QATAR UNIVERSITY

COLLEGE OF ENGINEERING

REAL-TIME TWEET SUMMARIZATION MOBILE APPLICATION

BY

NAZAR S. SALIM

A Project Submitted to

the Faculty of the College of

Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computing

June 2018

COMMITTEE PAGE

The members of the committee approve the Project of Nazar S. Salim defended on May 25, 2018:

Abdelkarim Erradi

Project Supervisor

Tamer Elsayed

Project Co-Supervisor

Qutaibah Malluhi

Committee Member

Osama Halabi

Committee Member

ABSTRACT

Salim, Nazar, S, Masters:

June: 2018, Master of Science in Computing

Title: Real-time Tweet Summarization Mobile Application

Supervisor of Project: Abdelkarim Erradi

Project Co-Supervisor : Tamer Elsayed

With the emergence of the massive volume of content through social media platforms, users are getting overwhelmed with information, though searching for the topic will give you filtered information that interests you. Yet, if the user is subscribed to multiple topics one of them might shadow that topic of interest. The project was created to address this issue through offering a mobile application for users to define their topics of interest.

The application named Real-time Twitter Summarization (RTS) offers a novel approach where user gets to not only choose the topic, but to decide on frequency and relevancy of the pushed tweets related to the topic. The application also provides a real time summarization, and offers notification once topic related novel tweet was created. These functionalities are solutions that were not provided in similar applications.

This project has not only been developed to be fully functional, but to also be usable in the simplest format. Scalability of the tweet summarization Engine was tested , to check if the application layer did cause delay or not.

It is important to mention that this work is an extension to the previous work of Suwaileh, Reem and Hasanain, Maram [7] submitted as a participation of “Real-Time Summarization Track” [3].

Keywords: *Mobile Application; Real-Time Summarization; Scalability; RTS; Information Retrieval; Information Filtering*

DEDICATION

I would like to dedicate this project to Science.

And as a sign that you can do anything.

If Nazar was able to pull it off, you sure could do it too.

ACKNOWLEDGEMENTS

I'd like to start my acknowledgment by thanking all my friends who have been waiting for me to finish the Master's degree to hang out with them again, and the answer to ”are you joining us tonight?”, Chang from ”sorry. Can't” to ”I'm there waiting for you”. Knowing that without the support I received from my family I would not have reached this point I would like to thank them form the bottom of my heart.

Additionally, I must appreciate the help of
- Dr. Tamer for allocating from his time to follow up with me on this Project, despite his pre-equipped schedule.
- Reem Ali Suwaileh, for the orignial work of the RTS engine, and for helping with the kick starting the project.
- Khaled Nabil Kadoura, for his input on the design that helped in taking the application to a new level.
- Ammar Al-Qmash for being there in my low points.
- Yazan Tayih for the ionic framework tutorials.
- Mariam Salim and Tasnim Ismail, for proofreading the report

TABLE OF CONTENTS

DEDICATION . . . iv

ACKNOWLEDGEMENTS . . . v

LIST OF TABLES . . . x

LIST OF FIGURES . . . xi

1 INTRODUCTION 1

1.1 Problem Statement . . . 1

1.2 Solution Overview . . . 1

1.3 Contributions . . . 2

2 BACKGROUND AND RELATED WORK 3

2.1 Background . . . 3

2.1.1 Tweet Stream . . . 4

2.2 Related Work . . . 4

2.2.1 Filtering Service . . . 5

2.2.2 TREC Real-Time Summarization Tools . . . 6

2.2.3 Twitter Deck . . . 7

3 REQUIREMENTS ANALYSIS 9

3.1 Functional Requirements . . . 9

3.2 Non-Functional Requirements . . . 10

3.2.1 Usability . . . 10

3.2.2 Scalability . . . 11

4 SYSTEM DESIGN 12

4.1 System Architecture 12
4.1.1 Subscription Service 13
4.1.2 RTS Engine 13
4.1.3 RTS Mobile Application 14
4.1.4 FireStore 14
4.2 RTS User Interface 14
4.2.1 User Experience (UX) design 15
4.2.2 User Interface (UI) design 15
4.3 Data Schema 18
4.3.1 Topics 18
4.3.2 User 19
4.3.3 Tweet 19
4.4 Communication between RTS components 20
4.4.1 Subscription Service Rest API 21
4.4.2 FireStore API 22

5 IMPLEMENTATION 23

5.1 Subscription Service 23
5.1.1 Load balancing 23
5.2 RTS Engine 24
5.2.1 Configuration options 24
5.2.2 Sync the Topics 24
5.3 Mobile Application 25
5.4 Hosting 25

6 EVALUATION 26

6.1 Performance Evaluation 26
6.1.1 Setup 27
6.1.2 Evaluation Results 27
6.2 Usability Testing 29

6.2.1 Task Analysis 30
6.2.2 Usability Testing Results 31

7 CONCLUSION AND FUTURE WORK 33

7.1 Conclusion 33
7.2 Future work 33

REFERENCES 35

Appendix A FIGURE SNAPSHOTS OF TWEETDECK 37

Appendix B USER EXPERIENCE MATERIAL 39

B.1 Persons 39
B.1.1 Persona Yazan 39
B.1.2 Persona Baraa 40
B.1.3 Persona Husam 41
B.1.4 Persona Abdol 41
B.1.5 Persona Khaled 41
B.2 Scenarios 42
B.2.1 Download/Sign up and Subscribe to the topics of interest 42
B.2.2 Abdol Learned about the Application 43

Appendix C USER INTERFACE MATERIAL 45

C.1 Screens Design 45

Appendix D HIERARCHICAL TASK ANALYSIS 47

D.1 Add Topic Simple 47
D.2 Add Topic Advanced 48
D.3 Edit Topic 49
D.4 Sign Up 50
D.5 Navigate to old date 51
D.6 Navigate to old date 52

D.7 Filter Topics 53
D.8 Remove Topics 53

LIST OF TABLES

3.1 List of Functional Requirements 10

4.1 Topic Attributes Description 20

4.2 Subscription Service Rest API List 21

4.3 List of the used FireStore API 22

LIST OF FIGURES

2.1 A high-level overview of the pipeline explained in [7] 4

2.2 A high-level overview of the pipeline explained in [7] 7

3.1 The Use Case Diagram of the RTS System 9

4.1 Over View on the System Components and Architecture 12

4.2 Login Screen and Sign Up 16

4.3 Login Screen and Sign Up with invalid filling 17

4.4 Guide sample and Topic Creation from 18

4.5 Sample Tweet, Filter selecting a topic 19

4.6 The Entity relation digram (ERD) 21

6.1 Graph showing time in milliseconds per the number of active clients requests 28

6.2 Time processing time (milliseconds) versus the number of topics 29

6.3 Summary of Usability survey answers 31

A.1 The type of columns that can be added to your TweetDeck 37

A.2 Example of a Screen Dashboard TweetDeck 37

A.3 Highlight of the filter Options of a Screen Dashboard TweetDeck 38

C.1 Screen Login and tweets Listing 45

C.2 Screen Topics Listing and topics Form 46

D.1 Hierarchical task analysis for Add topics 47

D.2 Hierarchical task analysis for Add topics 48

D.3 Hierarchical task analysis for Add topics 50

D.4 Hierarchical task analysis for Sign Up 51
D.5 Hierarchical task analysis for Navigating to old date 51
D.6 Hierarchical task analysis for Navigating to old date 52
D.7 Hierarchical task analysis for Filter Topics and Remove Topics 53

CHAPTER 1: INTRODUCTION

The recent information explosion caused by increased amount of published information and user-generated content has led to information overload. With that, the challenge of filtering out relevant information from general and repetitive content came about. Someone could have multiple interests; yet if one of those interest has a high volume of material, it will shadow out the other topics. In 2014, the text retrieval conference (TREC) introduced a new track titled "Real-Time Summarization Track" [3] to address this issue. The track focuses on both; real-time filtering of the tweet stream and identifying the relevant and novel tweets. This is to notify the user in real-time.

1.1 Problem Statement

The focus of a Real-time Tweet Summarization (RTS) System is to address the mentioned issue by providing a platform for users to define their topic of interest and get notified of new novel tweets about them.

1.2 Solution Overview

The project presented in this report extends the solution submitted by Qatar University IR Team. Section 2.2 describes the solution design in details, and how it addresses the aforementioned issue. The project extends it by developing a mobile application that works on an interface to utilize the functionality implemented in the filtering service aka the engine.

The solution is built with four main components: The mobile application, Subscription service, FireStore database and the Engine. The detailed description of the components and how they are implemented is described in Section 4.1 and Chapter 5 respectively.

The idea of this project is to put the use of the research done in [7] into a real application and expose it for users to experience. Additionally, it opens the chance to add feedback feature to the solution.

1.3 Contributions

The contributions of this project are

1. Development of the mobile application Real-Time Twitter Summarization (RTS) that utilizes the ”Scalable Real-time Tweet Summarization” Engine developed by [7].

2. Overcoming the drawback of available twitter summarization solutions by providing the following features in RTS

   - Defining their topics of interests in a straightforward way
   - Get notified about a new topic directly using push notifications in real time
   - Navigate through the history of pushed (selected) tweets

3. Introducing functionalities that allow control to the user where they can customize relevancy threshold, choose tweets frequency , add and remove topics of interest and thier keywords any time

4. Dev elope RTS in a way that is both Usable on the application level and scalable on the engine level through a well rounded experiment.

CHAPTER 2: BACKGROUND AND RELATED WORK

In this chapter; The concepts related to the project are introduced starting with (web 2.0) and the type of application that is introduced. Then, Twitter as a research opportunity is explained along with the topics of interest. After that, the research paper that this project is extending is described. Finally, this section is concluded with highlighting an exiting application that offers similar features to what this solution project proposes.

2.1 Background

The movement of the web from content-based to web applications and services (Web 2.0) has introduced a new level of content volume and research opportunities [9]. Twitter; the current leading macro blogging platform gets a lot of attention in the Text REtrieval Conference (TREC), offering multiple tracks - which are series of workshops focusing on a list of different information retrieval (IR) research areas- around it. Since 2011, tracks related to real-time search were offered, best summarized as a question of what are the most relevant tweets about topic X, given a stream of tweets [3]. Each year TREC offers it with another flavor of challenges. In 2014, Temporal Summarization Track was offered which focuses on getting updates about highly significant selected predefined events, that need to be analyzed on a stream of provided documents. Then, participants are asked to submit a summary of these events; where the optimal summary is the one that covers all of the essential information about them with no redundancy [2]. In 2016 TREC introduced the latest version of a microblogging focused track: The Real-Time Summarization, and carried on to 2017 and 2018 [4]. The focus of this track is to consider a group of users who have a set of interests, and the system's task is to automatically monitor the stream of documents to keep the users up to date with the topics of interest.

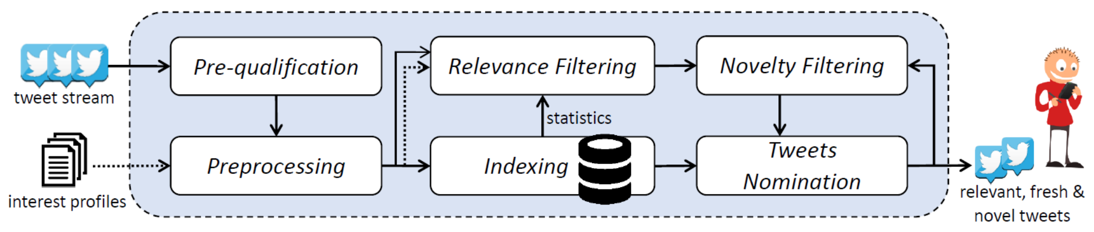


Figure 2.1: A high-level overview of the pipeline explained in [7]

2.1.1 Tweet Stream

To get access to a stream of statuses (aka tweets), Twitter offers two methods. The first method is the ”Sample Real-time Tweets”, which returns a small random sample of all public statuses [8]. The other method is ”Filtered Real-time Tweets”, this has two offerings: the standard free one that allows only one filter rule on a single connection, and requires disconnection to adjust the rule. A second method is an enterprise option that offers thousands of rules on a single connection, with no disconnection needed to add/remove rules using Rules API.

It is good to note that the ”Sample Real-time Tweets” is used in research to ensure neutralness. Also, if there is a challenge, all the participating teams would be getting the same samples. This helps in testing results accurately.

