

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Screen 3](#)

[Screen 4](#)

[Screen 5](#)

[Screen 6](#)

[Screen 7](#)

[Screen 8](#)

[Screen 9](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement data persistence](#)

[Task 3: Implement UI for Main Screen](#)

[Task 4: Integration with Bing Search API](#)

[Task 5: Integration with YouTube Data API](#)

[Task 6: Implement the widget](#)

[Task 7: Implement App variants](#)

[Task 8: Accessibility](#)

GitHub Username: shamim-ahmed

Content Finder

Description

An easy-to-use app that allows the user to receive news updates, photos or latest videos based on his interests.

Intended User

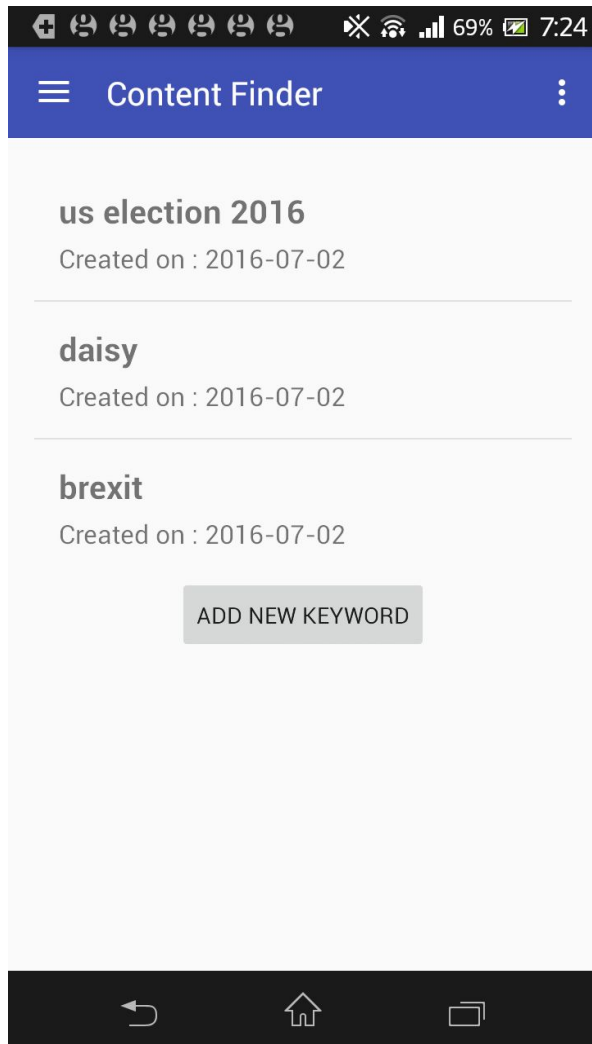
If you want to stay updated on specific topics that interest you, then you should definitely consider using this app.

Features

- The app will allow the user to specify one or more keywords, which will be persisted in local database.
- The home screen will show the keywords previously saved by the user.
- The user will be able to perform search based on the saved keywords.
- Search results from the following sources will be displayed:
 - News updates from Bing API
 - Photos from Bing API
 - Youtube videos
- The user will be able to mark a particular content as favorite. Basic information about the content (e.g. web URL, title, summary) will then be saved into local database for future reference.
- The app will have two variants: free and paid. The free version will display ads, whereas the paid version will be ad-free.
- Google Analytics will be used to measure user activities on different screens.

