SpoofSpotter

by

Ned Farrell, Bradley Davidson, Jeff Benton, Noah Muse

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Ned Farrell	4/24/2022	*Bradley Davidson*	4/24/2022
Ned Farrell	Date	Bradley Davidson	Date
Jeff Benton	4/24/2022	*Noah Muse*	4/24/2022
Jeff Benton	Date	Noah Muse	Date

Ryan Moore, Faculty Advisor — 4/24/2022 Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
April 2022

Table of Contents

Table of Illustrations

Abstract

The spread and unintentional consumption of misinformation has increased drastically in the past decade. The cause of this is not well known. According to a 2019 poll, 67% of Americans believe fake news causes a great deal of confusion, while only 26% percent of Americans feel “very confident” in their ability to recognize fake news. (Watson 2019) To combat this issue, we created SpoofSpotter. SpoofSpotter is a multi-platform application that is taking a proactive approach to combatting the unintentional spread and consumption of misinformation. Users are provided with a wide variety of curated news headlines, in which they will use their skills to determine which of these are not real. Failure to determine the fake news will direct users to a training module that will 1.) Teach them what they got wrong 2.) Teach them how to prevent this from happening in the future. 3.) provide other skills and information regarding dangers surrounding the spread and consumption of illegitimate information. We have successfully developed the core features for our app, allowing users to proactively learn how to spot fake information.

Senior Design Final Report
2021-2022

Project Name: SpoofSpotter

Project Summary:

In recent years, the spread of misinformation via social media and other online platforms has grown substantially. As this problem has gotten worse, massive corporations like Google, Twitter, Facebook, Reddit, etc. have allocated hundreds of millions of dollars towards combatting this across their various platforms. The most common way to fight this is some form of fact-checking, which is reactive by nature. Our team proposes the following proactive solution:

A multi-platform application that teaches users to proactively spot misleading information. By prompting users with a combination of real and fake news headlines. Users will attempt to determine the validity of these headlines, and upon getting any question wrong they will be given resources and information on how to spot misleading and/or fake headlines. The focus of this application is the teaching and training element that it provides. Quizzes are provided daily, prompting users with 4 options containing 3 real headlines and 1 fake or potentially misleading headline. Users will attempt to determine the fake option. Additionally, there is a continuous newsfeed that provides a curated feed of real and fake headlines – prompting the user to determine the validity of each. The inability to correctly determine the real or fake headline will result in the user being directed to a personalized training module based on the information they got incorrect. The goal of this app is to proactively teach users to spot misleading information in news headlines. Starting at the headline can eliminate the need

to even read an entire story. Machine learning will be used to generate headlines that will be used as questions and marked as fake. Machine learning will also be employed to determine whether scraped articles are misleading or not.

Problem Statement:

In the past decade, misinformation has become easier to share and unknowingly consume using social media and other online platforms. According to Ardèvol-Abreu and others, around 46% of people have unintentionally shared false or exaggerated information about COVID-19 at some point (Ardèvol-Abreu, 2020). To combat this, fact checking services have very recently been adopted by larger social media sites such as Facebook, Twitter, YouTube, Reddit, etc. The issue in this lies in the fact that fact checking can only happen after potentially false information has been shared. Depending on how quickly false information is flagged as false, a variety of users may have already digested the information as truthful. This can lead to a chain reaction of users posting false information under the guise of it being legitimate, purely due to the reactive nature of fact checking. Without proactive training in how to recognize false information, the spread of misinformation will continue to occur.

Solution:

SpoofSpotter is a multi-platform application that teaches users how to spot potentially misleading, biased, or incorrect information. To improve our users' ability to determine misinformation, SpoofSpotter provides a variety of news headlines, prompting users to differentiate real versus fake. Upon answering correctly, users gain points, streaks, and add to their personal and friends' leaderboards. If a user is unable to determine the validity of a news headline, they will then be coached on why they may have been incorrect. This provides them

with information on spotting, combatting, and general dangers/issues surrounding misinformation. The focus of this is to provide a proactive approach to addressing the issue of misinformation while also ensuring an engaging environment for users. This allows us to teach our users in a way that does not necessarily have to feel like learning.

Project Source:

Our team was created prior to coming up with any ideas for a project. Initially, our team consisted of one developer and two cybersecurity tracks. We had a brainstorming session and decided that we wanted to create a project that dealt with computer and/or internet literacy. We landed on making an app that trains users to spot fake information after talking about the new fact checking tools that Facebook and Twitter have recently implemented. From there, we decided to reach out for one more developer to aid with the software side of this project.
We completed a formal requirements analysis and adjusted our existing objectives for what we want the app to do. To adhere to more modern development practices, we remained agile when it comes to our development cycles. This means that we constantly received feedback from our peers and people from many demographics throughout the project's life cycle rather than trying to get a list of static requirements all in one go.

Project Objectives/Goals:

- Train users to spot fake information in real time
    - Create a page that users can use as a reference for how to spot fake information.
    - Redirect users to this page upon guessing incorrectly in a daily quiz.
    - Tailor the information on the page to be relevant to the question that they got wrong.

    - Personalize training results based on the type of question that was missed.
- Encourage regular engagement with SpoofSpotter
    - Implement a streak system that gives some sort of bonus or reward for daily engagement.
    - Create a leaderboard system so users can compete against themselves and their friends for the highest score.
    - Implement an analytics page for users to view their personal statistics and see what areas they are excelling at and where they can potentially improve.
- Supply an endless stream of headlines that users can use to get many different examples
    - Include a swiping function so users can swipe left for a fake or misleading headline, and right for something that looks real.
    - Include buttons for web users that may not use the phone application
- Maintain a curated database of relevant news headlines
    - Create a daily task that will scrape various news outlets for relevant headlines.
    - Filter out older headlines to keep the complete collection fresh.
    - Create relevant “fake” headlines using both automation and manual entry.
    - Use headlines from a variety of topics (sports, politics, science, etc. …)

Project Scope:

Our team developed a functional multi-platform application using industry-standard development practices. The Ionic framework is used in conjunction with the Angular TypeScript framework to develop this app. We used Google Firebase for both our authentication and

storage for things such as user information, quiz questions, and data regarding fake news. We abide by secure design practices by ensuring that our Firebase console is locked down and securely connected to our front-end application through security policies. A series of Python scripts have been developed to perform web scraping on various news sources to flesh out our collection of news headlines in our database. We also used Python scripts to manage our databases and ensure that they are only storing relevant news headlines by removing any information that is old. Machine learning scripts were written to both generate fake news as well as determine whether the scraped articles are real or misleading. These scripts are run daily from a Raspberry Pi, while the rest of our app is hosted on the cloud. We have multiple "game modes" to the app to test user knowledge. The daily quiz mode will present the user with multiple news headlines and the user must detect if there is any misinformation present. The quiz is to be of a fixed length and will give the user feedback on what they missed and how they can avoid it in the future. There is a "News Feed" mode where users are presented with a single news headline. They must then swipe left or right to determine whether the headline is fake. This is meant to imitate the gestures used in dating apps such as Tinder. Lastly, we present the user with statistics on commonly missed headlines and categories both from their own news feeds and from news feeds presented to our other users.

