Localr

Architecture and Design

Version 3.0

Topikós

4/12/20

Members:
Catane
Das
Gary
Jara
Le
McBride

TABLE OF CONTENTS

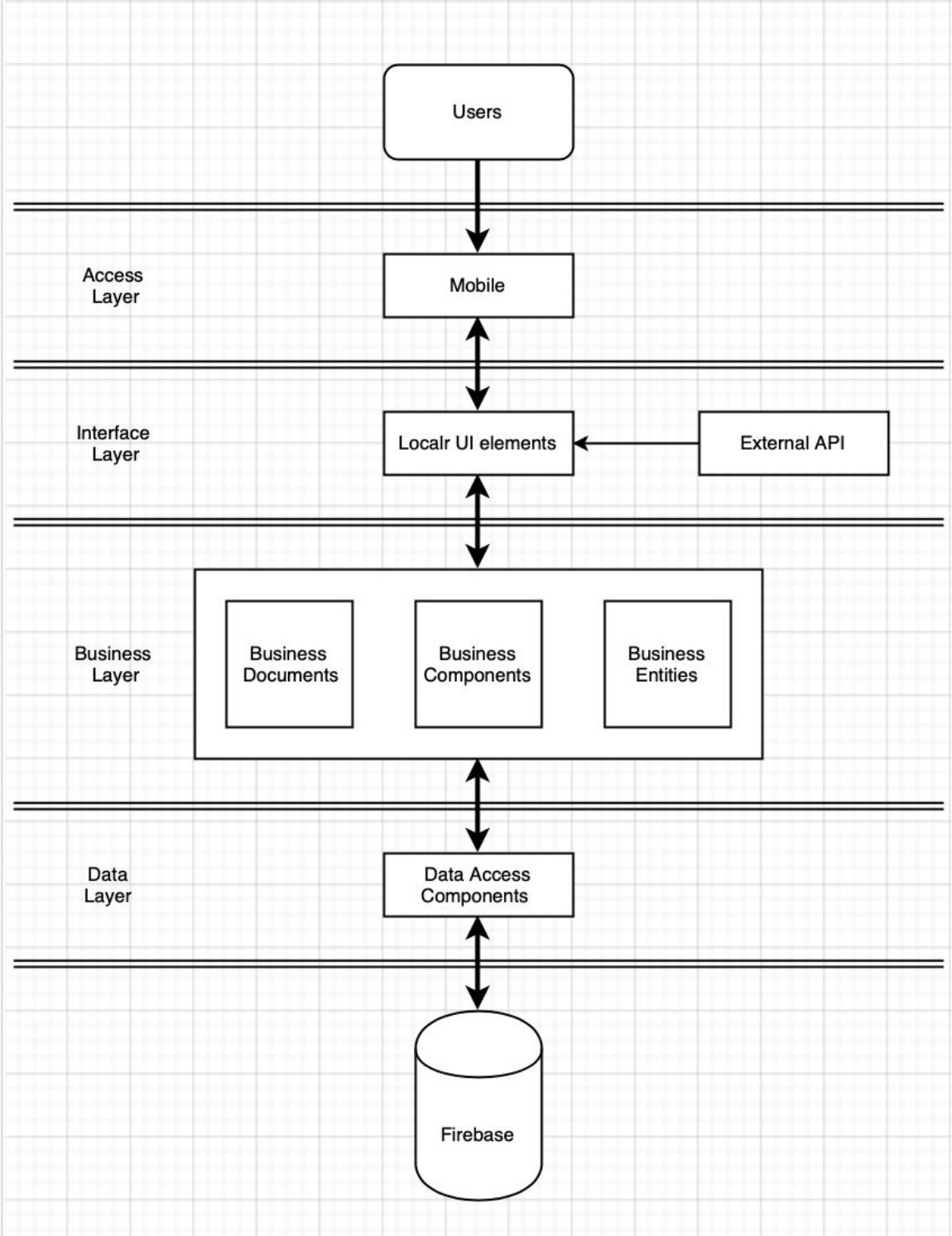
I. System Component Diagrams

We will be using layered architecture for our mobile application and using Firebase to develop it. Firebase runs in javascript and has SDK's available in Node.js, Java and Python. Since we are working on our application from scratch, we think that Firebase is a good way to start with the application. Also, it helps us in easy storing and retrieval of dynamic documents. Layered architecture which is also known as n-tier architecture, closely matches the conventional IT communication and organizational structures found in most businesses. Some of the benefits we have are its simplicity, consistency and browsability. These characteristics make our chosen architecture to be easy to implement, maintaining consistency with code and other layered projects while keeping all the apps together. Some of the disadvantages are its hidden use cases, which makes it hard for us to determine just by checking the code implementation. Since it's layered, the dependencies are straightforward and they change conceptually into higher layers from a low layer infrastructure.

The platforms planned for are Android and iPhone using React Native. We chose React Native to hopefully decrease the learning required to create a cross platform application for android and iPhone. React Native also may allow to transition to a web app in the future if the libraries used support it. We will be using primarily Firebase and AWS as our back end, along with using Python for our machine learning. However depending on our needs for machine learning we may need more space and opt to use Mongodb or Postgresql for our machine learning data.
We will be using Firebase API, maps API and possibly multiple news API’s. Firebase uses an entity based object relational model and represents the data in the form of JSON.

