



THOMPSON RIVERS UNIVERSITY

SENG 3210 –
Applied Software Engineering

Voting System App

Nduonyi Jack Ukitetu (T00647975)

Sam Elgert (T00609291)

Almat Bolatbekov (T00667332)

29th March 2023

Table of Contents

1 Introduction	5
2 Design Problem	6
2.1 Problem Definition	6
2.2 Design Requirements	6
2.2.1 Functions	6
2.2.2 Objectives	6
2.2.3 Constraints	7
3 Solution	8
3.1 Solution 1	8
3.2 Final Solution	10
3.2.1 Features and the software architecture	12
3.2.2 The system interfaces	14
3.2.3 The user interface design	15
3.2.4 The requirements traceability matrix	15
3.2.5 Environmental, Societal, Safety, and Economic Considerations	16
3.2.6 Limitations	17
4 Team Work	17
4.1 Meeting 1	17
4.2 Meeting 2	18
4.3 Meeting 3	18
4.4 Meeting 4	18
4.5 Meeting 5	18
4.6 Meeting 6	19
5 Conclusion and Future Work	19
6 References	21
7 Appendix	22

- The table of contents should be automatically generated by selecting "References/ Table of Contents." Remember that the table of contents should not have an entry of the "Table of Contents" itself.
- Proofread the text for typing and grammar mistakes.
- Follow the IEEE Bibliography style for the references by selecting "References/ Citations & Bibliography/ Style".

List of Figures

Figure 1. Solution 2 UI.

Figure 2. Solution 2 Activity Diagram.

Figure 3. Final Solution Activity Diagram.

Figure 4. Final solution UI.

Figure 5. Component Diagram for the OVS.

Figure 6. Class Diagram for the OVS.

List of Tables

Table 1. Decision matrix chart for the considered alternatives.

Table 2. Requirement traceability matrix for the OVS.

1 Introduction

With the rise of pandemic diseases such as COVID-19 and as our connectivity among people from different walks of life increases. Physical distance has to be maintained so that the spread of these diseases is mitigated. But this leads to an issue on how decisions can be made in a large group if a physical meeting is not an option. Making your opinion will have to be remote. This will also reduce the cost of in-person voting and give access to people with disability. “In allowing them to easily vote from home, online voting decreases their cost of voting to a much greater extent than for a person without mobility restrictions”[1].

An Online Voting System (OVS) will be used that will allow managers to post topics where people can then vote on them or write their own opinions. It also will provide a real-time dashboard that summarizes the voted topic's current statistics. It will allow users to participate from anywhere in the world. Hence increasing a more representative outcome.

This report is divided into the following sections: the design issue, potential solutions, teamwork, project management, conclusion, and recommendations for future work. The requirements, functions, objectives, and restrictions are all described in the section on the design challenge. The solution, which includes our iterations and the OVS features, concerns, and constraints, comes after design. Both the teamwork and project management parts discuss our teamwork as well as the structure and timing of the projects. All of the sections will be examined and any possibilities for further research will be assessed in our conclusion. We will then take a moment to reflect on the project as a whole and how it relates to our commitment to lifelong learning.

2 Design Problem

This section has the following two subsections:

2.1 Problem Definition

Write a problem statement of the project.

The COVID-19 pandemic caused significant global disruptions, and effective tracking and management of the disease were critical to minimizing its impact. However, decisions still will need to be made. Where people for their comfort vote on topics created by managers to make decisions. The current system used is not reliable for everyone and may be too complex for everyday users. The absence of a system with these capabilities has led to difficulties in communication and decision-making with analytics.

There is a pressing need for a comprehensive Online Voting system that provides accurate, timely, and relevant data to the public, votes and managers. Such a system should integrate data from various voters or users. This proposed system should also enable users to create polls which can then be voted on where the data will be saved on firebase. Additionally, the OVS should be accessible to all stakeholders and provide user-friendly dashboards and visualizations to facilitate decision-making and communication. Developing an OVS that matches and exceeds these criteria and the criteria listed below this will lead to user satisfaction.

2.2 Design Requirements

This section has the following three subsections:

2.2.1 Functions

- The User shall be able to create an account in the OVS
- The User shall be able to log in
- The Manager shall be able to create the post (topic for discussion)
- OVS shall present all topics for the User and Managers
- The Users and Managers shall be able to vote
- The data shall be stored in the database

2.2.2 Objectives

1. Develop a user-friendly and responsive Android app that allows users to remotely access and participate in voting for specific topics.
2. Implement an intuitive and easy-to-use Discussion Topic Management system that enables managers to create, manage, and delete topics for voting.

3. Design and implement a real-time dashboard that provides managers with current statistics on the voted topic, including the number of votes, percentage of participation, and other relevant metrics.
4. Ensure that the OVS complies with the performance standards for mobile applications, such as fast response times, low latency, and minimal battery usage.
5. Ensure that the OVS is secure and supports users' authentication and privacy mechanisms to prevent unauthorized access.
6. Ensure that the OVS is easily modifiable, allowing for the addition of new UI components with the minimal development effort.
7. Ensure that the OVS is compatible with mobile devices with Android API level 19 (KitKat) or higher API levels.
8. Document the design, implementation, and testing of the OVS, including technical specifications, system architecture, and user manuals.
9. Ensure that the OVS complies with applicable standards and regulations, such as data protection laws, security protocols, and ethical and societal considerations.
- 10.

2.2.3 Constraints

- **CO_1:** The OVS shall be enabled on mobile devices with Android API level 19 (KitKat) or higher.
- **CO_2:** The OVS shall enable users to vote or report their opinions remotely using the OVS about a particular topic.
- **CO_3:** The OVS shall enable managers to create new topics and perform discussion topic management.
- **CO_4:** The OVS shall provide managers with a real-time dashboard that summarizes the current statistics of the voted topics.

3 Solution

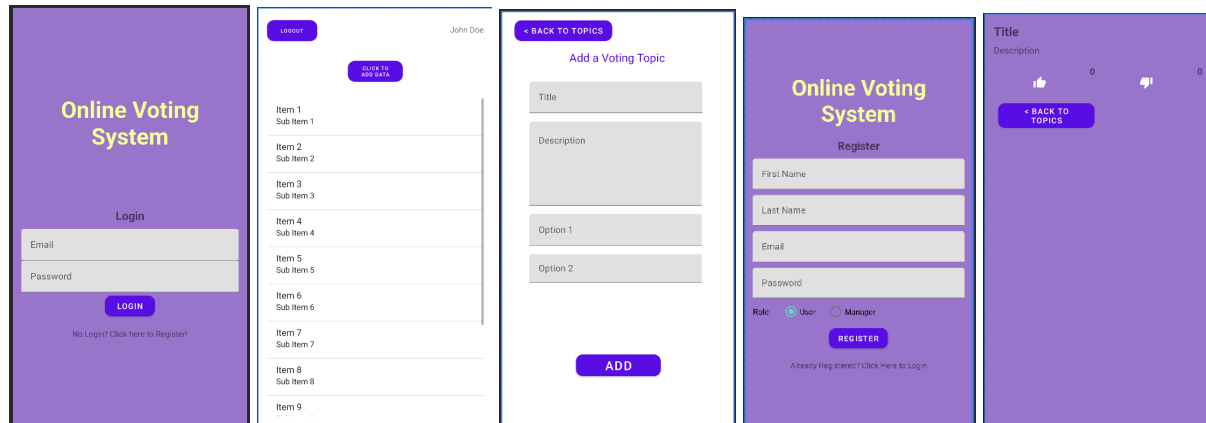
This section of the report will describe how the final solution was following an interactive process that identified the most important aspects of the OVT and the features to prioritize in the development of the first prototype. The final solution of the OVT application has functions where users can log in, register, and view post topics created by managers. The system is designed to have two types of accounts, managers and users. Managers have the ability to create new post topics, while users can view the topics and vote for one of the two options provided.