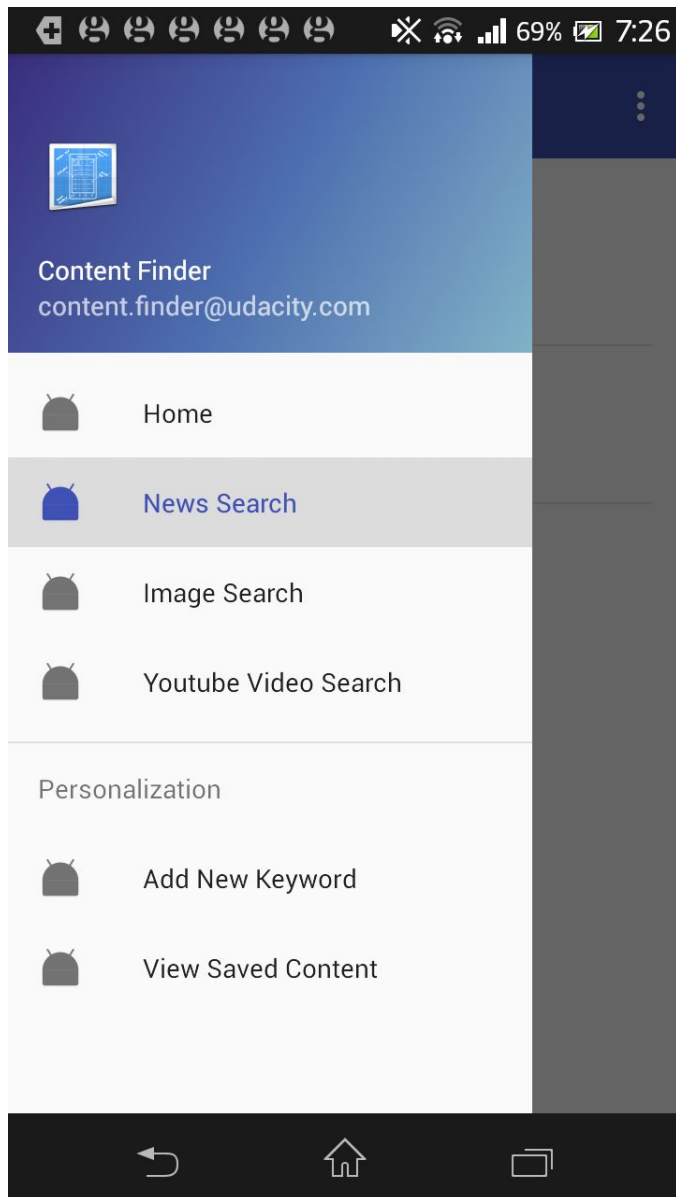
User Interface Mocks

Screen 1



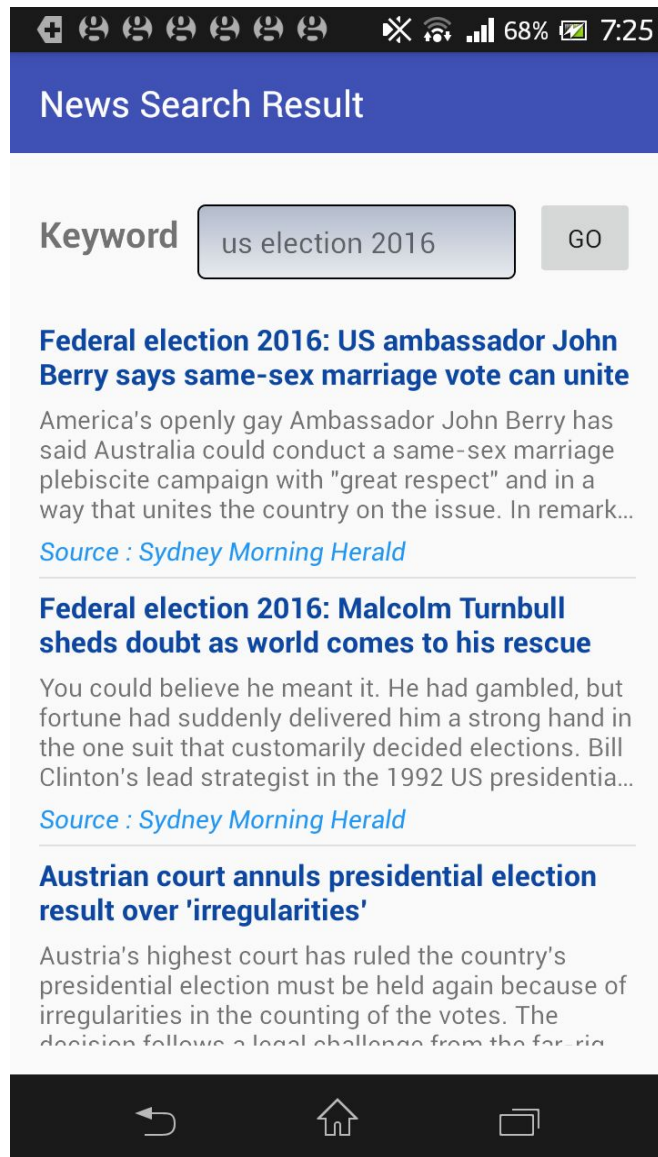
The first screen displays a list of the keywords previously added by the user, along with the button to add a new keyword.