Most of the underlying database configuration has been abstracted and it is accessed using an API. React Native Framework will access the Maps API to display a map and access locations on the map. It will also use the Firebase API to retrieve, save and authenticate user data and possibly our machine learning data. Our AWS server will host our React Native server to serve our data to our users.

System Component Diagram:



Use Cases

Use Case #1	User signs up for the app
Goal in context	Allows the user to sign up for the app
Scope and Level	Affects the user
Precondition(s)	The user must be successfully able to open the app when downloaded
Postcondition(s)	The user is successfully signed up and can proceed with the next page
Success end condition	The user is successfully signed up
Failed end condition	The user is not able to register to the app
Primary actors	Users
Secondary actors	None
Trigger	When the user chooses the signup button
Description	1. The user opens the app when it is downloaded successfully in their device. 2. The user is led to the welcome screen where they have the option to choose the signup button.
Extension	None
Sub - Variations	None
Related Information	
Priority	High
Performance	Approximately 2 minutes
Frequency	Once
Channels to actors	Welcome page

Open Issues	
Due date	End of Sprint 2
Superordinate	None
Subordinate	None

Use Case #2	User logs in to the app
Goal in context	The user is able to login to the app
Scope and Level	Affects the user
Precondition(s)	The user must have register before through the signup button
Postcondition(s)	The user will be able to use the app based on their preferences
Success end condition	The user will be able to login successfully.
Failed end condition	The user will not be able to login due to wrong login credentials or database connectivity issues
Primary actors	Users
Secondary actors	None
Trigger	When the user chooses the login button
Description	1. The user is directed to the welcome screen when they open the app. 2. The user chooses the login option 3. The user then is prompted to put in their login credentials 4. If it is successful,the user will be able to use the app based on the preferences set.
Extension	None
Sub - Variations	None

Related Information	
Priority	High
Performance	Approximately one minute
Frequency	Often
Channels to actors	Welcome page
Open Issues	
Due date	End of sprint 2
Superordinate	None
Subordinate	None

Use Case #3	Preferences
Goal in context	Allows the user to set their preferences to use the application.
Scope and Level	Affects the user
Precondition(s)	The user must have an account or successfully signed up to the app
Postcondition(s)	The user can use the app based on the category of news chosen
Success end condition	The user can view news stories based on the options chosen
Failed end condition	The user will be able to see irrelevant news stories or none at all
Primary actors	Users
Secondary actors	None
Trigger	When the user signs up successfully for the first time

Description	1. The user signs up for the app 2. When the sign up is successful, the user is led to the preference page where they are given with options to choose from for their proclivity.
Extension	None
Sub - Variations	The user will also be able to access the preferences page through the side menu option once when they log back in.
Related Information	
Priority	High
Performance	Approximately 1 minute (depends on the time taken to decide)
Frequency	Often
Channels to actors	Sign up page/ side menu
Open Issues	
Due date	TBD
Superordinate	None
Subordinate	None

Use Case #4	Pin article
Goal in context	The users are able to see the pinned articles on the map.
Scope and Level	Affects the user
Precondition(s)	The user must be able to successfully login or successfully sign up through the app.

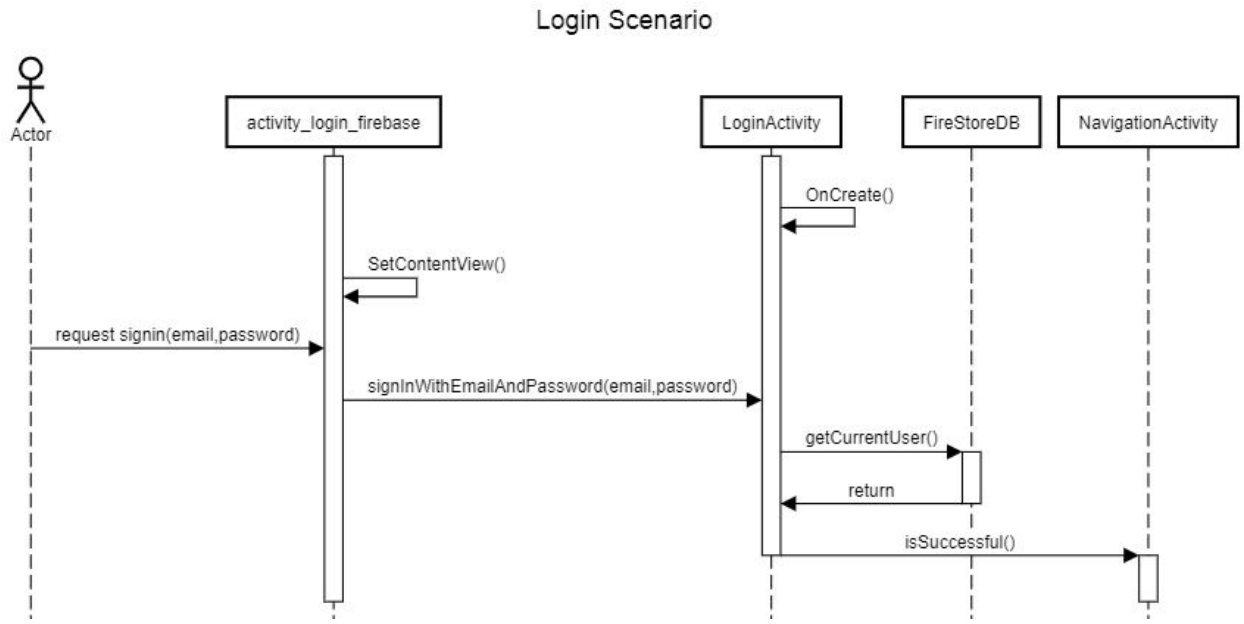
Postcondition(s)	The users are able to access the news article through the pins on the map.
Success end condition	The users can successfully view the news depending on the pin they choose
Failed end condition	No article or news is shown when the user selects a pin
Primary actors	Users
Secondary actors	None
Trigger	When the user chooses one of the pins on the map.
Description	1. The user successfully signs up or logs in to the app. 2. When the user is logged in, the user is displayed a map of their chosen location with pins all across the map. 3. The user chooses one of the pins of their interest to view the news for that particular location/ particular topic of their choosing.
Extension	None
Sub - Variations	None
Related Information	
Priority	High
Performance	1 - 5 second
Frequency	Often
Channels to actors	Map interface
Open Issues	
Due date	TBD
Superordinate	None

