Bilkent University

Senior Design Project

Cerebro

Project URL: kyorgancioglu.github.io/CerebroApp/

Low-Level Design Report

Project Members:

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

Feb 20, 2017

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.

Table of Contents

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.

1.1 Object Design Trade-offs

1.1.1 Complexity vs. Functionality

Cerebro is an application that provides different widgets which may increase the complexity of the application. Having too many options and functionalities can cause confusion for the users while using the application. Our aim to produce an application that is user friendly. It should be easy to use in order to reach high number of users and satisfying the users’ needs.

1.1.2 Performance vs. Complexity

We are planning to use different algorithms to analyze different type of medias. These algorithms will be designed for eliminating useless part of images that come from clients and finding only relevant part of medias. However, if we build up our algorithms in a very complex way, that will cause a performance loss in our system because analyzing single image takes too much time in this case. If a series of complex algorithms is applied to each images, our system will be less efficient, have worse performance and response time.

1.1.3 Compatibility vs. Extensibility

Our aim to reach big part of all Android devices especially the entry-level devices because we want to decrease the set-up fee and will be the accessible. In the market, there are Android devices, which have so many different versions. However, older versions do not provide flexibility according to 5.0 version. Our first priority reaches the large masses with supporting low-level devices due to cost constraints. Therefore, our application can be available for the Android 3.0 and the versions above it.

1.2 Interface Documentation Guideline

You can find the guidelines of class interfaces, which will be used in the next sections, down below:

Package Name	Name of the package
Class/Interface Description	Short description of the class/interface
Operation	Operation definitions provided by the class

1.3 Engineering Standards

We use Unified Modeling Languages (UML) and IEEE writing format as engineering standard in our reports. UML played important place in our graphical representation of the class diagrams and IEEE standard help us to create credible citations.

2. Packages

Cerebro app uses Model View Controller pattern for its interface. With MVC there will be also some other packages working together to form the program. These Packages are ClientPackage, ModelPackage, ViewPackage, ControllerPackage, UtilitiesPackage, ServerPackage.

2.1 Client

Client subsystem is the user part of the program. It is android based and has several parts.

2.1.1 Model

CerebroApp comes together with its Model classes. These classes have relationships with View and Controller classes and constructs the mains template of objects of the CerebroApp

- com.cerebroapp.model.widgetmodel
- com.cerebroapp.model.objectmodel

2.1.2 View

View objects are the visible part of the program to the user. It provides user with the interface elements. Main purpose of these classes are displaying the data from the model objects and creating the environment of editing that data

- com.cerebroapp.view.widgetview
- com.cerebroapp.view.objectview

2.1.3 Controller

A controller object is the active bridge between one or more view and model objects. It manipulates qualities of the model and draws it out with the View objects. It can also setup and coordinate tasks for the Cerebro app and manage lifecycles of objects.

- **Server Controller**

The server controller will manage the connection with API. Also it'll manage the user's settings and search history.

- **Connection Manager**

Connection controller will receive the request decides the requested activities based on request parameters and will delegate tasks to be performed based on the request parameters. The connection manager will manage the request with the network.

- **Search Controller**

Search Controller contains classes that will handle the interactions to start the search flow. Also, it will generate search queries and manage the API and server controller.

2.1.4 Utilities

- ***com.cerebroapp.utils***

This package contains all the classes that implement the functionalities that are provided by other sources such as Android OS. It is a wildcard package that provides many small pieces of different interfaces ranging from device camera access to image compression.

2.2 Server-Side Packages

The second half of Cerebro is its Server where the compute-intensive work is done. The names of the packages that belong to this subsystem start with the prefix ***"com.cerebroserver."***

2.2.1 Connector

-com.cerebroserver.connector

This package provides the interface to handle the connection, the API requests and responses between the server and its clients.

2.2.2 Image Processing

-com.cerebroserver.imageprocessing

This package provides the interface to match images in the coming requests to entries in the database. It runs the necessary operations like detecting interest points, creating local descriptors and feature vectors etc. It uses the interface provided by the database manager to run queries.

2.2.3 Database Manager

-com.cerebroserver.databasemanager

This package provides the interface to make queries to the database to find the matched image from the database. It is also responsible for simple database transactions, insertions, updates or deletes.

3.Class Interfaces

Package name: **com.cerebroapp.model.widgetmodel**
Description: Package for the model of the widgets. Provides basic functionality as an interface to be used by view and controller classes.
Operations: • getUniqueID() • getType() • getContent() • loadContent() • reload()

Package name: **com.cerebroapp.view.widgetview**
Description: Package for the view of the widgets. Provides basic functionality as an interface to be used by model and controller classes
Operations: • getUniqueID() • getSize() • setsize() • getView()

Package name: **com.cerebroapp.controller.serverconnector**
Description: Package for server connector that handles the API requests and responses exchanged between the client and the server.
Operations: • constructRequest(url, flags[], type, data[]) • pushRequests() • parseResponse()

Package name: **com.cerebroapp.controller.connectionmanager**
Description: Package for connection manager that provides basic HTTP methods to be used for connections with both the server and third party sources.
Operations: • GET (url, flags[], body) • POST (url, flags[], body) • PUT (url, flags[], body) • DELETE(url, flags[], body)

Package name: **com.cerebroapp.contoller.searchcontroller**
Description: This is the interface that allows the other drivers to surrender the control flow to the search controller. Once a search has started Search controller assumes full responsibility until search has been completed. the control flow can be interrupted with cancelSearch() and restartSearch().
Operations: • startSearchByImage() • startSearchByKeyword() • cancelSearch() • restartSearch()

Package name: **com.cerebroapp.utils**
Description: This is the wildcard package that contains all utilities that are provided by other sources such as Android OS. This package provides the interface that implements those utilities.
Operations: • startCamera() • captureImage() • closeCamera() • compressImage()

Package name: **com.cerebroserver.connector**
Description: This package handles the API request and connections between the server and the clients. This is the server side counterpart of connection manager and server connector.
Operations: • sendResponse() • popPendingRequest() • parseRequest() • constructResponse() • sendNotifications() •

Package name: **com.cerebroserver.imageprocessing**
Description: This package provides the necessary tools to match incoming searches. It uses the interface provided by database manager to query for matches.
Operations: • getSimilarityTreshold() • setDefaultSimilarityTreshold() • setDefaultAlgorithm() • matchBook() • matchMusic() • matchMovie() • match()

Package name: **com.cerebroserver.databasemanager**
Description: This package provides the interface to access database components. The two main components are the Media database and the device database. The device database holds all the information on mobile devices that use the app.
Operations: • runImageQuery(types[], matchingFunction(), interestPoints[], localDescriptors[], data[]) • runDeviceQuery(deviceModel[], deviceID[], deviceAppVersion[], deviceFirmvare[]) • updateTables()

4. Glossary

- **Android:** It is a mobile open source operating system which is developed and supported by Google Inc.
- **App**: Application
- **MVC**: Model-View-Controller
- **OS**: Operating System
- **UML**: Unified Modeling Language