3.1 Solution 1

Initially, the Voting System was proposed to be developed as a mobile app that would enable users to register, log in, create voting topics, and cast their votes for one option. To store the data, text files would be used, and there would be three separate files. The first one would be the "Users" file, which would contain all the relevant details about the user accounts. The second file, named "Posts", would store all the created posts along with their descriptions. The last file, known as "Votes", would record the post ID of the voted post and the user who cast the vote. This solution would very feasible for small-scale applications, However, there are a number of issues that could arise with storing data in text files. The main issue is with security vulnerability, text files are not very secure and anyone can see the data and change it. Secondly, the data management is very limited, with storing data in the text file it will be very hard to retrieve the information if it would be a very complex query. Overall, while storing data in text files can be a viable solution for small-scale applications, an online voting system application may require more sophisticated data management capabilities, security features, scalability, and concurrency control mechanisms

Solution 2

In the second solution Online Voting System was proposed to be developed as a mobile app that would enable users to register, log in, create voting topics, and cast their votes for one option, which is similar functionalities as Solution 1, However in this solution, the data is stored in Firebase real-time database. This solution gives more features in terms of security and data management capabilities. The design of the application is presented below in the pictures. Moreover, there is an activity diagram where you see that the User and Manager would have different pages. The user page is where users can view all of the posts and vote for one of the options, whereas managers can do the same however they can add the voting posts.

**Figure 1.** Solution 2 UI.

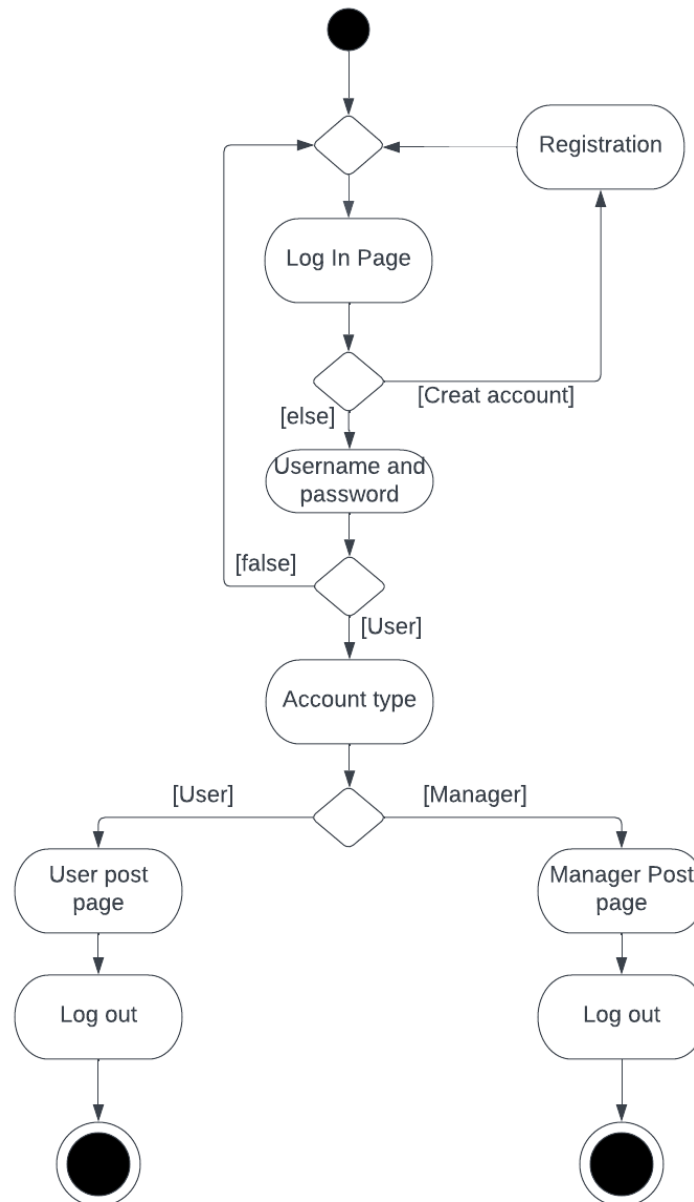


Figure 2. Solution 2 Activity Diagram.

After some consideration, it was decided that the Manager and User post pages should merge as the functionality is almost the same, and the application would just take unnecessary memory duplicating the pages' functionalities. Another consideration was a modification of the UI so that it will look more pleasant to the users.

3.2 Final Solution

The final solution is very similar to solution 2 and has the same functionalities, however, we merged the User post and the Manager post pages, so that it will be more efficient

and created a Post View page where users and managers can see the list of the created voting posts and can vote for them, including the addition of the post which can only be done by managers. In this solution, the UI of the OVT was also modified. Moreover, the voting page for the users now shows the pie chart where the total number of votes is shown, including the statistics of option 1 and option 2.

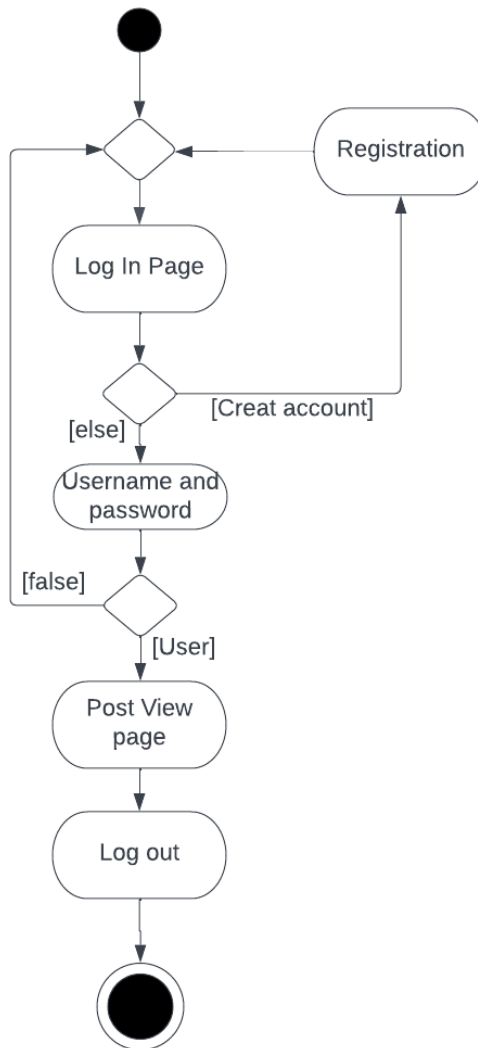


Figure 3. Final Solution Activity Diagram.

