




Fake News Detection Software

Lestyn Brooks


Introduction & Motivation

The purpose of this project is to determine whether an article found online is presenting truthful information.

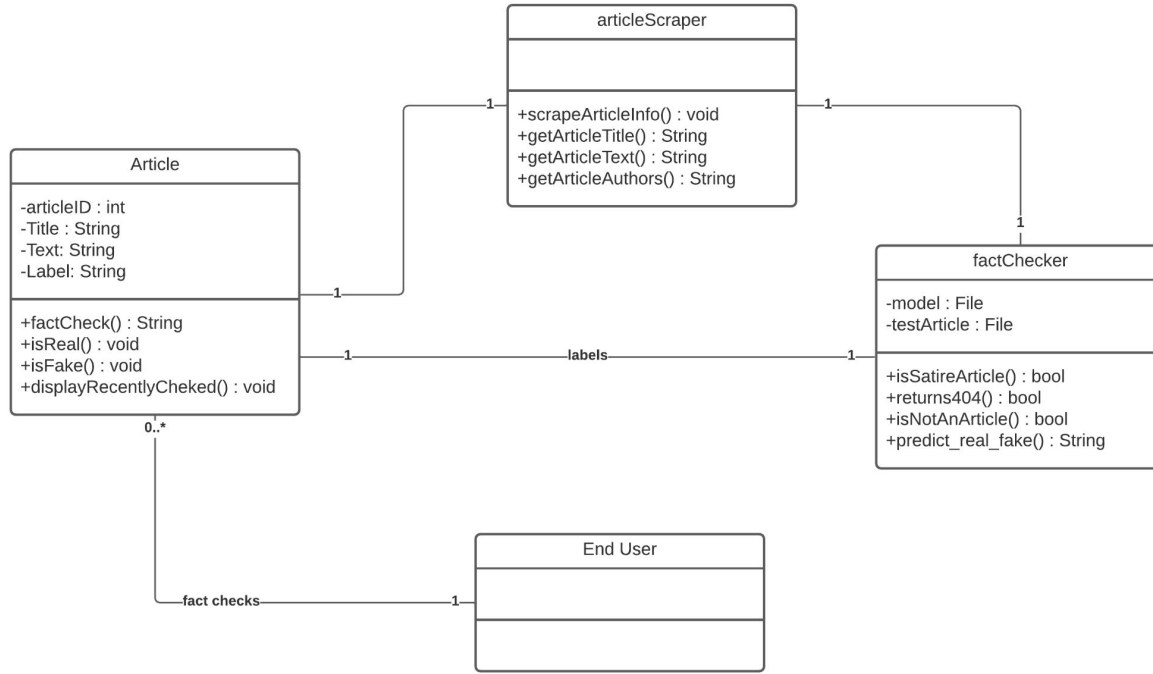
The reason I was motivated to do this as my Senior Project to begin with is because of how polarized the American political scene has become over the last few election cycles. The truth is, this country has become so divided that many elections have come down to who is hated the least by the majority of people. With many people on both ends of the political spectrum posting (and sharing) false and/or heavily biased information online, it's important for people to know if what they're reading is truthful or not.