Quick Project Timeline:

Task #	Task Name	Duration	Start Date	End Date
1	Consider testing methodology and project deployment	~1 week	9/20/2021	9/27/2021
2	Determine User Personas and Use cases	~1 week	9/27/2021	10/4/2021
3	Build functional framework	~3 weeks	10/4/2021	10/25/2021
4	Write Unit Tests	~1 week	10/25/2021	11/1/2021
5	Write Technical report	~7 weeks	9/27/2021	11/8/2021
6	Get feedback on our app design from peers/potential users	Ongoing	10/28/2021	3/21/2022
7	Iterate on feedback	Ongoing	10/28/2021	3/21/2022
8	Implement web server scripts	~1 week	11/1/2021	11/8/2021
9	Complete functional prototype	~7 weeks	10/25/2021	12/10/2021
10	Test prototype	~2 weeks	1/10/2022	1/24/2022
11	Update Final Report Draft	1 Week	1/10/2022	1/16/2022
12	All features implemented	~8 weeks	1/24/2022	3/21/2022
13	Add Testing Plan	1 week	1/30/2022	2/6/2022
14	Finalize IT Expo Abstract	1 week	2/6/2022	2/13/2022
15	Perform User Acceptance Testing	3 weeks	2/6/2022	2/27/2022
16	Create IT Expo Poster Draft	1 week	2/20/2022	2/27/2022
17	Finalize Poster after feedback	2 weeks	2/27/2022	3/13/2022

18	Finalize testing and development	~2 weeks	3/21/2022	4/4/2022
19	Record IT Expo presentation	1 week	3/27/2022	4/3/2022
20	In-class final presentation	1 day	4/4/2022	4/4/2022
21	Finalize Final report draft	1 week	4/3/2022	4/10/2022
22	Present at IT Expo	1 day	4/12/2022	4/12/2022
23	Submit Final Report to library	1 day	4/24/2022	4/24/2022

Technologies Used:

- Firestore: Relational database to store headlines, user information, activity logs, etc. …
- Firebase Authentication: Service platform to handle user logins and storage of passwords
- Python: Scripts to scrape news headlines from various websites.
- Ionic: Multi-platform app development tool to build the framework of our app. The activity layer will also handle the functionality of the front end. This will allow us to also make our project a web page.
- CapacitorJS: This plugin handles native implementations of features across multiple platforms in conjunction with Ionic.
- TensorFlow: A library from python that allows us to do natural language processing.
- Visual Studio Code: Source code editing software
- GitHub: Used as a central repository for application code and python scripts
- CircleCI: Used to run continuous unit testing on our GitHub repository.

Technical Architecture Diagram:

The following section will detail the technical architecture of the various components that work together to power SpoofSpotter. This section will include technical diagrams *(Figure 1),* our database schema *(Figure 5),* and sequence diagrams *(Figure 2-4)* pertaining to common tasks and core functionalities of SpoofSpotter.