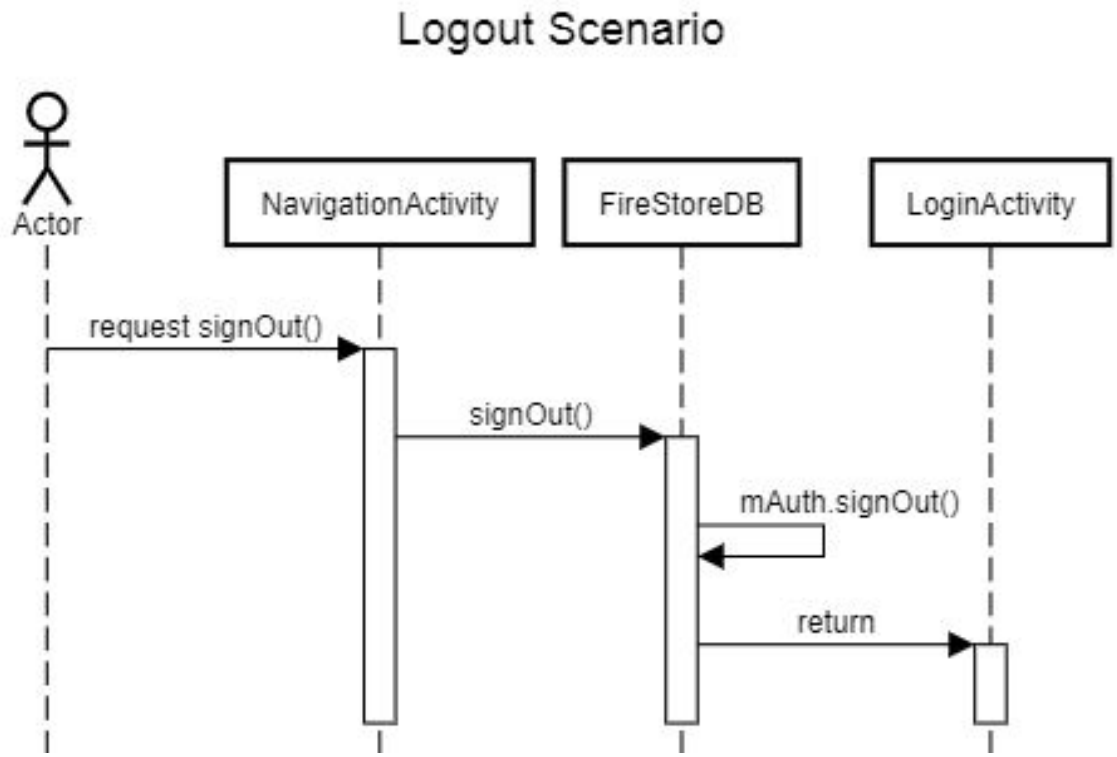
Subordinate	None

Use Case #5	Choose location
Goal in context	Allows the user to choose the desired location to view news articles
Scope and Level	Affects the user
Precondition(s)	The user must be successfully signed up for the app
Postcondition(s)	The user will be able to successfully choose the desired location
Success end condition	The user has access to the news articles based on their desired location
Failed end condition	The user is shown news articles from a random location or from their current location
Primary actors	Users
Secondary actors	None
Trigger	When the user is successfully signed up for the first time
Description	1. The user signs up successfully after they have downloaded the app. 2. After choosing their preferences, the user is asked to choose their current location or to choose a zipcode location of their choice to view the news articles of that area.
Extension	None
Sub - Variations	The users can change their location anytime through the side menu
Related Information	

