Bilkent University

Senior Design Project

Project short-name: cerebro

Analysis Report

Kaan Yorgancıoğlu 21302439
Dilara Ercan 21201256
Özgür Taşoluk 21301674
Nihat Atay 21102292
Furkan Salih Taşkale 21300878

Supervisor: Selim Aksoy
Jury Members: Ercüment Çicek, Uğur Güdükbay

Website: kyorgancioglu.github.io/CerebroApp/

Nov 7, 2016

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Index:

1. Introduction

Everyday thousands of new media are published in many different forms. Although there are some applications that help identify media by several ways, most of the mainstream apps fail to provide more than minimal information. Also most of them focus on a single type of media.

It is common that consumers come across a media product such as albums, movies, books in a shop and want to learn more about it. But there is no single easy way to access all the relative information in one place. Our solution to this problem is “Cerebro”. Cerebro is a mobile app for users to take advantage of. It is a platform that searches the media and puts together desired information by gathering them from trusted third party sources.

Cerebro uses the camera of the smartphone to capture the cover art of media such as book covers, album covers, movie posters, etc. It then searches the database for any matches and gathers all the information from the internet such as the artist, author or the director of the media, casts, release date, customer reviews, brief summary and etc. and displays them in a tidy smart way. This would allow users to quickly find information from their favorite sources, all at once.

2. Proposed System

2.1 Overview

Cerebro will be composed of two parts. A mobile application which is the frontend of Cerebro and a server side backend which will be responsible for matching and providing necessary information. The frontend will communicate the backend through an API.

The information displayed will be automatically gathered from third party sources by using scripts and APIs. The database will only include the most basic information required for matching. Therefore it will be easy to build up a large database.

The details screen is fully customizable through a user friendly gui which is accessed through the settings page. The details screen for each kind of media -books, music, movies, etc.- can be customized separately.

2.2 Functional Requirements

- Searching will be initiated by one button tap which will open the default camera
- If more than one matches occur users will be provided a list of matches
- User can tap on one of the matches list elements to go to the details
- On details page users will be provided with basic info accompanied by links to the product's web page
- The user can choose to see any other information (such as reviews) on the details screen for each type of media
- the contents of the details screen can be set through settings screen with the help of an interactive GUI a default setting will be provided
- media searching is done on the server side
- the search only returns a unique name for the media
- any other information is requested from the server by using the name of the media
- only the basic information is stored on the database any other info is obtained through running scripts on source sites
- The app will get the match by connecting to the server.
- The app will upload a preprocessed version of the image to the server. This will allow the app to use less data.

- The server will send back a list of matches.
- The app will construct the info page by using the data returned by the server to collect information from third party sources with the help of scripts.

2.3 Non-Function Requirements

- The application should be quick to launch
- The system's response time should be at most 3 seconds.
- The system should be available 7/24.
- The system should be able to respond 50 request per second.
- Cerebro should be user – friendly and easy to use
- Cerebro will not require an account to login
- The application will initially be released for Android
- The system will require Internet access
- The database should be large enough to match common searches
- The backend should be scalable to serve a large number of users if needed
- Both frontend and backend should be extensible so that new functionalities can be added in the future

2.4 Pseudo Requirements

- The system will be made of two parts. One is the app itself and the other is the server side.
- The app will be made to run on Android.
- The server will be built with Django framework.
- The server will have a database for image recognition.

