AIC – THE ANDROID AND IOS APP EXPLORER WITH COMPARISON

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Harini Chilamantula

SPRING
2015
AIC – THE ANDROID AND IOS APP EXPLORER WITH COMPARISON

A Project

by

Harini Chilamantula

Approved by:

__________________________________, Committee Chair
Meiliu Lu, Ph.D

__________________________________, Second Reader
Ying Jin

____________________________
Date

iii
Student: Harini Chilamantula

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

______________________
Graduate Coordinator

Ouyang Jinsong, Ph D.  Date

Department of Computer Science
Abstract

of

AIC – THE ANDROID AND IOS APP EXPLORER WITH COMPARISON

by

Harini Chilamantula

With the advent of mobile revolution after the introduction of IPhone in 2007, the momentum in smartphone adoption has been extraordinary. Later, the availability of Android OS based phones, at different price levels, further accelerated the smartphone revolution. Most important factor of the smartphone success is the availability of applications, which exploited the hardware & OS features of smartphones and provided the capabilities that consumers never thought of. App stores (introduced by Apple and Google) are the repository of validated, reliable and secure applications available for the consumers to download and use. This mobile revolution resulted in more than 3 million applications, to date, for the consumers to choose from. Choosing applications from such a humongous number of applications is quite overwhelming and daunting task for the consumers. It may result in lack of consumer interest in new applications and eventually stifle the software innovation.

AIC project solves a few of these problems by providing up to date and top ranking applications for the consumers to review and analyze. Further, this application provides both Android and IPhone applications in one place so user can evaluate the applications before buying their next phone. This tool can be used to analyze multiple applications and provide detailed analytics which are useful for both consumer and developer community. Another aim of this project is to
demonstrate real life application of data warehousing techniques and how these techniques solve some of the complex problems in an elegant way.

The application has a three-tier architecture as (1) Presentation layer (2) Application layer (3) Data layer. The tool presents web interface for the users to choose their favorite store, category and application. Application layer on the server processes the request. In data layer, the data is sourced from Apple’s App Store and Google’s Play Store with the help of APIs. Later extracted data is analyzed based on user request. The results are presented in a rich UI such as graphs and tables. This project involves web UI rendering, request processing, data analysis and comparison, API query generation, data collection and preprocessing, data transformation, data mart design, generating charts and tables etc. Various technologies such as PHP, HTML, JavaScript, JQuery, and MySQL are used in implementation of the project.

_______________________, Committee Chair
Meiliu Lu, Ph D.

_______________________
Date
ACKNOWLEDGEMENTS

I would like to thank Dr. Meiliu Lu for guiding me through this project and during my coursework. Dr. Meiliu Lu has been excellent mentor and her valuable suggestions helped me in gaining strong technical insights. She has been very considerate and accommodating during my project especially with my health situation.

I would like to thank Dr. Ying Jin for taking time to review my report and helping me with important feedback.

I would also like to thank Dr. Cui Zhang, Dr. Nick Faroughi, Dr. Arad, Dr. Ouyang Jinsong and Eric Merchant for helping and guiding me during various phases of my Master’s degree in Computer Science.

Finally, I would like to thank my family especially my husband Sandeep Venishetti for being with me and supporting me during my Master’s degree in Computer Science.
# TABLE OF CONTENTS

Acknowledgements........................................................................................................... viii

List of Tables ....................................................................................................................... xii

List of Figures ...................................................................................................................... xiii

Chapter

1. INTRODUCTION ......................................................................................................... 1
   1.1 Motivation .................................................................................................................. 1
   1.2 Goal ........................................................................................................................... 2

2. TECHNOLOGIES, LIBRARIES AND FORMATS ..................................................... 5
   2.1 Technologies .............................................................................................................. 5
   2.2 Libraries ................................................................................................................... 12
      2.2.1 IOS App Store API .......................................................................................... 12
      2.2.2 Android Market API ....................................................................................... 14
   2.3 Data Serialization ..................................................................................................... 16

3. SOFTWARE ARCHITECTURE AND DESIGN ..................................................... 21
   3.1 Application architecture and design ........................................................................ 21
   3.2 Presentation layer ..................................................................................................... 22
   3.3 Server-side application layer .................................................................................... 26
      3.3.1 Request processor ............................................................................................. 28
3.3.2 Sub-Request Generator ................................................................. 28
3.3.3 Aggregator .................................................................................. 29
3.3.4 Comp Analyzer ........................................................................... 30
3.3.5 Render Engine ............................................................................ 31
3.3.6 ETL Engine ................................................................................ 31
3.4 Data layer or Database architecture ............................................. 33