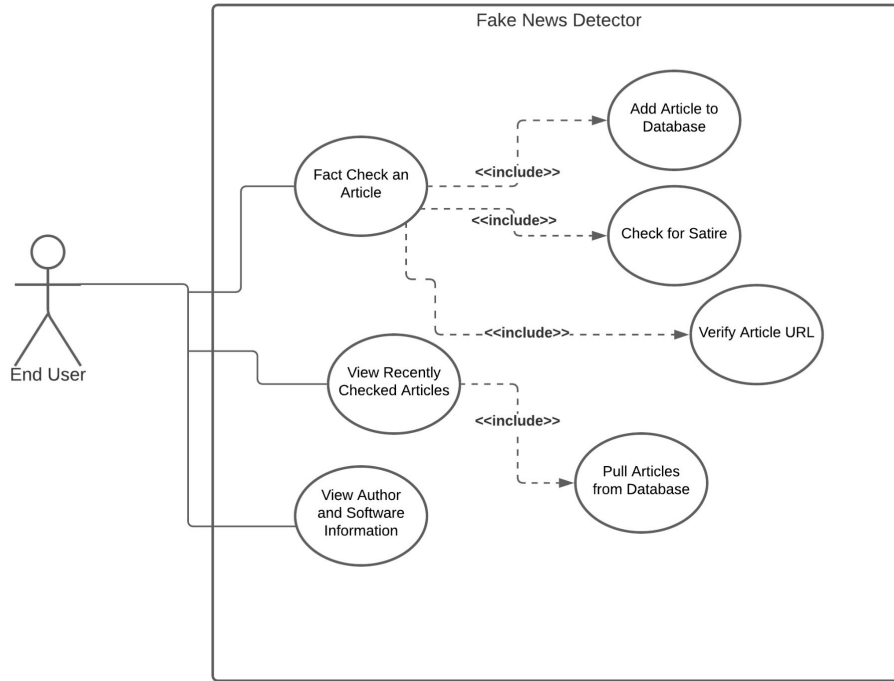
Software Requirements Specification (SRS) Highlights

- The application shall allow End Users to submit a link to an article so that it can determine the article's legitimacy using a machine learning algorithm.
 - The application shall inform the user if the article they submitted comes from a satirical source, such as The Onion or Babylon Bee, so that the End User knows that they are intentionally fake.
 - The application shall allow End Users to see what articles have been previously fact checked and what the results were so that the End User can have as much information as possible about what the algorithm is learning from.
 - Included in the repository is a full SRS document, where all of the remaining functional and non-functional requirements can be found.
- 