2.5 System Models

2.5.1 Use case Models and Scenarios

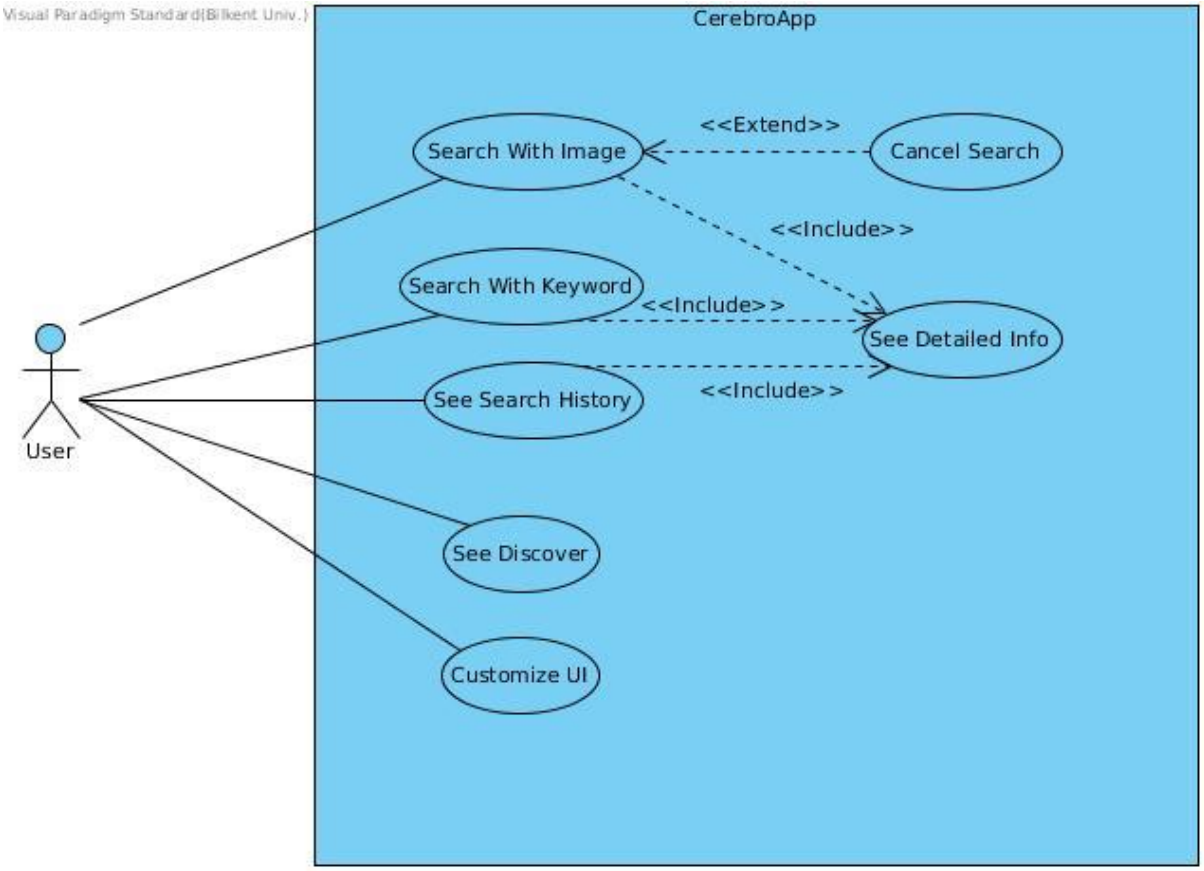

Figure 1: Use case diagram of the app.

Use Case Definitions

Use Case Name: Search With Image
Participating Actors: User
Entry Condition: User has started the application.
Exit Condition: User got the detailed information about the image he/she searched or returned to main screen.
Flow of Events:

1. User clicks Search With Image button.
2. User scans the image he/she desires.
3. Application starts to search the image.

3.1. User triggers Cancel Search use case if he/she wants to stop the search.
3.2. Otherwise application continues to search.

4. Application loads the results.

4.1. If multiple results found, applications shows the results as a list.

4.1.1. User triggers See Detailed Info use case to get detailed information about a result.

4.2. If only one result is found, application loads detailed information about the image.
4.3. Otherwise, application shows no results and returns to the main screen.

Use Case Name: Search With Keyword

Participating Actors: User

Entry Condition: User has started the application.

Exit Condition: User got the detailed information about his/her search or returned to main screen.

Flow of Events:

1. User clicks Search With Keyword button.
2. User enters the search query.
3. Application starts to search with given query.
4. Application loads the results.

4.1. If multiple results found, applications shows the results as a list.

4.1.1. User triggers See Detailed Info use case to get detailed information about a result.

4.2. If only one result found, application loads detailed information about the result.

4.3. If no result is found, application shows no results and returns to main screen.

User Case Name: See Search History

Participating Actors: User

Entry Condition: User has started the application.

Exit Condition: User is given his/her search history as a list.

Flow of Events:

1. User clicks the Search History button.
2. Application loads the previously searched images and keywords.

Use Case Name: See Discover

Participating Actors: User

Entry Condition: User has started the application.

Exit Condition: User is shown Discover page.

Flow of Events:

1. User clicks Discover button.
2. Application loads Discover page.

Use Case Name: Customize UI

Participating Actors: User

Entry Condition: User has started the application.

Exit Condition: New user interface customization is saved.

Flow of Events:

1. User clicks Settings button.
2. User clicks Customize Result Screen button.
3. User chooses which information he/she desires to see in result screen.
4. User clicks Save button.

Use Case Name: Cancel Search

Participating Actors: User

Entry Condition: User scanned an image and application started to search it.

Exit Condition: User is returned to main screen.

Flow of Events:

1. User clicks Cancel button.
2. Application loads main screen.

Use Case Name: See Detailed Info

Participating Actors: User

Entry Condition: User is given with results list.