2.2 Related Work

The RTS project in hand is an extension to work done in [7], Qatar University's IR Group submission for RTS 2016 Track. The core system components, donated as the filtering, service are described as part of the project's system architecture explained later in Section 4.1 of this report.

2.2.1 Filtering Service

The system design for the 2016 submission explained in [7] emphasizes on the light-weight and conservative filtering strategy; achieved through pipelining multiple stages: 1) Pre-qualification 2) Preprocessing 3) Indexing 4) Relevance Filtering 5) Novelty Filtering 6) Tweets Nomination 7) Profile Expansion. As highlighted in figure 2.1, The system adopts one-tweet-at-a-time processing model to ensure the shortest possible latency in making pushing decisions.

Pre-qualification

The core activity in the pre-qualification phase is to filter out tweets of a non-desired language, additionally the removal of Spam tweets - defined as tweets that have more than one URL or more than 3 hashtags-.

Preprocessing

After a tweet is marked as qualified, some processing takes place including special characters removal (e.g., emotion and symbolic characters), stop-words removal, URL removal, and stemming. Then, the indexing takes place where the system incrementally indexes all incoming English tweets for the evaluation period.

Relevance Filtering

While indexing the tweets, the relevance filtering takes place in parallel by using the vector space model to represent each interest profile the (title especially) and representing each incoming tweet as a vector using IDF-based term weighting scheme. Then, it computes the term weights using the number of tweets indexed at the time of constructing the vector, and the document frequency of the term. After that, the RTS system computes the relevance score of a tweet against each interests profiles using the standard Cosine Similarity function.

Novelty Filtering

Before sending the tweet to the broker, the RTS system should only push tweets that are relevant and novel to the user. Therefore, a novelty model has to be used to estimate the originality for each potentially-relevant tweet. The implementation of the engine examines the tweet against all previously-pushed tweets for a corresponding profile to estimate its novelty before deciding to push it to the corresponding user.

Introduced modifications

In this section, changes introduced to the existing engine implementation are highlighted. The first change was to make the relevance threshold customizable on the topic level in contrast to being pre-defined on the system level. Secondly, to modify the structure of the topic; such that it extracts the terms from the set of provided keywords of the topic, not the description. This was found to be more effective in experiments. Also, to modify that further such that users have the ability to define the keywords explicitly. Last but not least, the system was built with an assumption that all the topics are pre-known to it. In order to allow the usability of the system with a live application, the necessary configurations were done to sync the topics with the users' defined ones. All of that is detailed in section 4.1 describing the system architecture and chapter 5 explaining the implementation details.

2.2.2 TREC Real-Time Summarization Tools

To evaluate the submissions to the RTS Track a group of tools was created to mimicking a real scenario and give the assessors a way to evaluate the pushed tweets. That setup had 3 groups of components, first was the Mobile Assessment App, which mainly used to show the assessors the submitted tweets for each topic allowing them to rate it with 3 options: 0) not relevant, 1) relevant 2) relevant but repeated. Which in our case would be the mobile application offered to users to receive push notifications. The second component are the participating system where in the challenge each team would have developed their own engine connecting it to the evaluation broker (the third component)

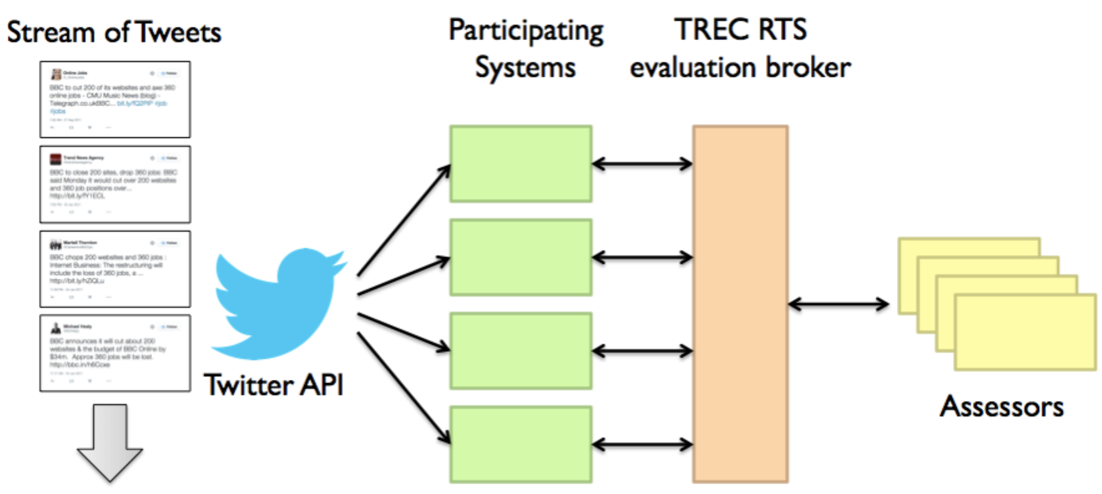


Figure 2.2: A high-level overview of the pipeline explained in [7]

to group and distribute the tweets to the assessors.

Figure 2.2 shows the data flow of the components, where each of the participating systems subscribes to twitter stream, and received the topic's definition from the broker, pushes assigned id and topic and tweet id to the broker when finding a relevant and novel tweet. then the broker manges delivering the tweets to the assessors.

The project in hand follows smiler flow, the main difference is that users create there own set of topics, and there is only one engine version, with multiple instance of it. We also replaced the broker with what we call the subscription service though it shares some functionality with the broker.

2.2.3 Twitter Deck

Twitter has acquired in mid-2011 an application that offers a dashboard. This dashboard consists of a series of customizable columns; each column provides information or stream of tweets based on its type. Figure A.1 highlights the selectable types, which include: mentions, direct messages, lists, trends, favorites, search results, hashtags, or all tweets by or to a single user. The one closest to the RTS System is the search column.

Search Column

The way the search column works is after it is added to the dashboard, it would show a search bar, where a search query is entered. Then, it would populate the column with search results ordered by most recent down to old tweets. The column would also be auto-updated when new tweets that satisfy the search query come. The search column also offers filter options as shown in figure A.3. On the content level, its possible exclude a set of keywords in the query, specify the language, and select only to display tweets that have a selected type of media such as (images, videos, gifs, etc..). On the user level, one of the filtering options is to select showing tweets by users who have been verified by twitter (have the blue tick next to their twitter handle); to allow seeing tweets by . Last but not least one of the filters available is on the engagement of the tweet, such that it is possible to specify the minimum number of likes/re-tweets/replies a tweet received to show in the column.

Drawbacks

Drawbacks of TweetDeck are not having real time updates, using basic filters and not checking for novelty. Following is their explanations.

The First drawback to TweetDeck is that users must stay logged in to access the content. Users are not provided with real-time updates while working on other applications. That is, users must still periodically check TweetDeck to obtain the latest updates. Second limitation is that the filters are basic, e.g. it only checks if a tweet has the term or not. Last, the novelty check is not available as an option.
Things that are offered in the RTS System and not in TweetDeck include the ability to be notified of new tweets that arrives without closely monitoring the feed. Also, novelty filter is applied as an essential step in selecting the tweet to push to the user. Not to forget using a more complex filter as provided by [7].

3.1 Functional Requirements

The RTS mobile application requirements focuses on to allowing users to define their own topics and have full control over the frequency of the push notifications.
As explained in Table 3.1 the main functions are; Managing Topics, viewing Pushed Tweets, Receiving Notifications, Signing up and Signing In.
Figure 3.1 shows the use case diagram identifying the engine to be the actor that pushes the tweets and the subscriber who can set the desired topics they are interested to be updated about.

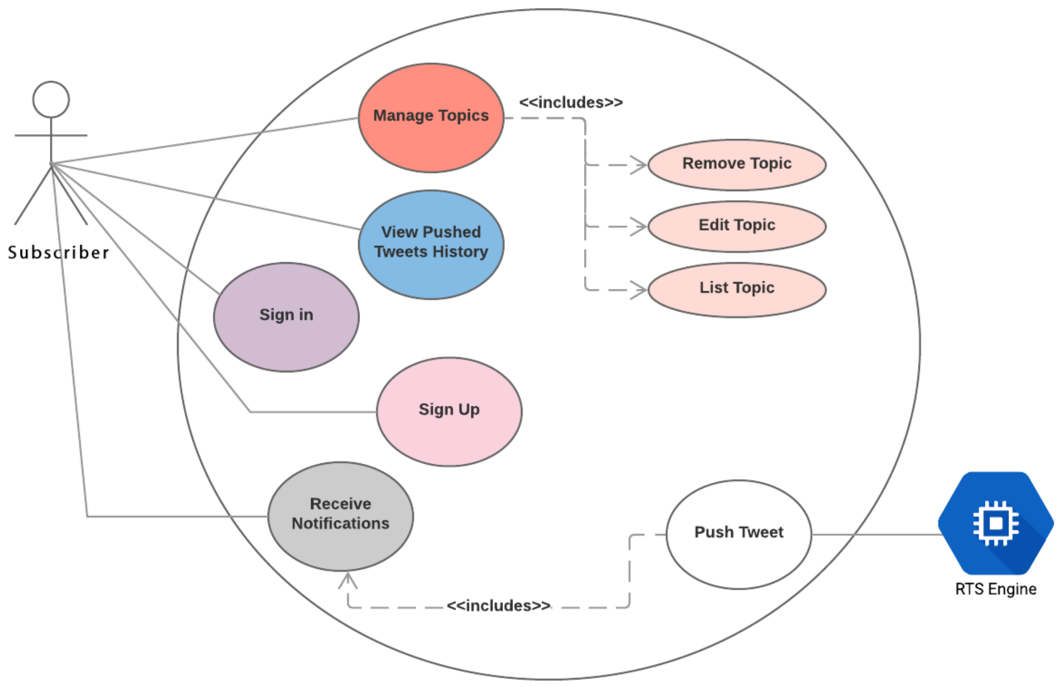


Figure 3.1: The Use Case Diagram of the RTS System

Table 3.1: List of Functional Requirements

Requirement	*Description*
Manage Topics	The ability to express their topics of interest, Navigate through the topics, edit and remove them. along the ability to customize the frequency and threshold settings per their desire.
Notification	Getting notified about new tweets without the need for having the application running.
View pushed tweet history	Access to a list of all sent tweets, with the ability to filter the tweets per a selected topic, look for tweets based on the pushed date.
Sign In	Simple user login to use the app functionality
Sign Up	The application should be open for public allowing new users to create accounts.
Push tweet	The processing that takes place between the engine getting the tweet till the subscription service connecting with the push notification service.

3.2 Non-Functional Requirements

Non-functional requirements (NFR) are requirements of how well the system should deliver its functionality. The key desired quality of RTS System are usability and scalability of the system.

3.2.1 Usability

The usability requirements for the application are as follows.

1. The interface has to be intuitive: Users should be able to learn the application on their own without the need for explanation. The application should offer the necessary guidance only if needed.

2. The number of clicks required to finalize any action should not exceed five clicks.

3. In case of error, the user has to be informed of the issue and the course of action to avoid it. For example, if there is an invalid field, the field that has a problem should be clarified, and the way to resolve the validity has to be explained in the error message.

3.2.2 Scalability

Scalability is the capability of a system to handle a growing amount of work by allocating more computing resources to accommodate that growth. Scalability can be achieved by running multiple instances of the comportments experiencing higher work load. Let us say one engine can handle approximately ten thousand topics, adding one more engine should allow twenty thousand topics and so on. The scalability requirements for our project is the ability to manage ten thousand topics per engine instance, along with the ability to perform sending one hundred tweets per sec when running a single instance of each component. Because the pipeline engine is scalable, adding more instances should be covered by it's scaling capability. This will be tested in section 6.2.2 then in the performance evaluation.

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

The developed RTS system follows the architecture shown in figure 4.1 that highlights the main components of the system:

a) RTS Mobile application: allows the users to define their topic of interest, receive notification directly on their mobile device and go through the history of pushed tweets.

b) subscription service: receives the created topics along with the edit and remove requests from the users and forwards them to the Real-Time Data Store (fireStore). The subscription service also acts as a balancer that divides the topics between Engine instances based on round-robin load balancing, along handling the push notifications for pushed tweets received from engine instances.

c) RTS Engine: listens to the stream of sample tweets and takes them through the pipeline explained in section 2.2.

d) FireStore: the database of the system offering the sync options and authentication.