Figure 1: Network Architecture

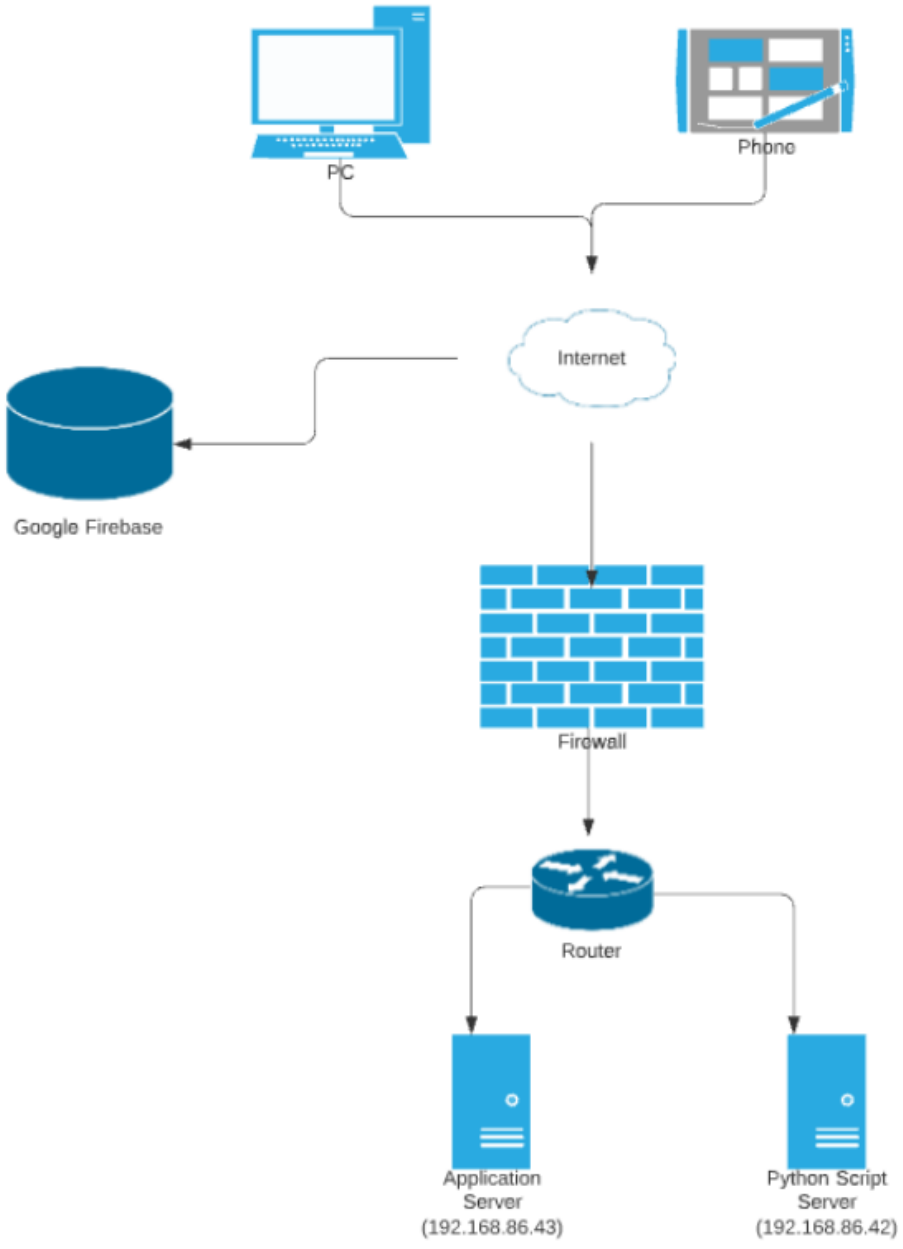


Figure 2: News Feed Sequence Diagram

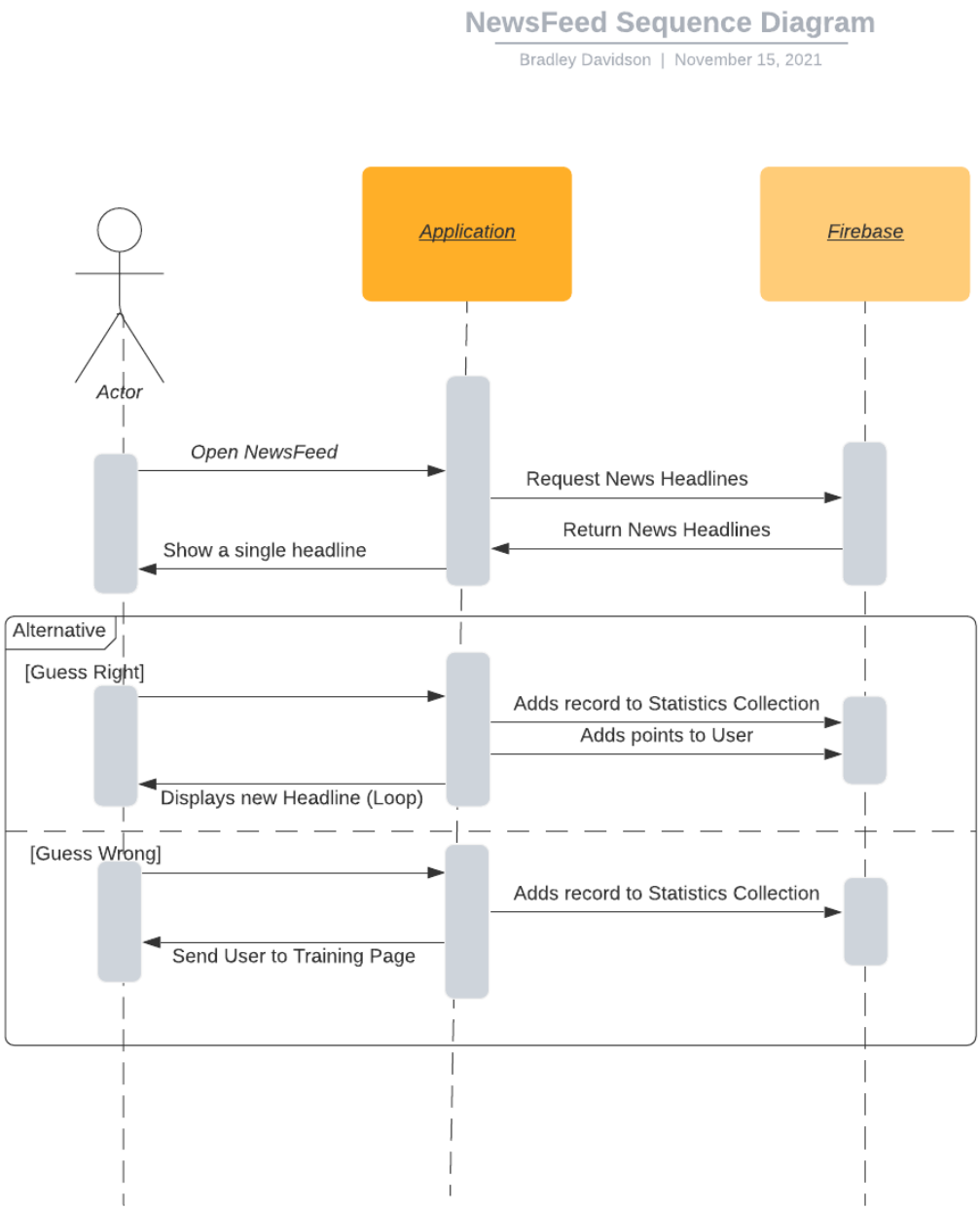


Figure 3: Daily Quiz Sequence Diagram