Screen 2



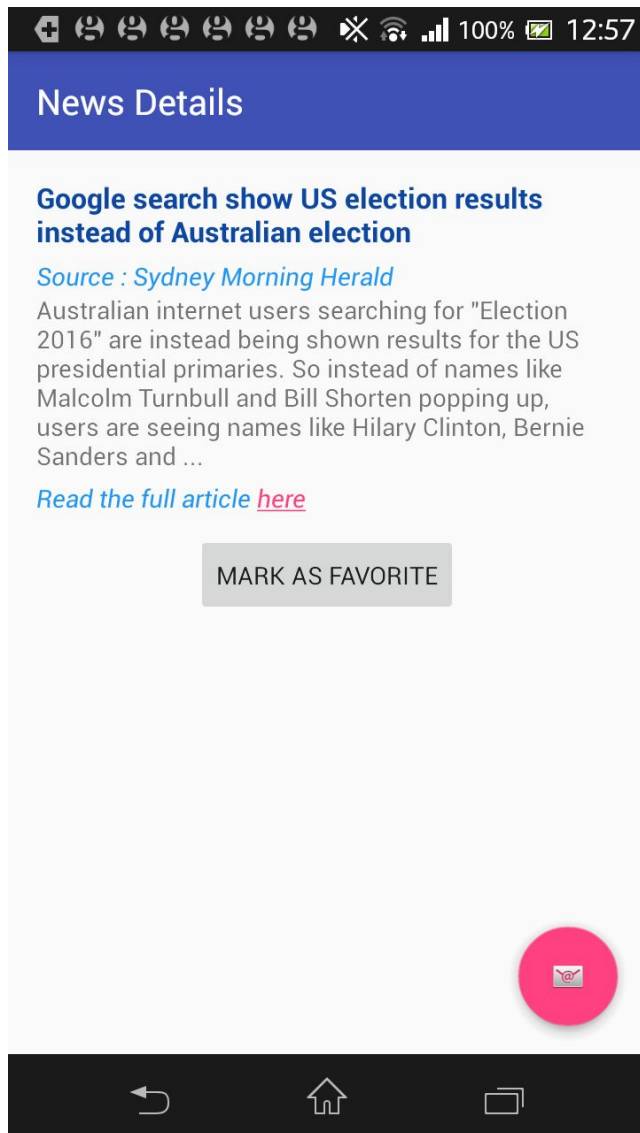
The navigation drawer allows users to find various screens easily.

Screen 3



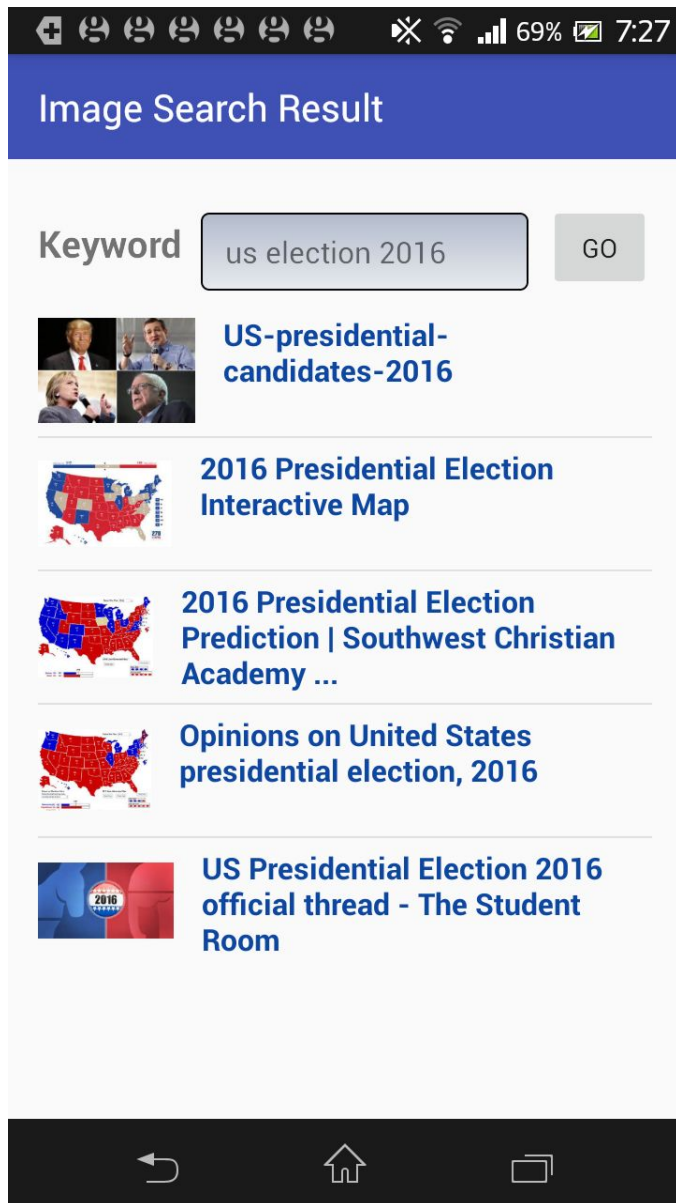
The news search screen shows latest news stories related to a particular keyword. There is also a drop down list that contains all the keywords previously added by the user. Clicking on the 'Go' button initiates a new search.

