Florida Atlantic University

CEN 4010 Principles of Software Engineering, Fall 2021

Milestone 3 Project Proposal and High-level Description

Strikers

Group: 20

Project Members

Nicholas Giacobbe (Team Lead)

ngiacobbe2020@fau.edu

Truong Nguyen

truongnguyen2019@fau.edu

Shay Reardon

sreardon2018@fau.edu

Alex Rodriguez

alexrodrigue2018@fau.edu

Luiza Trebesqui

ltrebesqui2018@fau.edu

Revision History

Version #	Date of Revision	Changes
1	09/28/2021	Initial Version
2	10/25/2021	Updated for Milestone 3

TABLE OF CONTENTS

Vertical Software Prototype

The prototype: https://lamp.cse.fau.edu/~cen4010_fa21_g20/striker_prototype/ The articles that show up on the homepage are obtained from the MySQL database. First, log into the admin account, the location under login on the navigation bar, to test the website. The username will be admin and the password is 12345. Once the user is logged in, the home page will display the username and a new navigation bar called profile that has all the articles submitted.

Executive Summary

Striker (preliminary name with domain for cheap) is a platform on which any journalist, blogger, or writer can post articles. Readers will have to pay a small fee to get behind the paywall of each article. This solves many problems facing journalism, text media, and the user experience.

Until recently, print media relied on advertisers and classifieds to generate most of their revenue. With the emergence of the internet, there has been a massive move towards a subscription-based model as most advertisers use different avenues outside new sites and newspapers. The big problem for readers is that there is a limit to how many sites they are willing to subscribe to. The emergence of substack also complicates the landscape as individual writers have generated sizable incomes through a direct subscription model and large followings generated on social media.

With the general decline in institutional trust, we feel that there is no longer as much value for writers to be attached to major newspapers. The direct model is much more advantageous for big names in journalism, but we feel it is not as beneficial for readers.

Our platform will allow users to read a much greater variety of content compared to a substack subscription while still allowing writers to receive a much more direct stream of revenue than working exclusively for a newspaper. It also solves the problem of there being no connection between the amount and quality of content and the subscription price. Our model creates the incentive for writers to be both prolific and create quality material.