4. IMPLEMENTATION ............................................................................. 37

4.1 Data Mart implementation ............................................................... 37

4.1.1 Connect ....................................................................................... 37

4.1.2 Create ........................................................................................... 38

4.1.3 Look Up tables creation ............................................................... 38

4.1.4 Insert ............................................................................................ 39

4.1.5 Iphone Category Dimension ....................................................... 41

4.1.6 Android Category Dimension ..................................................... 42

4.1.7 Fact Table ..................................................................................... 43

4.2 Implementation of Application User Interface ............................... 45

4.2.1 Welcome Page ............................................................................. 46

4.2.1a Project Overview ................................................................. 46

4.2.2 Help Page ................................................................................... 47
5.4.1 IOS App Comparison ................................................................. 61
5.4.2 Android App Comparison ............................................................. 66
5.4.3 IOS Apps Vs Android Apps ............................................................ 70

6. CONCLUSION ..................................................................................... 74

6.1 Conclusion ..................................................................................... 74
6.2 Future Work ................................................................................ 75

References ............................................................................................. 77
LIST OF TABLES

Table 1. Valid Search parameter keys for App Store API ................................................................. 13
LIST OF FIGURES

Figure 1. Application availability on AppStore and PlayStore .................................................. 1
Figure 2. ETL process for loading data from multiple sources to Data warehouse ...................... 10
Figure 3. Search sample example for Google Play Store using Market API ................................... 15
Figure 4. Category Search sample example for Google Play Store using Market API .................. 16
Figure 5. Example of proto message containing information of “Person” ................................ 18
Figure 5a. Example JSON data structure of “Person” .......................................................... 20
Figure 5b. Block diagram of AIC project architecture .......................................................... 21
Figure 5c. Main page flow control ......................................................................................... 23
Figure 6. IOS app store search flow control ............................................................................ 24
Figure 7. Android app store search flow control ...................................................................... 25
Figure 8. Comparative analysis flow control ........................................................................... 26
Figure 9. Application layer architecture ................................................................................... 27
Figure 10. Snowflake schema for the database implementation .............................................. 35
Figure 11. Welcome page ....................................................................................................... 46
Figure 12. Project overview page ............................................................................................ 47
Figure 13. Help page ................................................................................................................ 47
Figure 14. Feedback page ....................................................................................................... 48
Figure 15. Contact us page ..................................................................................................... 48
Figure 16. Iphone Tab ............................................................................................................. 49
Figure 17. Iphone Application selection page ........................................................................... 50
Figure 18. Iphone Application results page ................................................................. 50
Figure 19. Android Tab ................................................................................................. 51
Figure 20. Android Application Selection .................................................................... 52
Figure 21. Android Application Result page ............................................................... 52
Figure 22. Comp Applications page ........................................................................... 53
Figure 23. Comp Applications page with populated compare lists ............................... 54
Figure 24. Comp results page with comparison data .................................................... 55
Figure 25. Comp results page with comparison graphs ................................................ 56
Figure 26. Use Case study of compare ....................................................................... 57
Figure 27. Data modelling details of the AIC project ................................................... 58
Figure 28. Consumer User perspective UML .............................................................. 59
Figure 29. Developer User UML .................................................................................. 60
Figure 30. System design UML ................................................................................... 61
1. INTRODUCTION

1.1 Motivation

The introduction of IPhone by Apple in 2007 changed the mobile communication world in many ways. The primary reason for IPhone’s phenomenal success is the App Store that provides millions of applications via secured, reliable and verified source. It enabled innovative and breakthrough mobile phone usage models. Eg: Browsing, Games, Books, Online shopping, Navigation, Fitness Applications, Music etc. Later Android was introduced by Google in 2009 and made it open source. Several variants of Android phones were introduced by number of companies. Developer community sensed the momentum of consumer affinity towards latest iOS and Android based phones and jumped on to developing innovative applications. This resulted in exponential growth of applications available in both iOS App Store and Android Play Store as shown in Figure 1.

![Available Applications in Appstore and Playstore](image)

Figure 1. Application availability on AppStore and PlayStore
Further, introduction of tablets computers based on the iOS and Android resulted in several more variations of applications. Productivity Apps and big screen devices (Tablets, Convertible laptops etc.) helped these iOS/Android devices to make their way into Business/Enterprise world as well.

The number can only spike with the increased adoption of mobile computing by Enterprise users and consumer. While it is great that there is a huge pool of applications available to serve various needs of customer/user, it also poses huge challenge to the user when it comes to finding an application that works best suits one’s purpose.