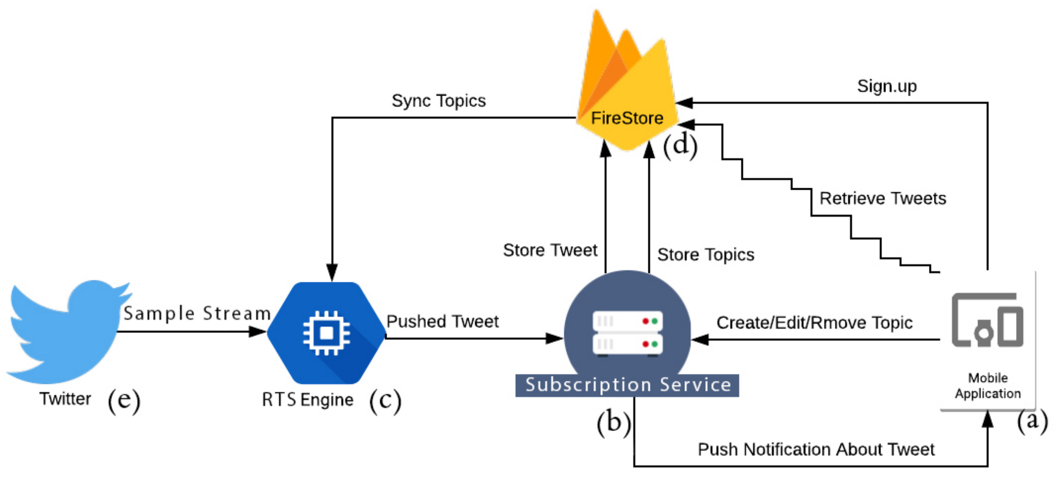


Figure 4.1: Over View on the System Components and Architecture

e) Twitter: micro-blogging platform the RTS engine listens to. The following subsections explain in details the roles of each of the components along with the communication channels between the components. Chapter 5 on the other hand covers in depth the implementation details.

4.1.1 Subscription Service

The Subscription service component of the RTS system acts as the intermediary between the three other components. It is responsible for balancing the number of topics between multiple running instances of the engine. It also handles the interaction with the mobile application by offering the server side validation for the requests coming from the users. Additionally, it maintains storing the topics and tweets in the database server. Furthermore, it connects with the push notification providers through OneSignal [5] a high volume, cross-platform push notifications and email service that works for iOS, Android, and Web. Further implementation details of the Subscription service can be found in section 5.1.

4.1.2 RTS Engine

The RTS engine is the core component of the RTS hence the given name. While active it subscribes to two streams
a) reading twitter stream and taking each tweet through the processing pipeline explained in section 2.2
b) second reading the sync stream from the Real-time Database FireStore that notifies the Engine about every addition, modification and removal of a topic. The engine stores every topic as a profile object that has the topic details including title and the keywords where it extracts the terms to analyze it against incoming tweets. More information on the profile data structure and the sync mechanism can be found in section 5.2 of the implementation chapter.

4.1.3 RTS Mobile Application

The main contribution of the project is the mobile application that allows the users to receive push notifications, manage their topics, change their preferences in terms of notification frequency, minimal tweet to topic and minimal relevance level. The application also allows the user to view the history of pushed tweets. It is also bundled with authentication and profile management features. Section 4.2 highlights the user interface design in details and further implementation description can be found at section 5.3.

4.1.4 FireStore

This project uses Google Firebase API [6] as both a database component to store the tweets, topics through the Beta version of new storing feature, and to manage the user sign-in/sign-up through the authentication service. The Real-time store comes pre-equipped subscription-based API that allows the other components of the system to fetch data directly from it while applying the necessary data access rules: that ensures that users have access only to the tweets and topics of their own.

4.2 RTS User Interface

The UI is both the first in line from importance for the success of the project and the second in line when it comes to development prioritizing. Behind the UI, there are two over-lapping sides to it. First the look and feel including the colors and the interactive items of the application which are the various form fields and buttons. The other side is the User Experience (UX) which focuses on the flow of the user between interfaces/screens, the intuitiveness of the icons used and the usability of the UI elements. A simple way to know if a design has failed is when it requires a manual to learn how to use it. The continuation of this section will focus on the work done to design the UX first then moves to highlighting the UI design of the application.

4.2.1 UX design

To develop a solid user experience, user-centric software design (UCSD) an approach where all the system functionalities and UX/UI are built with the user in mind. To achieve UCSD the project started with developing the persona of the system users. Which describes the application users by their interests, their level of technology savviness, and the information they are looking for. Then, moves to the second phase which is creating usage scenarios for the created persona¿ The scenarios get translated to user flow and interfaces. Follows is a highlight of one example of to demonstrate the activities taken. Section B.1 of Appendix B lists of all persona and the scenarios that were created can be found in Section B.2.

4.2.2 UI design

The UI design is translating the outcomes of the UX exercise into graphical components. In this section we will be showcasing the screens of the application taking you through the application in a demo format to explain the intention and the possible scenarios. The first take will be the flow of an error-free scenario, at each step we will be highlighting the components, the user's intentions, and possible interaction options. A full list of screens design can be found in Appendix C. Note that the highlighted screens are screen-shots from the implemented application.

Sign Up

The first screen that the user finds after installing the application is the sign in/sign up Page. The first screen new users see is shown on the left side of figure 4.2. These screens allow them to pick a quick authentication option through the common social media platforms (Google, Facebook, Twitter). This acts as a sign up if it is used for the first time, and as a login, if the user has signed up before. In our scenario the user would like to use his/her email to sign up, so he/she clicks on the signup button that pulls the form shown on right side of figure 4.2.

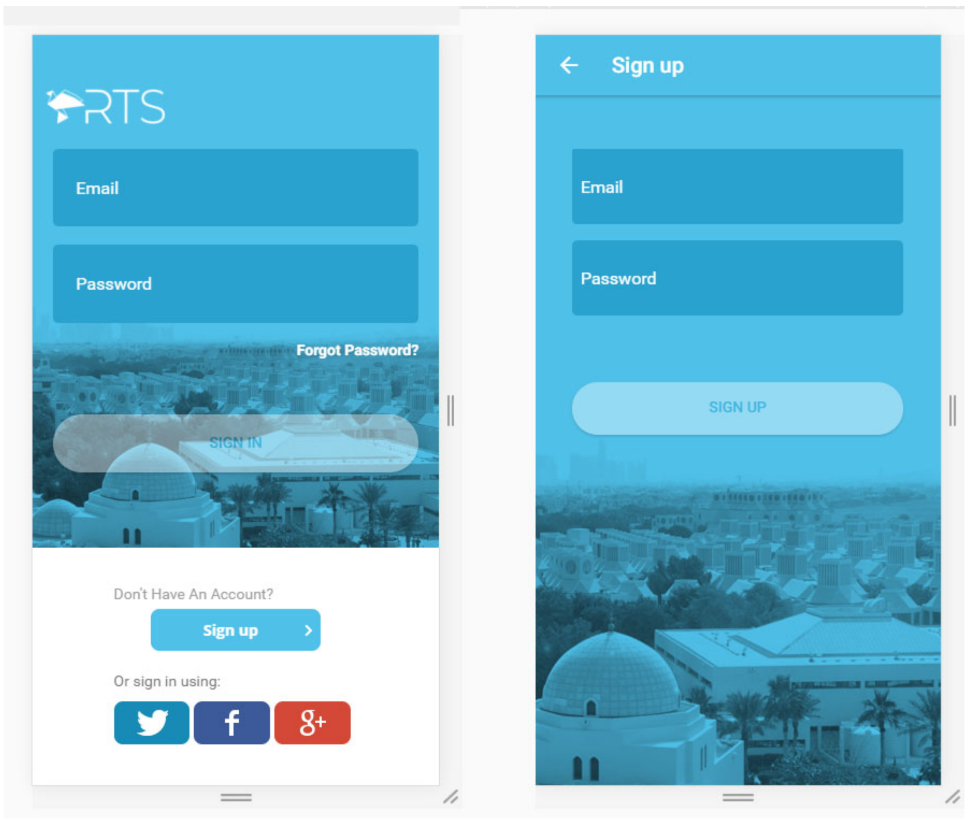


Figure 4.2: Login Screen and Sign Up

As shown in the figure the login and sign-up buttons are disabled with a level of transparency if the user clicks on them, he/she will be notified that he/she needs to fill the form with valid input in order to accept the action. Figure 4.3 shows it clearly along with the calcification massages.

First time user Login

After sign up, new users are taken to the topics list page to define their first topic, guided through the screen with highlighters tour. Figure 4.4 shows the guiding message on the left side. Note that the screenshot shows the end of the animation. The idea is that the application welcomes users and direct them to the add topic button, the right side of figure 4.4 shows the form which has the following input items:

1. *Title*: which is a given name for the topic

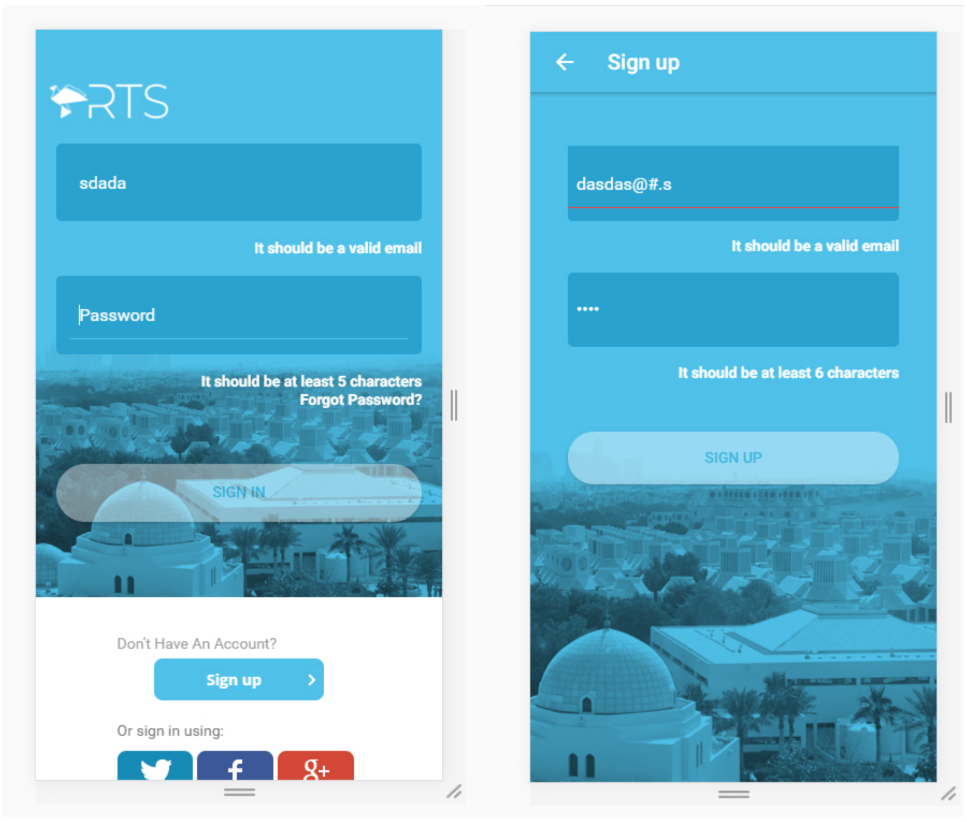

RTS
sdada
It should be a valid email
Password
It should be at least 5 characters
Forgot Password?
SIGN IN
Don't Have An Account?
Sign up
Or sign in using:
Sign up
dasdas@#.s
It should be a valid email
••••
It should be at least 6 characters
SIGN UP