Use Cases

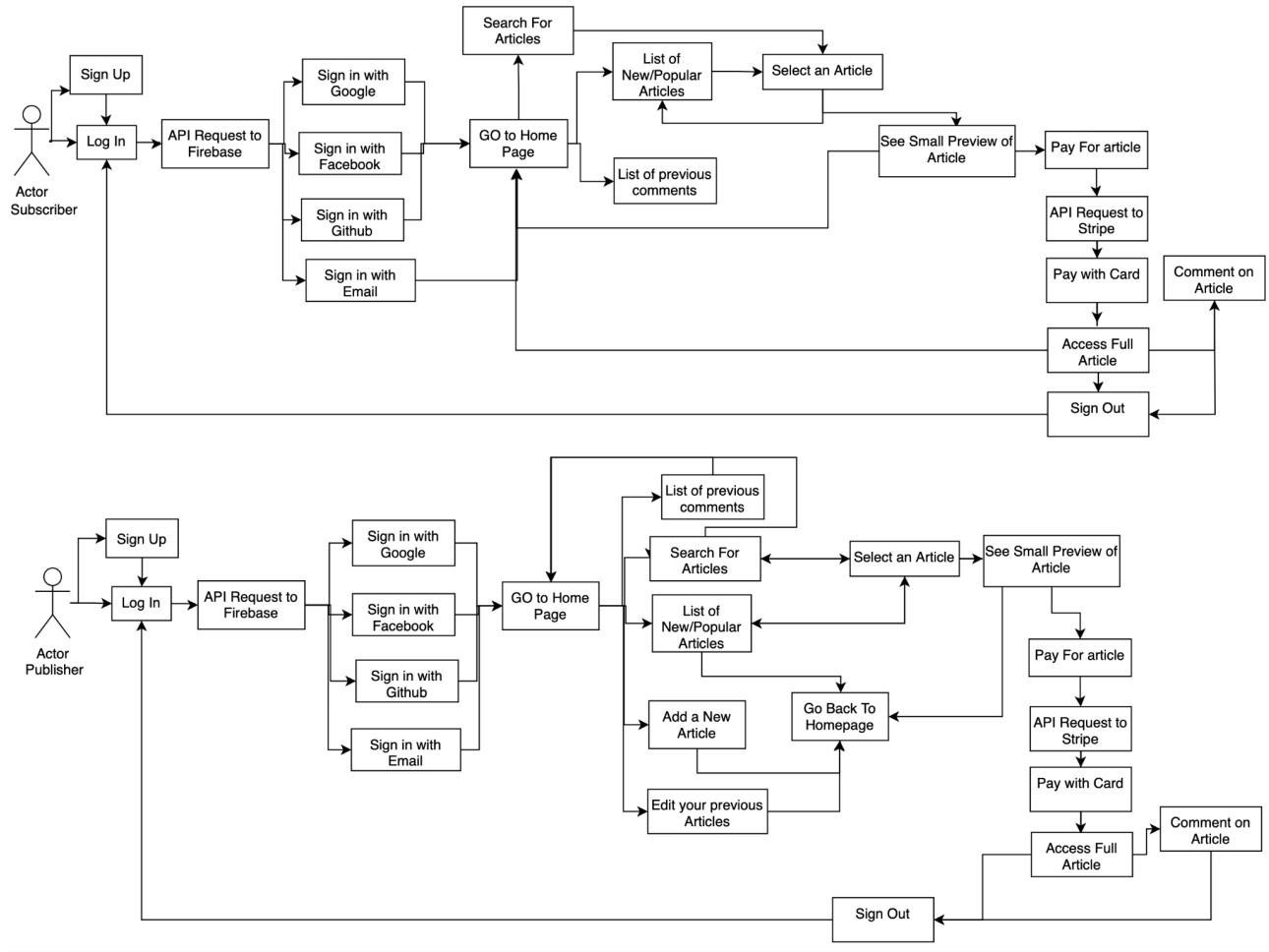


Fig. 1: Use Case Diagrams

Use Case - Search

The user wants to use the search capabilities to find a newspaper article. The user arrives at the Home Page and uses the search bar field to enter search criteria. The system displays the search results, and the user chooses among the articles displayed.

1. Description:

The use case describes how the search feature functions.

2. Actors:
    2.1. User
    2.2. Systems

3. Preconditions:
    3.1. User has internet connection
    3.2. System is available
    3.3. User has an active account

4. Primary Flow of Events:
    4.1. User goes to the website
    4.2. The Homepage of the website appears
    4.3. User enters search criteria for the article
    4.4. A web page containing a list of the relevant search results is displayed to the user
    4.5. Terminate Use Case: Search

5. Alternate Flows:
    5.1. User Enter Invalid Characters or Format into Search Bar
        5.1.1. If during step 4.3 the user enters invalid characters or search criteria
            5.1.1.1. Website returns an error stating that the user input is invalid
            5.1.1.2. Return to step 4.2
    5.2. There Are No Articles Matching the User's Criteria
        5.2.1. If during step 4.3 there are no articles that match the conditions entered by the user in the search bar
            5.2.1.1. Website returns an error stating that there are no articles matching the search conditions
            5.2.1.2. Return to step 4.2

Use Case - Comment

The user is on a page and wants to comment about the article displayed on the page. The user enters a comment and hits the submit button.

1. Description:

The use case explains how the user will post a comment for other users to view.

2. Actors:
    2.1. User
    2.2. Systems

3. Preconditions:
    3.1. User has internet connection
    3.2. System is available
    3.3. User has an active account
    3.4. User is logged in

4. Primary Flow of Events:
    4.1. User goes to the website
    4.2. The Homepage of the website appears
    4.3. User logs in
    4.4. User goes into a page containing an article by selecting an article on the homepage or by using Use Case: Search
    4.5. User pays for the article using Use Case: Paywall
    4.6. User access a page displaying the full article
    4.7. User clicks on the Comment Button
    4.8. User enters comment and clicks the submit button
    4.9. System thanks user for the comment
    4.10. Terminate Use Case: Comment

5. Alternate Flows:
    5.1. User does not have a valid account or is not logged in
        5.1.1. If in step 4.5, the user does not have a valid account or if the user is not logged in
            5.1.1.1. The webpage requesting the user to create an account and/or log in will be displayed
            5.1.1.2. User creates an account or logs in
            5.1.1.3. Return to step 4.4
    5.2. User Exceeds the Maximum length of Comments
        5.2.1. If during step 4.8 the user exceeds the maximum length allowed for comments
            5.2.1.1. Website returns an error stating that the comment exceeds the maximum length
            5.2.1.2. User decreases the length of their comment and resubmits the comment
            5.2.1.3. Return to step 4.9

5.3. User Selects an article that the User Has Not Paid For

5.3.1. If in step 4.7, the user tries to comment on an article that has not been purchased

5.3.1.1. The webpage will display an error and prompt the user to purchase the article with Use Case: Paywall

5.3.1.2. If user pays for the article return to step 4.6

5.3.1.3. If user does not pay for the article return to step 4.4

Use Case - Paywall

The user wants to purchase a newspaper article displayed on the page. The user will enter a credit card to buy the article using Stripe API.

1. Description:
The use case describes how the paywall feature works

2. Actors:
    2.1. User
    2.2. Systems

3. Preconditions:
    3.1. User has internet connection
    3.2. System is available
    3.3. User has an active account
    3.4. User is logged in

4. Primary Flow of Events:
    4.1. User goes to the website
    4.2. The Homepage of the website appears
    4.3. User logs in
    4.4. User selects an article on the homepage or by using Use Case: Search
    4.5. Type of payment appears through the Stripe API. Stripe API will allow the user to select their preferred payment method, enter their credentials, and notify the user and the system that the payment is complete.
    4.6. User access a page containing the full article
    4.7. Terminate Use Case: Paywall

5. Alternate Flows:
    5.1. User Does Not Pay for Article
        5.1.1. If during step 4.5 the user does not pay for the article
            5.1.1.1. Website displays a message stating that to access the full article, the user must pay for the article
            5.1.1.2. Return to step 4.2

Use Case - Post

The user wants to publish an article on the website. First, the user will click the add an article button displayed on the homepage for authorized journalists. The user will then post the newspaper article to the website by clicking the publish button.