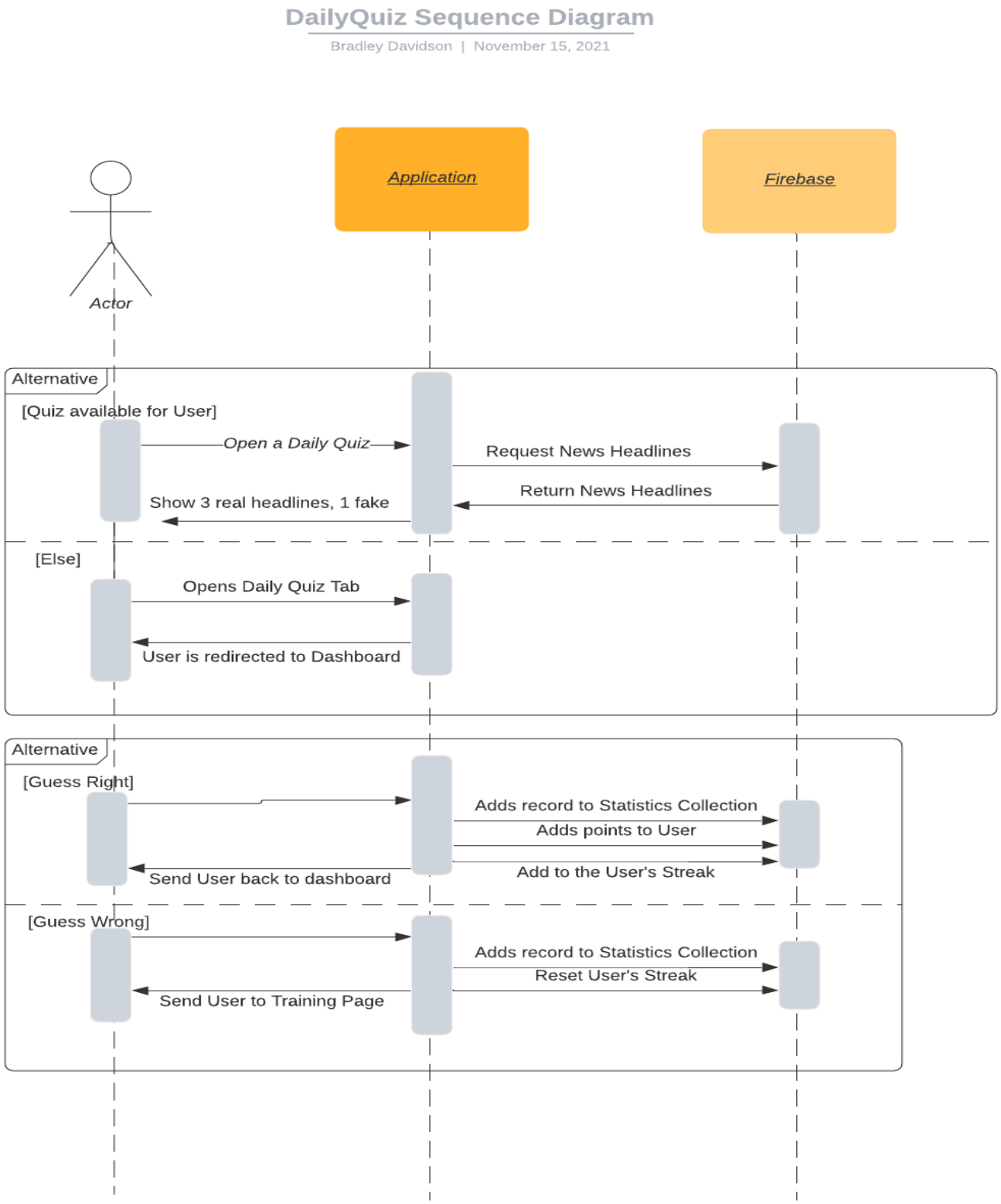


Figure 4: Manage Account Sequence Diagram

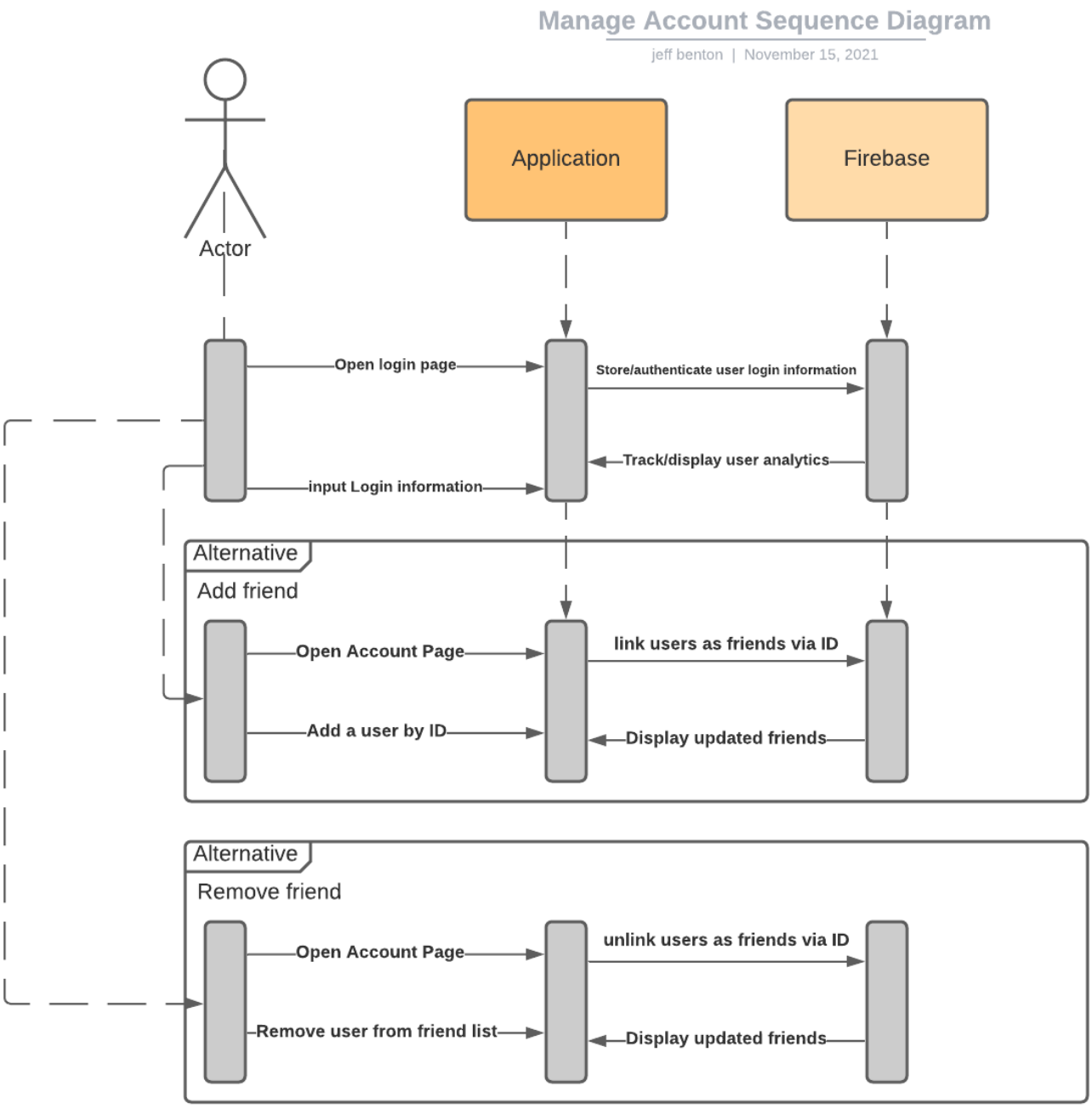
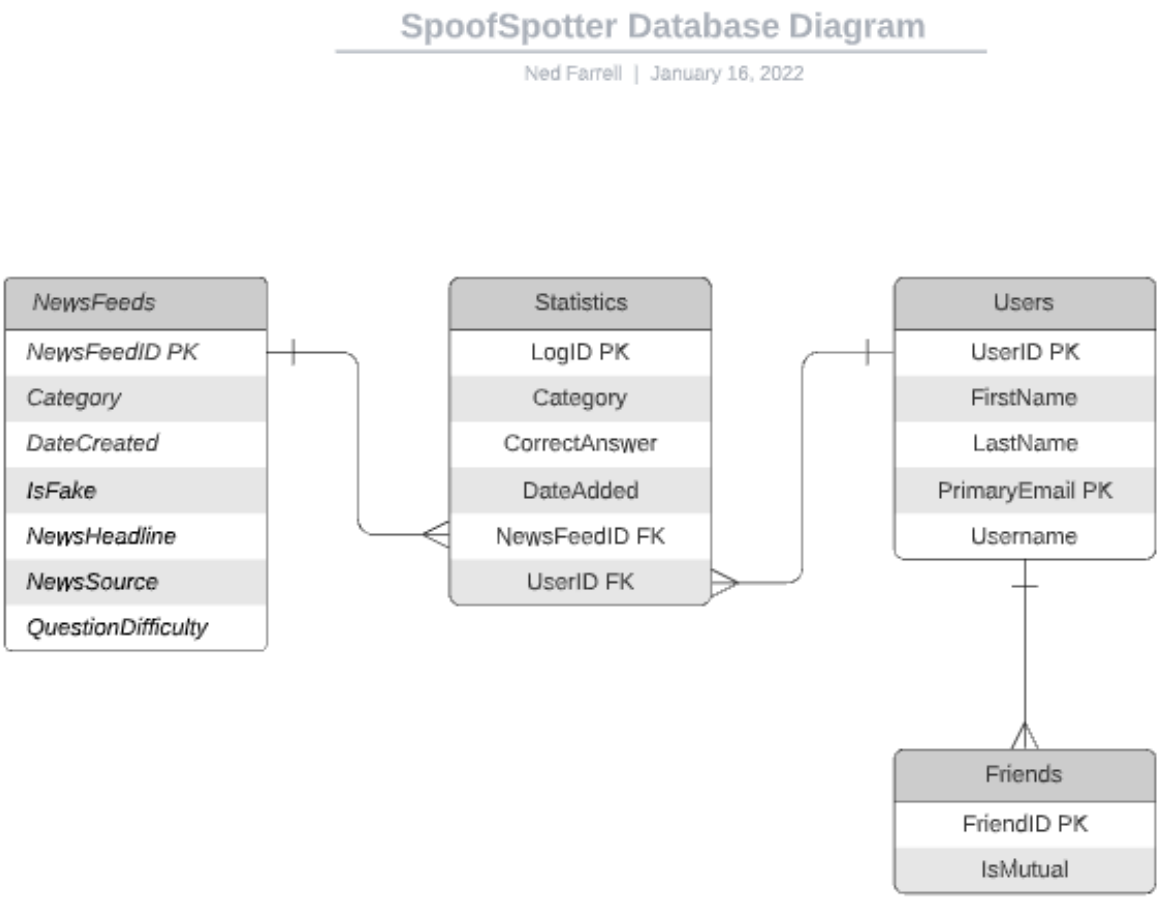