Exit Condition: Detailed information about a result is shown.

Flow of Events:

1. User clicks a specific result.
2. Application shows necessary information about the result.

2.5.2 Object and Class Model

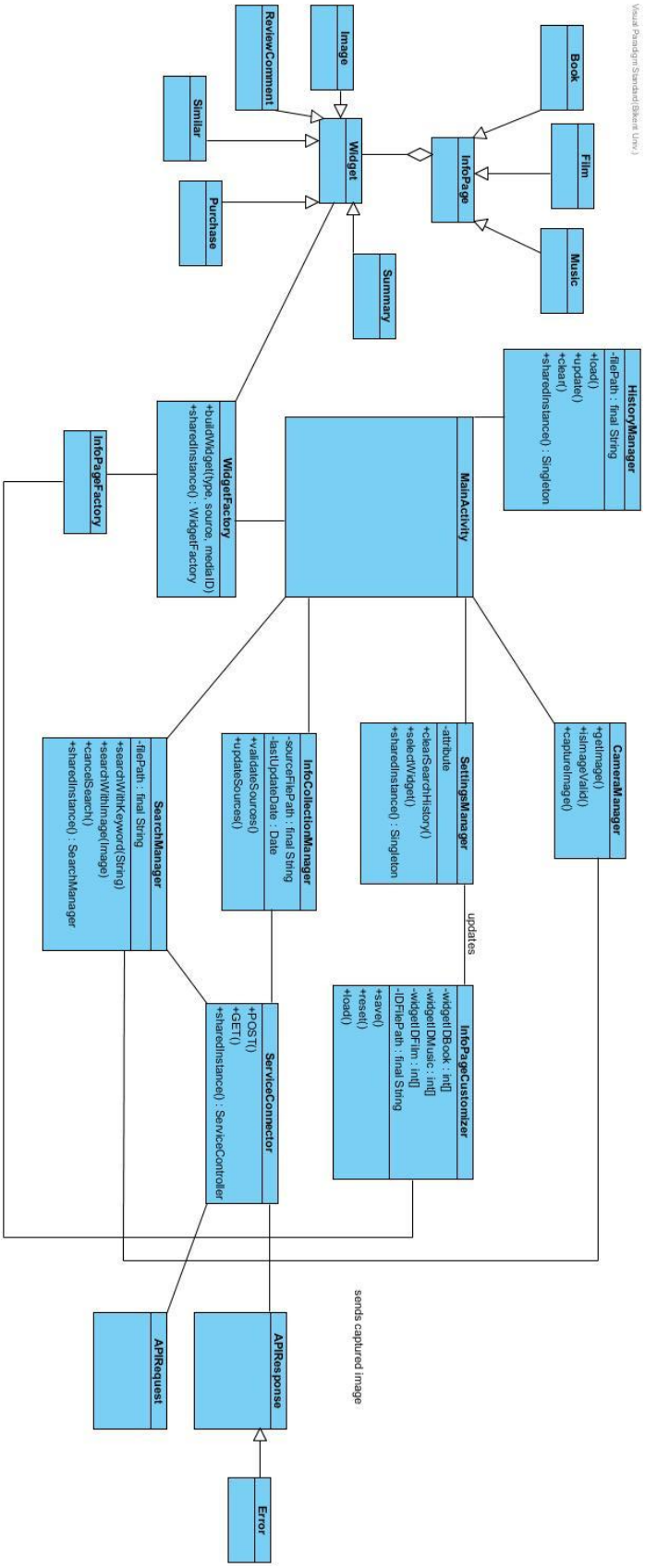

Figure 2: Class diagram of Cerebro app.