1. Description:

The use case explains how the user will post new articles to the website.

2. Actors:
    2.1. User
    2.2. Systems

3. Preconditions:
    3.1. User has internet connection
    3.2. System is available
    3.3. User has an active account
    3.4. User logs in
    3.5. User is an authorized journalist

4. Primary Flow of Events:
    4.1. User goes to the website
    4.2. The Homepage of the website appears
    4.3. User logs in
    4.4. User clicks on the Add an article Button
    4.5. User enters an article and clicks the post button
    4.6. System thanks user for uploading an article
    4.7. Terminate Use Case: Post

5. Alternate Flows:
    5.1. User Is Not Logged In
        5.1.1. If in step 4.3, the user is not logged in

5.1.1.1. The webpage requesting the user to log in will be displayed
5.1.1.2. User logs in
5.1.1.3. Return to step 4.4

5.2. User is Not an Authorized Journalist
5.2.1. There won't be any options to add articles displayed to the user
5.2.2. User will be required to submit a form requesting to become an authorized journalist
5.2.3. If a user receives an email stating that they have been accepted as an authorized journalist. Return to step 4.3
5.2.4. Otherwise, user will not be able to post any articles

5.3. User Exceeds the Maximum length of an article
5.3.1. If during step 4.5, the user exceeds the maximum length allowed for articles
5.3.1.1. Website returns an error stating that the article exceeds the maximum length
5.3.1.2. User decreases the length of their article and resubmits the article
5.3.1.3. Return to step 4.6

5.4. There is an Error with the Article Submission
5.4.1. If there is an error with the system and it results in an error during the article submission in step 4.5.
5.4.1.1. The webpage will display an error stating that an error has occurred during the submission and request that the user resubmits the article
5.4.1.2. Return to step 4.5

Data Definitions

Name	Meaning	Usage	Comment
Visitor	actor	Use case scenarios	This person is new and not familiar with the website.

User	actor	Use case scenarios	The person knows how to use the service. A regular of the website.
Non-member	actor	Use case scenarios	Have an account but no subscription to the website.
Member	actor	Use case scenarios	Have an account and subscription to the services of the website.
Author	actor	Use case scenarios	Uploader of news and information to the website.
Account	Data	Use case scenarios	Store user information.
New	Data/service	data and site user services	For users to read. Owned by the author (the uploader).
Upload	Data/service	data and user services	For authors and bloggers to upload the news and articles to the website.
Rating	service	Site user service	Allow users with an account to rate the articles.
Comments	service	Site user service	Allow users to leave their options.
Photo	Data/service	data and user services	For the user to see and the author to upload them.
Search	service	Site user services	Allow users to find the input keyword.

Log in	service	Site user service	Allow users to comment, rate, and access certain articles based on the account.
Sign up	service	Site user service	Allow users to create an account for the website.
Paywall	service	Site user service	Users will need to pay to see the article.
Website	User Interface	User interface	Front end display for user interaction.
Homepage	User Interface	User interface	First thing that users see when visiting the website.
Information Page	User interface	User interface	Page that is displayed when an activity is clicked.
System	platform hardware and services	Use-case scenarios	The mySQL database, all code, front end design and back end supporting services.
Striker	Domain Name	Use-case scenarios	Name that represents all web pages and the web site
Striker.com	Production server	Use-case scenarios	The server that stores all of the data.

High Level Functional Specifications

Non-Member Expectations

1. **Creating Account- Priority-1**
   - **1.1.** The system shall allow the user to create an account by storing UserID, Password, Date of Birth, First Name, Last name, and answer to security question/phone number and if the account is for a user or content creator (done automatically based on email). The system should not allow the User to Create an account if the UserID was chosen by the user already exists in the System's Database. Also, the system shall prevent the user from creating an account if the user's chosen password does not match the re-enter password field. The system shall prevent the creation of the user's account unless all entries are filled.
   - **1.2. Stimulus Response Sequence**
     - 1.2.1. User enters a UserID (same as email)
     - 1.2.2. User enters a Password
     - 1.2.3. User re-enters Password for confirmation
     - 1.2.4. User shall enter their First and Last Name
     - 1.2.5. User shall enter their date of birth
     - 1.2.6. User shall provide to an answer to given Security Question or provide their phone number
     - 1.2.7. System shall check is email is authorized to be for content creator privileges
     - 1.2.8. System shall check if UserID is available
     - 1.2.9. System shall validate Password
     - 1.2.10. System shall store user Name, date of birth, and answer to selected security question/phone number and if they are a reader or content provider
     - 1.2.11. System shall confirm that the account was created to the User
     - 1.2.12. System shall automatically redirect the user back to the home page
     - 1.2.13. If the user is a content creator they shall be directed to an additional page to enter more information regarding receiving payment.
   - **1.3. Functional Requirement Label**
     - 1.3.1. REQ 1 Creating Account
2. **Browsing Stories- Priority-1**
   - **2.1.** The system shall sort news stories based on the date in an ordered list or a popularity algorithm. Content creators and users will not be able to alter the order of the list as an algorithm does this within the system. The

stories will have an image, headline, and a small summary associated with them.