Software Design

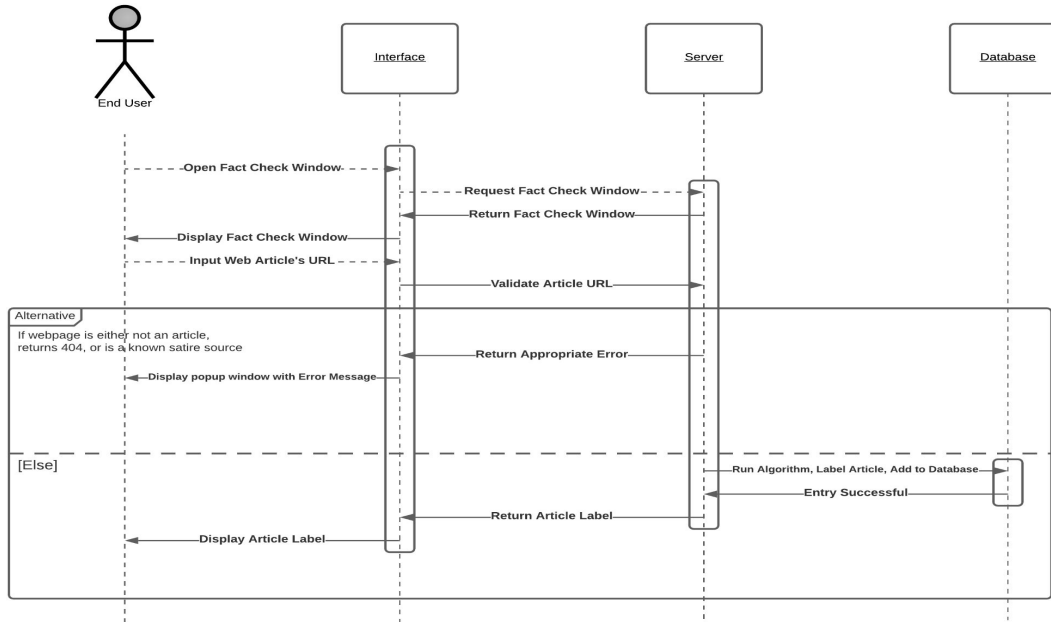


Software Design, Continued



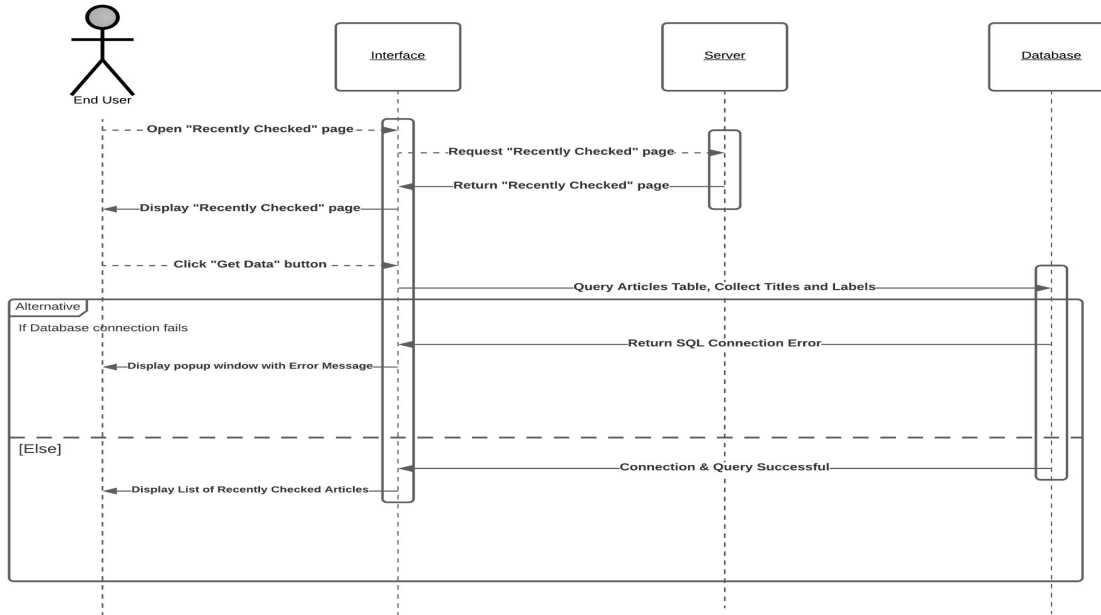
More Software Design

End User Fact Checks an Online Article



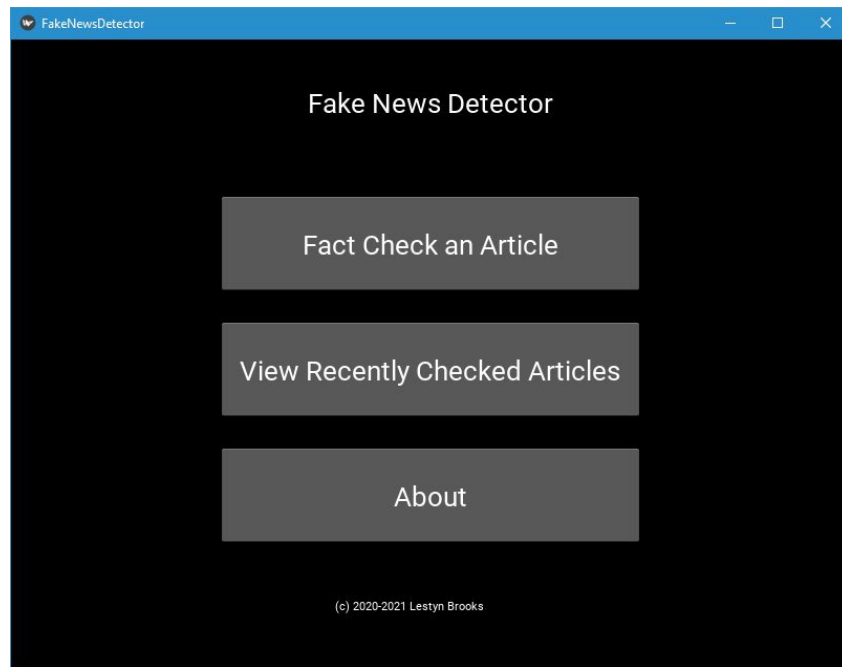
Even More Software Design

End User Views Recently Checked Articles
from the Database



The Solution

- With ease of use in mind, I chose to make a GUI for this software using the Kivy library in Python
- Python was also the language of choice because of its vast amount of machine learning libraries, such as scikit-learn



The Solution - Fact Checking

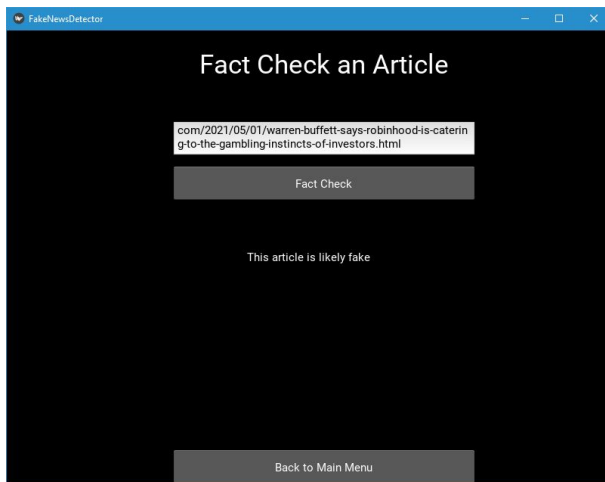
BuildModel.py (key component)

```
X_train, X_test, Y_train, Y_test = train_test_split(data['text'], data.target, test_size=0.4, random_state=42)

pipe = Pipeline([('vect', TfidfVectorizer()),
                  ('tfidf', TfidfTransformer()),
                  ('model', PassiveAggressiveClassifier())])

model = pipe.fit(X_train, Y_train)
prediction = model.predict(X_test)
```

GUI Prediction



FactCheck.py

```
import pickle

model_file = "final_model.sav"
