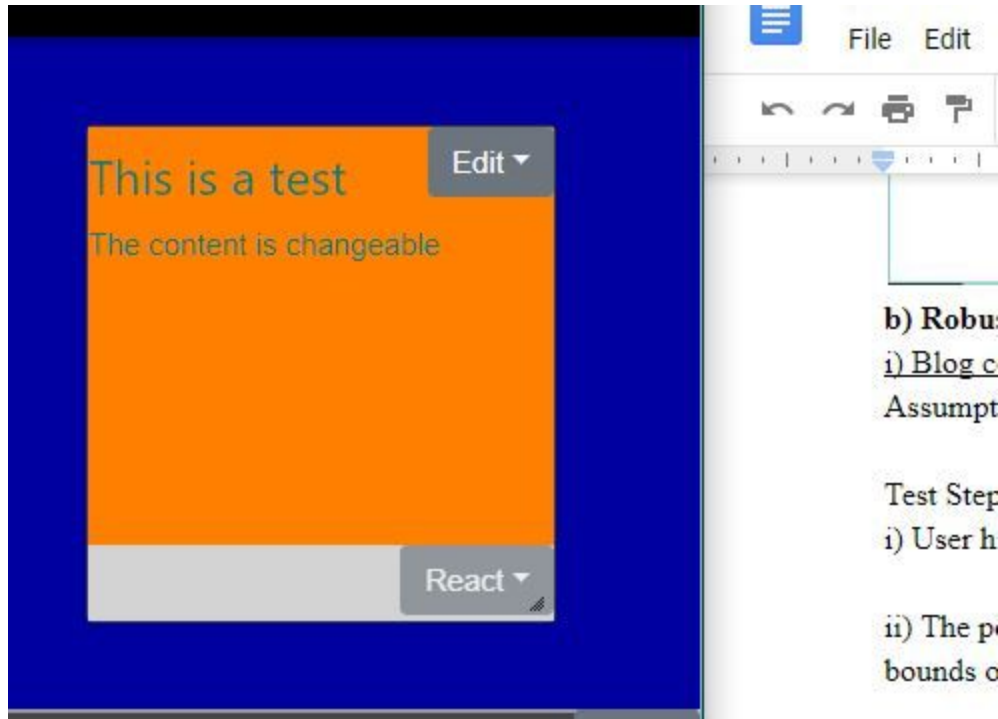
ii) The post remains flush against the right edge, and does not disappear behind the rightmost bounds of the window.
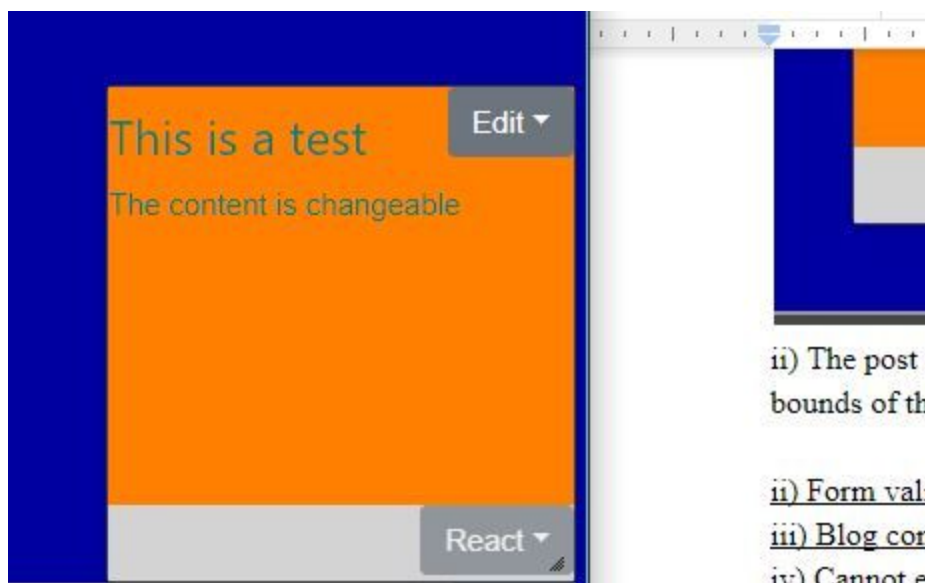
<u>ii) Form validation for create blog</u>

Assumptions: User has an account and has not already created a blog

Test Steps:

i) User does not input anything in the "Blog Name" field on the "Create Blog" page

Blog Name

Enter a unique name for your blog

Identifying Picture

◉ Get image from URL ○ Get image from Library

ii) User hits "Submit". The submission is rejected and validation appears under the blog name requesting input

Blog Name

Enter a unique name for your blog

Identifying Picture

⚠ Please fill out this field.