2.2. Stimulus/Response Sequence - Browsing Stories

2.2.1. Logged in user can see homepage with list of stories they may choose to purchase

2.2.2. System generates list of stories based on date or popularity

2.3. Functional Requirement Label

2.3.1. REQ 2 Browsing Stories

3. Search Stories- Priority-2

3.1. The user will be able to enter an article title and the system will return a story with the matching title

3.2. Stimulus/Response Sequence - Search Stories

3.2.1. User enters the article title in the search bar.

3.2.2. System returns article with matching search term.

3.3. Functional Requirement Label

3.3.1. REQ 2.1 Search

Reader Member Requirements

4. Choose the Selected Story- Priority-1

4.1. The system has provided a list of news stories available for purchase. The user may choose to click on any story where they will be taken to a paywall associated with the story.

4.2. Stimulus/Response Sequence-Choosing Story

4.2.1. User views list of stories provided by the system and added by content providers.

4.2.2. User clicks the story they would like to purchase.

4.2.3. System takes the user to a paywall associated with the story.

4.3. The user will be able to search for a story by typing keywords into a search bar, the system will return a list of stories with matching keywords.

4.4. Stimulus/Response Sequence-Search

4.4.1. User will type the keyword in the search bar located on the header and hit enter.

4.4.2. System will return list of articles with keyword

4.4.3. User may select story using the select story method mentioned above

4.5. Functional Requirement Label

4.5.1. REQ 3.1 Selecting Story.

4.5.2. REQ 3.2 Search.

5. **The Paywall- Priority-2**
   - **5.1.** The system has provided a paywall associated with the chosen story. The user shall enter their payment information and once that information is entered and processed by a third party payment system the system will associate the user and the chosen story and allow the user access and view the page.
   - **5.2.** Stimulus/Response Sequence-Choosing Story
     - 5.2.1. System takes the user to a paywall and provides a third party payment service to fill out.
     - 5.2.2. The user will fill out their payment information
     - 5.2.3. Once the payment has been processed, the system will associate the story with their account
     - 5.2.4. Once the story is associated with the account, the will be provided with a button to view the story
   - **5.3.** Functional Requirement Label
     - 5.3.1. REQ 4 Selecting Story.
6. **Viewing Story- Priority-1**
   - **6.1.** The user shall be able to view the page if they have permission to view the story which occurs when they have already paid for it. When the user clicks the link provided to the story on the list of stories they will be directed to the story. If they have not paid for the story they will be instead taken to the paywall associated with the story. The Stories will be created by content creators and will be stored in a database within the system.
   - **6.2.** Stimulus/Response Sequence-Viewing Story
     - 6.2.1. User clicks link to story
     - 6.2.2. System checks if the user has paid for the story.
     - 6.2.3. If the user has the permission to view the page the system will allow the user to view it.
     - 6.2.4. If the user does not have permission to view the story the system will take them to the paywall associated with the story.
   - **6.3.** Functional Requirement Label
     - 6.3.1. REQ 5 Viewing Story.
7. **Commenting on a Story- Priority-1**
   - **7.1.** Underneath each story will be a comment section where users will be able to comment on the story they read.
   - **7.2.** Stimulus/Response Sequence- Commenting on a Story
     - 7.2.1. User types in a comment within a comment field
     - 7.2.2. The user will be able to click post and the comment will be saved an uploaded to the comment section

7.2.3. The system will save the comment to a database and will post it on the page

7.2.4. Other users will be able to view the comment when they visit the story.

7.3. Functional Requirement Label

7.3.1. REQ 6 Commenting on a Story.

8. Profile-1

8.1. The user will have a profile page they will be able to access by clicking on a link within the page. The profile page will provide a list of articles they have purchased. The user will also have an option to be able to edit their password. Usernames will not be able to be changed unless requested through customer support.

8.2. Stimulus/Response Sequence - Profile

8.2.1. User clicks profile link on header

8.2.2. System takes the user to the profile page that contains a list of articles they have purchased.

8.2.3. The user may click an option to change their password and will be asked to enter their old password, and the new password twice to verify it.

8.2.4. System will Save the old password.

8.3. Functional Requirement Label

8.3.1. REQ 7 Profile

Content Provider Requirements

9. Creating Content Provider Profile- Priority-1

9.1. The profile creation will be the same as a reader but the content provider will be further asked to fill information about receiving payouts. Content Providers will have all the same permissions associated with reader accounts but will have the ability to create content.

9.2. Stimulus/Response Sequence- Profile(content provider)

9.2.1. The user will fill out profile and when the users is verified to have content provider privileges they will be asked by the system to provide payout information

9.2.2. Once the additional information is provided the system will give the account access to a page where they can create articles.

9.2.3. The system will return the user to the homepage automatically

9.3. Functional Requirement Label

9.3.1. REQ 8 Creating Content Profile

10. Creating Content- Priority-1

10.1. Content providers will have an additional button on the homepage to take them to a page where they can write content. On the creating content page there will be a wysiwyg text editor that will generate html pages that will be saved to the Mysql database.