Figure 4.3: Login Screen and Sign Up with invalid filling

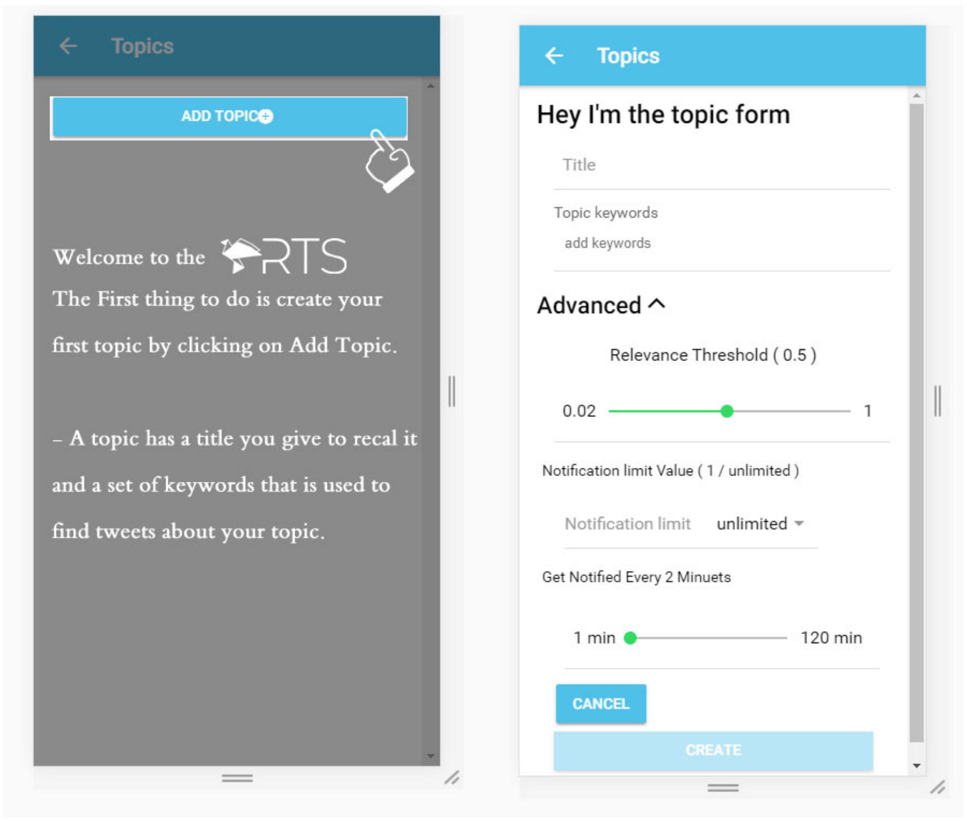
2. *Keyword*: This is the place to add a list of keywords that the tweets get evaluated against

3. *Advanced*: which are the items related to the frequency of push notification and the relevance threshold

The Create button has been disabled until the required fields are filled (Title and one keyword is essential) the rest has default values.

Viewing Tweets/filter options

Once a user creates a topic they are taken to the home page. As shown in figure 4.5 the page shows them an example tweet that tells them this is how a tweet is presented. The right side shows the pop-up for selecting among the topics created by the user. Additionally by clicking on the calendar icon, the user will be presented with a calendar to limit out the displayed tweets. Note that by default, the application shows users


Topics

ADD TOPIC

Welcome to the RTS
The First thing to do is create your first topic by clicking on Add Topic.

– A topic has a title you give to recal it and a set of keywords that is used to find tweets about your topic.

Topics

Hey I'm the topic form

Title

Topic keywords
add keywords

Advanced

Relevance Threshold (0.5)
0.02 — 1

Notification limit Value (1 / unlimited)
Notification limit: unlimited

Get Notified Every 2 Minuets
1 min — 120 min

CANCEL
CREATE

Figure 4.4: Guide sample and Topic Creation from

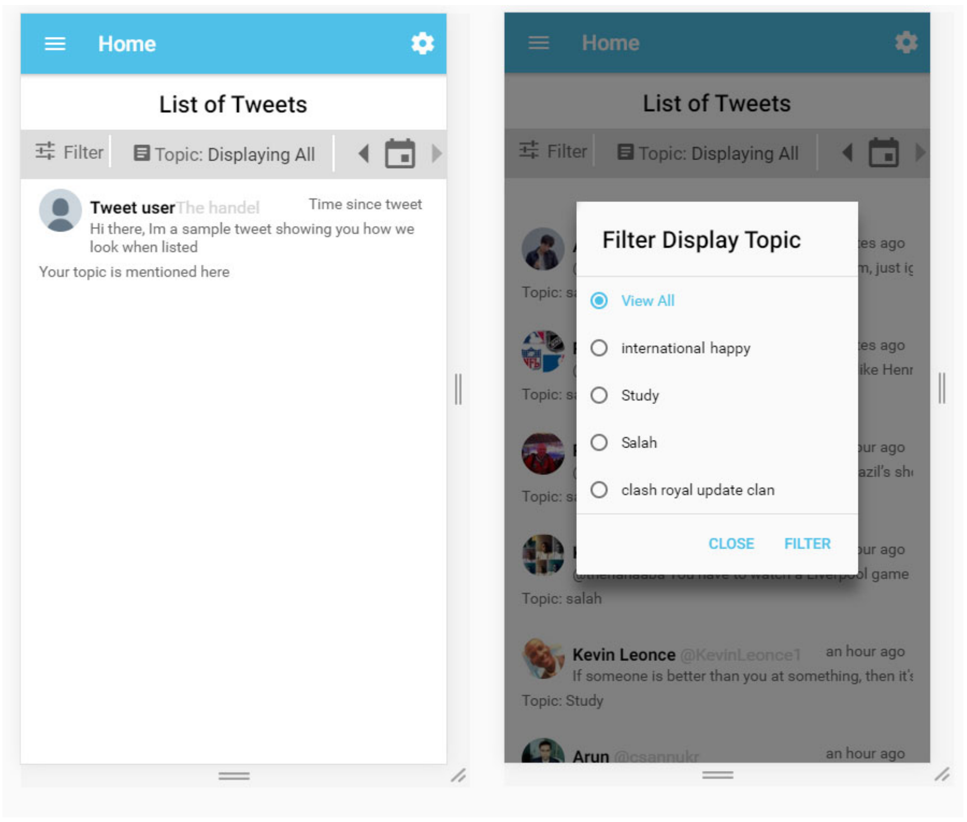
current day's tweets. The arrows allow the user to move to the previous days, and if he/she is at a day prior, the next button will be enabled.

4.3 Data Schema

The system has three major entities: topics, subscription and tweets. In this section, we will highlight the relations of each entity with the others, cover the attributes and their usage within the application. Figure 4.6 shows the entity relation diagram.

4.3.1 Topics

The topics are the expressed representation of user's interest. As shown in section 4.2.2, a topic has two sides: one concerned with the tweet relevance and the other is related to the push notification. Each user could have zero to many topics, and a topic has one user subscribed to it. This allows storing the customization related information in the same



Hi there, Im a sample tweet showing you how we look when listed

Your topic is mentioned here

If someone is better than you at something, then it's

Figure 4.5: Sample Tweet, Filter selecting a topic

object as the terms. If the system introduces topic sharing, then a topic customization entity needs to be added to the schema. Table 4.1 bellow shows a detailed description of the topic attributes, their data types and their usage.

4.3.2 User

The user entity allows linking the topics and tweets to the selected user. It also stores the user's device identifier used by the subscription' service when pushing notifications to the mobile device of the users. Furthermore, the authentication services are managed with the users' entity.

4.3.3 Tweet

The tweets collection holds the minimal necessary content to display the tweet to the user, also, it stores the user's id used by the application to apply the access rules such

Table 4.1: Topic Attributes Description

Attribute	Data Type	*Description*
topicID	String	An auto-generated identifier used to link a tweet to the topic and used to reference the topic for edit and removal requests
uid	String	used to reference the owner of the topic, and reflects on the access rules
keywords	Array[string]	Caries the list of keywords that is used to test an incoming tweet against the topic
title	String	The title is a way for the user to identify the topic. used when filtering and when picking it to edit/remove
delta	int	delta is the number of minutes between each push notification and the previous one.
limitPer	character	Stands for Limit Per and the possible values are `'d':Per Day, 'h':Per Hour, '0' no limit applied;`
limitVal	int	Stands for Limit Value and is used with the limiPer to identify the limit of tweets per hour or day. or ignored if set to unlimited.
relevance-Threshold	float	used in the RTS Engine after calculating the cosine relevance value of tweet terms and topic keywords to determine if it passes the minimal relevance required.

that users only see the topics pushed to them. Additionally the tweet has a snapshot of the tweet object taken from Twitter that is used to display related media of the tweet. Furthermore, it also holds the Twitter user object to show the name, handle and the display photo of the user.

4.4 Communication between RTS components

In this section, the Application Programming Interface (API) between different components, mainly the subscription service and the fireStore, is highlighted. Details about the implementation specification will be explained in chapter 5.

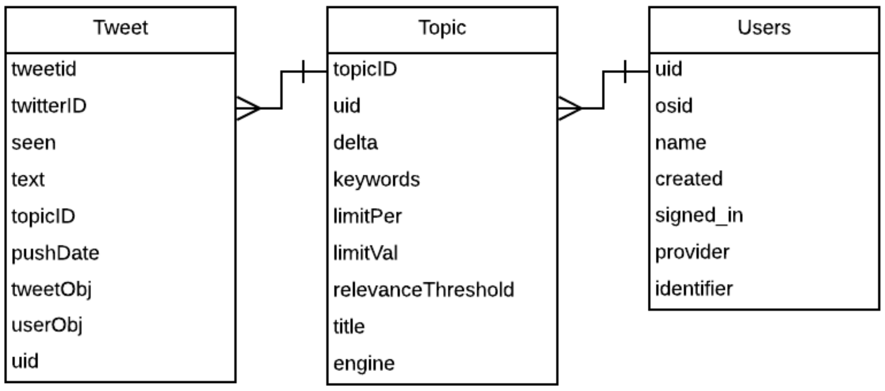


Figure 4.6: The Entity relation digram (ERD)

4.4.1 Subscription Service Rest API

As discussed in previous sections the subscription service takes care of topics creation, modification, and removal. The service offers all of that through a group of rest API listed and explained in table 4.3. The subscription service also offers a set of methods for the RTS engine to push tweets, and communicate the RTS engine instance status.

Table 4.2: Subscription Service Rest API List

Verb	*URI*	*Description*
Get	/rts/topic/:id	Used to retrieve a single topic object identified by the topicId
Get	/topics	used to retrieve all topic objects
Get	/engine/init	used to inform the server about the start of a new RTS engine instance, which then will reassign some of the topics to it and lower the amount for the others.
Get	/engine/heartbeat	used to confirm if the engine is still running in case it has not sent any tweets for a defined time.
post	/tweet/:topic/:tweetid /:engine	used to reference the owner of the topic, and reflects on the access rules

Table 4.3: List of the used FireStore API

Verb	*URI*	*Description*
Java SDK	FirebaseApp .initializeApp	Used to initialize connection with the firebase services and reference the collection
Java SDK	initLoad.get() .getDocuments()	used to retrieve all topic objects
Java SDK	topicsRef .addSnapshotListener	used to create an Event listener to creation/modification and deletion of topics.
AngularFire SDK	DB.collection()	used to create a refinance object to the collection
AngularFire SDK	query.get().then (querySnapshot)	used to execute a retrieval query.

4.4.2 FireStore API

In order to retrieve and update data on the FireStore, an SDK for a good set of platforms is offered by the platform. Further details about the used SDK's are found in the description of the components in Chapter 5. The SDK provides the capabilities of the real-time database allowing the components to be notified of each change without the need to actively reaching and retrieving the full list of items.

CHAPTER 5: IMPLEMENTATION

In this chapter, presents the implementation detail of RTS Component, detailing how they are communicating and highlighting some of the implementation decisions that were made and the reasoning behind them. For the subscriber service, it was built in nodeJS to utilize the non-blocking, event-driven I/O paradigm that it offers.RTS Engine was created on a JavaEE mainly since the original work extended from [7] was written in Java. Additionally it provides a mature ans stable platform. As for the mobile application, a Hybrid mobile application framework was used that helps to utilize web development skills along offering one source code while addressing multiple mobile platforms including Android and ios.

5.1 Subscription Service

The main functionality of the subscription' service is related to receiving users topic request including the creation of new ones, along with modification and removal requests. On the creation of a new topic, the application is required to send the user id as a URI query value, e.g. `?uid='theUserId'` The rest of the topic object is sent in the body of the request allowing flexibility when adding complex data including the array of keywords. Once the topic object is received, it gets first checked for the required information, that is, the title and the keywords. Additionally, the validity of the provided user id is checked against the fireStore by merely retrieving the user object for the given `uid`.