2.5.3 Dynamic Models

2.5.3.1 Sequence Diagrams

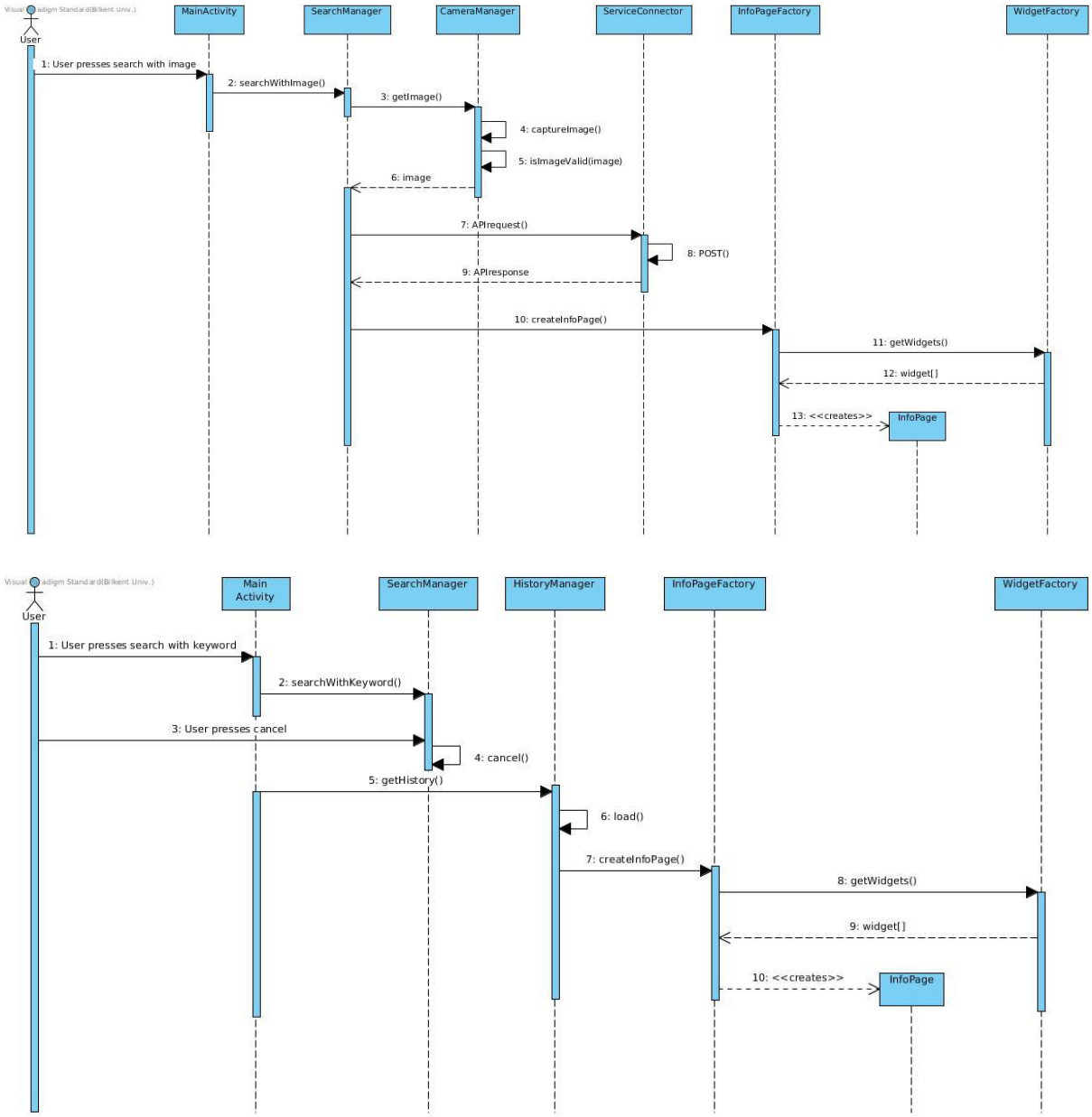

Figure 3: Sequence diagram of the app.