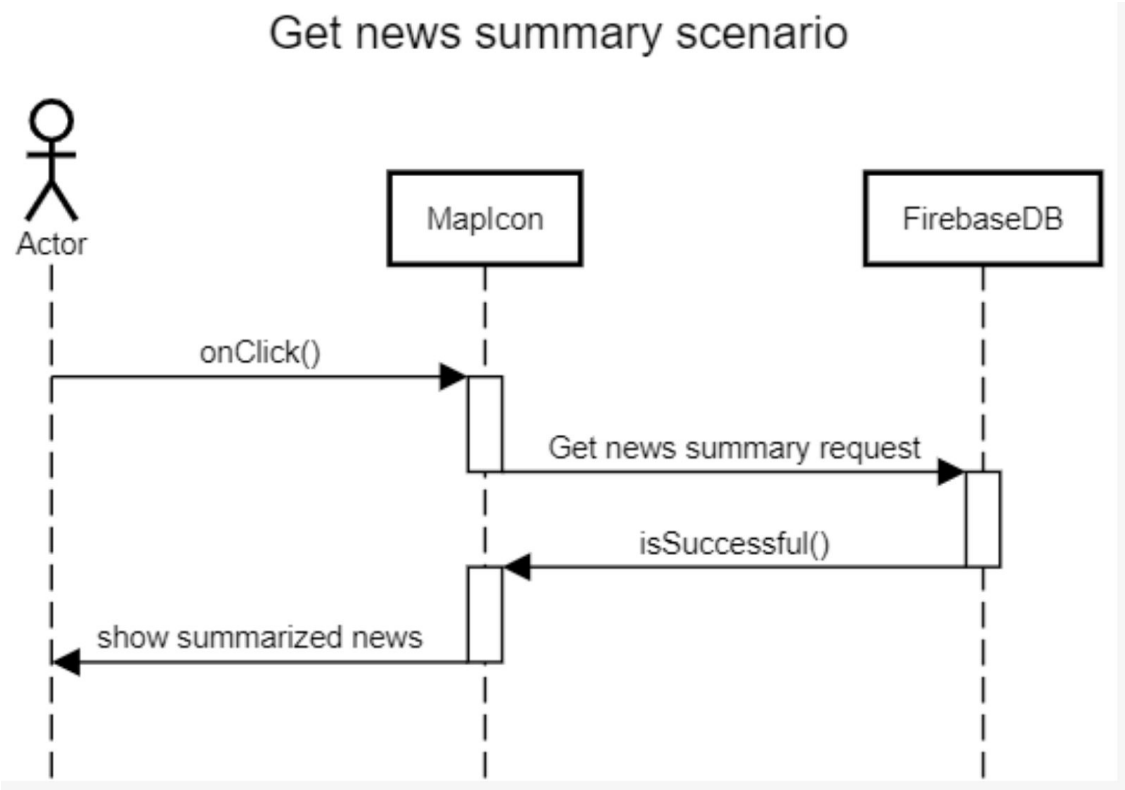
Priority	High
Performance	1 - 5 seconds
Frequency	Often
Channels to actors	Welcome page/Side menu
Open Issues	
Due date	TBD
Superordinate	None
Subordinate	None