Screen 4



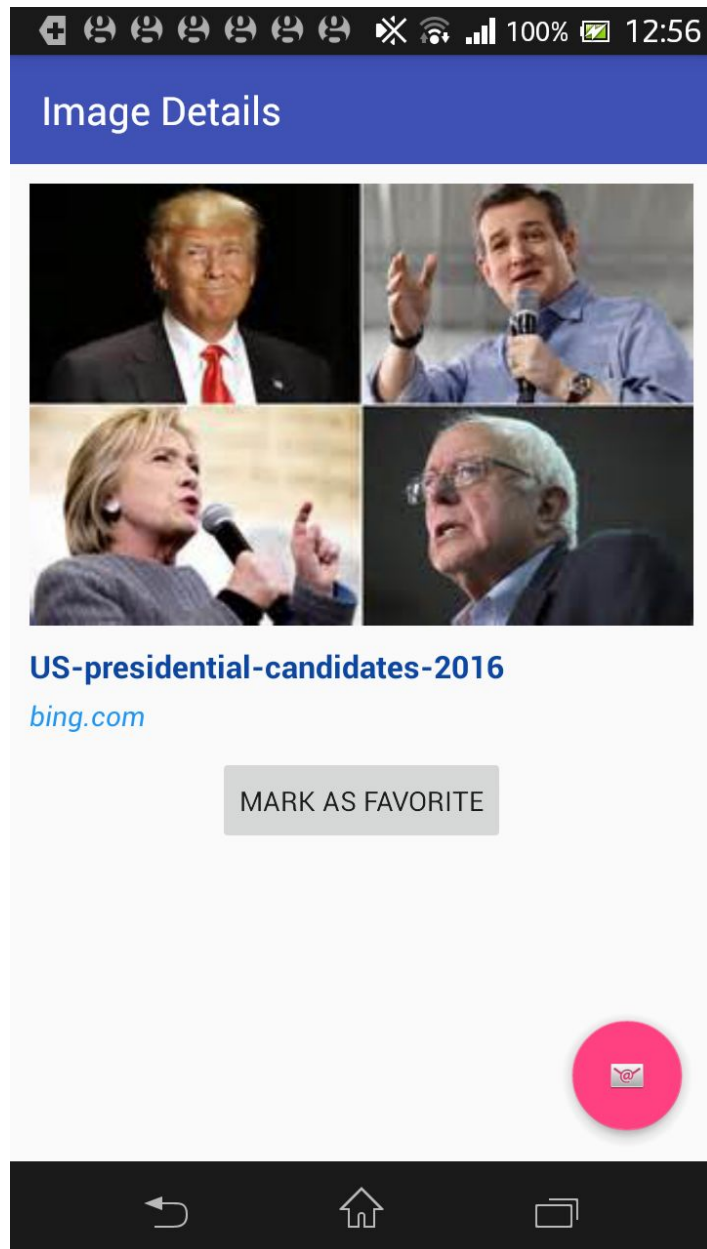
The news details screen shows the title and summary of a particular news story. A link to the original news story is also provided. The 'Mark as Favorite' button allows the user to save the news story in local database for future reference.