Figure 5: Database Schema



User Personas:

Figure 6: The Lonely Grandparent persona

The Lonely Grandparent

Job Title
Retired

Age
Age 65 or older

Highest Level of Education
High school degree or equival

About

- Grandma Betsy lives by herself
- She has 3 children and 11 grandchildren that are spread acros the country
- Betsy spends a lot of time on social media to catch up with her children, grandchildren and friends.
- Most of the news and media she consumes is from Facebook

Goals

- Consume, discuss and share news, media, video clips, etc. to friends online.
- Interact with her family whenever she can
-

Environment

- Betsy spends a lot of time on her phone or tablet
- Loves to share and talk about all types of different news stories they see on facebook
- Can sometimes mistakingly share incorrect or biased information without realizing it.

Figure 7: The Weary Student persona

The Weary Student

Job Title
Full-Time Student

Age
19

Highest Level of Education
High School Diploma

About

- Dan is a full time student, enjoying his first year at the University of Cincinnati
- He lives with his best friends on campus and spends a lot of time socializing with new groups
- Dan hates social media and has a hard time believing some things he reads online

Goals

- Explore various news topics
- Learn to spot misinformation or potentially biased news soruces/articles
- Stay caught up on news, world politics, sports, local happenings, etc.

Environment

- Dan has to frequently use different forms of social media to stay caught up with school activities
- He has to spend a lot of time researching topics for school assignments
- Surrounded by a lot of differing views/opinions while he is trying to form his own.

Figure 8: The Political Enthusiast persona

Use Cases:

Figure 9: Answer Question use case

Name	Answer Question
ID	UC_001
Description	User determines whether headline is truth or false
Actors	User
Organizational Benefits	Fulfill purpose of application
Frequency of Use	At least once a day
Triggers	The user is presented with a quiz question
Preconditions	
Postconditions	Answer is evaluated
Main Course	1. User answers correctly a. If user is logged in, increment streak value

	2. Next question is presented
Alternate Courses	AC1 User answers incorrectly 1. User answers incorrectly 2. User is redirected to the page informing them how to spot misinformation
Exceptions	

Figure 10: Add Friend use case

Name	Add Friend
ID	UC_002
Description	User adds another user to their friends list
Actors	User
Organizational Benefits	Increase retention rate by allowing users to interact and compete with each other
Frequency of Use	
Triggers	The user hits the add friend button
Preconditions	User is logged in
Postconditions	The friend added is visible in the user's friend list and database is updated
Main Course	1. User hits the Add Friend button 2. Dialog box prompts user to input friend ID 3. User enters the ID of their friend 4. User hits Add 5. FriendsList table is updated with a new listing 6. System redirects to friends list
Alternate Courses	AC1 User adds friend from their profile 1. Friend is added and remains on the profile screen 2. FriendsList table is updated with a new listing AC2 User is not logged in 1. Redirect user to login screen with error message 2. User logs in 3. Return user to main course step 2
Exceptions	EX1 Friend does not exist 1. System notifies user of error 2. Return user to main course step 2

Figure 11: Remove Friend use case

Name	Remove Friend
ID	UC_003

Description	User removes friend from their friends list
Actors	User
Organizational Benefits	Allows users to choose who their leaderboard is comprised of
Frequency of Use	
Triggers	User selects remove friend on friend's profile
Preconditions	User is logged in and is friend with this user
Postconditions	Friend is removed from user's friend list and database is updated
Main Course	1. User navigates to friend's profile 2. User hits "Remove Friend" 3. FriendsList table is updated with a new listing 4. Screen is updated and user remains there
Alternate Course	
Exceptions	

Figure 12: Log In use case

Name	Log In
ID	UC_004
Description	User enters their username and password to log in
Actors	User, Authentication Service
Organizational Benefits	Users can save and track their progress
Frequency of Use	
Triggers	User hits log in button or performs action that requires logged in state
Preconditions	User has an account already
Postconditions	User is logged in
Main Course	1. User enters username 2. User enters password 3. User hits "Log In" button 4. User is now logged in
Alternate Course	AC1 User enters incorrect information 1. User hits "Log In" button 2. Error message is presented notifying user that the username/password combination is wrong
Exceptions	

Figure 13: Create Account use case

Name	Create Account
ID	UC_005
Description	User enters information and registers in the database