10.2. Stimulus/Response Sequence - Creating Content

10.2.1. The user will be able to see a button to click on the homepage to create content

10.2.2. The user will click the content creation button

10.2.3. The user will be redirected to a page to create content.

10.2.4. The user will type the article in the text editor

10.2.5. When the article is completed the user will hit the submit button

10.2.6. The editor will generate an html page and will save the content to the mysql server and users will be able to view the article on the homepage

10.3. Functional Requirement Label

10.3.1. REQ 9 Creating Content

Support

11. Contacting Customer Service- Priority-3

11.1. Users will be able to contact the developers via email. Users shall type in their name, email, and their query. System shall store these fields and submit it once the user clicks the send button. System shall prevent the User from contacting developers directly. Users shall have to submit a ticket and the ticket shall be redirected to the correct personnel.

11.2. Stimulus/Responsive Sequence

11.2.1. User shall navigate to Contact Us link on the footer

11.2.2. System shall redirect the user to a contact page

11.2.3. User shall fill in Name, Email, and type up their Query

11.2.4. User shall then click “Send” button underneath the Query box

11.2.5. System shall store the information and submit

11.2.6. System shall state that the information was sent and thank the user

11.2.7. System shall a button to redirect the user to Home page

11.3. Function requirement label

11.3.1. REQ 10 Contact Us

List of Non-Functional Specifications

Performance:

1. **Load Speed:** The ability to move between web pages within the system should take around 600 MS to 1.3 seconds to complete. Reaching the main feed should take around that time frame as well as loading into articles.
2. **Responsive Time:** The system will remain responsive no matter the resolution or type of device in place. Whether it be a resolution of 1920x1080 or 1792x828, the site will remain responsive.

Security Requirements:

1. **Account Based System:** The site, Striker, would be using an account system in order to keep track of comments and other rating metrics. To ensure the protection of user personal data:
    a. A security question and phone number will be needed to create an account.
    b. The security question will be required in order to reset the password if needed.
    c. Passwords will be required to be 8 characters long and include both uppercase and lowercase characters.
    d. Password hashing will be put in place to protect the user's account from malicious intent.

2. **Payment Protection:** With payment processing being a part of our system, protection would need to be put in place to ensure the safety of our users. To ensure this, the site will not store any information regarding a user's preferred payment process.

3. **User Privacy**: In order to ensure the privacy of the user, all user data will only be accessible to the user and the development team. The user would also have the option to omit their full name for a different display name in order to ensure anonymity.

4. **Spam Protection:** In order to ensure the safety of the user from bot spam accounts:
    a) A captcha will be asked on the creation of an account using the reCAPTCHA API.
    b) To protect against spam in comments and articles, both of these systems will require an account to access and use.

Storage:

1. **Data Storage:** All of the data used within this system will be stored within the MySQL database in the LAMP Server.
2. **File Storage:** Files will primarily be held within the LAMP server. All files used will be backed up to GitHub and Google Drive to ensure its safety.

Availability:

1. **Uptime:** The website, Striker, will be accessible 24 hours a day, 7 days a week.
2. **Downtime:** This system does not expect much downtime to occur unless LAMP is unavailable. If downtime is necessary, an announcement will be shown on the page prior and a redirection to a separate page stating the site is down will occur during downtime.

Ease of Use:

1. **Training:** Minimal training time would be required in order to use our system. The only required training would be how to create and upload articles to our site. Past this, the site would be user-friendly and usable by new users.

Interoperability Requirements:

1. **Browser Compatibility:** This site, Striker, would display and run properly on web browsers Google Chrome, Mozilla Firefox, and Safari.
2. **Device Compatibility**: This system should display and run properly on multiple devices, such as mobile devices, desktop computers, and laptops. Design would stay consistent throughout desktop computers and laptops, with it slightly deviating on mobile.

Fault Tolerance:

Exception Handling: If an error were to occur at any moment, an error code will be displayed to the user. Using this error, the user will be able to troubleshoot the issue.

Supportability Requirements:

1. **Code Standards:** This system will be programmed using HTML, CSS, and PHP in order to accomplish its goals. Each group member will be using the data names mentioned previously in code and documentation.

Expected Load:

1. **User Load:** The site will be able to handle a maximum of 35 people before performance begins to degrade. Even if performance begins to degrade, no crashes are expected to happen due to user load.

High-Level System Architecture

1. **FAU LAMP Server:** We will use the provided group account on FAU's LAMP server to host the entirety of Striker so that it can be accessed by anyone who has the link to the website.
2. **Laravel:** This will be the PHP framework that will be used to write the backend code for Striker. It is written in PHP and will be used to securely interact with both the frontend and the MySQL server.
3. **Bootstrap:** This will be responsible for the frontend of Striker, as it makes designing a fashionable, but also functionable website quick and easy.
4. **Brackets:** This will be the go-to text editor for HTML, CSS, PHP, and JavaScript.
    a. HTML — will be the backbone for Striker; it would not work without it.
    b. CSS — will be used to implement Bootstrap for styling purposes.
    c. JavaScript — will be responsible for intractable elements throughout Striker.
    d. Personal Home Page (PHP) — will be the language used for server side functionality to interact with the database and allow for real time edits in the tables within the Mysql database.
5. **TinyMCE:** This API will be used as the "What You See Is What You Get" (WYSIWYG) text editor that will be used to allow authors to create content.
6. **MySQL:** This will be what comprises Striker's database and will also be used for anything regarding addition or modification of data in the database.
7. **JIRA:** Will be used in order to coordinate tasks and responsibilities between all of the group members that will be working on the project together.
8. **GitHub:** Will be used as a code repository to allow easy collaboration — whether it be addition, deletion or modification of code. It will also allow us to keep track of changes of the code base in a chronological order
9. **Google Firebase Authentication:** The API that will take care of logging in and authentication; making it swift and trivial for any and all to log into Striker.
10. **Stripe:** The API that will be responsible for everything related to payments, providing a secure and trusted method of paying for articles on Striker. It also will allow Striker to keep all critical financial information separate from it's database.