Screen 5



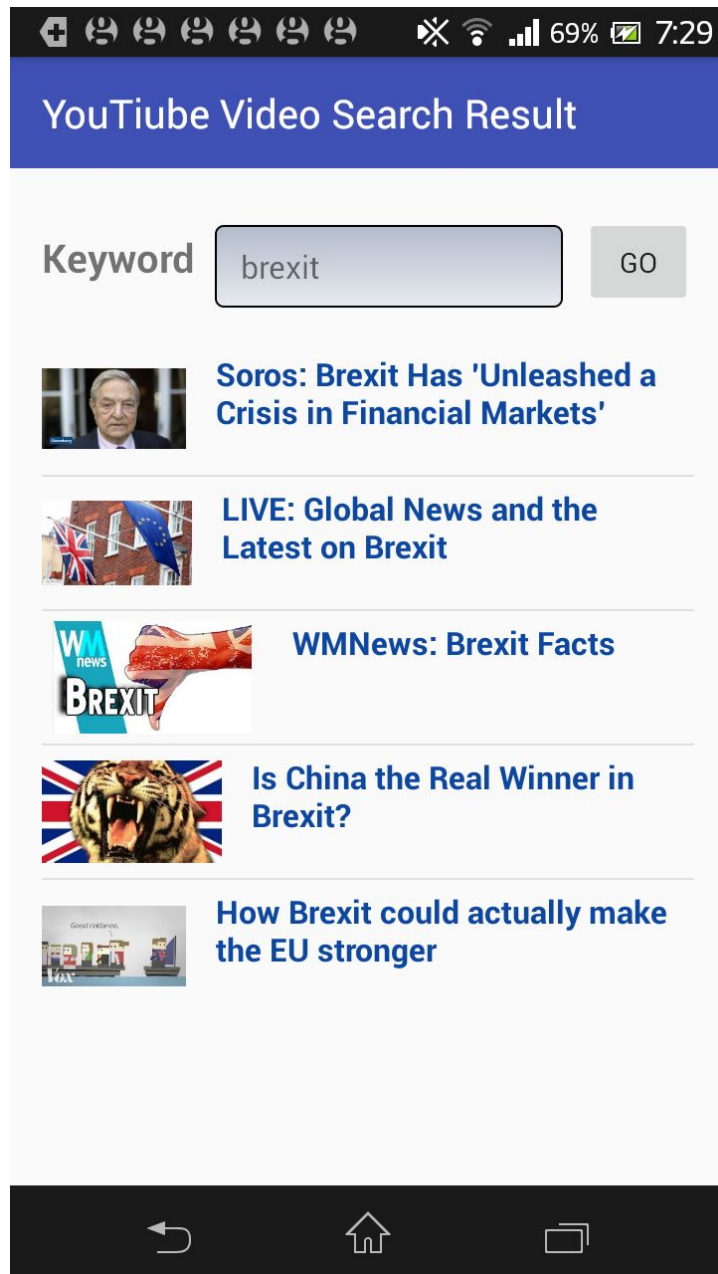
The image search screen shows latest images related to a particular keyword. There is also a drop down list that contains all the keywords previously added by the user. Clicking on the 'Go' button initiates a new search.