		Solutions					
		Solution 1		Solution 2		Final Solution	
Criteria	Weight	Score	Partial Score	Score	Partial Score	Score	Partial Score

Functionality	0.40	6/10	0.24	5/10	0.20	8/10	0.32
Simplicity	0.25	2/5	0.10	4/5	0.20	3/5	0.15
Cost	0.20	10/15	0.14	11/15	0.15	11/15	0.15
Safety	0.15	4/10	0.06	9/10	0.14	9/10	0.14
Sum	1.00		0.54		0.69		0.76

Table 1. Decision matrix chart for the considered alternatives.

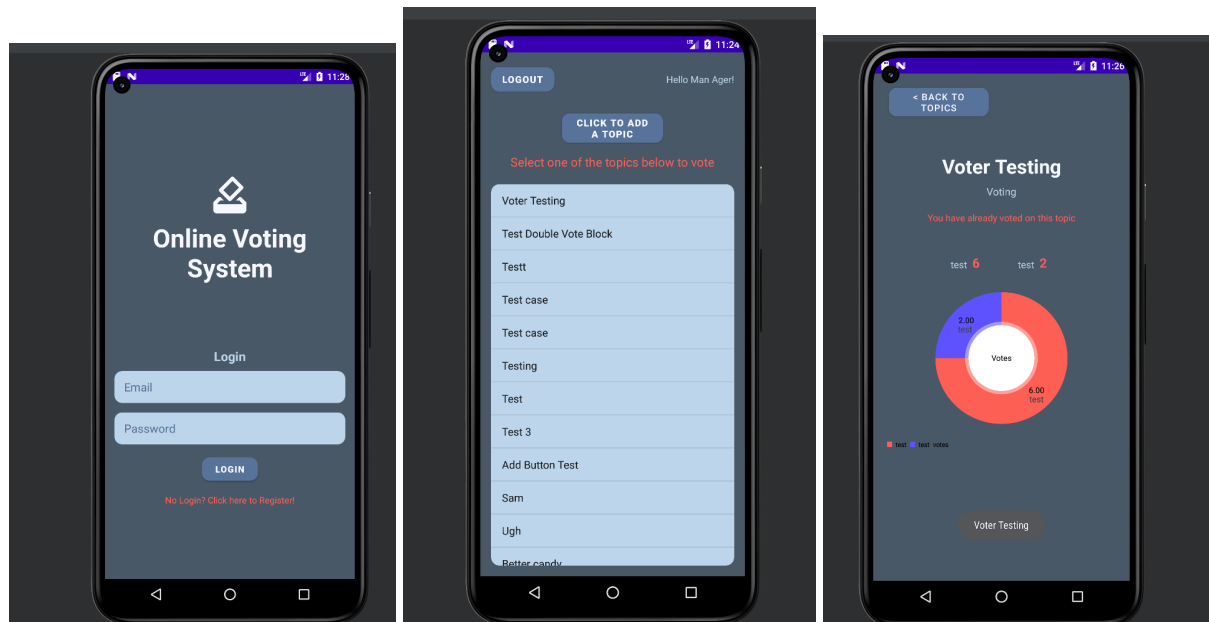


Figure 4. Final solution UI.

3.2.1 Features and the software architecture

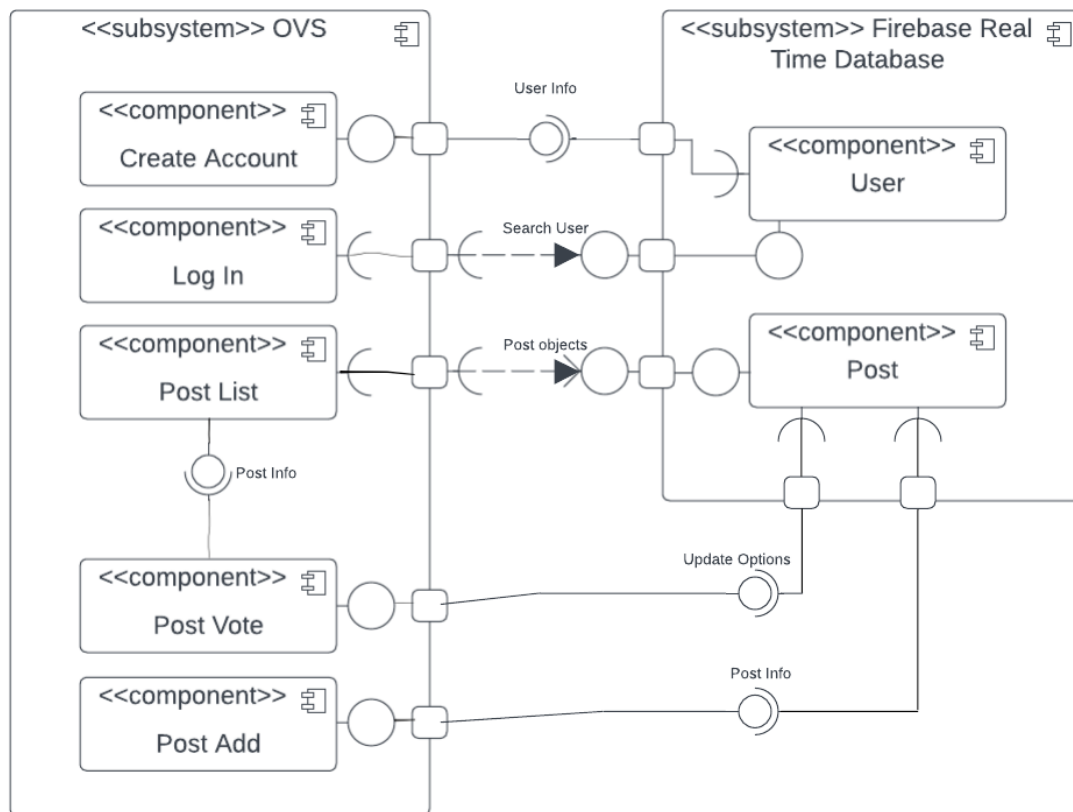


Figure 5. Component Diagram for the OVS.