with open(model_file, 'rb') as file:
    model = pickle.load(file)

test_article = "articles.csv"

def predict_real_fake():
    prediction = model.predict([test_article])
    return prediction[0]
```

The Solution - Article Scraping

```
def scrapeArticleInfo(url):
    article = Article(url)

    article.download()
    article.parse()

    with open('articles.csv', mode='w') as article_set:
        fieldnames = ['title', 'text']
        article_writer = csv.DictWriter(article_set, fieldnames=fieldnames)
        article_writer.writeheader()
        article_writer.writerow({'title': article.title, 'text': article.text})

def getArticleTitle(url):
    article = Article(url)

    article.download()
    article.parse()

    return article.title
```

```
def getArticleText(url):
    article = Article(url)

    article.download()
    article.parse()

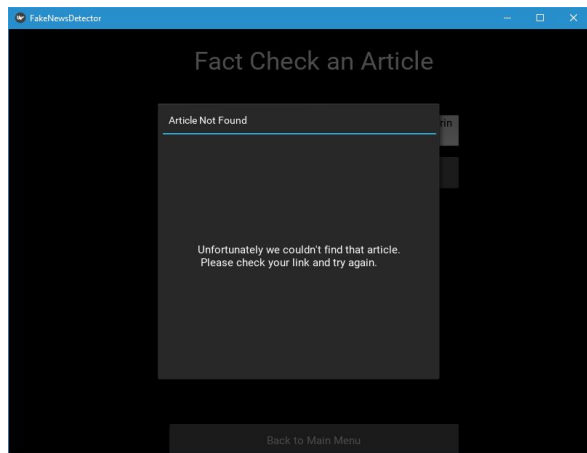
    return article.text

def getArticleAuthors(url):
    article = Article(url)

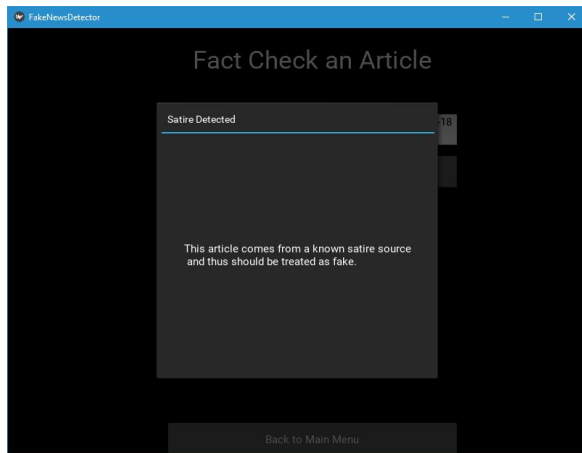
    article.download()
    article.parse()

    return article.authors
```