5.1.1 Load balancing

The load balancing takes place in two scenarios. First when a new topic is added the tag of the least-sized engine (the instance that has the minimal amount of topics) is assigned to the topic, further details of how the tag is used in the engine instance will be shown as part of subscribing to the firebase stream in section 5.2.2. The second scenario where load-balancing takes place is when a change in the engine instance is identified, an instance has not confirmed active for the defined maximum idle period, a new instance is added by calling `/rts/engine/inti`.

5.2 RTS Engine

This Section, describes how the topics get synchronized between the database and the array of topics stored in the RAM of running thread. We will start by pointing out the configuration options available for the instance regarding the startup connections and the ones that affect the the behavior of algorithm, controlling if the engine expands the topics terms or sticks with the provided terms by the user.

5.2.1 Configuration options

The original configuration options had lots of variables that affect the frequency of tweets submission, and where set to affect the system as a whole (a common value used across all topics) that was one of the first contributions of the mobile application allowing the users to define their preferences. The other set is related to the location of the index files and the baseurl of the broker (subscription service) in case it was not hosted on the same host.

5.2.2 Sync the Topics

As mentions earlier the initial version of the RTS Engine used to have predefined set of topics build to test the algorithm, so the first course of action was to create the `FireBaseUtil` a utility class the works on top of the the FireStore SDK. The utility receives an instance of the `ScalableFilter` (the main logic of the pipeline) to access the list of topics, Then modify the topics list according to the listening event, to either add as new topic as new ones arrive or perform changes on the a topic on a modification event, The last option is removal where the list gets modified to skip the removed topic.

5.3 Mobile Application

The mobile application is as a Hybrid App allowing the export to multiple platforms including IOS and Android. That is made possible through the use of Ionic framework that allows developing the app using web technologies, the app runs in a `cordova` a container that acts as an interface between native calls and web technologies; through running a web view with exposed functions that translate to native calls of the used plug-ins. That said, the application code base is mainly web-based, along with some native code provided by the selected plug-ins. The Ionic framework comes bundled with angular a web framework that uses data binding and component-based web interface.

5.4 Hosting

In order to achieve a 24/7 uptime, both the RTS engine and the subscription' service were redeployed on a Virtual Machine (VM) purchased from DigitalOcean a cloud computing platform the offers PAAS. The standard droplet was selected which offers 2 GB of Memory, one vCPU Processor, 50 GB in storage and a total bandwidth of 2 TB per month. The OS Selected was Ubuntu 16.04.4 LTS GUN/Linux 4.4.0-119-generic x86 64.

To move the RTS Engine and run it on the VM the project had to be converted to a maven project a software project management tool that deals with the dependencies. And in order for the mobile application to reach it, a domain name was used to point at the server.

CHAPTER 6: EVALUATION

To evaluate achieving the project's functional and non-functional requirements we ran a set of tests. In this chapter, we will explain the evaluation setup. For the functional and usability test, a series of user testing was conducted with HCI-based techniques, where we ask a group of testers to use the application in the format detailed in section 6.2, to confirm achieving the functional requirements and measure the users' satisfaction with the application. For the scalability test, we evaluate each component of the project to determine the saturation point where the performance starts to degrade. The conducted experiments test the system at the level of single instance per component then checks the impact of adding more instances. Unit testing was also used to validate that the elements of each component work correctly. Series of integration tests were also carried on to ensure that the overall system is working correctly as a whole. Though the RTS engine was tested originally to be scalable [7], the performance evaluation covered in this chapter demonstrates that the modifications made did not impact the scalability of the system.

6.1 Performance Evaluation

To evaluate the system scalability , we needed to run a series of tests generating increasing workload. First we tasted the RTS Engine ability to handle multiple edits, creation, and removals of topics concurrently along the ability to run on a large set of topics. To achieve that we used Locust, a python-based load testing tool. Locust is completely event-based working on a set of defined light-weight processes, and therefore it's possible to support thousands of concurrent users on a single machine. Followed is a description of the evaluation setup followed by the results.

6.1.1 Setup

To validate the ability to handle concurrent requests a group of clients was created such that depending on a configuration number they receive, they will execute a series of calls to the RTS subscription service which will create new topics in the FireStore and directly reflect in the list of topics in the RTS engine. To evaluate the effect of the number of topics on the execution time, two test accounts where created for the VM users to use when creating new topics. The created topics had a dense number of terms randomly picked from a pool of trending terms; each topic would have three to seven terms. And the measures were made on 3 ranges set of numbers created topics:50-100, 100-1000 and 1000-5000. Each is including an equal number of terms 5 or 7. They are randomly mixed yet the sequence is stored to repeat the test with the same setup ten times and take the average, worst and best timing for further analysis.
The RTS engine instance the test run on was hosted on the machine detailed in section 5.4. And the machine that generated Locust clients was running a Windows 10.0.17134 64 bit Operating System, x64-based processor. Installed memory (RAM) of 16.0 GB and 6th Generation i7 Intel core.

6.1.2 Evaluation Results

Subscription Service

Figure 6.1 shows the ability of the subscription service to handle concurrent clients. The graph shows three lines.
1) the blue dots raising is the number of active users; where the tool used spawns them gradually to allow seeing the effect as the number of the current users increase.
2) The second purple dashed line is active time highlighting the average time needed for each client to complete a cycle of the commands instructed to call.

(i) *Login*: the initial startup scenario create to establish the connection and act as the testing user client. It also includes fetching the users list of the 1st 50 topics.

(ii) *Create*: the topic creation scenario in which the spawned client selects a random sample of the extracted trending topics terms.

3) Last the green solid line reflects the time needed for the client to get initialized (time to run the starting sequence which includes login and retrieving the list to topics). The green solid line is relatively stable meaning that the start up sequence (`login`) did not get effected by the number of active users, averaging around 200 millisecond. On the other hand the time to execute the active calls ranged between 340 millisecond to a minimal of 160 milliseconds, showing a clear variation depending on the number of concurrent calls. However as the number of users reached 50 concurrent active users the startup calls stopped; as its the configured number of clients. This shows a direct effected of the start up sequence on the execution time of active users. That is related to the topics fetching section of the `login` scenario.

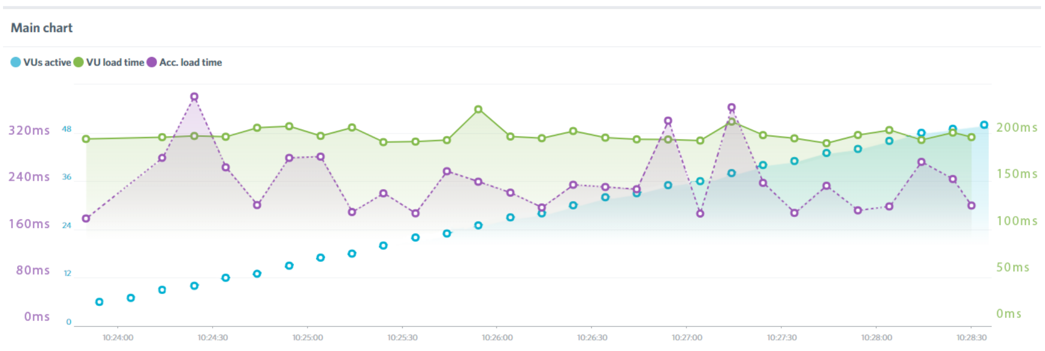


Figure 6.1: Graph showing time in milliseconds per the number of active clients requests

Effect of the number of topics on the engine execution time

Positive outcomes where shown in the evaluation of the light-weight and conservative filtering strategy of the engine pipeline implementation, as shown in Figure 6.2, which plots the number of topics in the x-axis, and the average execution time of the tweet filtering in milliseconds in the y-axis. Plotting the number of topics starting from 50 topics to 5,000 topics. For each number there are three bars: the average of all tweets' processing time to select a tweet to send to the users donated in blue, the average time for qualified tweets in orange, and the average time of unqualified tweets (the one that do not pass the filters) in the gray color. The results clearly show how the RTS engine is not effected by the number of topics and the average processing time only slightly increases with the addition of a thousand topics.

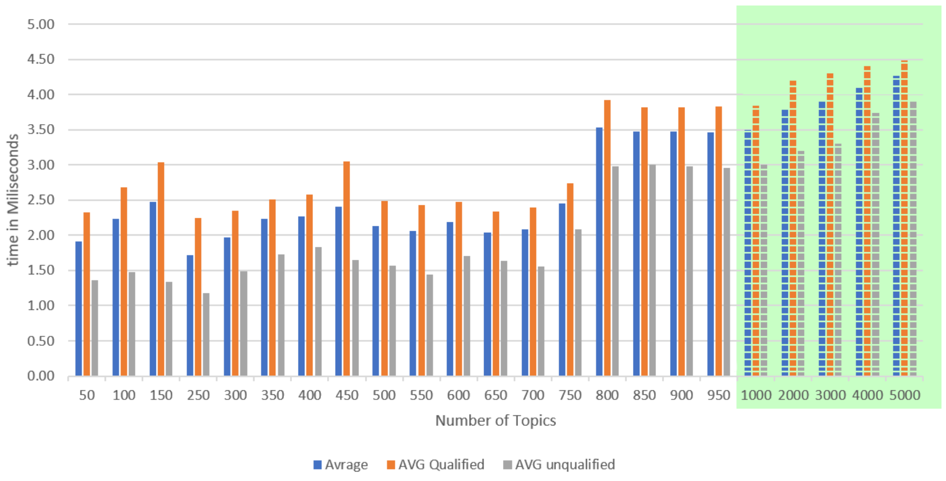


Figure 6.2: Time processing time (milliseconds) versus the number of topics

6.2 Usability Testing

The usability testing focused on both making sure the application is usable, and ensuring the functionalities are all working. To create the test, we started with task analysis as part of establishing the correct conceptual model. We carefully went through the functional requirements and translated them to hierarchical task analysis, detailed in Appendix D. Then, we asked a group of 20 testers to download and use the app, to validate their satisfaction level with the application. The tests are divided into three sets:

1) Given no clue about what the application is about.

2) Briefly informed about the application and the problem it is trying to resolve.

3) Given a detailed explanation of how the application works and the functionalities it provides.

6.2.1 Task Analysis

The following group of tasks were created as a translation of the functional requirements. Followed is the highlighted description of the task aim. Detailed task steps tree can be found in Appendix D. The questions related to each task were as follows:

Sign up

The aim of this task is to ask the tester to create an new account, as an initial step that will allow the other tasks to take place.

Add topic-simple

We split topic creation into two tasks, in this one the testers are asked only to give the topic a title and list out their keywords.

Add topic - advanced

The advanced topic creation focuses on the testers' ability to understand the meaning of the options and being capable of setting the right values. That includes the threshold slider and the tweet frequency.

Filter topics

This task takes place few days later, based on the fact the the tester should have created at least two topics prior to starting this task, the tester is then asked to select filtering to tweets of the first topic.

Navigate to older date

The aim of this task is to confirm the intuitiveness of navigating through the tweets history.

Edit topic

The task covers both navigating though the topics and editing them.

Remove topic

The aim of this task is to evaluate the ability to remove topics, and if the users get the correct confirmation along the reflection on the tweets list.