Sequence Diagrams

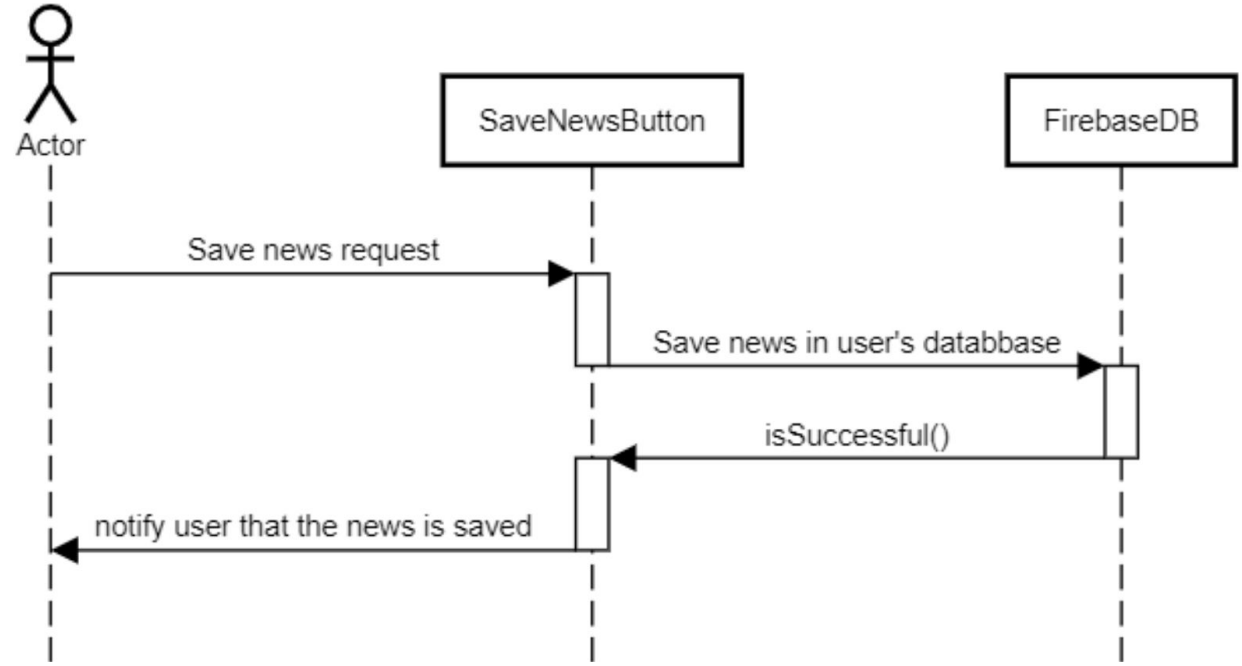


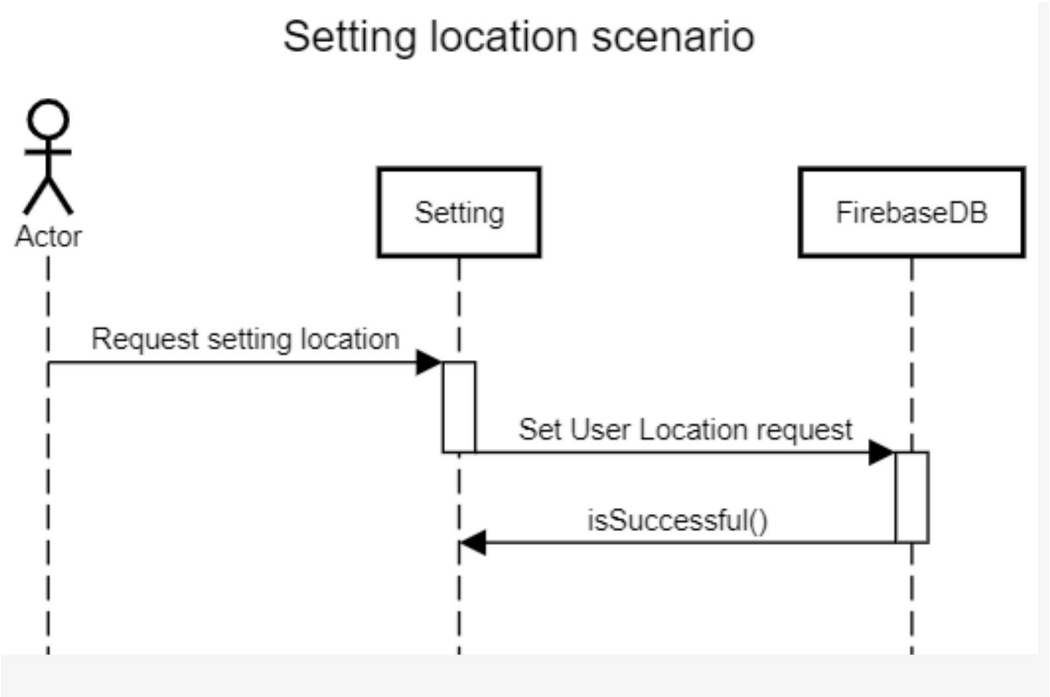


Get news summary scenario

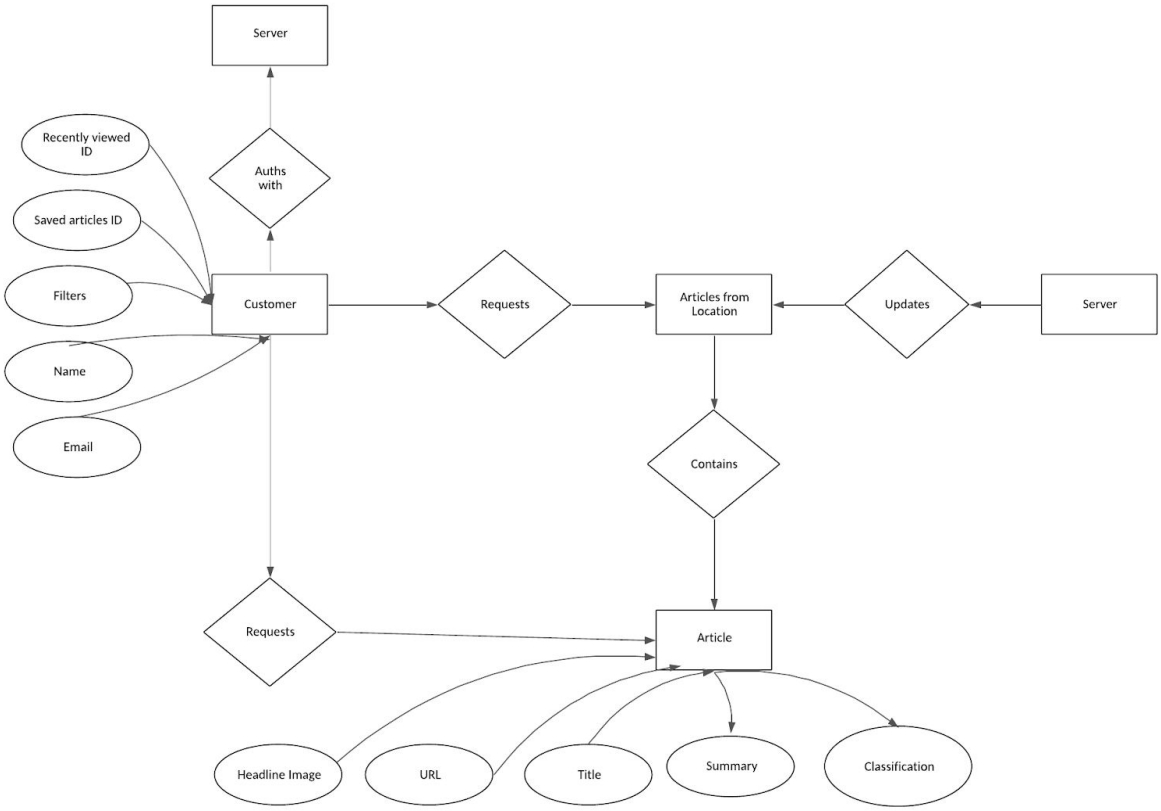


Save news scenario