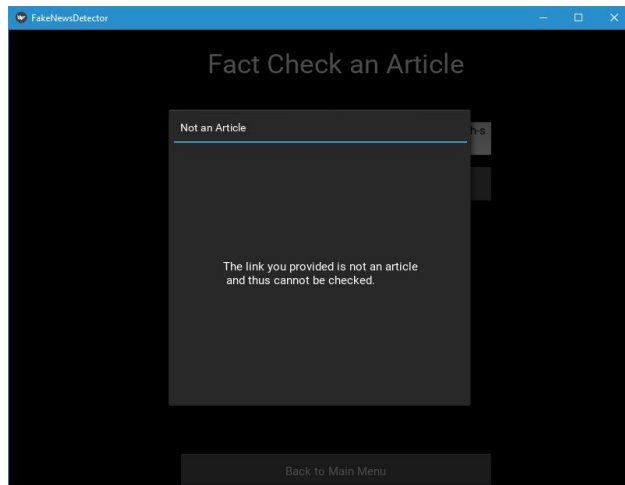
The Solution - Error Handling



```
def isSatireArticle(self):
    satireSources = [...]
    x = 0
    articleIsSatire = False
    while x < len(satireSources):
        if satireSources[x] in self.articleURL.text:
            satireArticle()
            articleIsSatire = True
            break
        else:
            x += 1
    return articleIsSatire
```



```
def returns404Error(self):
    notFound = False
    r = requests.head(self.articleURL.text)
    if r.status_code == 404:
        error404()
        notFound = True
    return notFound
```



```
def isNotAnArticle(self):
    notArticle = False
    if not getArticleAuthors(self.articleURL.text):
        notAnArticle()
        notArticle = True
    return notArticle
```

The Solution - Recently Checked Articles (Database)

Title	Label
May Day protesters demand more job protections amid pandemic	fake
More perilous phase ahead for Biden after his 1st 100 days	fake
India launches effort to inoculate all adults against COVID	fake
Texas holds special election to replace Rep. Wright, who died of COVID	fake
Number of Americans fully vaccinated tops 100 million	fake
Venezuela's 'doctor of the poor' beatified in small ceremony	fake
Determined volunteers still search for capsized ship missing	fake
US to restrict travel from India over COVID starting Tuesday	fake
Get data	Back to Main Menu

```
def displayArticles(self):
    try:
        connection = create_server_connection("localhost", "root", pw)
        cursor = connection.cursor(buffered=True)
        sql_query = "SELECT Title, Label FROM Articles.Article ORDER BY ID DESC LIMIT 8"
        cursor.execute(sql_query)
        connection.commit()
        rows = cursor.fetchall()
        for row in rows:
            self.layout.add_widget(Label(text=str(row[0]), halign='center', text_size=(350, None)))
            self.layout.add_widget(Label(text=str(row[1]), halign='center', text_size=(350, None)))
    except mysql.connector.Error as err:
        print("Error {}".format(err))
```

Questions Before I Continue?