6.2.2 Usability Testing Results

The initial tests highlighted a number of usability defects in the system. First, unexpected actions from the users like clicking the back button while filling the form resulted in closing the from instead of confirming with them first. This was addressed right away resulting in improvements of the results of the second round of testing. Figure 6.3 highlights a survey results based on the testers feedback containing rating the following questions:

1. I was able to preform the task (Y/N)
2. I was able to figure out what to do easily (1-5)
3. The application described the errors when they happened (1-5)

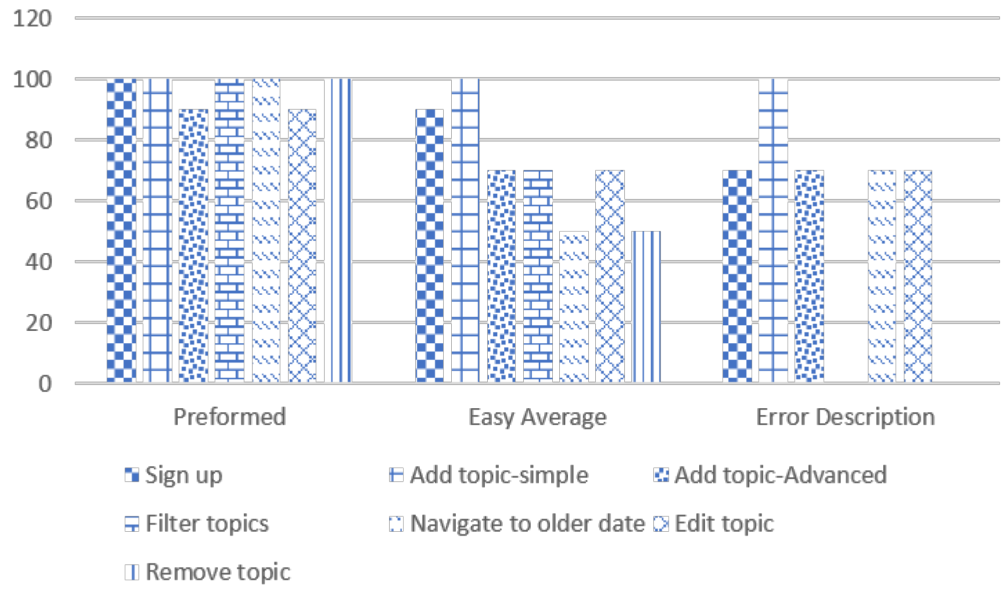


Figure 6.3: Summary of Usability survey answers

Analysis and actions

The first group of bars shown in Figure 6.3 reflect the success rate of the tester to preform each of the tasks. Five of the tasks had a success rate of 100%; the others had about 90%. The second group refereed to the easy of use with a majority of 70% easy having both navigating to older days and removing a topic as the hardest. For navigation, the introduction of a calender date pick helped improve the results as the initial setup required the tester to click multiple times to reach the defined date. For the remove topic task, the fact that the tester needed to edit the topic to find the delete button was counter-intuitive and the solution was to introduce the delete icon in the topics list page.

Last group is related to the error description, majority of the tasks had 70% score, note that both filter the topics and removing a topic Tasks were inapplicable to answer this question: where the tasks required no input, hence not plotted in the Error description. From these results, we were able to conclude that the application successfully achieves the desired functionality, yet there is need for future usability improvements.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

In conclusion, RTS mobile application was successfully developed, to extend the Scalable Real-Time Tweet Summarization RTS Engine [7]. Offering the users the ability to of defining their own topics of interests in a straightforward way. Also, get notified about a new tweets directly using push notifications in real time. Additionally, the ability to navigate through the history of pushed (selected) tweets.
The application has successfully achieved the usability requirements despite few usability issues. A group of 20 users tried the application, and five of the tasks had a success rate of 100% the others had about 90% with the majority of testers scoring 70% on for the feedback on error. Testers mentioned that the tweets they received were in fact relevant to their desired topics of interest and many have reported that they will continue using the application after the test.
In terms of scalability of the RTS engine achieved the desire of processing 5k topics the tweet with an average of 5 milliseconds. Providing that the application layer did not affect the scalability of the engine. Overall, this project provided further functionalities that were not developed in other tweet summarization.

7.2 Future work

For the future work, we recommend: More in-depth analysis of the system scalability, looking at the effects of changing the parameters of topics. Develop RTS System further by providing an administrative dashboard to monitor the platform overall performance. Additionally, follows is a list of recommended future work:

1. *Common Topics* Offering a set of topics created by the administrators based on common keywords for users to subscribe to, and define their frequency of notification for.

2. *Topic Keywords Suggestion*: When filling the keywords auto complete option should be available

3. *Interact with tweet*: This means that the user should be able to re-tweet/like/comment on the tweet directly from the application if their account is linked with Twitter, or taken to Twitter application if required.

4. *Tweets Collections*: Creating a functionality in the application that allows the user to select from the recommended tweet and place it as part of a collection, go back to it faster then navigating to its date. More like favorite list of tweets.

5. *Apply heuristics for a fair load balancing*: Knowing that topics have a different set threshold and the limit is not the same across all, some engine instances might end up idle if they satisfied all the topics they have.

REFERENCES

[1] Personas, scenarios, user stories. https://www.slideshare.net/InteractionDesign/personas-scenarios-user-stories-38054661. (Accessed on 05/13/2018).

[2] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. Trec 2014 temporal summarization track overview. Technical report, NATIONAL INST OF STANDARDS AND TECHNOLOGY GAITHERSBURG MD, 2015.

[3] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the trec-2014 microblog track. Technical report, MARYLAND UNIV COLLEGE PARK, 2014.

[4] Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. Overview of the trec 2016 real-time summarization track. In *Proceedings of the 25th text retrieval conference, TREC*, volume 16, 2016.

[5] Qatar University Office of Graduate Studies. ONESIGNAL - MULTI-PLATFORM PUSH NOTIFICATION SERVICE. https://onesignal.com/, May 2018.

[6] Navdeep Singh. Study of google firebase api for android. *International Journal of Innovative Research in Computer and Communication Engineering*, 4(9):16738–16743, 2016.

[7] Reem Suwaileh, Maram Hasanain, and Tamer Elsayed. Light-weight, conservative, yet effective: Scalable real-time tweet summarization. In *TREC*, 2016.

[8] Twitter Team. Docs — twitter developers. https://developer.twitter.com/en/docs, 2018. (Accessed on 04/09/2018).

[9] Carsten Ullrich, Kerstin Borau, Heng Luo, Xiaohong Tan, Liping Shen, and Ruimin Shen. Why web 2.0 is good for learning and for research: principles and prototypes. In *Proceedings of the 17th international conference on World Wide Web*, pages 705–714. ACM, 2008.

APPENDIX A: FIGURE SNAPSHOTS OF TWEETDECK

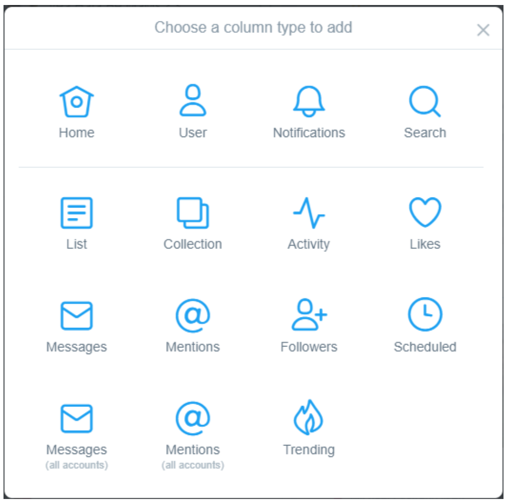


Figure A.1: The type of columns that can be added to your TweetDeck

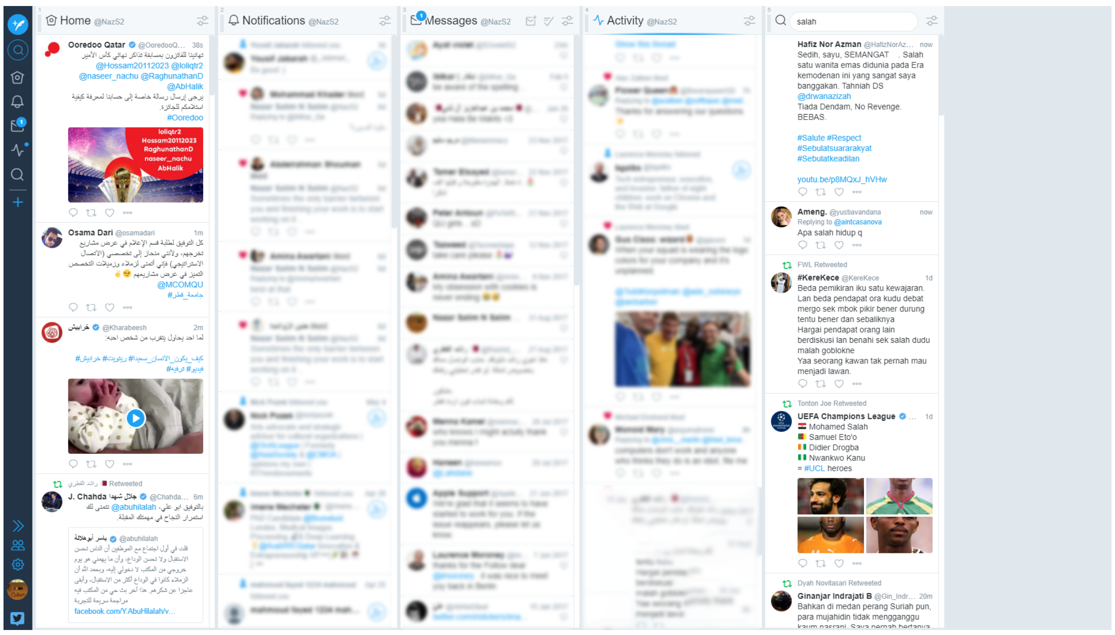
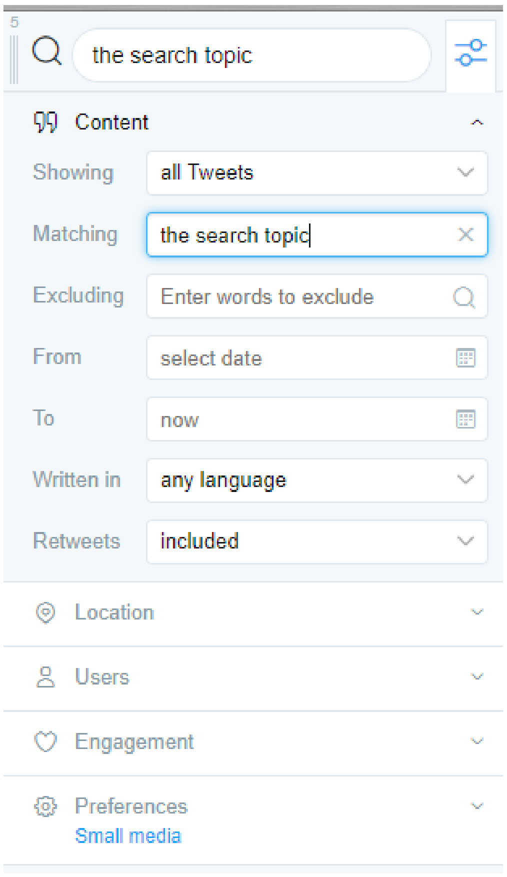


Figure A.2: Example of a Screen Dashboard TweetDeck



5

the search topic

Content

Showing: all Tweets

Matching: the search topic

Excluding: Enter words to exclude

From: select date

To: now

Written in: any language

Retweets: included

Figure A.3: Highlight of the filter Options of a Screen Dashboard TweetDeck

APPENDIX B: USER EXPERIENCE MATERIAL

B.1 Persons

This document is created as part of the UX analysis of the users. Aimed at highlighting a group of users of the application which will be used as a reference when making some design in the design. Reading what was done around Personas Creations in the sides of Interaction Design 2014 [1]

B.1.1 Persona Yazan

Age: 23 years old

Gender: Male

Working Status: Shifting between Jobs

Relation Status: Single or so he says.

Moto in life: Nothing is true…

Languages: Arabic, English

Uses Twitter: (has an account. But never use it) only for searching but rarely.

wears a Watch ..

Topics of interest: Politics.

News and Article

Twitter Interests:

Scenarios: Things happening at his location. (Qatar-¿ Doha, Exact location motazah alsadd)

where he sometimes sees a fire and what's to know what have cussed it.

Would like to get updated about Sryan political war updates.

as an example of low interest is the immigration of siryan people.

General interest:

- Movies

- TV. Series

- New Tech *Phone*: *type*: Android Samsung Galaxy S6

Have it with him: Always

Favorite color: Red then Black.

B.1.2 Persona Baraa

Age: 27 years old

Gender: Male

Working Status: Project coordinator QCharity

Relation Status: Single

Moto in life: beard is all you need.

Languages: Arabic, English

Uses Twitter: No.

No Watch ..

Topics of interest: Beards, Politics, Crypto, fashion/Style.

Using Facebook or youtube + Googling it to get updated on things that happen in Syria, Along the need to know about a set of Crypto Currencies

Scenarios: Things happening at his location. (Qatar-¿ Doha, Exact location motazah alsadd) where he sometimes sees a fire and what's to know what have cussed it.