11. **Supported Browsers:** Striker will be compatible with as many browsers that are in use by many end-users, such as Google Chrome, Safari, Firefox, and Microsoft Edge.

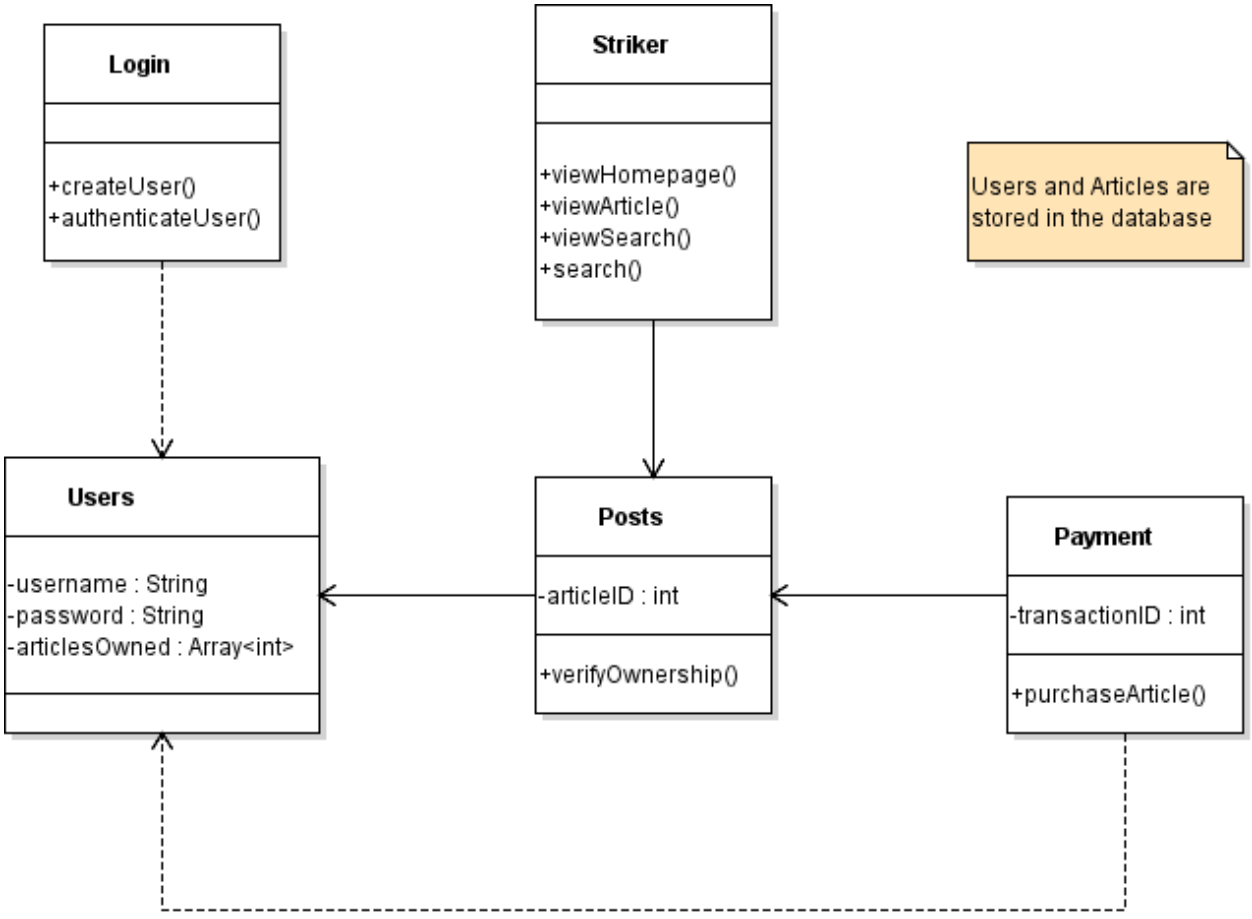
Database Organization:

Table Name	Key	Name	Data Type	Null	Description
Users		first_name	varchar(20)	N	
		last_name	varchar(20)	N	
	U	username	varchar(20)	N	Not Applicable
	U	email	varchar(70)	N	
		password	varchar(20)	N	Not Applicable
	P	firebase_uid	varchar(30)		
		auth_question1	int	N	
		auth_ans1	varchar(255)	N	
		auth_question2	int	N	
		auth_ans2	varchar(255)	N	
Posts	P	p_id	varchar(30)		
		firebase_uid	varchar(30)		