2.5.3.2 Activity Diagram

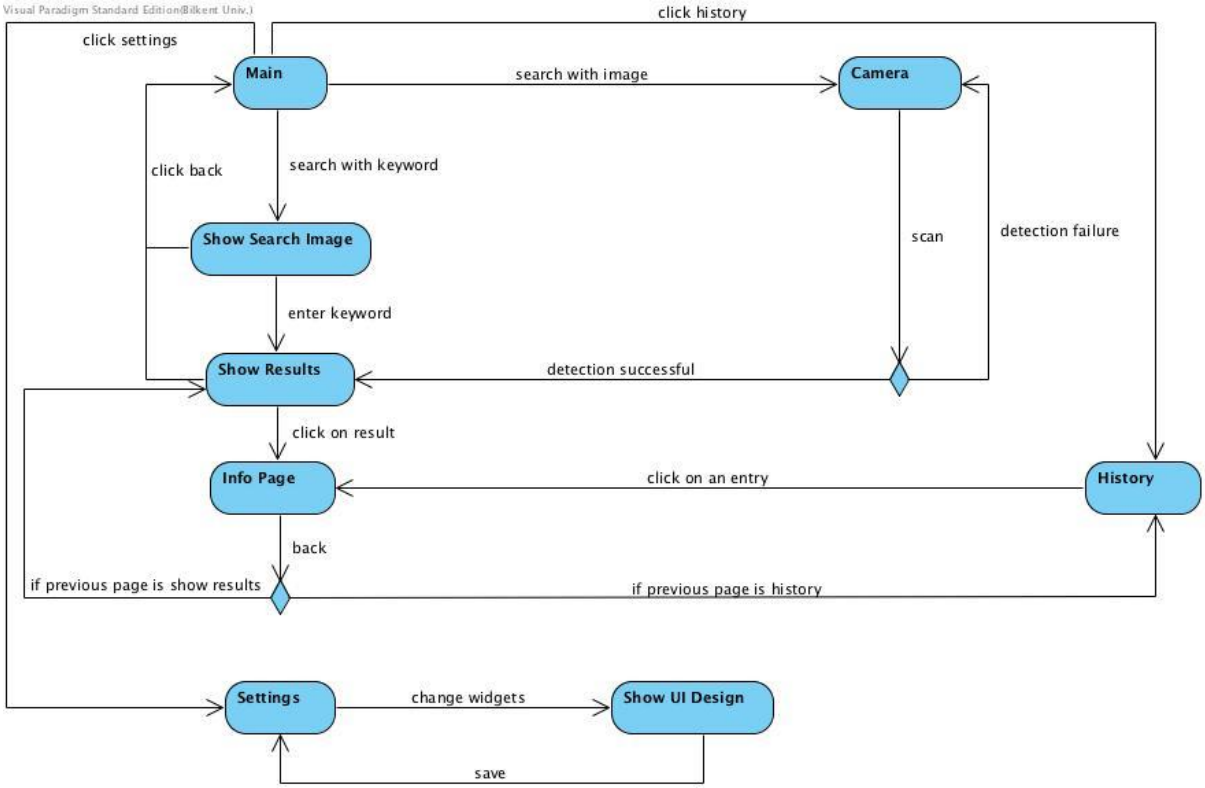

Figure 4: Activity diagram of the proposed system.