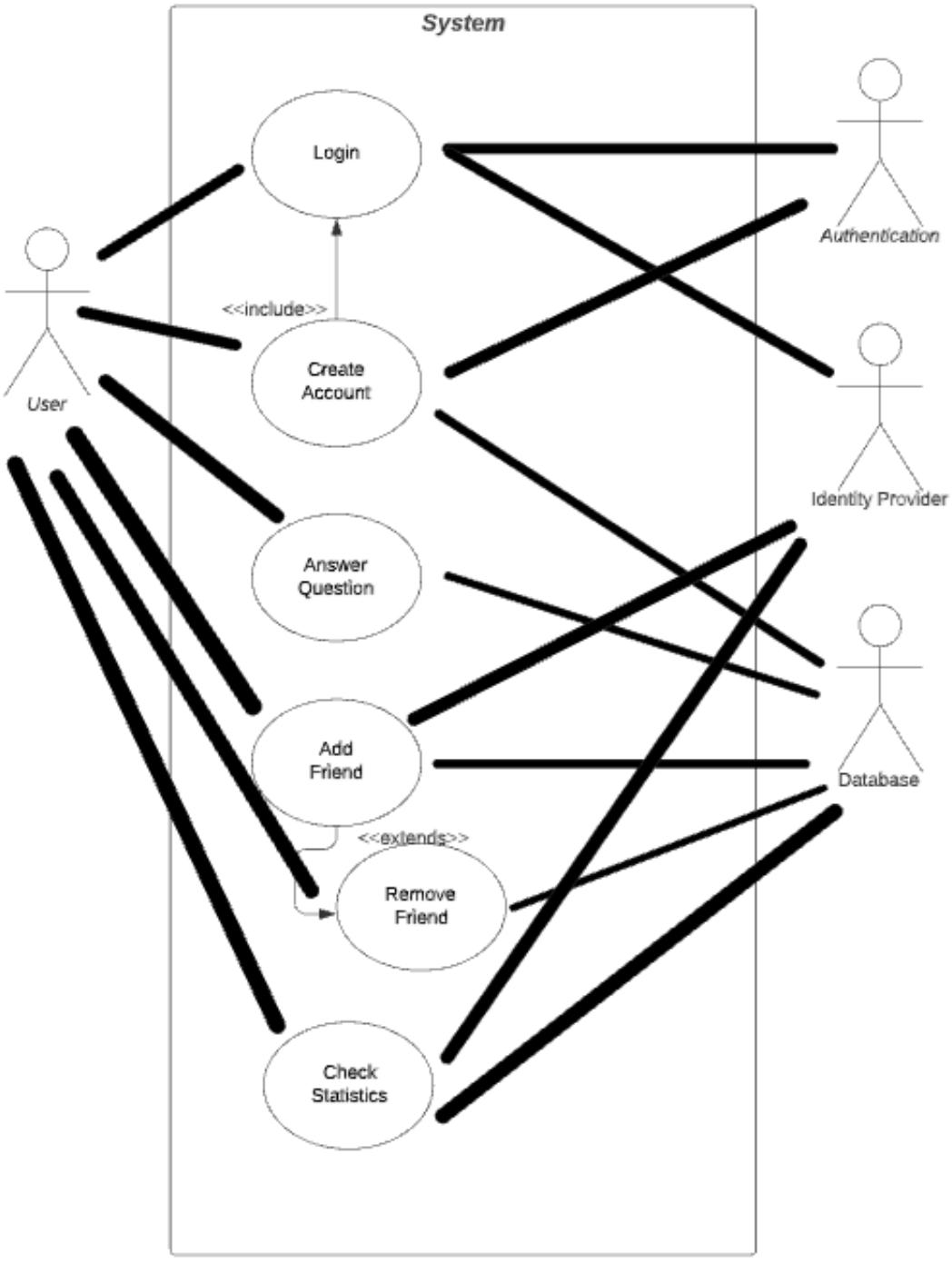
Actors	User, Firestore
Organizational Benefits	Users can save and track their progress
Frequency of Use	Once
Triggers	User hits “Create Account” button
Preconditions	User is not logged in
Postconditions	User is logged into new account and database is updated
Main Course	1. User enters email address, username, password a. First and last name are optional 2. Database is updated with user information 3. Password is stored securely with Firestore 4. User is redirected to home screen logged in
Alternate Course	
Exceptions	

Figure 14: Check Statistics use case

Name	Check Statistics
ID	UC_006
Description	User checks their statistics on questions answered
Actors	User, Firebase
Organizational Benefits	User can gain understanding of their progress
Frequency of Use	
Triggers	The user navigates to the Statistics page
Preconditions	
Postconditions	
Main Course	1. User navigates to Statistics page 2. Firebase retrieves information on user's questions 3. Chart and statistics are shown to user
Alternate Courses	AC1 User is not logged in 1. User navigates to Statistics page unlogged in 2. User is shown anonymous data for other non-logged in users
Exceptions	

Use Case Diagram:

Figure 1515: Use case diagram



Testing Plan:

Overview:

This section contains information regarding testing and evaluation of the SpoofSpotter application, along with a general overview of the quality assurance plan. The goal of this plan is to ensure high quality code with a streamlined process of testing. Our agile testing methodology will cover functional and nonfunctional testing. Functional testing includes unit testing, integration testing, system testing and acceptance testing and will be primarily handled by the implementation of CircleCi and with the help of Ionic's built-in UI testing feature. Non-Functional testing will include performance, security, usability, and compatibility testing. This plan will ensure that SpoofSpotter is running optimally for its users.

The use cases that we will be testing will be:

- Answer Question
- Add Friend
- Log In
- Check Statistics

We want to test these cases because they are what we consider to be the key features of our app. Without these working, we will not be able to meet our main goals of the app. We will additionally use these cases in our user acceptance testing so that we can ensure that the app runs smoothly for actual users and that all our UI elements are intuitive and easy for users to use. Through this testing, we can ensure that the app is both functional and usable.

Methodology:

Our project implements continuous testing through the integration of CircleCI. This allows us to run and document unit tests upon each commit to our GitHub repository. An added benefit to the usage of CircleCI is that it marks commits within our repository as passing or failing, so we know ahead of time whether pushing a commit to production will break aspects of our code.

We chose this approach so that our tests will be completely automated, which will prevent us from ever forgetting to run tests manually. Additionally, since CircleCI is a web-based platform all members of our group will be able to easily access both our testing procedures and our testing history.

For user acceptance testing, we provided peers with a set of important tasks to complete, with minimal guidance as to how to complete these tasks. This is to test both the functionality of our app as well as the usability of our UI. We will then get back to these users and ask whether they were able to complete each task successfully. Additionally, these actions should be logged to our system.

Scope:

Our testing primarily focused on service level functions and our connection to Firebase. Examples of things that were tested for our project include the following: retrieving a daily quiz for a user, retrieving a news headline, adding a friend, or authenticating a login request. As our app is made through Ionic, our app automatically integrates Angular's unit testing for UI components. We still manually ensure that UI components look correct on all platforms but testing whether the component loads should happen automatically.

For user acceptance testing, we are essentially using the same use cases as our unit testing. As our unit testing already covers whether UI components are loaded in, the user acceptance testing is more so that we can ensure that the components are both responsive and easy to understand. Completing these test cases requires users to do things like navigate to different pages within the app, create login credentials for the app, use swipe gestures, and use the skills that our app is trying to develop to critically read news headlines (allowing us to test the effectiveness of our app as well).

Objectives:

The main objective of testing SpoofSpotter and our QA plan is to discover any issues within code, the potential existence of bugs, errors, functionality, feature, UI/UX issues, or anything that may impede on the successful operation of the application.

a) All major use cases should be accounted for.
b) All logic happening within the service level should be tested.
c) All UI components should be compatible with Ionic's cross-platform distribution.
d) All app-breaking bugs should be resolved before IT Expo.
e) Testing should happen within separate branches before merging into our production branch.
f) Users should understand the app's functionality and features.