Screen 6



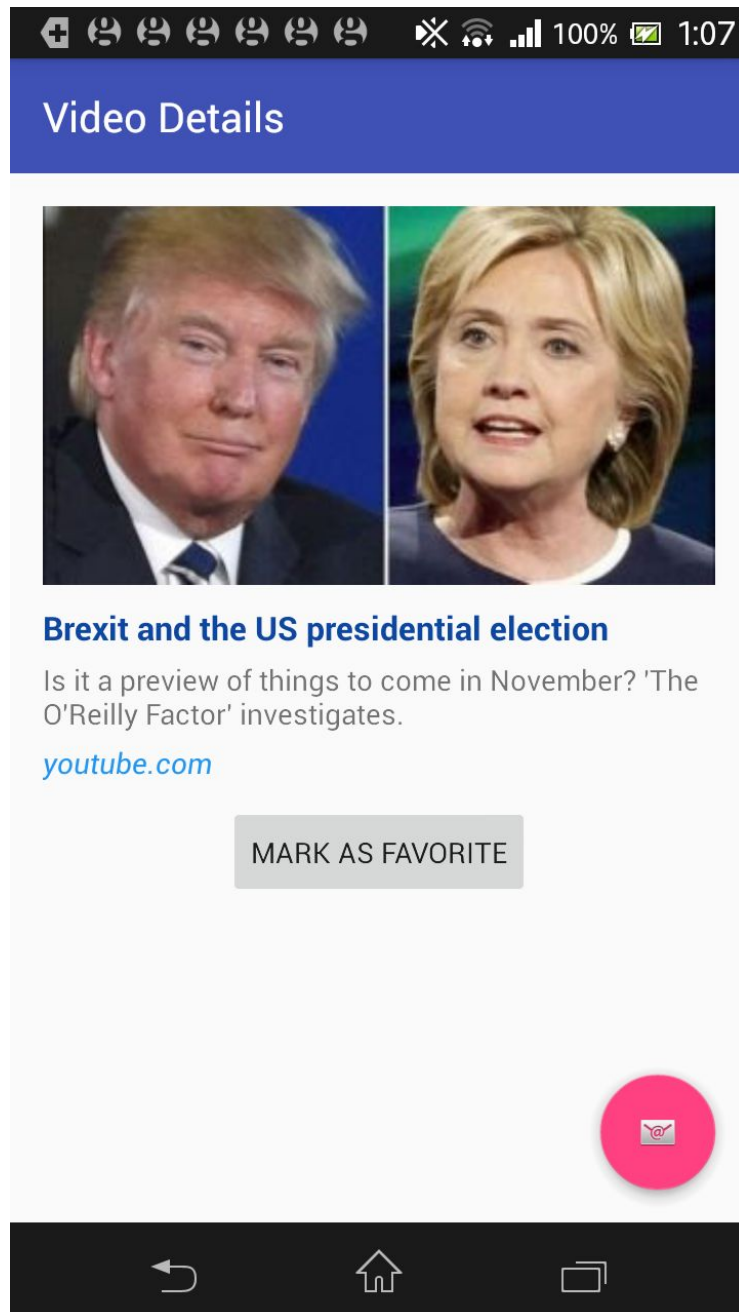
The image details screen displays the details of a particular image, including its title and source. The 'Mark as Favorite' button allows the user to save basic information about the image (e.g. URL, title, description) into local database for future reference. A 'Share' button is also provided so that the original image link can be shared on social media.

Screen 7



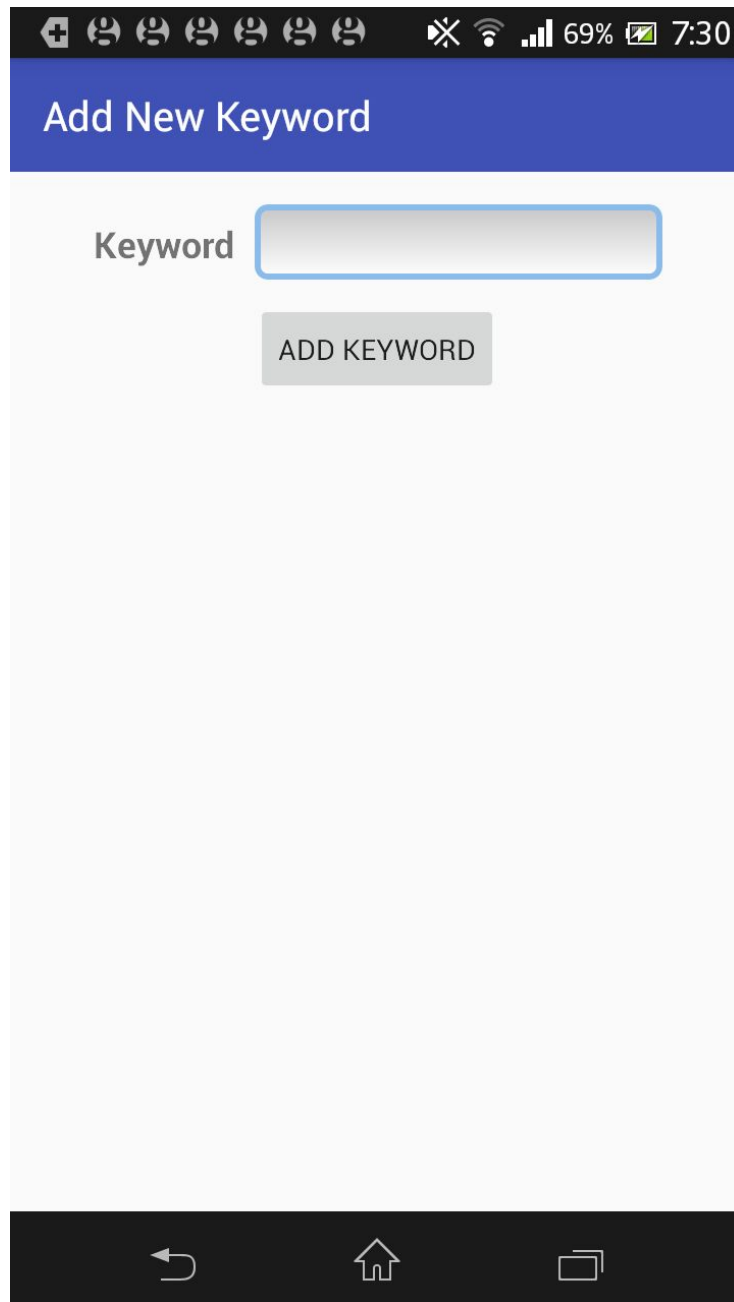
The video search screen shows latest YouTube videos related to a particular keyword. There is also a drop down list that contains all the keywords previously added by the user. Clicking on the 'Go' button initiates a new search.