◉ Get image from URL ○ Get image from Library

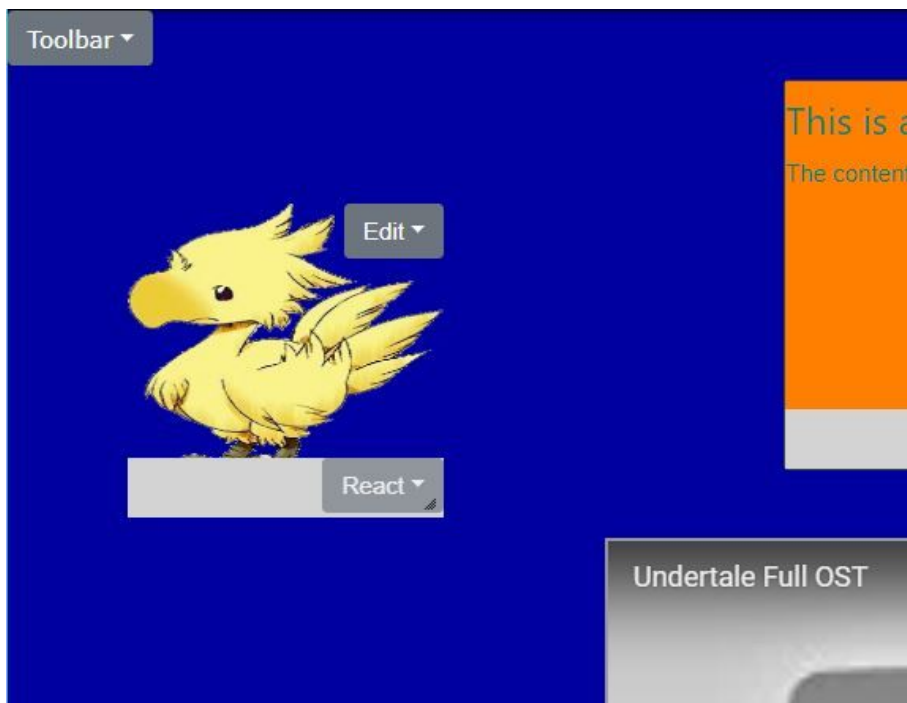iii) Blog content cannot be resized below a certain size

Assumption: User is on the "Blog View" page for their own blog

Test steps:

i) User enters into edit mode and verifies original size of image
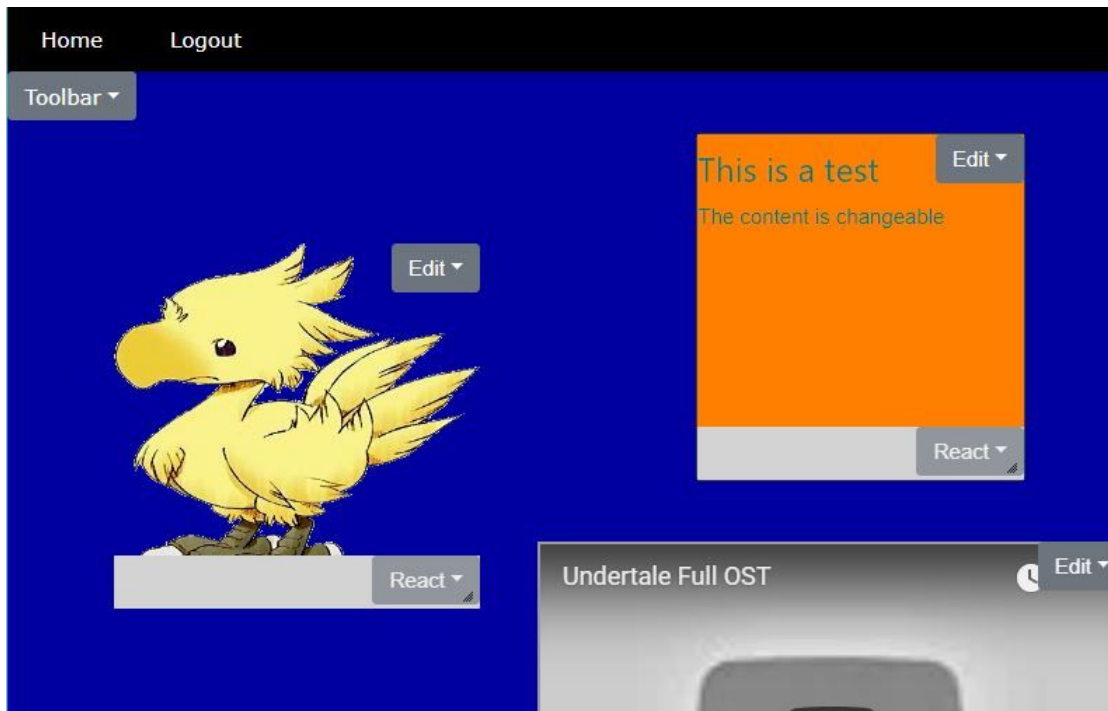
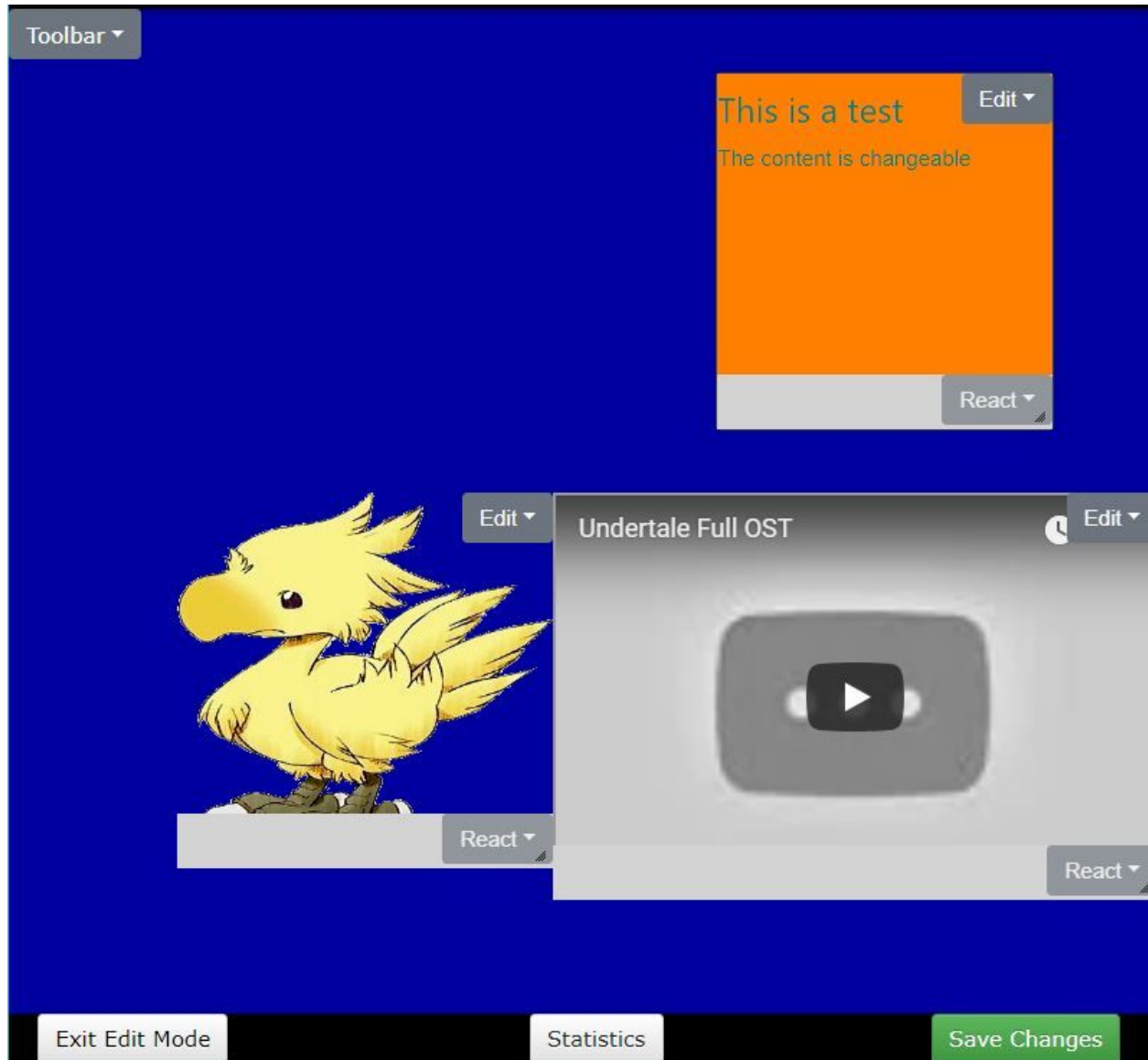ii) User attempts to resize image as small as it can be, but it doesn't go below a certain size