2.5.4 User Interface Mock-ups

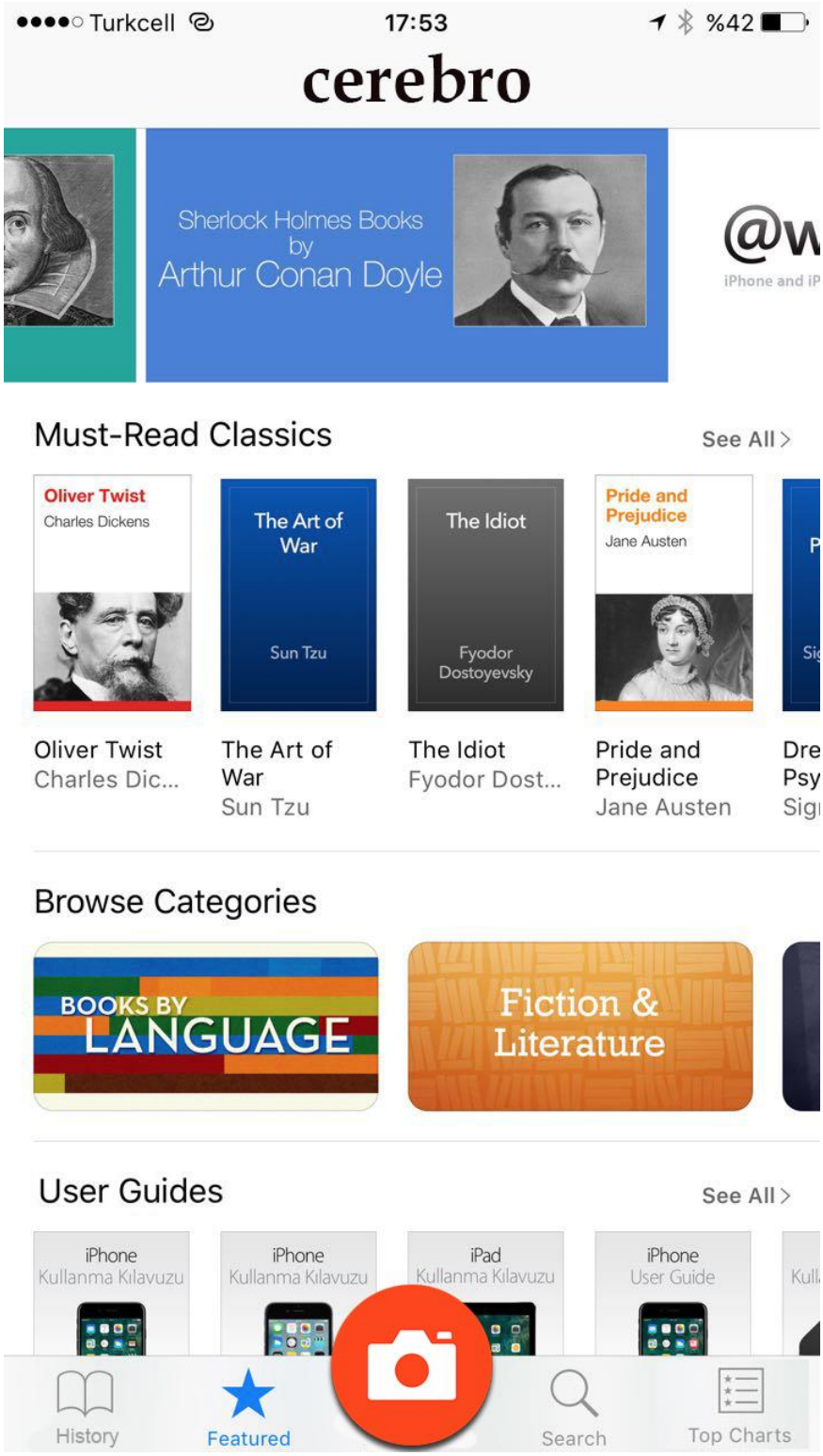

Figure 5: The featured page of the cerebro app. This screen will allow users to discover new media.

Figure 6: The search by keyword screen of Crebero.