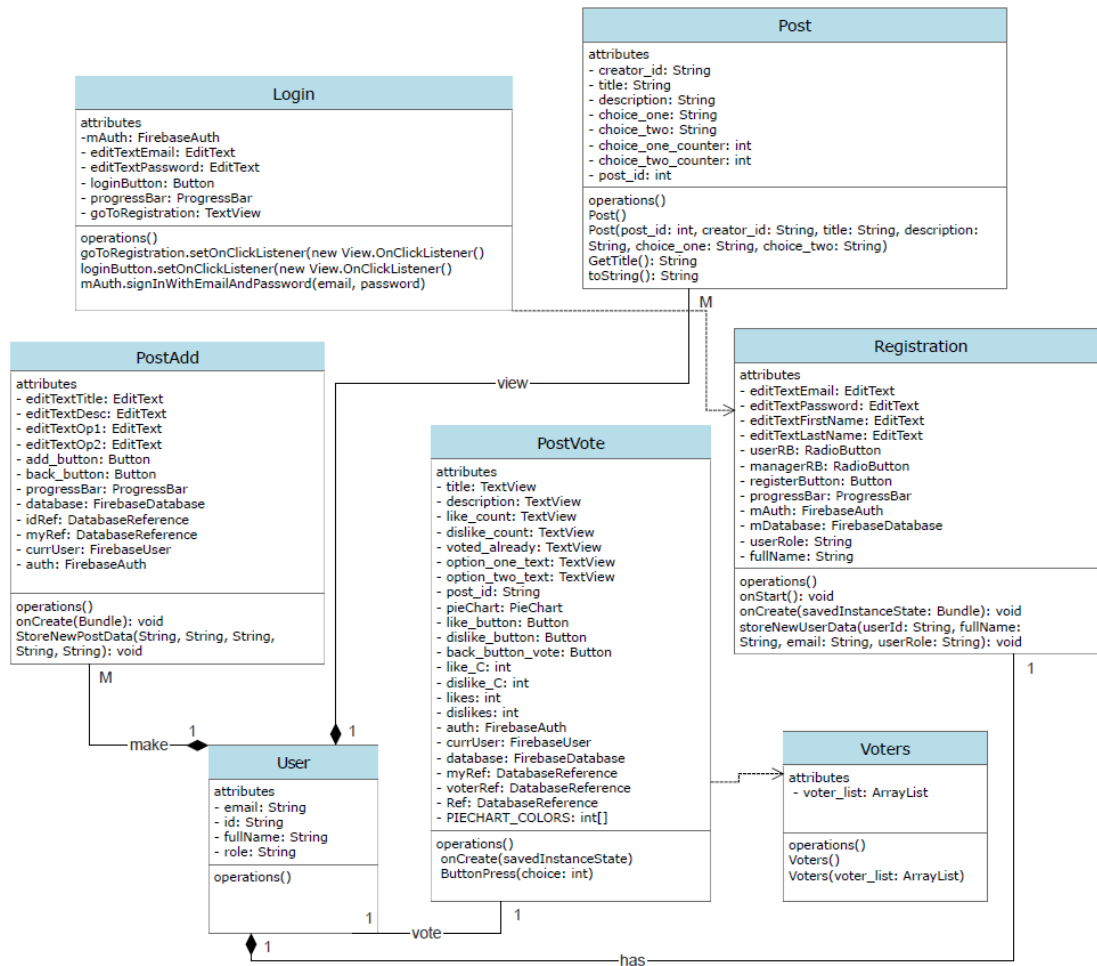


Figure 6. Class Diagram for the OVS.

3.2.2 The system interfaces

Our system currently has no temporal events, however, it has many different signal events. Our system has data-driven signal events. The data-driven events are triggered by data changes in a database reference and they result in data being updated and displayed in the app. This subsection covers the external data-driven components

Data-Driven Event Signals:

- New post added: When the post data changes in the database, the system gets a response where it updates the list of voting topics
- New vote recorded: When the vote data changes the system gets a response with the updated voter list which then blocks the current user from voting twice.

3.2.3 The user interface design

The functionality of the system is dependent on user interactions with our interface components. The business-driven events are triggered by things like clicking buttons and they result in some data being sent to our Firebase database. This subsection these user-driven events and the system responses to these events.

User Driven Events:

- Register Button: Checks registration details are valid and then sends the new authorized user details to the Authorization database.
- Login Button: Checks if login information is valid and then returns the user details from the Authorization database.
- Add Topic Button: Creates a new topic Post object from the information the user inputs into the form. The object is then saved into the real-time database under the posts reference.
- Post Voting Buttons: These add the current users' unique ID to a list of voters in the post object. The post object is then updated with this new list in the real-time database.

3.2.4 The requirements traceability matrix

ID	Ass. ID	Business Need, Justification	Project Objective	Requested By	WBS Element	Specification	Design	Test Cases
1	1.1	Register page test	Need a registration to make an account	User	5	Finished	Finished	TC_01
2	1.2	Login page test	A login page for created account	User	8	Finished	Finished	TC_02
3	1.3	List view (clickable) test	To ensure list view is clicked with an intent link	User	15	Finished	Finished	TC_03
4	1.4	Add topic test	Manager	User	12	Finished	Finished	TC_04

			should be able to add new voting topics					
5	1.5	Vote topic deletion	Vote topic can be deleted	Jack	2	Not started	Not started	

Table 2. Requirement traceability matrix for the OVS.

3.2.5 Environmental, Societal, Safety, and Economic Considerations

3.3.5.1 *Environmental considerations*

For sustainability, the app should maintain its use over a long period of time. Low battery and processor usage.

3.3.5.2 *Societal considerations*

For societal impacts, the app should help track votes on a topic to which decisions can be made. Informing the user about the results of the votes.

For ethics, the app should be user-friendly, meaning the app should not be hard to operate. They will also be read-aloud features to help people with seeing disabilities.

3.3.5.3 *Safety considerations*

For health and safety, we want to keep computers safe; therefore we want a protected app that is not at risk of non-breachable hacks. The app will keep voting anonymous for the interface, making voting safer. The app will also provide a 2FA authentication to ensure illegal login of accounts.

For regulatory compliance, the app should follow the law of app-making, and not disregard any rules against it for it to make public.

3.3.5.4 *Economic considerations*

For economic factors, we consider how many people download the app and how much funding is required for the mass upload of the application. This means a low-cost app is appropriate. The maintenance fee will also be considered to lower the cost.

For reliability, our app is consistent and accurately displays each vote when the app is opened. It will also have a maximum downtime time of 15 minutes. Information on the app is updated with every click.

For aesthetics, our app should keep professionalism and a decent app interface with no contrasting colors. With a simple format making it easier to use.

3.2.6 Limitations

The OVS application has several limitations. One of them is the inefficient creation of posts. The database stores the value which represents the number of all posts that are stored there. When the manager creates a new post, the application requests this counter from the database to declare the id for that particular post, consequently adding the new post to the database. This limitation affects the efficiency of the application, moreover, the application can create posts with the same id if two managers will create a post at exactly the same time.