Further, application developers are also finding it very tough to increase user adoption and also monetize their applications. It is becoming very difficult to provide unique features of the applications and market the same due to millions of applications available on various stores.

In addition to the above mentioned challenges, the main motivating factor of the project is to find an efficient solution to design and implement system to analyze such large collections of applications and data. In the modern age of digital world, the amount of raw data is overwhelming, so efficient and scalable database system is required to manage the data. Data-warehousing techniques solves majority of real-life problems and offers platform for software innovation.

1.2 Goal

To solve some of the issues in rising number of applications in various stores, variations of the applications across different types of devices, the AIC project has been developed.

The goal of the project is to showcase the potential of a AIC project for iOS and Android applications, provide top list of applications in each of the famous categories, provide details of
each application such as App name, description, size, latest release, price, reviews and download link etc. The goal is also to provide latest and update ratings and rankings of the store apps. A developer or a user can also be benefited with a comparative analysis of various applications of different stores or even same store and among various applications. Comparative analysis of applications is another key goal of the project.

Apart from frontend, the goal of the project is to efficiently retrieve the data, process the data, transform and store the data through an efficient database architecture and schema. The key goal of the data mart of the project is to provide filtered, useful and curated data of applications of various stores. Each app record has set of attributes like price, release, version, size, reviews, download link etc.

Performance of the AIC project is another key emphasis of the project. The AIC project has several purposes for both consumers and developers.

App Discovery and Details: Consumers can find top applications among various categories, price, reviews and application size etc. One can take informed decision of which application to use and download, which type OS, the next phone would be based on etc. The AIC project pulls the real-time reviews, ratings and rankings to be most up to date, all with great store performance.

Search Results: Users may be able to search a specific app in the stores. A matching/relevant list of applications is obtained and presented to the user.

Application Analysis: Developers may be able to use this platform to find what users expectation of new applications, how certain applications are made to top list of applications, what are key factors in success or failure of various applications, how an application could be priced to be
successful and how to the application should be developed to be of minimum size (which is a key factor for space constrained mobile devices) etc.

**Brand Protection:** Developers can also use this store to figure out any brand violations from other developers, or any patented feature violation etc. Instead of visiting multiple stores, Developer can save time and effort due to the curated data of AIC project.

**Market Analysis:** In this fast changing mobile communication world, to be successful in the long term, a Developer can’t be complacent with one’s released application. He/She has to continuously monitor the market trends and should monitor competing applications for the type of features added, the changes in various versions, the reviews, salient features that consumers like etc. That helps developer to think and add even more innovative and unique features to stand out of the bunch of applications. The AIC project provides curated data and provides comparative analysis of various applications in both iOS and Android applications. One can choose multiple applications from each store to do analysis. It provides detailed differences and graphs.

**Advertising:** AIC project is a great platform to advertise and market the applications from new developers. Marketing salient features of the new application would result in greater adoption, greater revenue, popularity and feedback.

As this chapter introduces the project, Chapter 2 details some of the technologies, libraries and formats used in the project. Chapter 3 describes the software architecture and design of the AIC project. Chapter 4 provides the Implementation details. Use case study is provided in Chapter 5. Chapter 6 concludes this report.
2. TECHNOLOGIES, LIBRARIES AND FORMATS

Before detailing the project design architecture and implementation, it might be helpful to provide some details of the technologies used in the project, libraries used for extracting application data from app stores and the various data formats.

2.1 Technologies

**PHP:** PHP is server side scripting language that allows developers to create dynamic content, rich UI, send/receive cookies, database operations, command line scripting, standalone desktop applications. PHP code can be mixed with HTML code or can be combined with various other web frameworks or templating engines [1]. PHP is supported on all major operating systems like Linux, Windows, Mac OS etc. It is also supported by most of the web servers like Apache, IIS etc. Also, PHP supports procedural programming as well as object oriented programming model or mix of both. PHP can output HTML, images, PDF files and even flash content. Strongest and most significant feature of PHP is its support for several databases. PHP offers several easy-to-use features to write database enabled web page using either database specific extensions (MySQL extensions) or using abstraction layers like PDO or using Open Database connection standard extensions (ODBC). PHP has support for communicating to other services using protocols like IMAP, COM, HTTP, LDAP, SNMP, OP3 etc. It also has rich regular expression capabilities for text processing [2].