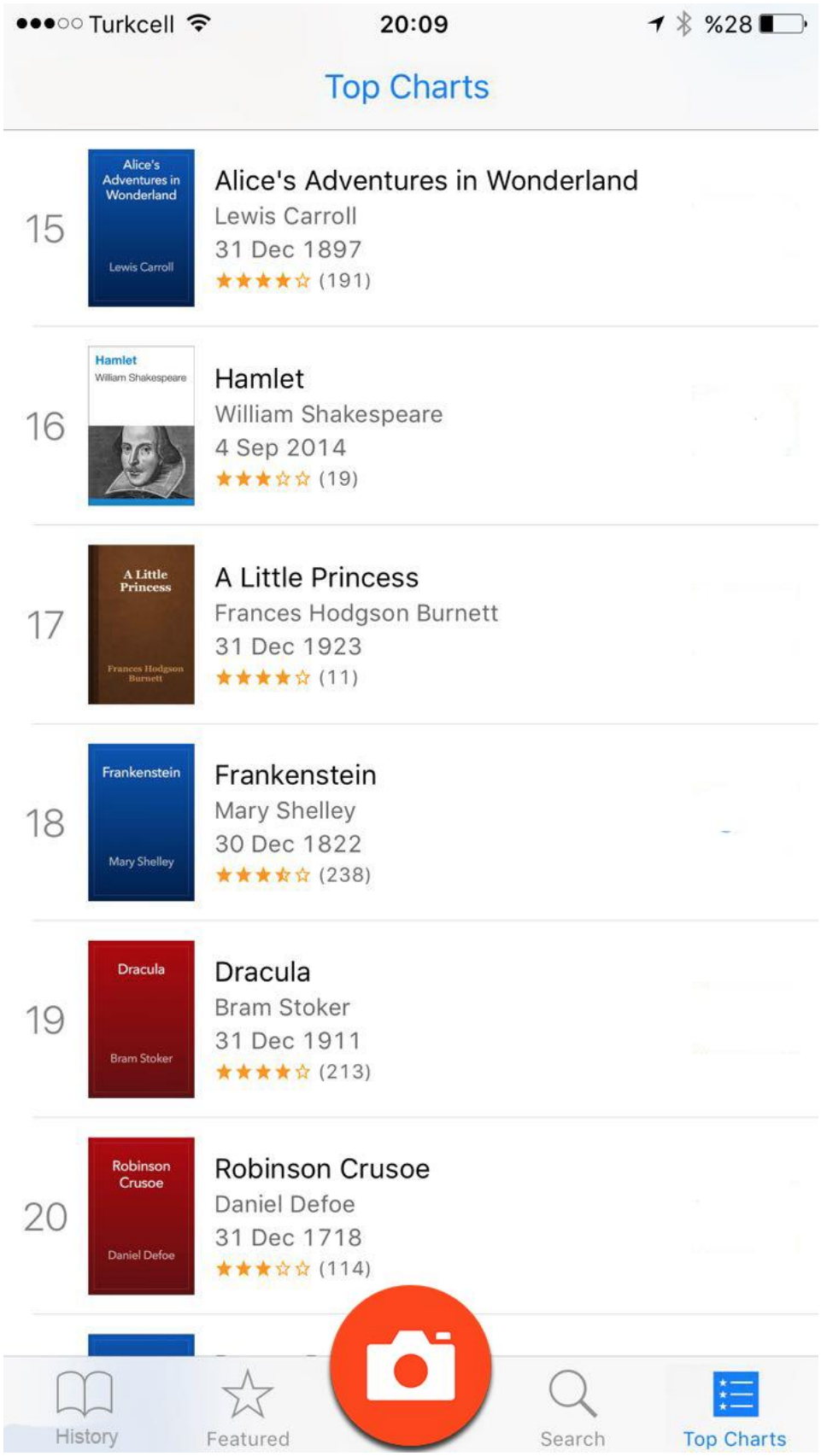

Figure 7: Top charts screen which is similar to featured screen. This screen will display the top searched media.

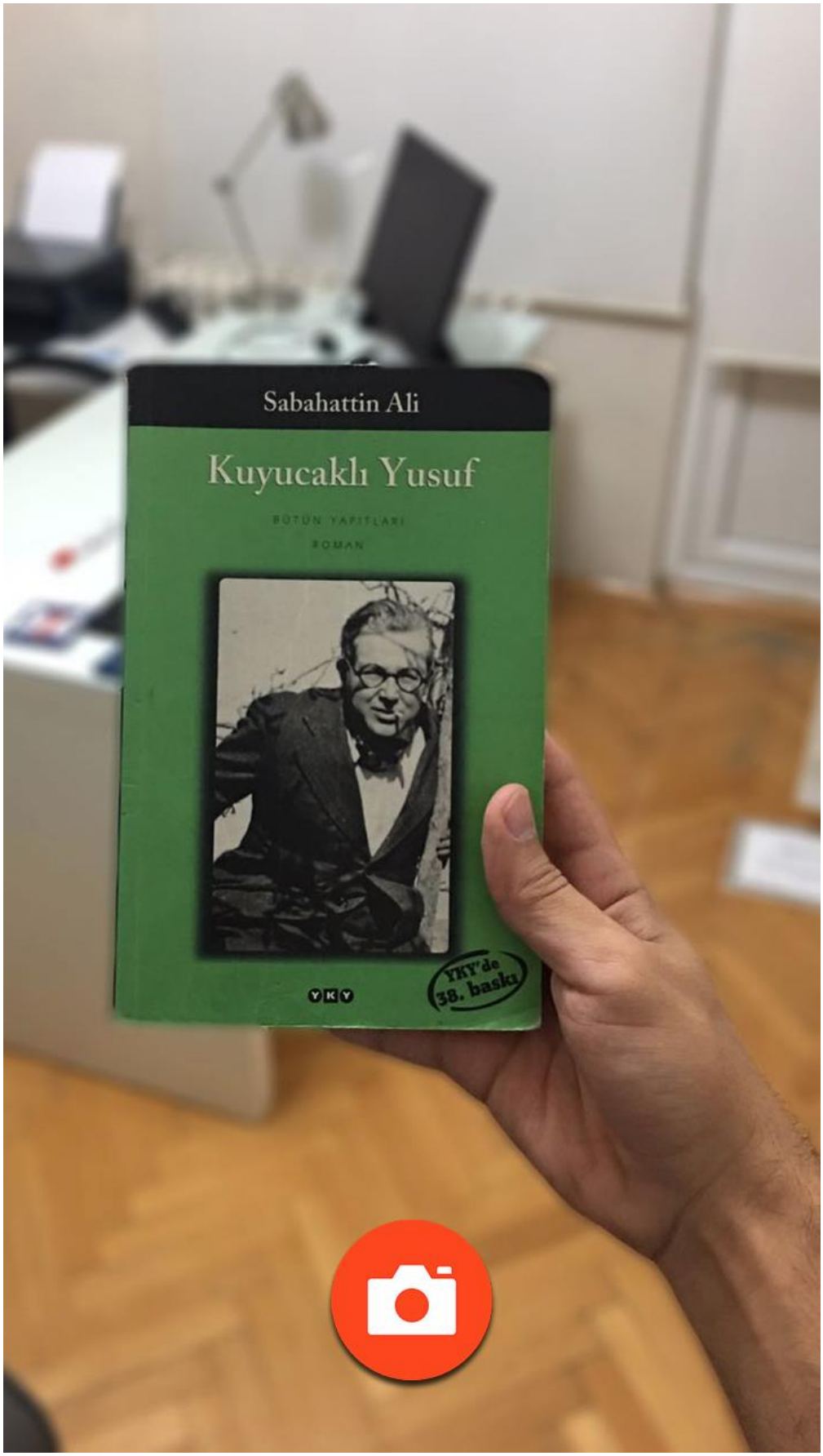

Figure 8 : The search by image screen of Cerebro app. This screen allows the user to capture the image of the media.