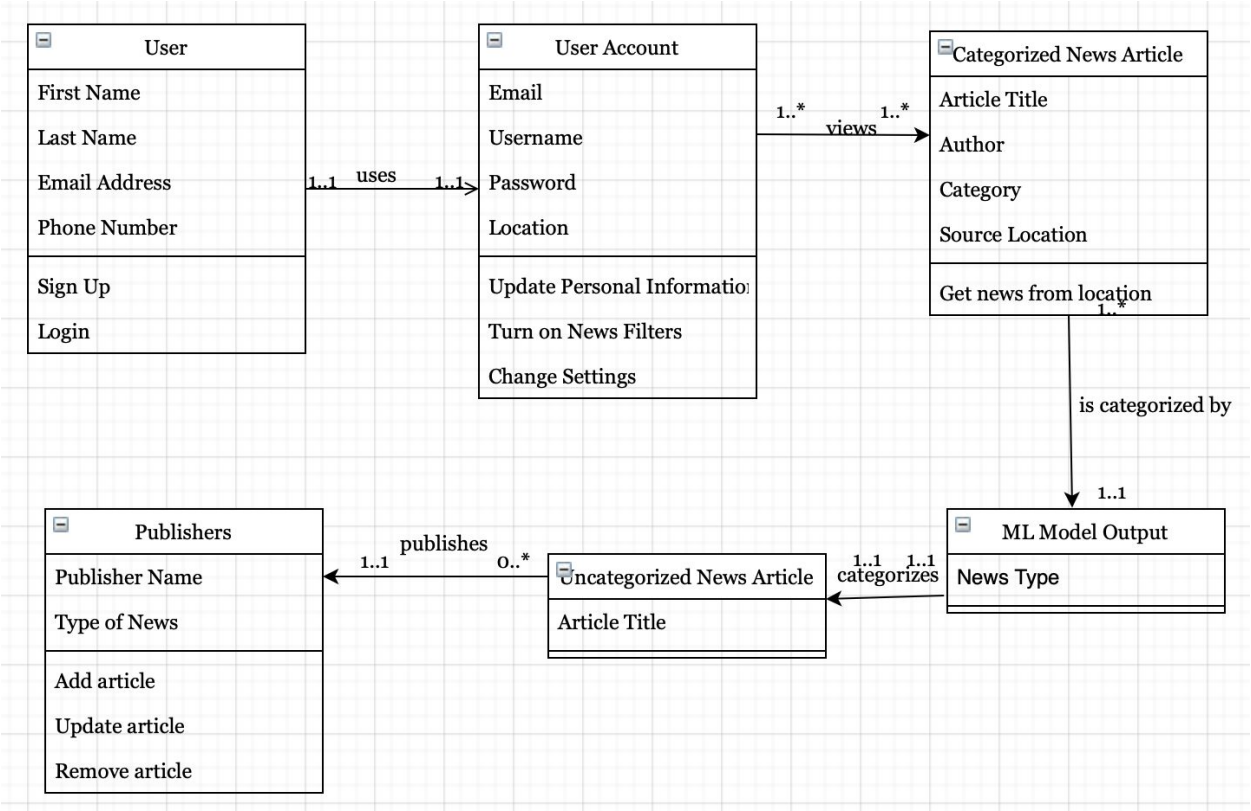



Setting location scenario
Actor
Setting
FirebaseDB
Request setting location
Set User Location request
isSuccessful()

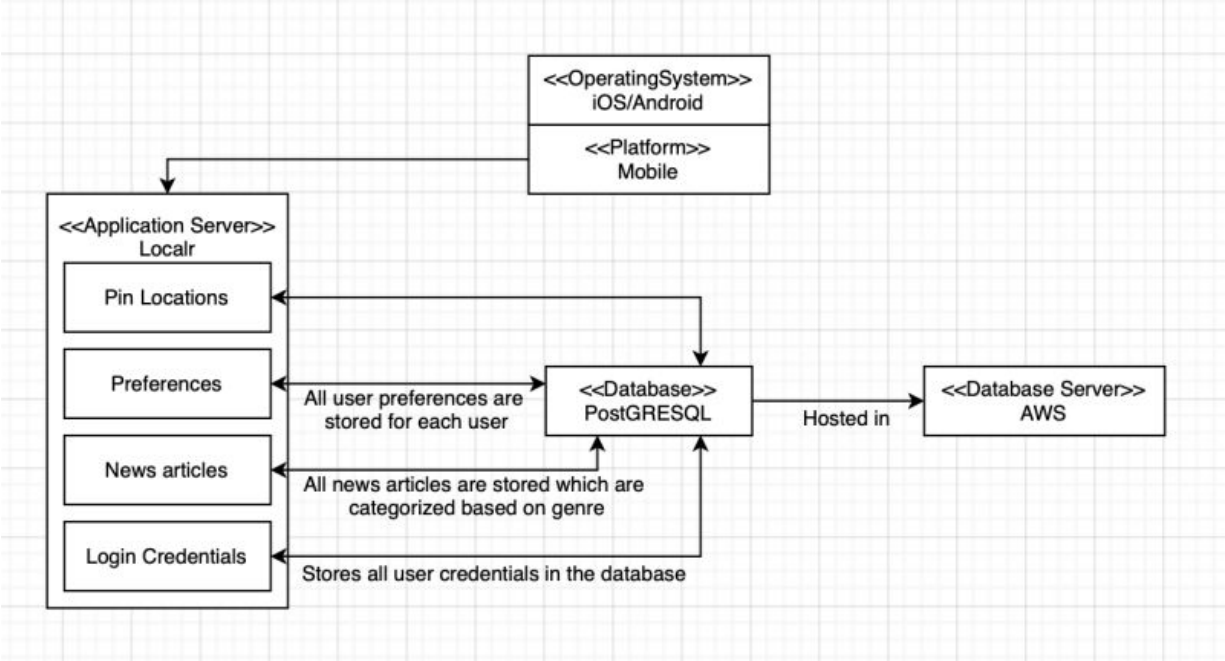
Entity Relationship Diagram:



Class Diagram



Deployment Diagram



II. Trade-Off Analysis

SERVER/COMPONENT CHOICE

Criteria	Choice 1	Choice 2	Choice 3
	AWS	Windows Azure	Rackspace
Team Knowledge	+	-	-
Availability of Resources	+	+	+
Security	+	+	-
Uptime	-	-	+
Ability to Hold Storage Required	+	+	+
Integration Between Components	+	-	-
Complexity (Learning Curve)	+	+	-
RANK	1	2	3

The team has decided to choose AWS as its server due to the amount of knowledge we have on the platform compared to the other two servers listed. Another big reason why we choose AWS over Windows Azure and Rackspace is due to security. AWS uses IAM (Identity and Access Management) while Azure uses ActiveDirectory and Rackspace uses RBAC (Role-Based Access Control). Although all these servers have effective means of security management, AWS has more flexibility in terms of geographic replication due to its availability of location in sixteen regions. Another reason why we are more inclined to use AWS is due to its integration between other components in our project, AWS can support our databases and article information gotten from the scrapers. As for complexity, AWS is easier to understand and access due to the many available resources online, therefore, making the learning curve not too hard.

DATABASE SELECTION

Criteria	Choice 1		Choice 2		Choice 3	
	Firebase	AWS	MongoDB	AWS	PostgreSQL	AWS
Team Knowledge	+	+	-	+	-	+
Database Searching	+	+	-	+	+	+
Security & Maintenance Features	+	+	+	+	-	+
API Integration	+	+	-	+	-	+
Migration between iOS & Android	+	+	-	+	+	+
Affordability	+	+	-	+	+	+
Compliance w/ JSON files	+	+	-	+	+	+
Ease of Use	+	+	-	+	-	+
RANK	1		3		2	

The team chose to work hand-in-hand with Firebase due to team members' previous experience and knowledge of it. In terms of other criteria, using Firebase is more efficient in accessing and easily searching through databases; it has more security and maintenance features, including more user authentication features that will greatly benefit our project and keep the databases secure. Since we plan to also use Maps API, Firebase is also a good choice for holding the database since Google's Map API has great integration through it. Firebase also supports both iOS and Android whilst MongoDB has reportedly had issues with cross platform data handling.

FRONTEND FRAMEWORK SELECTION

Criteria	Choice 1	Choice 2	Choice 3
	React Native	**Angular**	**Vue**
Team Knowledge	+	+	-
Availability of Resources	+	+	-
Migration between iOS & Android	+	+	+
Affordability	+	+	+
Complexity (Learning Curve)	+	-	-
Reusability	+	-	-
RANK	1	2	3

The team chose to go for React Native due to its cross-platform support and due to its multiple help and code documentation available online, it has less of a learning curve to tackle compared to the other two frameworks. As for reusability, due to the way React Native is designed, we can reuse code we already wrote for other parts of the application; meanwhile, both Angular and Vue have issues with replication of components, which is detrimental to the way our app works. Another major reason why we chose to use React Native is due to its customizability and accessibility, whilst Vue would be harder to use especially since most of its documentation is written in Chinese, causing a major language barrier issue which would then affect our time spent working on development.

LANGUAGE SELECTION

Criteria	Choice 1	Choice 2	Choice 3
	Python	Java	R
Team Knowledge	+	+	-
Availability of Resources	+	-	+
Integration between Components	+	+	-
Credible Library Support	+	-	+
Speed	-	+	-
Complexity (Learning Curve)	+	+	-
RANK	1	2	3

The team chose to use Python as its main coding language in terms of Machine Learning mainly due to the knowledge we already have of the language, the visual libraries it has, and the ease of the syntax. Also, Python is known for its immensely credible amount of credible libraries we can use for our training as well as the resources available. We would not have troubles with the learning curve with Python due to the large community of support we can get as compared to R. Another reason we chose to use Python is due to its integration with our other components such as our web scrapers and statistical code. The only notable downside with using Python is the speed at which we run code through it.

SERVER VS SERVERLESS

At the moment, we are running with locally hosted servers on our personal laptops as we develop the frontend/backend components of our project. This can be done due to the simplicity of our project at this point of time. We, as a team, have not yet exhausted the amount of storage that would require larger memory/database space. Eventually, once we have developed a more sufficient and accurate amount of information based on users, articles, and machine learning data, we plan to move to a serverless design with the use of more cloud services.

III. Machine Learning Model

Objective
Classify articles by topic so users can filter their news maps.

Selected Model (1st Iteration)
Binomial and Multinomial Naive Bayes treating each headline as a "Bag-of-Words".