**JQuery:** JQuery is a cross platform lightweight small, fast and feature rich JavaScript library. The emphasis is on the easy to use JavaScript library [3]. It contains features such as HTML document traversal, HTML/DOM manipulation, CSS manipulation, HTML event methods and event handling, eSffects and animation, Ajax and utilities with a simple API. It is supported in several popular browsers. It wraps several common tasks (requiring several lines of code) under
methods that can be readily used. It also provides developers to create plug-ins on top of the
JavaScript Library. The modular approach to the JQuery helps in the creation of rich dynamic
webpages. It can be either downloaded or can be included it from a CDN (Content Delivery
Network).

**JavaScript:** JavaScript is a cross platform object oriented lightweight language. It is popularly
used as part of browsers whose implementation allows client-side scripts to interact with user,
browser control, asynchronous communication and document display content manipulation. It
can also be used on server side with runtime environments like Node.js[4]. Core JavaScript have
common language features and standard library of objects like Array, Date, Math and control
structures. Client-Side JavaScript extends the core language by supplying objects to control a
browser and its DOM (Document Object Model (DOM) such as those to place elements on
HTML form, actions like mouse click, form input and page navigation etc. Server-side JavaScript
extends core language by supplying objects relevant to server environment, such as extensions to
allow applications to connect to database, file manipulations on server etc.[5]

**HTML:** HTML is a markup language for describing the web pages and its contents. The
language comes up with set of markup tags one can use to describe the HTML documents. Each
HTML markup tag describes different document content. Along with Tags, attributes help in
creating flexible and custom web pages. HTML elements form the building blocks of all
websites. HTML allows images and objects to be embedded and interactive forms can be created.

**CSS:** Cascading Style Sheets is a style sheet language for describing the look and format of a
document in markup language like HTML. Along with HTML and JavaScript, CSS is the most
important technology used by websites to create stunning websites and rich user interfaces for
web applications. The primary purpose of the CSS is to separate the content from the presentation
of a document. This provides great accessibility, flexibility and control in presentation of
document. CSS helps in separating presentation from HTML content and by provides formatting
instructions for each HTML element. The separation of formatting and content helps in
presenting same content in different styles for different rendering options. [6]

MYSQL: MySQL is open source Relational Database Management System (RDBMS) widely
used on the servers which is ideal system for both small and large applications. MySQL uses
Structured Query Language (SQL) and has support on wide number of platforms. MySQL server
system provides capability to add, access and process data store in a computer database. MySQL
is very fast reliable and easy to use. Owing to MySQL being RDBMS, where database tables are
built by separate tables and build relationship than one big one, it offers high performance and
reliable database system. Users can setup the rules to establish relationships between different
data fields such as one-to-one, one-to-many, unique, required or optional and pointers between
different tables. The database enforces user rules so that the application would see consistent,
unique and up-to date data. MySQL is developed, distributed and supported by Oracle [7].
MySQL forms the central component of popular LAMP, XAMPP web application software stack.

Tools:

NetBeans: NetBeans is an Integrated Development Environment (IDE) that lets developers
quickly develop, debug and deploy applications based on various languages such as Java, HTML,
HTML5, JavaScript, CSS, PHP, Python, C, C++ etc. It comes with built-in source editors,
analyzers, convertors, compilers, debuggers and source revision control systems. NetBeans IDE
indents, highlights source code, refactor code and code templates with code generators. NetBeans
platform allows applications that can be developed from a set of modules and can be extended to
third party developers. NetBeans platform offers common services that can be reused to develop
applications and lets developers focus on specific logic to their application than common UI tasks. The features of NetBeans platform range from User Interface management, user settings management, storage management, window management, wizard framework, visual library and Integrated development tools [8]. NetBeans comes with several modules such as NetBeans Profiler, GUI design tool, NetBeans JavaScript editor (which extends support for JavaScript, Ajax and CSS).

**XAMPP:** XAMPP is free and open source cross-platform web server solution stack. It comes with stack of Apache HTTP server, MySQL database, PHP interpreter and Python interpreter. It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for development and testing purpose. Everything to set up web server such as server application (Apache), database (SQL) and scripting language (PHP) is included in the stack. XAMPP is cross platform stack that works equally on Linux, Mac and Windows. As most webservice deployments use same components as in XAMPP, it becomes easy to deploy the code developed on a local server on to a live server. It also comes with a control panel that gives complete control over all installed XAMPP components. It allows use start/stop different modules, launch UNIX shell, access windows explorer and monitor processing running in background etc. XAMPP comes with SMTP and FTP server included as well.