Whenever the user creates an account there is a free choice of choosing which account type he wants to make, which can be a limitation for the Online Voting System. The application does not check the person that wants to make the manager account, which can lead to many managers and the number of created posts, resulting in high consumption of memory

Another limitation of the OVS is no option for editing, updating, or deleting posts by managers. The only way to change or delete data is through the database, which cannot be accessed by managers. This can be a problem for a manager, for example, the manager created the post, but he observed that he made an error while creating the post, and there is no possible way to update the post or delete it.

The limitations that are described in this section suggest the obvious potential for future features as the application is developed further.

4 Team Work

4.1 Meeting 1

Time: February 26th, 2023, 4:30 pm to 6:00 pm

Agenda: Distribution of Project Tasks

Team Member	Previous Task	Completion State	Next Task
-------------	---------------	------------------	-----------

Sam Elgert	N/A	100%	Creation of the Firebase and bitbucket
Almat Bolabekov	N/A	5%	App coding
Nduonyi Jack Ukitetu	N/A	5%	Report documentation.

4.2 Meeting 2

Time: March 7th, 2023, 10:00 am to 10:15 am

Agenda: Review of Individual Progress

Team Member	Previous Task	Completion State	Next Task
Sam Elgert	Creation of the Firebase and bitbucket	10%	App coding
Almat Bolabekov	App coding	20%	App coding
Nduonyi Jack Ukitetu	Report documentation.	20%	App coding

4.3 Meeting 3

Time: March 16th, 2023, 10:00 am to 10:15 am

Agenda: Review of Individual Progress

Team Member	Previous Task	Completion State	Next Task
Sam Elgert	App coding	40%	App coding
Almat Bolabekov	App coding	40%	App coding
Nduonyi Jack Ukitetu	App coding	40%	App coding

4.4 Meeting 4

Time: March 16th, 2023, 10:00 am to 10:15 am

Agenda: Review of Individual Progress

Team Member	Previous Task	Completion State	Next Task
Sam Elgert	App coding	40%	App coding
Almat Bolabekov	App coding	40%	App coding
Nduonyi Jack Ukitetu	App coding	40%	App coding

4.5 Meeting 5

Time: March 19th, 2023, 6:00 pm to 6:15 pm

Agenda: Review of Individual Progress

Team Member	Previous Task	Completion State	Next Task
Sam Elgert	App coding	85%	App coding
Almat Bolabekov	App coding	85%	App coding
Nduonyi Jack Ukitetu	App coding	85%	App coding

4.6 Meeting 6

Time: March 25th, 2023, 4:00 pm to 4:15 pm

Agenda: Review of Individual Progress

Team Member	Previous Task	Completion State	Next Task
Sam Elgert	App coding	100%	App coding, Report documentation
Almat Bolabekov	App coding	100%	App coding, Report documentation
Nduonyi Jack Ukitetu	App coding	100%	App coding, Report documentation

5 Conclusion and Future Work

As a group, we achieved multiple big milestones. We coded all the desired requirements for a voting app with android studio using Java. Our voting app has many features including a fully functional registration and login system, and the ability to vote on topics or create topics to vote on if you are a manager which is all saved in a real-time database on firebase. We learnt many cooperative methods in how to work well as a group while both programming together and writing this report. Our final design has many functions that we wanted to include and is not missing one key feature. As this design followed the engineering process for us to reach the final solution. The features that we have included for the app are; registration and login, vote add and a voting page where users can vote on a topic, this will allow a user to vote once. The objectives we achieved are based on requirement fulfillment, performance, safety and efficiency. For the cost, with little going to maintenance cost. Secondly, for safety, the main goals we accomplished were that our code is non-breachable and provides a safe and secure password system both of which were achieved. Finally, our code is only missing none of our efficiency goals and they were all accomplished as the code runs at amazing speeds with few halts and has a very basic interface so that anyone can use it without any issues.

Our final design is intended to be used as a general-use safe for an average household, business corporation and national polls, therefore more planning and research would have to be done to make the UI more user-friendly and a better data structure algorithm in our database. Improvements to the code for next time would be good for making it more modular and less coupling of classes and more cohesion. Secondly, if a vote is created with all info empty it still adds it to the database this will be fixed for future work. Upgrading the code so that one can vote without wifi and it will be added to the database when the user is connected to the internet. As well as possibly programming a notification system to send a notice to a mobile device when a new poll is added or the voting results are out. Also making a change that does not cause a vote to have the same id if created at the same time. Fixing the OVS to make options for editing, updating, or deleting posts by managers.

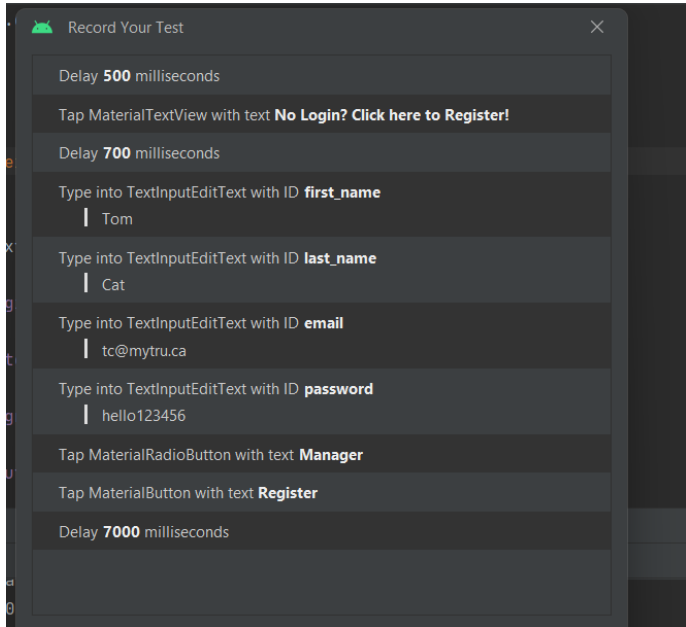
6 References

[1] M. Wigginton and D. Stockemer, "Does the Introduction of Online Voting Create Diversity in Representation?," SAGE Journals Home, 2021. [Online]. Available: <https://journals.sagepub.com/doi/full/10.1177/14789299211064450>. [Accessed: 29-Mar-2023].