Our objectives for user acceptance testing are as follows:

a) Ensure that the key functions of our app are clearly defined.
b) Verify that our UI components are intuitive
c) Test that our application works on multiple devices with different OS versions.

Test Logs and Procedures:

Figure 16: Test Logs and Procedures

Use Case #	Test Case	Input	Expected Output	Actual Output	Pass/Fail	Date
001 (Answer Question)	Answers Question	Correct answer	Present "Correct" Alert -> get next question	Present Correct alert -> get next question	PASS	10/27, 11/12, 1/25
001	Answers Question	Incorrect Answer	User is redirected to a page teaching them to spot misinformation.	Present "Wrong" alert -> present next question, or navigate to training	PASS	10/27, 11/12, 1/25
001	Answers Question	Correct Answer	Daily answer Streak is increased	Streak increase	PASS	12/25
002	Add Friend	The user adds a friend that doesn't exist.	System notifies user of the error -> returns to the add friend page.	System notifies user of the error -> returns to the add friend page.	PASS	2/06, 3/13
002	Add Friend	User adds a friend that exists	User is now "Following" their new friend, can see friend's stats	User is now "Following" their new friend, can see friend's stats	PASS	2/06, 3/13
004	Log In	User enters a correct username and password	The user logs in.	User is logged in	PASS	11/12, 12/06, 3/13
004	Log In	User enters an invalid username or password.	The user is notified that their entries aren't properly formatted	The user is notified that their entries aren't properly formatted	PASS	11/12, 12/06, 3/13

004	Log In	User enters information that doesn't exist.	The user is notified that the login information is incorrect.	The user Is notified that the login information is incorrect.	PASS	11/12, 12/06, 3/13
006	Check Statistics	User answer	Statistics page is updated with correct information.	Statistics page is updated with correct information.	PASS	1/25, 2/07, 3/13
005	Create Account	User registers a new account	Account is created and stored in firebase	Account is created and stored in firebase	PASS	1/25, 2/07, 3/13

User Acceptance Test Logs:

Figure 17: User Acceptance test logs

ID	User	Device	Test Case	Pass/Fail	Comment(s):	Difficulty of Task (1[easy]-5[difficult])
001	Ned	Android	Answer Question	Pass	Answered question correctly and system responded appropriately	2
002	Tyler	Web	Create Account, Login	Pass	User successfully created a new account and logged in.	1
003	Tyler	Web	Answer Question (Newsfeed and DailyQuiz)	Pass	User completed a daily quiz and multiple questions in the news feed.	2

004	Tyler	Web	Check Statistics	Pass	User successfully checked their personal statistics after answers questions	3.5
005	Andrew	iPhone	Create Account, Login	Pass	User successfully created a new account and logged in.	1
005	Andrew	iPhone	Answer questions (NewsFeed and DailyQuiz)	Pass	User Successfully completed the daily quiz and newsfeed portions of the application.	2
006	Steven	Web	Answer questions (NewsFeed and DailyQuiz)	Pass	User Successfully completed the daily quiz and newsfeed portions of the application.	2

Testing Review:

Through testing, we know how our app functions as of now and were able to get a better understanding of how it should work. We want to continuously test and make sure that everything meets our expectations.

Change Management Plan:

- Who can request changes:
    - Team members
    - Advisors
    - Users
- What circumstances allow a change:
    - Users can request a change if there is a feature that they would like implemented
        - User feedback will provide a beneficial communication channel that will allow us to listen to and implement user suggestions.
    - Users can request a change if there is a critical bug that affects the way the app behaves
        - Any major bugs or prohibiting issues within the application should be discussed and acted upon immediately within the group.
        - Smaller bugs/QOL issues/features will be handled based on the severity of the problem. The team will determine the severity of individual issues and solve them orderly based on their findings.
    - Developers can request a change for any reason
        - Any code changes should be well documented
            - Ensuring there are no issues committing/pushing when submitting changes to code.

        - Major changes should be discussed and agreed upon, prior to finalizing a decision regarding a potential change.
            - Major changes may include but not be limited to:
                - Feature addition/deletion
                - Changing of technologies used
                - Any major changes to overall applications theme, look, or views.
        - Minor changes should be discussed prior to finalizing a potential change.
            - Minor changes may include but not be limited to:
                - Small coding changes
                - Minor tweaks/QOL improvements
    - Advisors can request a change if there is a technical aspect that needs to be addressed
- How we will triage the change:
    - Creating/logging a change request
    - Review change request
    - Assessing the change risks
    - Approval of change
    - Implementing the change
    - Closing the change request
    - Review the change

- How will changes be communicated:
    - Major changes should be discussed and agreed upon prior to any final decisions regarding potential changes.
    - Minor changes should be notified to the rest of the group, detailing what was changed, why, and when the change occurred. (Change log)
    - Changes will be well documented.
    - If the change is related to the contract, the advisor should be notified

Budget:

This analysis will detail the financial requirements of SpoofSpotter. Hardware Costs, Software Costs, Misc. Expenses, and Labor Costs will all be considered in this analysis. Workstations, Internet, Photoshop, and our server are noted in the analysis and were not purchased exclusively for the project but are vital in its completion. Please note that at our current capacity, SpoofSpotter is using the free plan from Firestore. Allowing 1GB of data storage, 10k authenticated users, 600,000 reads, 1.5M writes – but this price can increase anywhere from $5-$35 a month depending on the amount of information we are storing. We do not anticipate having to pay for this service, but it should be noted.

Figure 18: Budget analysis table

Budget Analysis				
Item	Unit	Unit Cost	Total	Annual Costs
Hardware				
Workstation	4	$750	$3000	
Device/Workstation to act as server (Laptop)	1	$300	$300	
Software				
FireStore	1	Free	Free	Free
Photoshop	1	19.99	$19.99/mo	$199
GitHub, VS Code, Microsoft Teams, Android Studio	1 instance Per member	Free	Free	Free
Misc.				
Internet	1	60.99	60.99/mo	$738.88
Labor				
Project Development Team	Members	Rate/hr	Hours	Total
	4	$20	350	$7000 (one time)
Project Maintenance	4	$20	180	$3600 (per year)
Total	Hardware		$3,300	
	Software		199.99	
	Misc		738.88	
	Labor		$10,600	
	Total		$14,838.87	

Problems Encountered and Analysis of Problems Solved:

Problems Encountered:

1) Issues with automated testing: While our project is integrated with CircleCI, giving us automated testing every time we make changes to the project, Firebase has made any testing outside of initialization testing difficult. By nature of how Angular projects implement karma for unit testing, the actual components are often loaded with live data (As a result of code within the initialize functions of the components) making testing reliant on actual dynamic data instead of mocked objects. This could potentially be addressed by switching off karma for our testing, but this would be out of character for Angular projects.
2) Challenges involved with multi-platform interfaces: Ionic is a very powerful tool that has allowed us to develop for a variety of different platforms simultaneously with minimal platform-specific worries. The challenge that this brings up, however, is that user interfaces for mobile applications are fundamentally different than interfaces for web applications. There are ways in which platform specific styling can be applied, but this comes at the cost of maintainability as it is much harder to maintain two versions of an interface.
3) Security considerations: As ionic/angular is a completely client-side codebase, there are certain things we must figure out regarding security. Specifically, we are unsure of how to secure our Firebase config keys exactly. While we can hide them from prying eyes in our GitHub repository by simply putting them into a .gitignore file, we still must contend with the fact that variables from the environment.ts file will be served to the client

when they use the application. From preliminary research, it looks like there is no great way to secure API keys in the front end. Rather, one of the most common suggestions is to keep the front end insecure but lock down the backend to only work from certain domains and/or only work with certain credentials. There are steps we can take right now towards securing our app, but this is an ongoing process.