**Apache HTTP Server:** Apache HTTP server runs in the background under an appropriate Operating System supporting multi-tasking and provides services to other applications that connect to it such as client web browsers. Initially developed to work with Linux Operating System, it was quickly adapted to other operating systems like Windows, Mac, Solaris, and FreeBSD etc. The Apache binary running under UNIX is called HTTP daemon and under win32 is called Apache.exe. Though original Apache core is fairly basic and contains a limited number
of features, it was made quite powerful with added functionality using many modules that extended the server’s capabilities. Functionality that is not required can easily be removed making the server small, light, very fast, consuming less system resources, fewer security holes and less memory. Apache server can be easily integrated to other open source applications such as PHP, MySQL making it even more powerful. It is a key component of XAMPP stack. Apache server offer number of services and using various protocols such as HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), Domain Name Service (DNS), File Transfer Protocol (FTP). Apache’s key task is to establish communication over networks and it uses TCP/IP protocol. Apache server comes with configuration flexibility using configuration files in which directives are added to control the server’s behavior [9].

**Master Data Creation Process:**

**ETL:** ETL is essential component to load data into Data-Warehouses and DataMart. It stands for Extract, Transform and Load [10]. ETL defines a process of how the data are loaded from multiple source system to the data warehouse as shown in Figure 2. ETL processes are widely used in data integration, data migration and data management (MDM) processes.
Figure 2. ETL process for loading data from multiple sources to Data warehouse.

**Extract** part of ETL defines how data is extracted from multiple data sources, homogeneous or heterogeneous sources, makes it accessible for further processing. The key objective of the extract step is to retrieve all the required data from the all the sources with as minimum resources as possible. Efficient definition and design of extract process is required to avoid negatively affecting the source systems in terms of performance or response time. Each source system may use different data organization and format. The extraction phase’s goal is to extract and present data into a single format appropriate for transformation processing.

**Clean**: While clean is not exclusively defined in ETL process, it is important to clean to the extracted data to ensure the quality of data in the data warehouse. Sometimes, it is considered part
of extraction phase. During Cleaning step, several tasks as performed such as creating unique identifiers, type conversion of Null into standardized NA, converting extracted into standardized format, validating specific data (such as address fields) etc. Specifically, data validation involves confirming whether data pulled from sources have expected values. If data fails validation rules, either it is rejected and reported back to source system or modify the data to pass validation rule to accept for next phase.

**Transform:** In Transform step, transformation of data is performed to reconcile the data across multiple sources, perform calculations or string parsing, improvise data with external lookup information and convert it to match format required by the target system. During transform step, sets of rules are applied to transform the data from source system to the target system. It may include several tasks such as conversion of the data to appropriate dimensions, deriving new calculated values, encoding free-form values, translated coded values, transposing or pivoting, standardization, disaggregation of multiple columns, splitting columns into multiple columns, type and unit conversion to match target system, joining data from multiple sources, data aggregation, sorting and generating surrogate keys for further processing. Sometimes this step also involves applying advanced validation rules [11].

**Load:** This step involves loading the resulted transformed data into various business intelligence applications such as Data Warehouse, Data Marts or Online Analytical Processing (OLAP) applications. During this step, it is ensured that the load process is performed correctly and efficiently with minimum system resources. Generally, the target of load process is database [11]. Generally constrains defined in the database schema as well as in triggers are activated after the data load is done. This is to improve the performance of ETL process.
2.2 Libraries

2.2.1 IOS App Store API

Apple’s IOS app store [12] database is accessed using App Store API provided by Apple. The App Store API allows users/developers to provide a qualified URL embedded with a search query for the content request from App Store and returns the data in JavaScript Object Notation (JSON) format. The content requested can be virtually everything available publicly on App Store and iTunes store such as apps, books, movies, music, podcasts, games etc. Developer can also request content based on ID-based lookup.

Example of fully qualified URL looks something like following:

https://itunes.apple.com/search?parameterkeyvalue

Where parameter key value can be one or more key value pairs that gives details of the query.

When multiple key/value pairs need to be passed as parameters, the key/value pairs need to be concatenated (using &) with <key>=value format.

Example of fully qualified URL with multiple key value pairs as parameter looks something like following:

https://itunes.apple.com/search?key1=value1&key2=value2&key3=value3

Valid Keys are illustrated in the table 1:
### Table 1. Valid Search parameter keys for App Store API