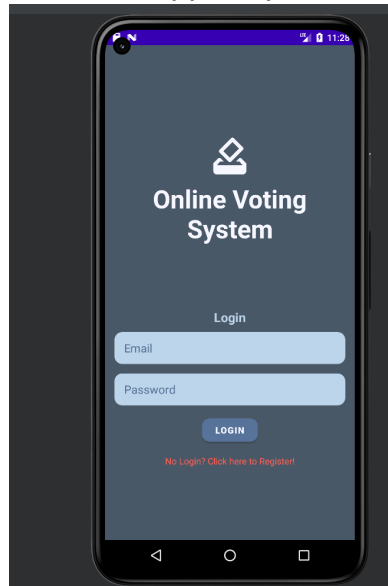
7 Appendix

Test cases

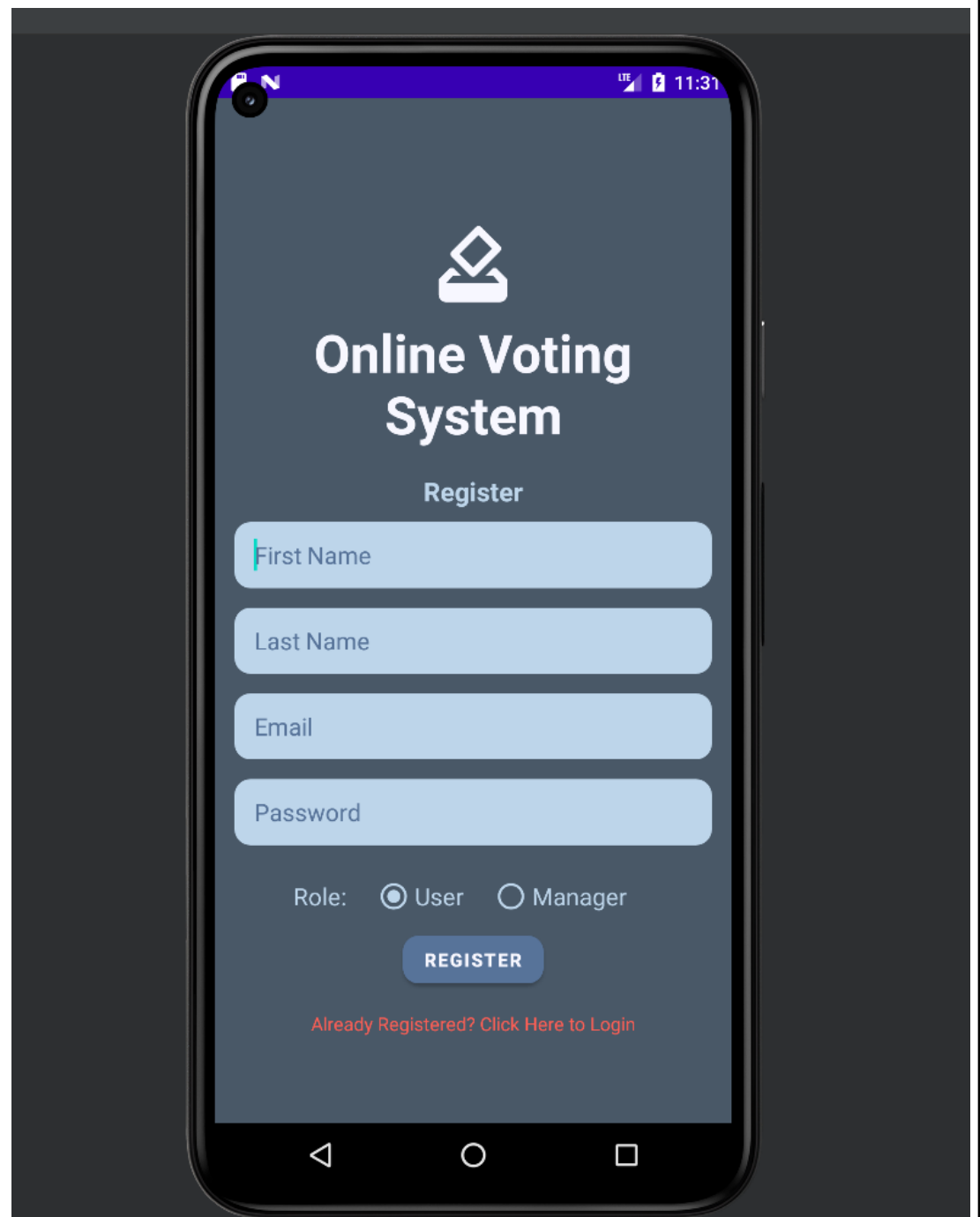
Test case ID	Title
TC_01	Register page test
TC_02	Login page test
TC_03	List view (clickable) test
TC_04	Add topic test

Test case ID	TC_01
Title	Register page test
Description	For new users of the app. An account must be created for them to vote or view topics.
Preconditions	Must not have an account registered on that email.
Postconditions	Account created successfully will be displayed. Hence the user can now log in to vote or view topics to be voted on.
Running steps	<p>Steps:</p>  <pre> Record Your Test - Delay 500 milliseconds - Tap MaterialTextView with text No Login? Click here to Register! - Delay 700 milliseconds - Type into TextInputEditText with ID first_name Tom - Type into TextInputEditText with ID last_name Cat - Type into TextInputEditText with ID email tc@mytru.ca - Type into TextInputEditText with ID password hello123456 - Tap MaterialRadioButton with text Manager - Tap MaterialButton with text Register - Delay 7000 milliseconds </pre>

When the app is open.



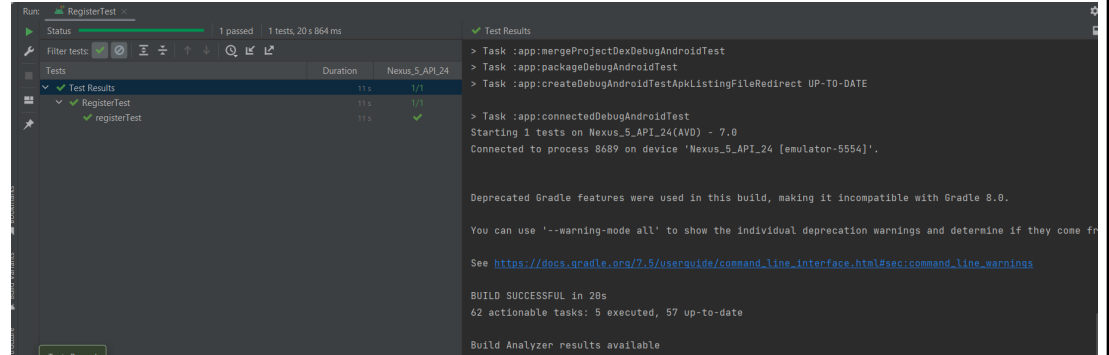
The user will click on the register here text which will take them to a sign-up page.