Would like to get updated about Syrian political war updates. As an example of low interest is the immigration of Syrian people.

General interest: - Movies - TV. Series - Music Phone: type: Android Samsung Galaxy J7 Have it with him: Always 24/7

Favorite color: Blue and Black.

B.1.3 Persona Husam

Age: 25 years old

Gender: Male

Working Status: Researcher

Relation Status: Single

Moto in life: Everything has a price

Languages: Arabic, English

Uses Twitter: Yes, only to get news

Topics of interest: crypto-currencies and cars

Uses Facebook, Twitter, Instagram and Youtube *Twitter* Interests: Following local people for local news and some celebrities in my topics of interest

in general, would like to get updates about new laws related to crypto, has an eye on new ICO's that gets talked about in the social media. He added: I would like to know about crypto news, things related to bitcoin and the future of currencies, business decisions, financial market and the world news generally General interest: - Movies - New Tech - Cars Phone: Type: Iphone 6s Have it with him: Always

Favorite color: Black and White.

B.1.4 Persona Abdol

Age : 25 years old

Gender: Male

Working Status: Part time

Relation Status: Single

Moto in life: WaQa3na Fe El Fa5

Languages: Arabic, English

B.1.5 Persona Khaled

Age : 23 years old

Gender : Male

Working Status: hired on a project.

Relation Status: Single

Moto in life: WaQa3na Fe El Fa5

Languages: Arabic, English

B.2 Scenarios

Step one: Listing out all the Scenarios on high level. Step two: Detail the steps as a Scenario Step three: Create the site-map to address the scenarios. (might be on-line through a UX application)

B.2.1 Download/Sign up and Subscribe to the topics of interest

Baraa heard about the application from the developer and says that he is interested in using the application, Baraa get the application downloaded to his phone directly. – Note It will be free for him -

Then Baraa looks for the logo of the application and recognizes it based on the logo he saw last time he was with the developer. *(Opens the application)*

After opening the application, Baraa sees a splash screen that has the application logo. Then signup/login page shows, *(The login page shows few options like (using Twitter, Gmail, Facebook Or email))*

Baraa Selects to use Facebook option to log in. Facebook shows him a page to confirm giving access to the application. Then gets redirected back to the application home page.

The Application shows Baraa a quick explanation of how the application works and thanks him for joining our testers group. Also, it highlights for him that his actions an anonymous? This information is kept private within the application and used only to evaluate the effectiveness of the application. Then moves him to his profile creation step.

In the profile Creation page. Baraa is presented with some predefined set of topics to select from. Done in a way similar to the topic selection in Quora and Tumblr. Baraa is also shown the option to create his own self-created topic. (it highlighted as an advanced option, yet it tells him that its recommended) Once Baraa is done selecting the topics, The Frequency settings is shown to him and the desired style of display.

Pages List

1. Pages Mentioned in the scenario:
2. Splash Screen
3. Login/signup Page
4. Welcome Page (For First timers)
5. Create Profile.
6. Success page.
7. Highlighted tweets. (empty option)

B.2.2 Abdol Learned about the Application

Abdol Heard from his friend that they are developing a research-based application that would summarize tweets and send it to him based on his interests. He goes through similar Steps as Baraa to create and customize his profile. And been using the application for a week with a set 60 Reamended tweets. While he was out with some of his finds a topic came and he said yea I read it on a tweet that was sent to me. He then attempts to go through the app to get the exact tweet. He starts with opening the application, He is then auto-logged into the use and redirected to the dashboard. Which highlights for him some summaries (Average info sent per day/week and total until the day) and along with a colander based listing of the tweets that were sent. Abdol then scrolls to the left to go back a week and two days and found the ten tweets that were sent to him on that day. Scrolls down and sees it as the 7th. Reads it to his friends and closes the application.

Pages Mentioned in the scenario:

1. Login page (yet here its skipped)
2. Dashboard page
3. Welcome Page (For First timers)
4. Create Profile.
5. Success page.
6. Highlighted tweet. (empty option)

APPENDIX C: USER INTERFACE MATERIAL

C.1 Screens Design

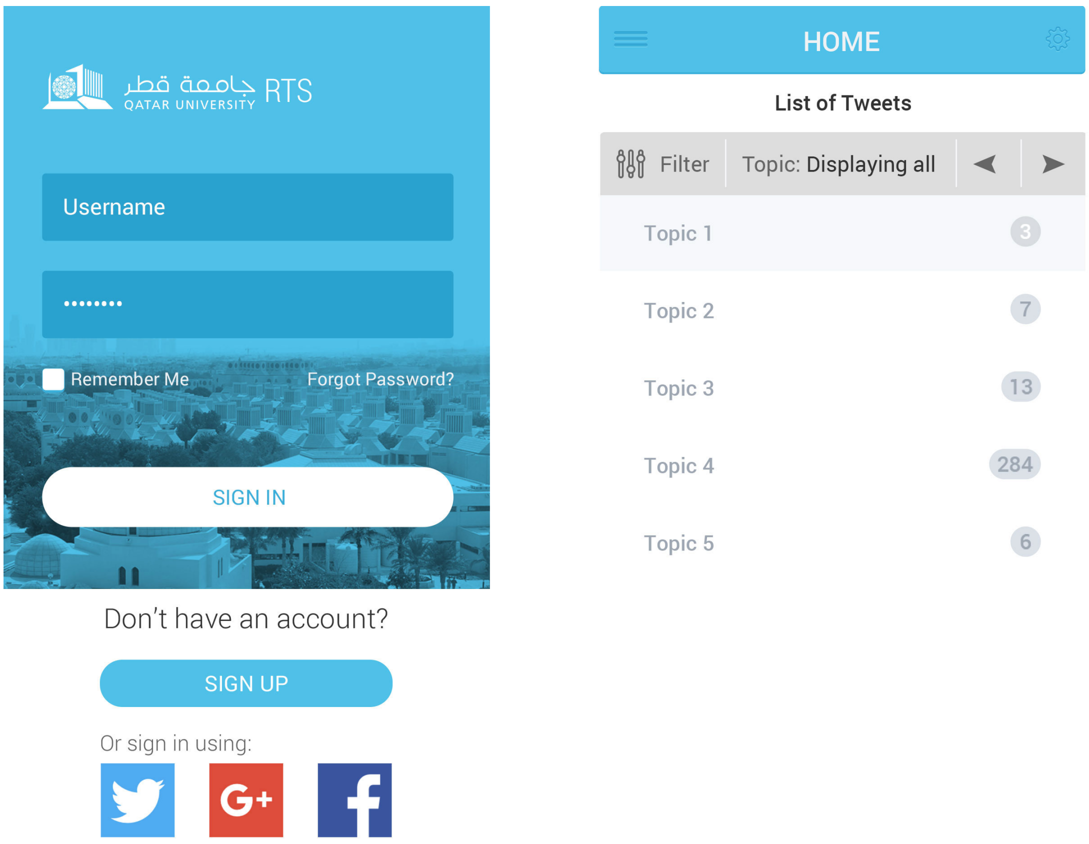


Figure C.1: Screen Login and tweets Listing

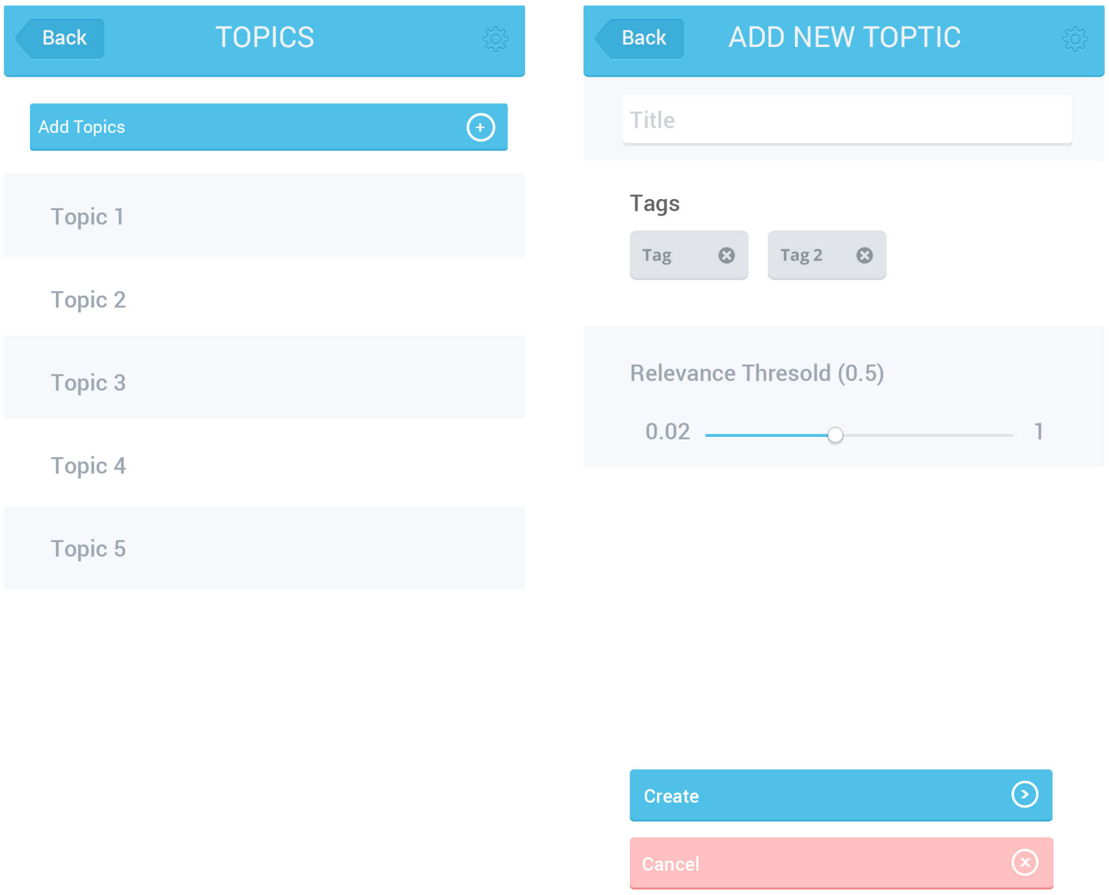


Figure C.2: Screen Topics Listing and topics Form

Followed is the Task analysis for the UI Test Tasks.

D.1 Add Topic Simple

Do: 1, 1.1, 1.1.1, 1.1.2, 1.1.2.1, 1.1.2.2, 1.1.3

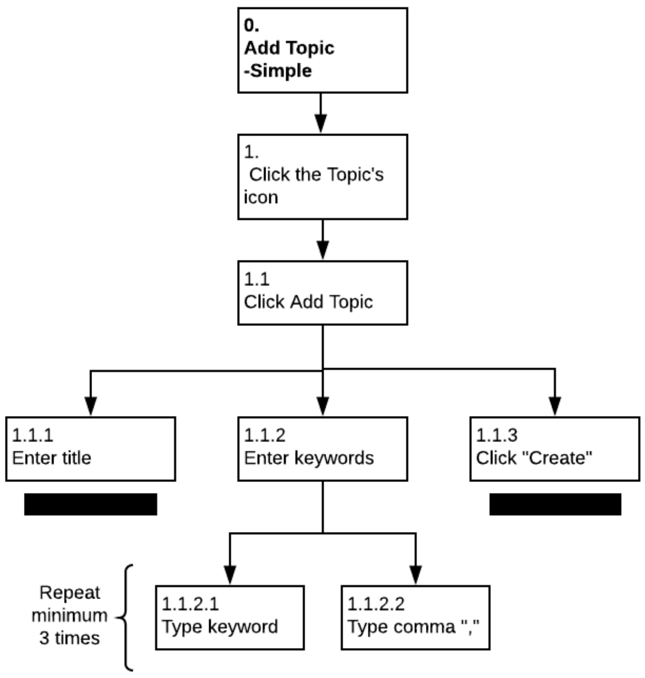


Figure D.1: Hierarchical task analysis for Add topics

D.2 Add Topic Advanced

Do: 1, 1.1, 1.1.1, 1.1.2.1, 1.1.2.2, 1.1.3, 1.1.3.1, 1.1.3.2, 1.1.3.2.1, 1.1.3.3, 1.1.4