<table>
<thead>
<tr>
<th>Parameter Key</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>search string for APP store</td>
<td>Yes</td>
</tr>
<tr>
<td>Country</td>
<td>country specific store to be accessed</td>
<td>Yes</td>
</tr>
<tr>
<td>Media</td>
<td>the type of content to search: valid content are movie, apps, software, ebook, music, audiobook, book etc.</td>
<td>No</td>
</tr>
<tr>
<td>Entity</td>
<td>type of results returned. Eg: bookAuthor for book type of content</td>
<td>No</td>
</tr>
<tr>
<td>Attribute</td>
<td>attribute you want to search relative to type of media.</td>
<td>No</td>
</tr>
<tr>
<td>Callback</td>
<td>Name of callback function to use when returning search results</td>
<td>No</td>
</tr>
<tr>
<td>Limit</td>
<td>Limits the number of search results</td>
<td>No</td>
</tr>
<tr>
<td>Lang</td>
<td>language of search results</td>
<td>No</td>
</tr>
<tr>
<td>Version</td>
<td>version of search</td>
<td>No</td>
</tr>
<tr>
<td>Explicit</td>
<td>whether explicit content allowed</td>
<td>No</td>
</tr>
</tbody>
</table>

The App Store API results content in JSON format, either a collection of name/value pairs, known as object or an ordered list of values such an array.

One can also generate lookup query based on look up ID instead of search query. For example, the following is a fully qualified URL for lookup-based query.

Eg 1: Look up Jack Johnson by iTunes artist ID:

https://itunes.apple.com/lookup?id=909253

Eg 2: Look up Yelp Software application by iTunes ID:

https://itunes.apple.com/lookup?id=284910350

Eg 3: Look up a book by its 13 digit ISBN:
2.2.2 Android Market API

Android Market API is open source 3rd party API to access Google PlayStore to search for an application and also extract application information. Android Market API supports JAVA, RUBY as well as PHP and uses Google Protobufs. Given that it is third party API, a Google Account is required. The API requires a valid android device ID to be provided by developer.

The API can perform following tasks:

- Category Search
- Complete App details
- Search with key word
- Top Apps search
- Get comments & Screenshots

To search using the Android Market API, one has to include login details, device id and protocol buffers along with MarketSession.

Example search from Market API [12] is shown in Figure 3:
Figure 3. Search sample example for Google Play Store using Market API

Similarly one can also create category search request as shown in Figure 4 example code snippet from Android Market API:
Figure 4. Category Search sample example for Google Play Store using Market API

More code snippets and relevant API functions can be found in Android Market API [13].

2.3 Data Serialization

With the advent of high-level languages such as Java, C# and developer friendly platforms (Visual Studio, Android, XCode) supporting the idea of high level and abstract representation of real life problems into respective programs, the complexity of data representation is ever increasing. At the same time the need for cross-platform compatibility and hassle-free user experience of applications across different devices and with different Operating System is also increasing. To support the transmission, storage, and reconstruction of such data objects across
multiple and different system/computer environment, Data serialization techniques [14] are employed. It is a process through which complex data structures are translated into stream of bits that can be stored, transmitted and reconstructed into original objects. Same serialization technique should be used for both translating to raw stream of bits and also re-construction into complex objects. Serializing an object is also called as marshalling and deserialization of objects are also called as unmarshalling.

Data Serialization techniques are particularly useful for implementation of Remote Procedure calls, to support multiple and different platforms and for implementing applications that are architecture independent and memory layout independent.

Few such serialization format and techniques that are commonly used and especially those employed in this project are as following.

**XML:** XML (Extensible Markup Language) [15] is most popular representation supported by several platforms. It is a markup language with set of rules for data encoding; XML is based on XML Schema (W3C). The most salient characteristic of XML over other prior data formats is that it is both human-readable and machine-readable format made it very popular. The key goal of XML is to simplify and increase usability across several platforms and over Internet. The XML defines several constructs such as Unicode character, Tags, markups, attributes, Elements etc. that gives the power to the user to define complex data structures as well as allow high level of flexibility. XLST (Transformation language) for transformation of different outputs, XPath/XQuery (querying language) are all based on XML format. With several APIs as well as binary support and being standardized, it quickly became popular among developer community.

**Protocol Buffers:** Protocol Buffers [16] is another method of serializing data with emphasis on performance, simplicity and extensibility. Relatively new, the Protocol buffers are developed by
Google, with wide cross platform support. The key aspect of this method is based on Interface description language, where one describes structured data and an application is employed to generate the equivalent source code for parsing the structured data. Thus, this enables protocol buffers to be able to support across wide variety of platforms. It has wide support in several popular languages with compilers available in C++, Java, Python, JavaScript, PHP and Ruby etc.

The key design goal of the protocol buffer is performance, ease of use and less verbosity. Protocol buffers are flexible, efficient and have automated way to serialize structured data. Developers can define the specification of how the data that needs to be serialized by defining it in protocol buffer message types in .proto files as shown in Figure 5. Every protocol buffer message is a logic record of data with name-value pairs.