Info about the user can now be entered and registered afterward.

```
ViewInteraction textInputEditText = onView(  
    allOf(withId(R.id.first_name),  
        childAtPosition(  
            childAtPosition(  
  
withClassName(is("com.google.android.material.textfield.TextInputLayout"),  
ut")),  
0),
```


	<pre> 0), isDisplayed()); textInputEditText.perform(replaceText("Tom"), closeSoftKeyboard()); ViewInteraction textInputEditText2 = onView(allOf(withId(R.id.last_name), childAtPosition(childAtPosition(withClassName(is("com.google.android.material.textfield.TextInputLayout")), 0), 0), isDisplayed())); textInputEditText2.perform(replaceText("Cat"), closeSoftKeyboard()); ViewInteraction textInputEditText3 = onView(allOf(withId(R.id.email), childAtPosition(childAtPosition(withClassName(is("com.google.android.material.textfield.TextInputLayout")), 0), 0), isDisplayed())); textInputEditText3.perform(replaceText("tc@mytru.ca"), closeSoftKeyboard()); ViewInteraction textInputEditText4 = onView(allOf(withId(R.id.password), childAtPosition(childAtPosition(withClassName(is("com.google.android.material.textfield.TextInputLayout")), 0), 0), isDisplayed())); textInputEditText4.perform(replaceText("hello123456"), closeSoftKeyboard()); </pre>
Actual output	sddsad

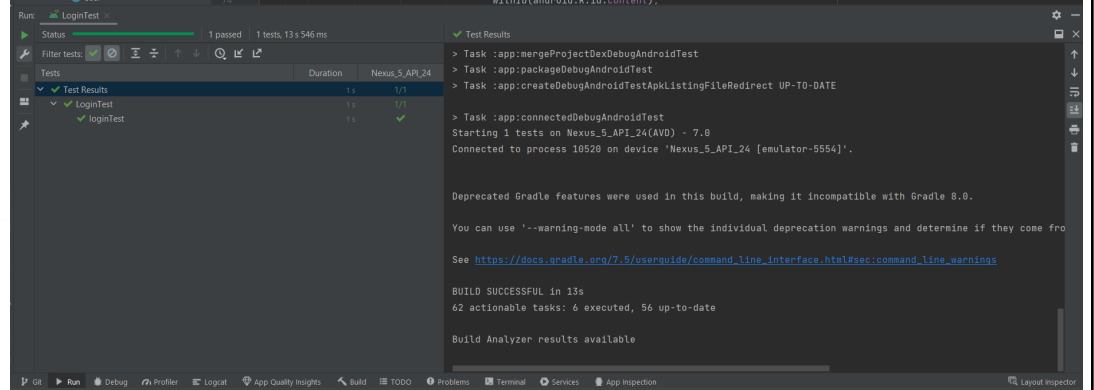


The test was successful for creating a new account with an email which has not been already registered.

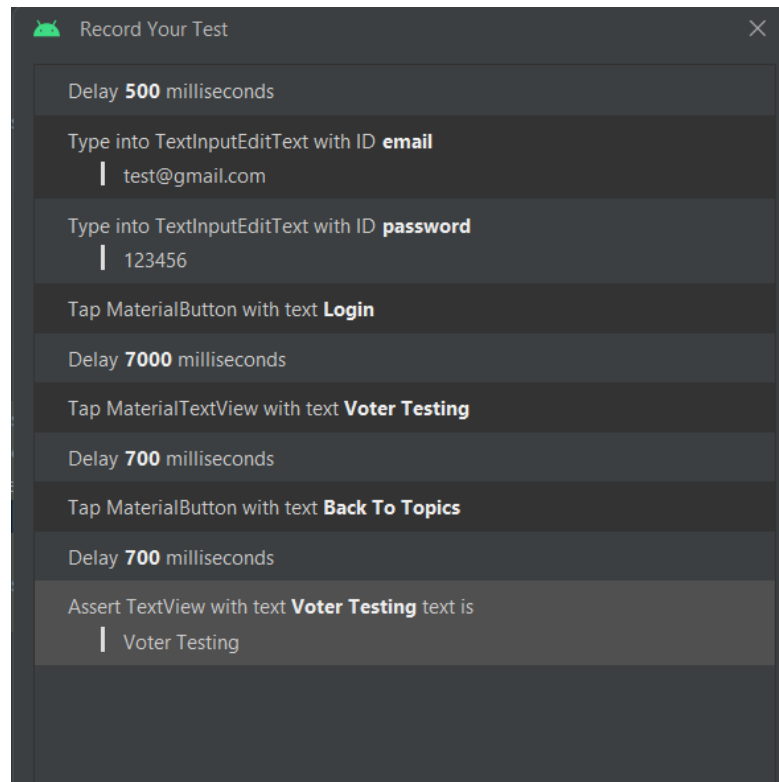
Test case ID	TC_02
Title	Login page test
Description	For a returning user who logs out. sign-in will be required to vote and view topics.
Preconditions	The user must have a registered account with a valid name and password.
Postconditions	A message box will display “Log in successfully” and will be taken to the topic page.
Running steps	Steps:



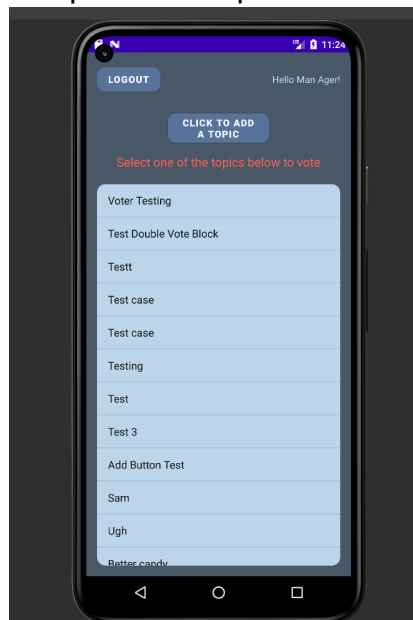
27

	<pre> 0), 0), isDisplayed()); textInputEditText.perform(replaceText("test@gmail.com"), closeSoftKeyboard()); ViewInteraction textInputEditText2 = onView(allOf(withId(R.id.password), childAtPosition(childAtPosition(withClassName(is("com.google.android.material.textfield.TextInputLayout")), 0), 0), isDisplayed())); textInputEditText2.perform(replaceText("123456"), closeSoftKeyboard()); </pre>
Actual output	<p>The log-in test case is successfully.</p>  <p>The screenshot shows the Android Studio interface with the 'Run' tab selected. The 'Test Results' panel on the right shows a green checkmark for 'LoginTest' and 'loginTest'. The 'Logcat' panel at the bottom shows the following log messages:</p> <pre> > Task :app:mergeProjectDexDebugAndroidTest > Task :app:packageDebugAndroidTest > Task :app:createDebugAndroidTestApkListingFileRedirect UP-TO-DATE > Task :app:connectedDebugAndroidTest Starting 1 tests on Nexus_5_API_24 (AVD) - 7.0 Connected to process 10520 on device 'Nexus_5_API_24 [emulator-5554]'. Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0. You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from See https://docs.gradle.org/7.5/userguide/command_line_interface.html#sec:command_line_warnings BUILD SUCCESSFUL in 13s 62 actionable tasks: 6 executed, 56 up-to-date Build Analyzer results available </pre>