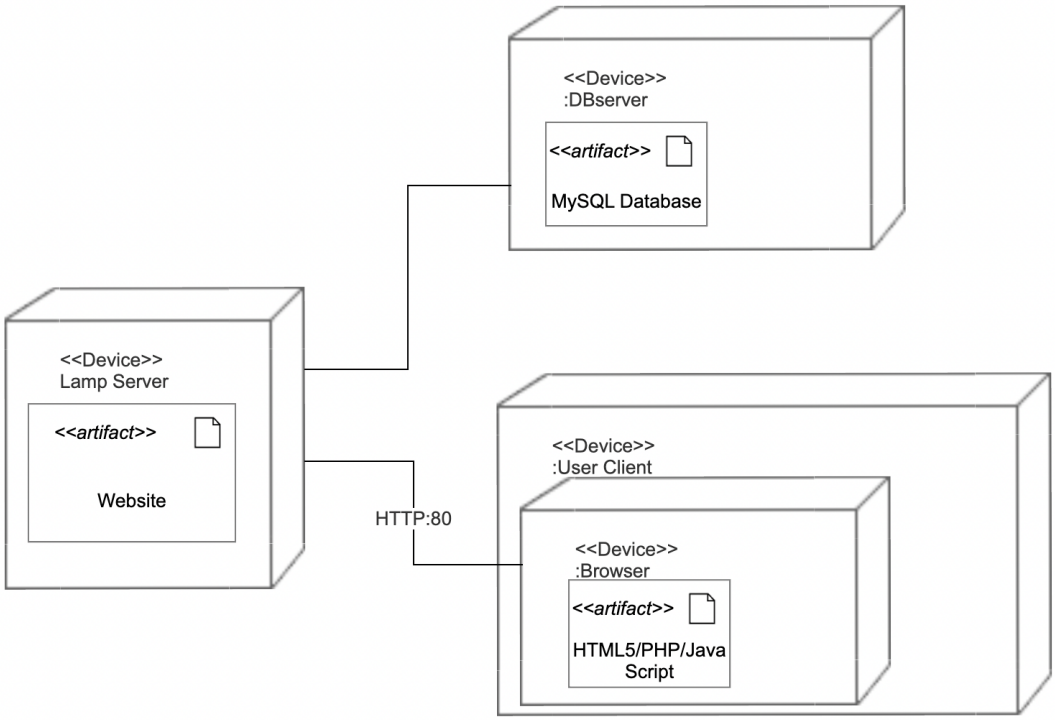
		p_title p_content p_posted_by p_date price	varchar(255) text varchar(255) timestamp double		Limit to 50,000 char
Comments	P	c_id firebase_uid c_content c_posted_by c_date	varchar(30) varchar(30) text varchar(255) timestamp		Limit to 1,000 char
Payments	P	transaction_id firebase_uid p_id transaction_date price	varchar(255) varchar(30) varchar(30) timestamp double		

High Level UML Diagrams

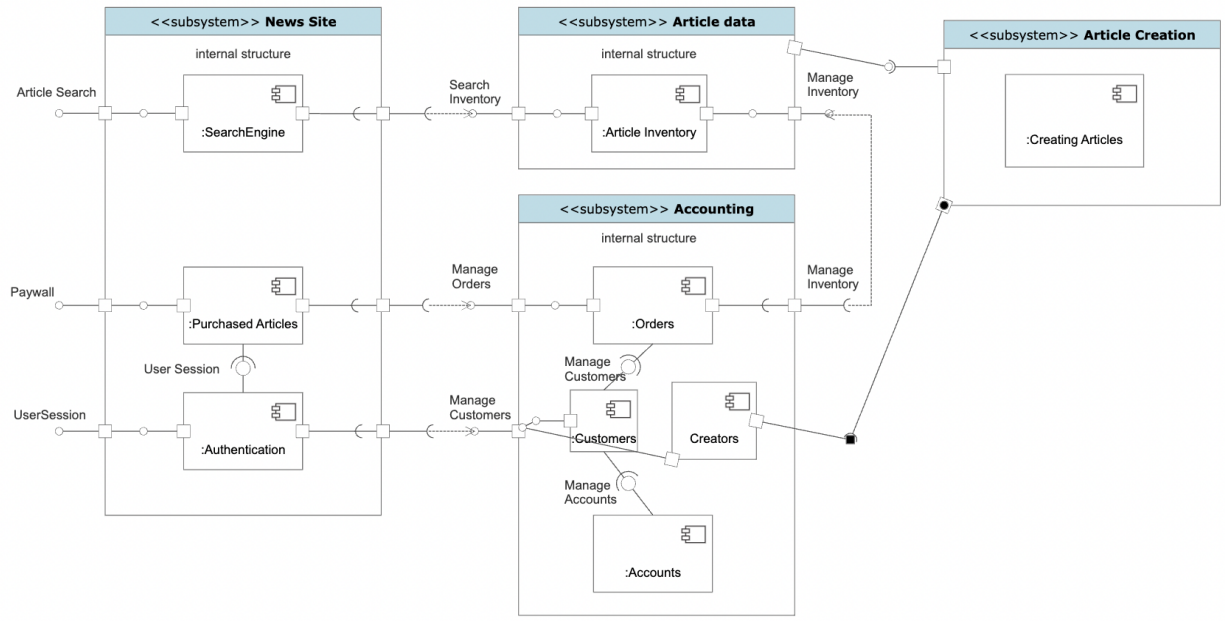
Class Diagram:



UML Deployment:



UML Component:



Key risks

1) Skills risks (do you have the right skills),

It is challenging to work with the provided outdated software. To learn more about programming, everyone agreed to work with python, and we don't have much experience with the language in developing software.