Example [17]:

```protobuf
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}
```

Figure 5. Example of proto message containing information of “Person”.

Once the message is define, we can use SerializeToOstream() to serialize the data.

Eg:  person.SerializeToOstream(&output)

To reverse serialize, one can use parse api like ParseFromIstream():

Eg:  person.ParseFromIstream(&input); where input is has serialized data.

Protocol Buffers have several advantages over XML data structures such as

- Protocol buffers are easy to use
- Simpler and smaller
- Lot more faster than XML
- Less ambiguous to serialize and de-serialize
- Much simpler to programmatically access

**JSON**: JavaScript Object Notation is an open standard and independent data exchange format developed for representing data structures. It is human readable and language independent. Data structures in JSON are based on key-value pairs. JSON is generally used to serialize and transfer data over network. It has been extended from JavaScript language and uses file extension of .json. When compared to XML, JSON is less expressive or less verbose and simpler. The example JSON data structure is shown is Figure 5a.
Example:

```json
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 25,
    "height_cm": 167.6,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
```

Figure 5a. Example JSON data structure of “Person”

Powered with information of all the technologies, API libraries, data formats used in the project, we now delve into the details of the software methodology architecture and design of the project, implementation details of the project and use case scenarios.
3. SOFTWARE ARCHITECTURE AND DESIGN

This chapter introduces the software architecture and design of the project, illustrates how the application is partitioned, designed, and provides the details of each functional unit and all the functional units interact. This unit also illustrates various flow diagrams of how a request is propagated and how the final objective is achieved. This chapter will also provide the details of the database architecture, how performance is optimized and how the database is optimized.

3.1 Application architecture and design

Key objectives of the architecture of the project are to make it modular, easy-to-use, flexible, performance optimized and user friendly. Though the AIC project is an academic project, the real life use-case scenarios and advantages abundant and well described in later chapters. To enable any future expansion, the architecture is well defined and designed in such a way that there is scope for future expansion of the project, including a commercial project if it is viable.

The project is partitioned into (1) client-side execution or Presentation layer (2) server side Application layer (3) Data layer as shown in Figure 5b.

![Diagram of AIC project architecture](image)

Figure 5b. Block diagram of AIC project architecture
3.2 Presentation layer

Providing rich user experience is one of the key goals of the project and presentation layer is designed accordingly. The web pages are divided into two sets of pages. First one is a set of informational pages such as Home page, Project overview page, Help, Feedback and contact us pages. The other one is working set of pages (such as Iphone, Android, compare, case study and App store data modelling) that implements the AIC project. This is to separate the actual mobile application project and its related documentation or developmental process.

The informational set of pages is provided with documentation and also ability to receive feedback that is being recorded in database. The feedback can be used for further improvement of the project and its capabilities. Application overview provides the details as in this report.

The working set of pages implement the AIC project. The pages are vibrant and colorful with informative and entertaining live icons/images from both iOS App Store and Android Play store. The pages can be rendered using slider mechanism, which is typically found in modern web designs. The pages are categorized into: Iphone: to provide the list of categories for iOS applications, their top ranking applications are loaded dynamically and based on the user selection or a search string, the application details are rendered. Android: to provide the list of android Play store application categories, list of top applications dynamically fetched and based on user selection, application details are rendered. Compare: Compare page can be used to compare any number of applications from either of iOS App store or Android Play store. It renders the comparison tables and colorful graphs for certain fields where it makes sense. CaseStudy: provides example case study information. App Store DataMart: page provides data modelling or DataMart details of the AIC project.

Next few sections describe the flow of control in the presentation layer:
Figure 5c shows that the flow control starts from *start* state and loads main page. From there, based on user choice, it can transition to either IOS App Store, Android Play store or Compare page. An Intermediate welcome page is also reached where important information like project overview, user guide, help, feedback and contact us pages are provided for the user.

Figure 5c. Main page flow control
Figure 6 shows the flow control of IOS application search flow. In this flow, when a user selects a specific category, the backend application layer loads the top applications in the category for the user to select or takes a search string. Based on the input, the application details are fetched. In this flow, it checks whether a new data extract is required or not based on the update time tags.

Similarly, Figure 7 shows the flow control of Android application search flow. The flow is similar to IOS application search flow.
Figure 8 shows the comparative analysis control flow. This flow details entire compare flow, where the presentation layer provides various categories to pick an application from. Later, it loads top applications in the selected category for the user to pick one or more applications from the store. These are added to the compare list. These options are provided for both iOS App Store and Android Play store applications. Once the user selects all the required applications and submitted for comparison, application layer sends the compare data through render engine and presentation layer displays the application details as a table and as well as graphs or charts.
3.3 **Server-side application layer**

The server side application layer is partitioned into several sub-blocks as shown in Figure 9:

- Request processor and Analyzer
- Sub-Request Generator
- Aggregator
- ETL (Extract, Transform, Load) Engine
- Comp Analyzer
- Render Engine

It is important to note that the sub-blocks/engines mentioned above are logical blocks and code sections in the implementation.