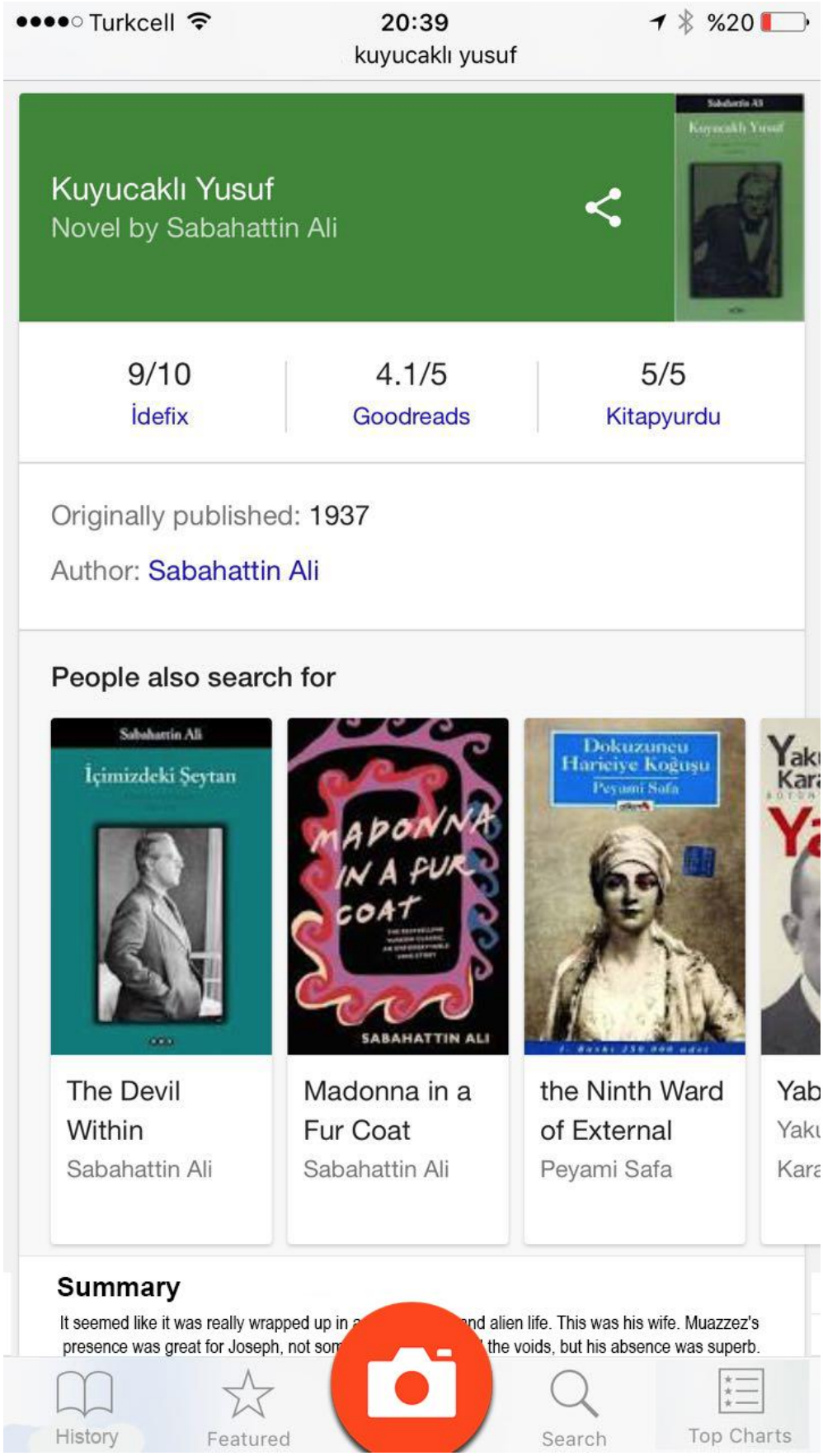

Figure 9: The information page for the searched media. It contains the selected widgets.

3. Glossary

- Widget: It is the little tools that contain desired information about the searched media. It has several types such as image, purchase links, similar items, information/summary.
- InfoCollectionManager: It is the manager class that handles the information collection from third party sources with the help of scripts.
- InfoPageCustomizer: It is the class that handles UI design changes made by user.

4. References

- The screen mockup images are inspired by popular apps including Google books, iOS appstore, and Google search engine.