2) Schedule risks (can you make it given what you committed and the resources),

Everyone did the assigned tasks and followed the schedule. There could be unexpected events to delay the working process.

3) Technical risks (any technical unknowns to solve),

VMware limited what users could do, it is difficult to do everything in it. Software might not integrate correctly could result in a serious setback.

4) Teamwork risks (any issues related to teamwork),

Everyone is well communicated and fast at giving feedback. Teamwork risk could be each individual could not complete the assigned part, due to personal problems.

5) Legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright).

Legal rights risks- might have issues with ownership of content.

Tell us how do you plan to resolve risks? The key is to resolve risks as soon as possible. Categorizing risk as above helps a lot in managing them. Be brief: identify the risk and explain (2-3 lines), list how will you address these issues (2-3 lines).

Risks are inevitable from a lengthy project. First, we plan out the who, what, and when, to get a general idea so that risks could be visualized inside of a scope. Secondly, the group constantly communicates if there is any problem, and assists each other to avoid unnecessary risks. Lastly, if the team could not resolve any of the risks listed above, we will communicate with the professor intermediately to figure out the best possible solution.

Competitive Analysis

While our idea is unique, and we believe that our platform has the potential to be disruptive to the media landscape, there are existing media outlets that perform a similar function. Our main competitive advantage is our monetization model, but it is still necessary to compete on similar metrics such as readability and ease of use. Our platform is most similar to substack or medium, where individuals create single-page articles and post them on the platform. The significant difference between them and our platform is that we do not tie the article to the author for readers to subscribe to. Instead, each article must be purchased individually. WSJ/NYT represents legacy media outlets that are on a subscription-based model.

The analysis of competitors' websites will focus on six main features (homepage, readability, comments, payment model, multimedia, ease of use, variety of content). The competitive analysis will utilize a numerical scale (1=bad, 2=poor, 3=fair, 4=good, 5=outstanding) and consists of three websites chosen for their focus text media.

Metrics	Substack	Medium	WSJ/NYT	Striker
Homepage	5	1	5	5
Readability	5	4	3	5
Comments	4	5	3	3
Payment Model	1	1	1	5
Multimedia	2	3	5	2
Ease of use	5	3	4	4
Variety of Content	2	4	4	5
	3.4	3	3.6	4.14

Striker(4.14)

No direct competitors are pursuing our model, and the relatively quick build time will likely not result in a site as refined as others. The main goal is to create a platform that has the potential to disrupt the media market and improve from there. Other goals in the future are to create a pricing algorithm based on the article's popularity and perceived quality to encourage content creators to make both quality and quantity. We believe that this platform has the potential to solve many of the problems related to the subscription-based model in that it is only feasible for a minority of creators to rely on subscriptions for their sole income source and the limits to how many subscription services people are willing to pay for. With the movement away from traditional media outlets and the general decline in trust among established institutions, we believe the door has opened for less established outlets to succeed. Alternative media is growing while traditional outlets are struggling to keep their viewers engaged. We think our platform can be an excellent avenue for lesser-known independent journalists to get their content out to a broad audience and make a decent living off of their work.

Substack(3.4)

Substack is a website where users can subscribe to different newsletters. It has excellent readability and ease of use but can get very expensive if you want to read other authors consistently. Subscriptions add up quickly at a price between 5-10 dollars a month, and there is no guarantee that an author will be prolific and create quality content consistently.

Medium(3)

Medium is a platform where people can pay to read more than five articles a month. However, we feel that while this is a good deal for readers to be able to read a large quantity and diversity of content, the vast majority of writers cannot generate significant income for their content as that 5 dollars is spread over many different articles. Therefore, our monetization model is much more direct and lucrative to attract better writers.

WSJ/NYT(3.6)

Legacy news outlets rely heavily on subscriptions and a large staff to create good content. However, with notable veterans of these outlets making up to 10 times their salary on Substack, we feel that they will start struggling to attract talent to create great content. Our model allows for the same direct payment model similar to Substack, but without the commitment that comes with a subscription.

Team Roles

Team Leader, Front & Back End Developer:

- Nicholas Giacobbe

Front End Lead, Group Lamp Server Admin:

- Alex Rodriguez

Back End Lead, Github Master:

- Luiza Trebesqui

Front & Back End Developer, Scrum Master:

- Shay Reardon

Front & Back End Developer:

- MySQL Admin - Truong Nguygen

Checklist

a. Team decided on basic means of communications - DONE

b. Team found a time slot to meet outside of the class - DONE

c. Front and back end team leads chosen - DONE

d. Github master chosen - DONE

e. Team ready and able to use the chosen back and front-end frameworks - DONE
f. Skills of each team member defined and known to all - DONE
g. Team lead ensured that all team members read the final M1 and agree/understand it before submission - DONE