Test case ID	TC_03
Title	List view (clickable) test
Description	When a user arrives on the topic page a list with all topics is displayed when click it sends the user to the voting page.
Preconditions	The user must log in with a valid account
Postconditions	The user will be sent to the voting page to vote on the clicked topic.
Running steps	Steps:



when a topic is clicked it will relay you to the appropriate voting page for that particular topic.



```
DataInteraction materialTextView = onData(anything())
    .inAdapterView(allOf(withId(R.id.ListViewPost),
        childAtPosition(
            withClassName(is("android.widget.LinearLayout")),
```

	<pre> 0))) .atPosition(0); materialTextView.perform(click()); </pre>
Actual output	<p>This test case was successful</p>  <p>The screenshot shows the Android Studio interface. On the left, the 'Test Results' pane shows a single test case 'listview_clickable' with a status of 'passed' and a duration of 12s. The main window displays the 'Test Results' tab, showing the test case 'listview_clickable' with a status of 'passed' and a duration of 12s. The 'Test Results' pane also shows the test case 'listview_clickable' with a status of 'passed' and a duration of 12s. The 'Test Results' pane also shows the test case 'listview_clickable' with a status of 'passed' and a duration of 12s.</p>

Test case ID	TC_04
Title	Add topic test
Description	This is a test case to ensure the add a topic button for the manager will correctly work.
Preconditions	Login is required to ensure only the manager will be able to add topics to vote on.
Postconditions	A new topic is created asking for a topic title, description and vote option. Which will then be voted on.
Running steps	Steps:

```

Delay 500 milliseconds
Type into TextInputEditText with ID email
| test@gmail.com
Type into TextInputEditText with ID password
| 123456
Tap MaterialButton with text Login
Delay 7000 milliseconds
Tap MaterialButton with text Click to add data
Delay 700 milliseconds
Type into TextInputEditText with ID editTextPostTitle
| Test case
Type into TextInputEditText with ID editTextPostDesc
| This is a test case
Type into TextInputEditText with ID button_1_description
| Working
Type into TextInputEditText with ID button_2_description
| cool
Tap MaterialButton with text Add
Tap MaterialButton with text Back To Topics
Delay 700 milliseconds
Assert Button with text CLICK TO ADD DATA exists

```

Login

We first log in using an email and password (snippet below)

```

ViewInteraction textInputEditText = onView(
    allOf(withId(R.id.email),
        childAtPosition(
            childAtPosition(

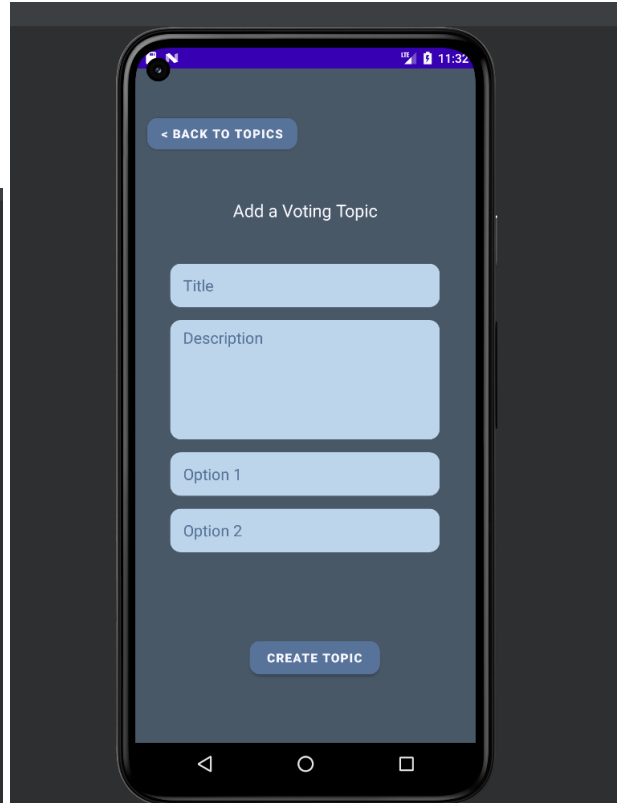
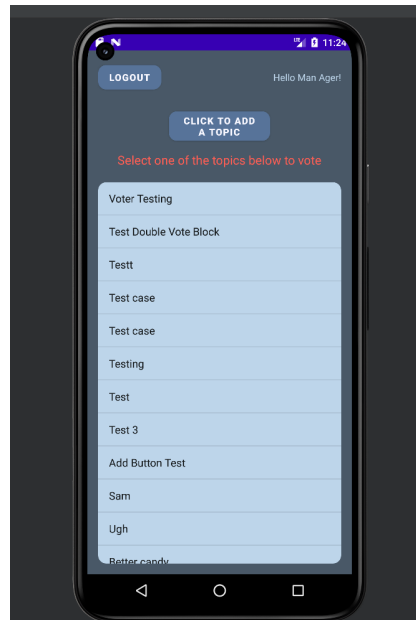
withClassName(is("com.google.android.material.textfield.TextInputLayout")) ,
0),
0),
isDisplayed())) ;
textInputEditText.perform(replaceText("test@gmail.com"),
closeSoftKeyboard());

ViewInteraction textInputEditText2 = onView(
    allOf(withId(R.id.password),
        childAtPosition(
            childAtPosition(

withClassName(is("com.google.android.material.textfield.TextInputLayout")) ,
0),
0),
isDisplayed())) ;
textInputEditText2.perform(replaceText("123456"),
closeSoftKeyboard());

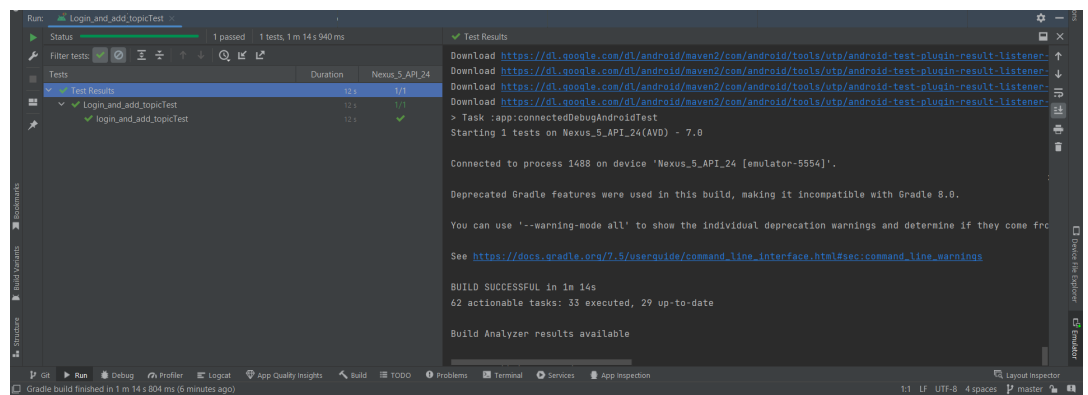
```

After the login was successful we then click on add topic



The Manager can then choose a topic title, description, options which will then be voted on.

Actual output



As seen in the picture above the test is successful and a topic has been created.