Data Sets
HuffPost News Category Data Set : News headlines classified by topic
https://www.kaggle.com/rmisra/news-category-dataset

Politics	32,739
Wellness	17,827
Entertainment	16,058
Travel	9,887
Style & Beauty	9,649
Parenting	8,677
Healthy Living	6,694
Queer Voice	6,314
Food & Drink	6,226
Business	5,937
Comedy	5,175
30 other topics, each with less than 5,000 headlines	75,670
Total	**200,853**

UCI News Aggregator Data Set : News headlines classified by topic
https://archive.ics.uci.edu/ml/datasets/News+Aggregator#

Business	268,666
Science and Technology	108,465
Health	45,615
Total	**422,746**

Model Deployment
Initially we will be training, testing and "deploying" our ML model from a personal laptop. Deployment means that the trained model reads unseen news headlines from a Python scraper, assigns appropriate topic labels to each headline, and writes the labeled headlines to a Google Firebase database.

In later iterations, we may need to employ a more complex model that requires training, testing and validation be moved to Google's Compute Engine. Additionally, the trained model will reside on a remote application server hosted by either Amazon Web Services or Google's Cloud Platform. We will choose the solution that offers free service the longest. For the purposes of this semester, the Python headline scraper will probably reside on the same virtual machine as the trained ML model.

References
Text Classification Using Naive Bayes, Shimodaira 2020,
https://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn07-notes-nup.pdf

Naive Bayes and Text Classification I - Introduction and Theory, Raschka 2014, arXiv:1410.5329

Tackling the Poor Assumptions of Naive Bayes Text Classifiers, Rennie et. al. 2003
https://www.aaai.org/Papers/ICML/2003/ICML03-081.pdf

IV. Data Visualization

Summary of model runs

Data Set File	Headline Count	Dictionary Size	Headline Count by Topic	Model Accuracy
hufffPost.json	200,853	55,649	ARTS 1,509 ARTS & CULTURE 1,339 BLACK VOICES 4,528 BUSINESS 5,937 COLLEGE 1,144 COMEDY 5,175 CRIME 3,405 CULTURE & ARTS 1,030 DIVORCE 3,426 EDUCATION 1,004 ENTERTAINMENT 16,058 ENVIRONMENT 1,323 FIFTY 1,401 FOOD & DRINK 6,226 GOOD NEWS 1,398 GREEN 2,622 HEALTHY LIVING 6,694 HOME & LIVING 4,195 IMPACT 3,459 LATINO VOICES 1,129 MEDIA 2,815 MONEY 1,707 PARENTING 8,677 PARENTS 3,955 POLITICS 32,739 QUEER VOICES 6,314 RELIGION 2,556 SCIENCE 2,178 SPORTS 4,884 STYLE 2,254 STYLE & BEAUTY 9,649 TASTE 2,096 TECH 2,082 THE WORLDPOST 3,664 TRAVEL 9,887 WEDDINGS 3,651 WEIRD NEWS 2,670 WELLNESS 17,827 WOMEN 3,490 WORLD NEWS 2,177 WORLDPOST 2,579	0.5087
huffPost_v01.json	5,000	9,081	BUSINESS 1,000 COMEDY 1,000 CRIME 1,000 POLITICS 1,000 TRAVEL 1,000	0.7990
huffpost_v02.json	25,000	20,337	BUSINESS 5,000 ENTERTAINMENT 5,000 FOOD & DRINK 5,000 HEALTHY LIVING 5,000 PARENTING 5,000	0.8054

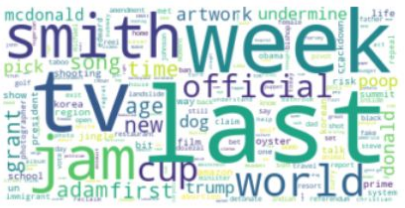
Summary of model runs (cont'd)

Data Set File	Headline Count	Dictionary Size	Headline Count by Topic	Model Accuracy
news_v01.json	465,045	59,686	business 115,967 entertainment 152,469 health 45,639 politics 32,739 science & tech 108,344 travel 9,887	0.9061
news.json	455,158	57,656	business 115,967 entertainment 152,469 health 45,639 politics 32,739 science & tech 108,344	0.9125
news_v02.json	59,322	25,325	business 9,887 entertainment 9,887 health 9,887 politics 9,887 science & tech 9,887 travel 9,887	0.9241
uci-news-aggregator.csv	422,419	54,637	business 115,967 entertainment 152,469 health 45,639 technology 108,344	0.9242
bal_news.json	160,000	36,381	business 32,000 entertainment 32,000 health 32,000 politics 32,000 technology 32,000	0.9261

Word Clouds

(The size of each word indicates its frequency of appearance in its respective data set)

huffPost.json



huffPost_v01.json

huffPost_v02.json

news.json

news_v01.json

news_v02.json

bal_news.json

uci-news-aggregator.csv

V. Risk Management

ID	Description	Mitigation Scheme	Severity Level	Date of Identification	Status
1	Incorrect Time Estimation	Have a person (Project Manager) take on the task of time management	High	3/9/2020	In Progress
2	Project Delays	Improve our time management (assign our tasks more specifically, ensure that we all understand the plan)	High	3/9/2020	In Progress
3	Conflicting Priorities	Have to discuss and come to an agreement about the design features and we plan to include and their implementation	High	3/9/2020	In Progress
4	Tradeoff between maximum functionality and maximum performance	Try to have a good balance of both (avoid overdoing on the features but want to still have some	High	3/9/2020	In Progress
5	Lack of Communication	Have a person (Project Manager) consistently reach out to each team member	High	3/9/2020	In Progress