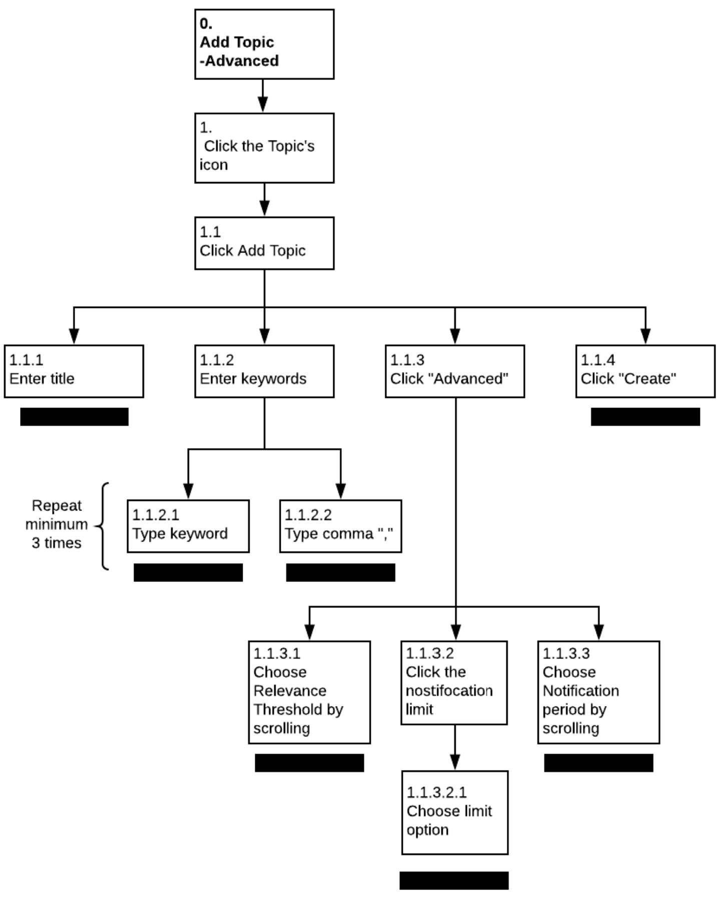

0. Add Topic -Advanced
1. Click the Topic's icon
1.1 Click Add Topic
1.1.1 Enter title
1.1.2 Enter keywords
1.1.3 Click "Advanced"
1.1.4 Click "Create"
Repeat minimum 3 times
1.1.2.1 Type keyword
1.1.2.2 Type comma ","
1.1.3.1 Choose Relevance Threshold by scrolling
1.1.3.2 Click the nostifocation limit
1.1.3.3 Choose Notification period by scrolling
1.1.3.2.1 Choose limit option

Figure D.2: Hierarchical task analysis for Add topics

D.3 Edit Topic

Do: 1, 1.1, 1.1.1, 1.1.4

or

Do: 1, 1.1, 1.1.1, 1.1.2, 1.1.2.1, 1.1.2.2, 1.1.4

or

Do: 1, 1.1, 1.1.3, 1.1.1.3.1, 1.1.4

or

Do: 1, 1.1, 1.1.3, 1.1.3.2, 1.1.3.2.1, 1.1.4

or

Do: 1, 1.1, 1.1.3 ,1.1.1.3.3 , 1.1.4

or

Do: 1, 1.1, 1.1.1, 1.1.2.1, 1.1.2.2, 1.1.3, 1.1.3.1, 1.1.3.2, 1.1.3.2.1, 1.1.3.3, 1.1.4

or

Do: 1, 1.1, 1.1.1, 1.1.2.1, 1.1.2.2, 1.1.3, 1.1.3.1, 1.1.3.3, 1.1.4

or

Do: 1, 1.1, 1.1.1, 1.1.2.1, 1.1.2.2, 1.1.3, 1.1.3.1, 1.1.4

or

Do: 1, 1.1, 1.1.1, 1.1.2.1, 1.1.2.2, 1.1.3, 1.1.3.3, 1.1.4

or

Do: 1, 1.1, 1.1.3, 1.1.3.1, 1.1.3.2, 1.1.3.2.1, 1.1.3.3, 1.1.4

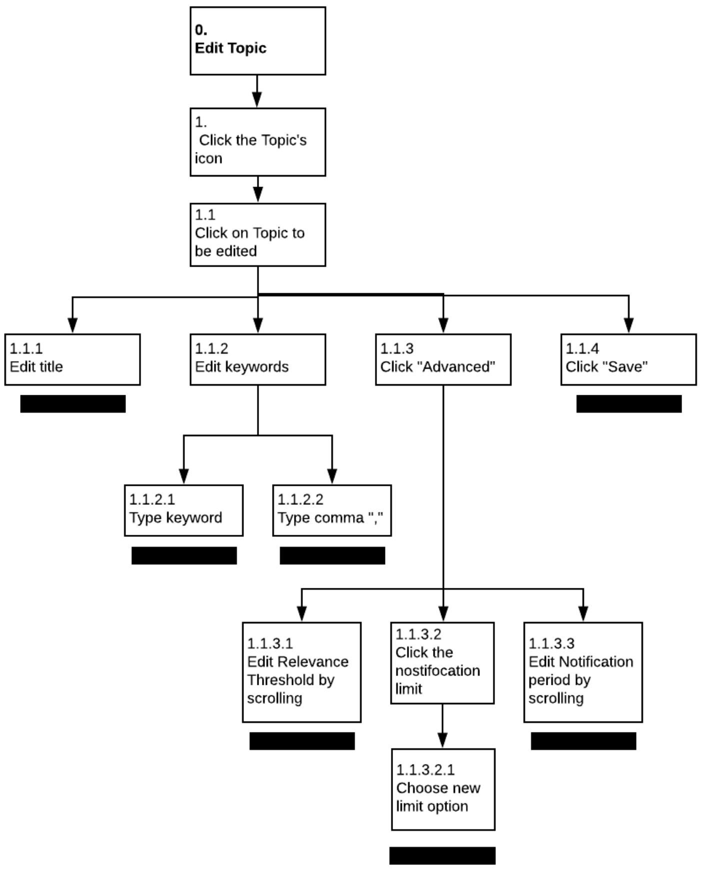

0. Edit Topic
1. Click the Topic's icon
1.1 Click on Topic to be edited
1.1.1 Edit title
1.1.2 Edit keywords
1.1.3 Click "Advanced"
1.1.4 Click "Save"
1.1.2.1 Type keyword
1.1.2.2 Type comma ","
1.1.3.1 Edit Relevance Threshold by scrolling
1.1.3.2 Click the nostifocation limit
1.1.3.3 Edit Notification period by scrolling
1.1.3.2.1 Choose new limit option

Figure D.3: Hierarchical task analysis for Add topics

D.4 Sign Up

Do: 1, 1.1, 1.2, 1.3

Or

Do: 2, 2.1, 2.2, 2.3

Or

Do: 3, 3.1, 3.2, 3.3

Or

Do: 4, 4.1, 4.2, 4.3

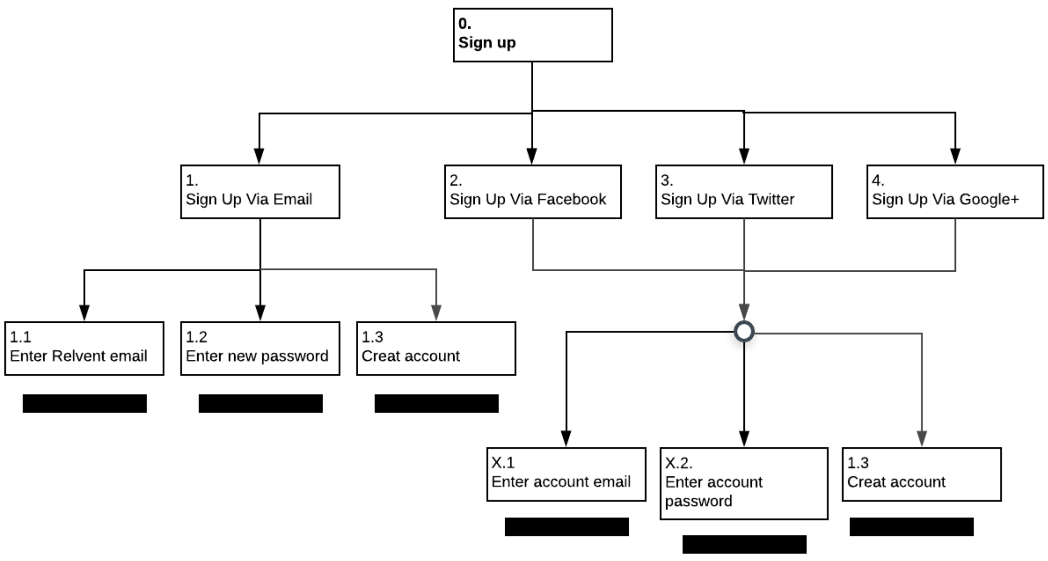


Figure D.4: Hierarchical task analysis for Sign Up

D.5 Navigate to old date

Do: 1

or

Do: 2, 2.1, 2.2

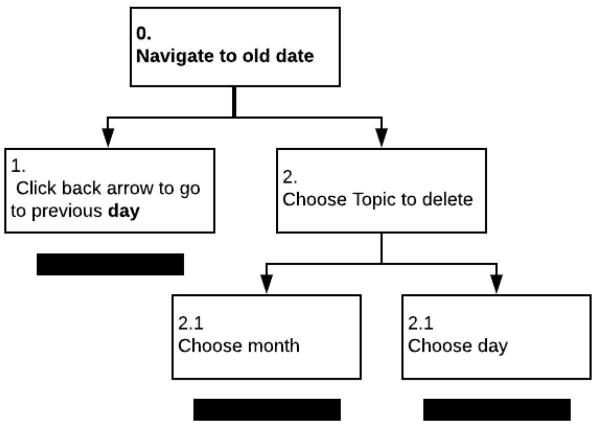


Figure D.5: Hierarchical task analysis for Navigating to old date

D.6 Navigate to old date

Do: 1

or

Do: 2, 2.1, 2.2

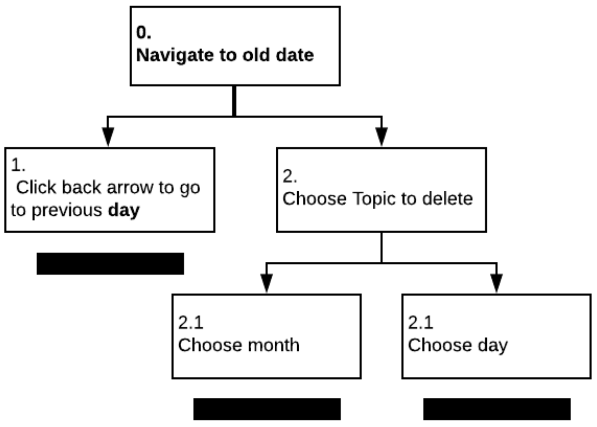


Figure D.6: Hierarchical task analysis for Navigating to old date

D.7 Filter Topics

Do: 1, 1.1, 1.1.1

D.8 Remove Topics

Do: 1, 1.1, 1.1.1

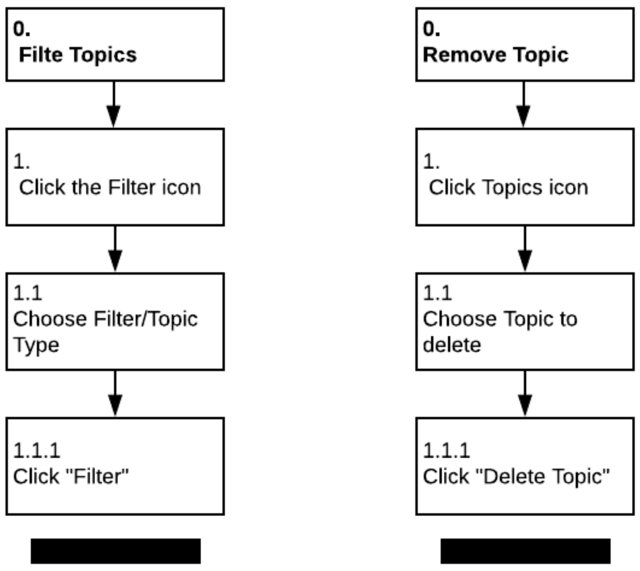


Figure D.7: Hierarchical task analysis for Filter Topics and Remove Topics