4) Challenges with project scope: When first presenting our project to our advisor, the plan was primarily to have quiz functionality in the app. In an effort to expand our scope, we opened our app to other platforms, added other "game" modes within the app, and elaborated on plans to expand the scope of our app into the realm of browser extensions. Some of the aspects of our scope expansion have been more successful than others. We have continued to look for avenues in which we may flesh out our project, but we believe that our initial expansion has been a success and net gain for our project. This will be an ongoing issue likely for the duration of our project.
5) Inclusion of machine learning: In the spring semester, our plan was to include machine learning to both generate news headlines to use in the quizzes and to spot misleading articles as they were scraped. Generating news headlines was a challenge as the product needed to be clean and readable for it to be believable as a real headline. Spotting misinformation on the scraped articles proved to be a challenge in terms of both finding an efficient way to spot and accuracy of reporting it.

Analysis of Problems Solved:

1) Implementation of the Ionic Framework: our technology stack has changed drastically from our first days of this project. We initially planned to just develop an Android app using Kotlin, but after some preliminary discussions with our advisor we decided to add cross-platform support. We tested various methodologies, namely: Kotlin + a standalone web app, ReactJS with React Native apps on mobile platforms, and finally Ionic with an Angular codebase. We decided to use Ionic, as Angular is a very readable codebase for web developers. Additionally, with Capacitor we can automatically port our web application to Android, iOS, and Windows devices.
2) Database solution: In the beginning of our development cycle, we went back and forth on which database solution we should implement. At first, we planned to have a SQL database and .Net Core web services hosted on a private web server for our project. Upon further consideration, we realized that the creation of a standalone .Net Core app was overkill and would add unnecessary complexity to our project. Instead, we decided to use Google's Firebase as a free cloud-based storage solution. This simplifies our project significantly as we can write our business logic within the same application as our front end. Additionally, we can take advantage of the fact that Firebase has native authentication features that allow us to automatically tie in user authentication with our storage calls.
3) Securing the Firebase config key: While this problem is not completely solved at this moment, we at least have a rough plan on how we can at least mitigate the issue. For now, we can at least move the environment.ts file holding the key into the .gitignore so

that it won't be included in our public repository. Next, we need to generate one set of credentials and keep them in a secure location, but also somewhere where only we can access them for reference. Then we obviously need to deprecate the old credentials so that even if they are found, they won't be able to access our Firestore solutions. We still need to research best practices for securely storing the config file or develop a different solution to this.

4) Inclusion of Machine Learning: As we determined that a browser extension wouldn't mesh well with our current iteration of the app, we pivoted to focusing on machine learning by doing the following:
    a) We did a lot of research on ways that we could generate headlines, and we settled on using Natural Language Processing. We followed a tutorial that originally would train a model using horror stories and would generate a plot to a new horror story using a part of one of the stories used to train. After changing the original script to generate headlines instead, it didn't produce very meaningful headlines. Giving the model more time to train and changing the technique to give it a seed text eventually got it to a state where it could produce somewhat legitimate headlines that could almost be believable
    b) The issue with the current setup in SpoofSpotter is that everything that isn't being scraped from satire news sites like The Onion or ClickHole are being marked as real with no way of knowing whether that is true or not. We first tried to implement a machine learning model that would classify a news article fake/real based on headline only. Unfortunately, this was inaccurate and wasn't

going to be an ethical method to implement, as we could incorrectly mark an article as misleading. The current solution does the exact same thing except it evaluates the body text of the article and uses 3 different machine learning algorithms to evaluate the validity of the article.

Conclusion:

In conclusion, SpoofSpotter provides a more proactive solution to the misinformation problem that our world is facing. The SpoofSpotter team has produced a fully operating application that has all its core functionality in working order.

In the duration of this project, there have been many lessons learned, one of the most important being how to work and succeed in an agile environment. Many of us are not yet accustomed to the fast-paced, iterative style that makes up this type of workflow. The project has evolved a lot since its initial formation, the ability to adapt when necessary has been a huge factor in our work to this point. Another important lesson that ties into the first is project time management. Staying consistent with our workflow allowed us to meet our goals and even surpass some of them.

Our general IT knowledge has increased significantly in the creation of SpoofSpotter. There have been moments for each of us where we were working outside of our comfort zones, but this has been beneficial to help us learn new skills in the process. We have learned about user authentication, working within a new framework like Ionic, working with live (web-scraped) data, an unfamiliar database structure, and much more. While we have all had experience working in teams through co-op experiences, the process of developing SpoofSpotter has enabled us to learn better communication skills and teamwork in general, especially with such a small team.

References

Ardèvol-Abreu, A., Delponti, P., & Rodríguez-Wangüemert, C. (2020). El profesional de la información. *Intentional or Inadvertent Fake News Sharing? Fact-Checking Warnings and Users' Interaction with Social Media Content*, *29*(5). https://doi.org/10.3145/epi

Brandtzaeg, P. B., & Følstad, A. (2017). Trust and distrust in online fact-checking services. *Communications of the ACM*, *60*(9), 65–71. https://doi.org/10.1145/3122803

INFORMABLE. (2020, February 25). https://informable.newslit.org/.

Sittmann, J., & Tompkins, A. (2020, July 17). The strengths and weaknesses of automated fact-checking tools. *DW Akademie*. https://www.dw.com/en/the-strengths-and-weaknesses-of-automated-fact-checking-tools/a-53956958.

SmartApp Soft House. (2021, July 15). *Fake news hero - quiz - apps on google play*. Google.] https://play.google.com/store/apps/details?id=it.smartappsofthouse.fakenewshero&hl=en_US&gl=US.

Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, Arnaud Legout. Social Clicks: What and Who Gets Read on Twitter?. ACM SIGMETRICS / IFIP Performance 2016, Jun 2016, Antibes Juan-les-Pins, France. https://hal.inria.fr/hal-01281190

Watson, Amy. "Confidence in Ability to Recognize Made-up News U.S. 2019." *Statista*, 29 June 2020, https://www.statista.com/statistics/657090/fake-news-recogition-confidence/.