iv) Cannot edit another person's blog

Assumption: User is on the "Blog View" page for their own blog

Test steps:

i) User can see an "Enable Edit Mode" button on their blog. Clicking it displays a "Toolbar" for editing, and places "Edit" dropdowns over all elements.

ii) User navigates to another blog. "Enable Edit Mode" button is missing, and there is no way to move or edit the content, only comment or react.
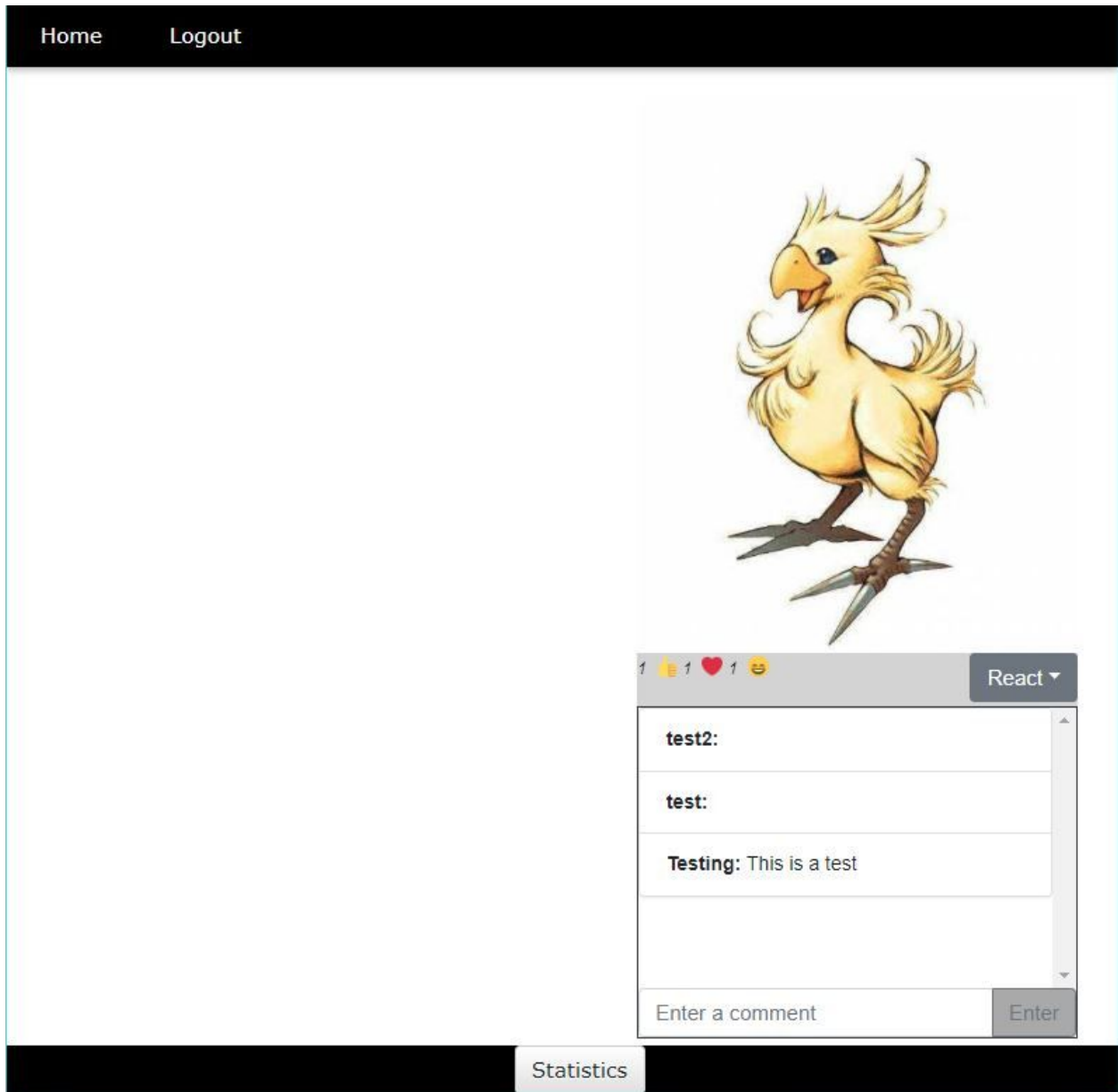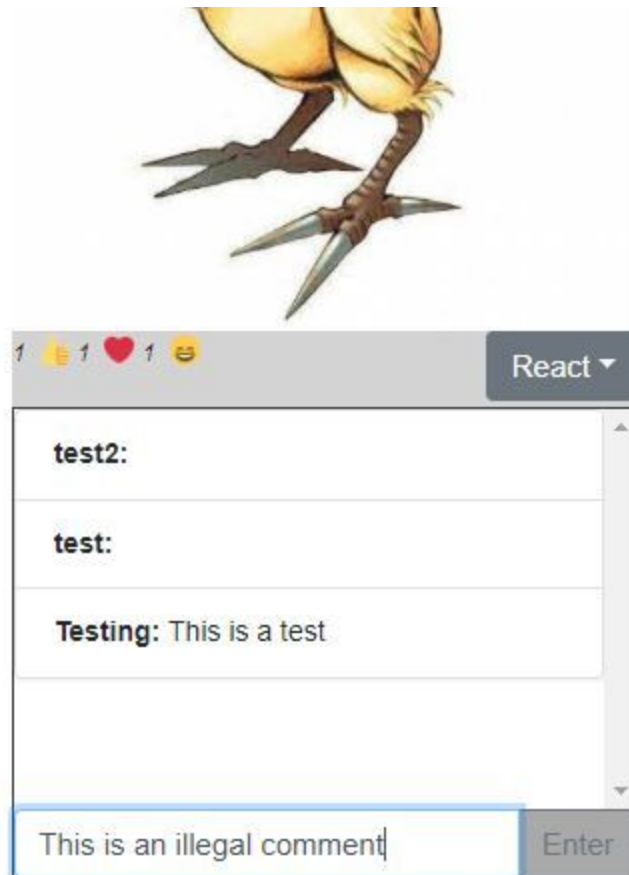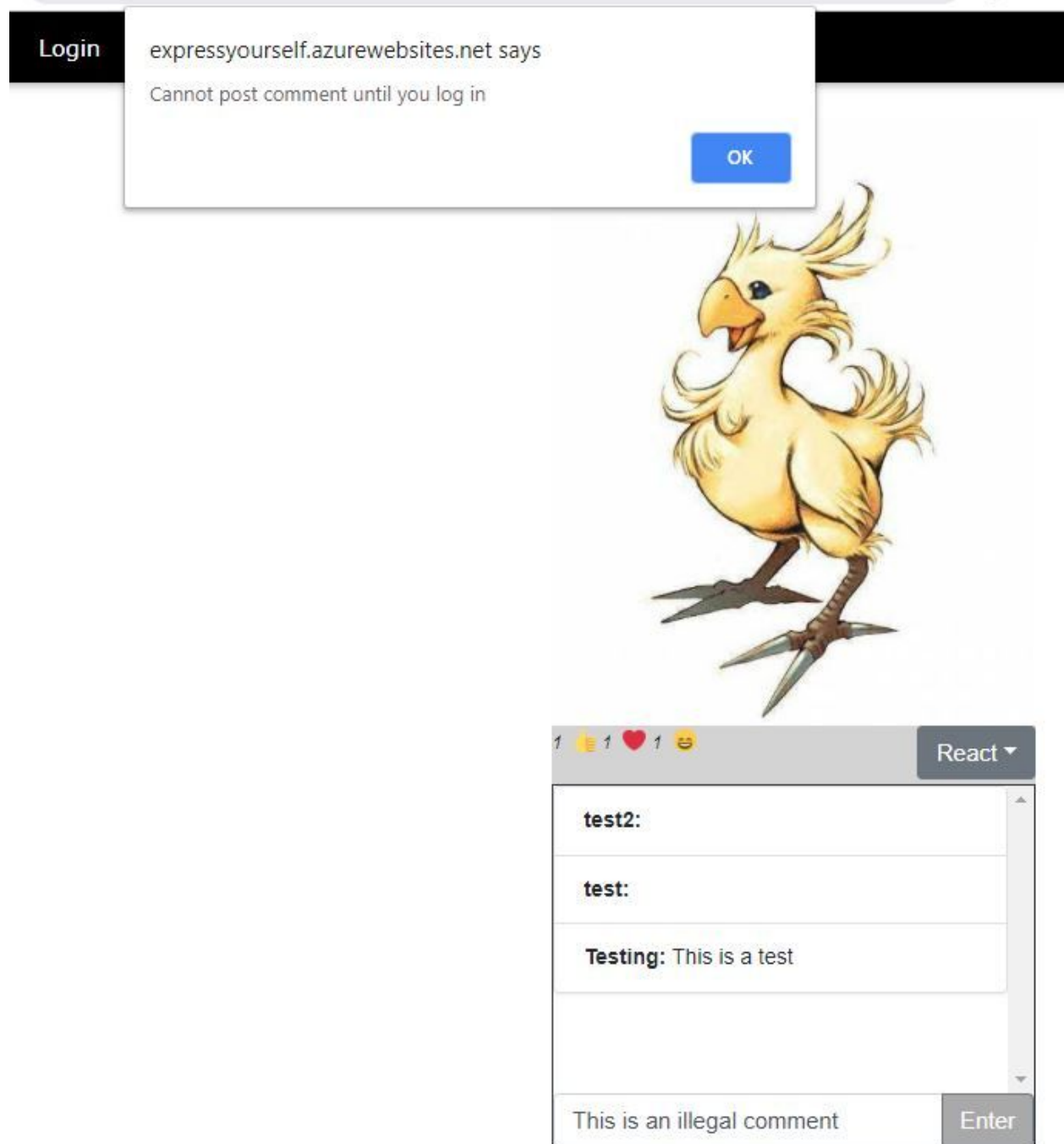
<u>v) Cannot comment without an account</u>
Assumption: User is on the "Blog View" page for someone else's blog and is not logged in