Screen 8



The video details screen shows the title, summary and thumbnail for the selected YouTube video. Clicking on the video thumbnail will launch a suitable app (e.g. YouTube app) that can play the video. The 'Mark as Favorite' button allows the user to save basic information about the video (e.g. video id, title, description) in local database for future reference. There is also a 'Share' button to facilitate sharing on social media.

Screen 9



The screenshot shows a mobile application interface for adding a new keyword. At the top, there is a status bar with various icons and the time 7:30. Below the status bar is a blue header with the text "Add New Keyword". The main content area is light gray and contains a label "Keyword" followed by a text input field. Below the input field is a gray button labeled "ADD KEYWORD". At the bottom of the screen is a dark navigation bar with three icons: a back arrow, a home icon, and a recent apps icon.

The 'Add New Keyword' screen allows the user to specify a new keyword and save it. Afterwards, the user will be able to perform search using that keyword.

Key Considerations

How will your app handle data persistence?

The app will store two kinds of data:

- The user-specified keywords.
- Basic information (e.g. web URL, title) for content marked as favorite.

In both cases, the data is stored in local SQLite database. A custom content provider will be implemented in order to support data persistence. A CursorLoader will also be used to run queries in the background.

Describe any corner cases in the UX.

- The app will adapt gracefully to devices with various screen resolutions.
- The app will use a navigation drawer to highlight the key screens. For devices with smaller screens, the user will be able to expand and collapse the navigation drawer. For devices with larger screens (e.g. tablets), the navigation drawer will always be visible.

Describe any libraries you'll be using and share your reasoning for including them.

Picasso will be used for rendering photos obtained via Bing API. Picasso hides all details of network communication and caching. The resulting code is often quite simple and maintainable.

Next Steps: Required Tasks

Task 1: Project Setup

- Create a project with an empty fragment.
- Add Picasso dependency in build.gradle file.
- Ensure that the build is working properly.

Task 2: Implement data persistence

- Define the database tables with all required columns and constraints.
- Implement the content provider.
- Implement CursorLoader for running queries in the background.

Task 3: Implement UI for Main Screen

- Build UI for main activity which displays a list of saved keywords.
- Build UI for the activity that allows the user to save a new keyword.

Task 4: Integration with Bing Search API

- Investigate how search operation can be implemented using Bing API and what the response format will be.
- Investigate how Bing API can be used to search for different kinds of contents (e.g. news, photos).
- Implement the AsyncTask that will send the search request to the server and make the result available to the relevant view.
- Implement the UI for search result and news/image detail screen.

Task 5: Integration with YouTube Data API

- Investigate how YouTube Data API can be used to search for videos related to a particular keyword and what the response format will be.
- Implement the AsyncTask that will send search request to the server and make the result available to the relevant view.
- Implement the UI for search result and video detail screen.

Task 6: Implement the widget

- Implement a widget that will show a list of contents that have been saved by the user recently.
- Clicking on the widget will launch the app and show the details of the saved content on relevant screen.

Task 7: Integrate with Google services

- Modify the build process to support two variants of the app: free and paid.
- Integrate the free variant with Google AdMob to display ads.
- Add Google Analytics support to the app so that user activity on various screens can be tracked.

Task 8: Accessibility

- Ensure that content description is present for UI elements and navigation using D-pad is supported.