Test steps:
i) User enters a value into the input field of the comment

ii) User hits "Enter". The comment is rejected and an error message is displayed.

c) Time-efficiency

i) Load blog content under 3 seconds

The performance test performed by "Pingdom Tools" will determine how long it takes to load the blog-view page. "I like chocobos blog page" was used to performed the test.

The load time - 2.42s

Page Size -697.2KB

ii) Upload image

Performance test will check the the time it will take for the web page to upload an image from the system library. This test was performed for five different times with five different images

The average load time - 1.15s

iii) Submit changes under 1 second

The performance test will determine the time taken to the submit changes and load the blog-view page after the blog owner is done altering and adding content to the blog page.

The load time - 39.4 ms

iv) Commenting under 1 second

Performance test will determine the period it takes for a user to add comment to a post and the display the content of the blog.

The load time - 42.1ms

v) Reacting under 1 second

The performance test will done to check the amount of time it takes the webpage add reactions, load and display the blog-page.

The load time - 613ms

8. Contributions

Task	MacDonald Contribution	Akinyemi Contribution
Problem Definition	100%	0%
Economic Feasibility	90%	10%
Functional Requirements	30%	70%
Use case Diagrams	0%	100%
Software Qualities	20%	80%
Software Architecture	20%	80%
Design Patterns	20%	80%
Class Diagrams	0%	100%
Programs	20%	80%
Technical Documentation	100%	0%
Acceptance Testing	66%	34%
Website Styling	60%	40%
Website Functionality	90%	10%
Website Setup	100%	0%
Report	50%	50%
Total	49%	51%