![Application layer architecture](image)

**Figure 9. Application layer architecture**
3.3.1 Request processor

Request processor processes and analyzes the request from the client side of the application, processes the request based on the type of request, type of App store (i.e. whether App Store or Play Store), initiates the subsequent action to Sub-Request generator. The request processor takes in the user request and decodes the type of app stores it may need to access for this request. It will also check for the type of category the request falls in. Request processor then generates a request object with appropriate member data. Based on this request, the request processor initiates a database retrieve and generates list of top 100 applications for this category that user may want to choose from, either for application details or for a comparative analysis. The resulting list is sent back to the Render engine. When retrieving the list of top applications from the database, it does check whether the data is up-to-date or stale. If not up-to-date, it will command sub-request generator to initiate app store search and update.

Once user choses a specific application for a given OS and category, request processor simply passes the request to next stages.

Comparative analysis engine is invoked if the request is of a detailed comparison of various applications from any of the stores.

This type of coarse level categorization helps in achieving performance and cleaner implementation.

3.3.2 Sub-Request Generator

Sub-Request Generator is rather a collection of functions each one specific to category of application and app store. This makes it possible to add some customization for the search request, adding metadata to the database and for also render engine. Sub-request generator takes the request from Request processor with the input of category, application store OS and name of
the application or a search pattern. It then applies any user specified customizations or app store and category specific customizations. Both the store APIs take different set of parameters for different category of applications. For an iOS app store search pattern, the parameters of media type, entity and attributes are populated in the Request object by the Sub-Request generator. For an Android Market API search pattern, Android Market API requires login connection and mobile device MAC id is required for it to make a connection with Android Play Store. So, Sub-Request Generator sets up a new MarketSession() connection, passes on login details and valid android device MAC id (provided in a file) to Market API. It also populates the parameters of ExtendedInfo, viewType, AppType with the Request object. It applies these customizations to the request object, sends the request and open MarketSession() object in case of Android to the Aggregator engine for further processing.

3.3.3 Aggregator

Aggregator is one of the key functional blocks of the application. This block is optimized for efficient processing and best performance. This block keeps track of most of QoS metrics such as last updated time, changes in list of top applications, the time when a next update is required. Such Metadata is maintained in database. When a request arrived, this block analyzes the request and checks if the data is updated for specific application. If the data is updated, it populates the request and initiates the render engine. If data is not recently updated, it initiates process of update from specific application store.

Aggregator understands the protocols of iOS App Store API and Android Market API. To this effect, it takes the request and creates command for appropriate store API with the details populated such as application name, category, level of information to fetch, version of app, type of search, attributes, media, entity, ExtendedInfo, viewType, AppType etc. Once the commands
are created, it invokes the corresponding API for fetching the application data from the specified stores.

It also sends the original request object to ETL engine so that it can decode the returned data appropriately (JSON decode or protocol buffers or XML decode). It also uses the request object so that the appropriate metadata (last updated time, list of top apps etc.) for the request is populated in the database once the request is received. This metadata is important for any future requests so as to optimize the store search and update costing performance, quality of service and user experience.

3.3.4 Comp Analyzer

Comp Analyzer is invoked by request processor upon decoding that the request type is a comparative analysis request. The request may include several applications from different categories and different stores. The Comp Analyzer takes a list of applications to be compared, the requested OS store (i.e. whether App Store or Play Store), category of each specific application. Comp Analyzer consolidates the queries for same dimension table to improve performance. It also checks the last updated time and if it is outdated, it invokes the Aggregator to update the data. Once after updated data is available, Comp Analyzer finds out the fields that need to be compared (fields set are fixed set for now but can be easily made programmable), extracts data from the database and processes the data for both table display and graphical display. While generating compare data, it extracts the application name, App icon, rating, size, category, price, download URL, screenshot, version and release date etc. It then passes this data to render engine. Being separate makes it easy to extend the functionality in future especially if user wants to filter out certain fields or sort app details based on certain fields etc.
### 3.3.5 Render Engine

As the name suggests, the primary task of Render Engine is to reformat the data to be able to display it in rich, efficient and user friendly way. This unit takes data from Request Processor or Comp Analyzer, formats the data based on the request and sends the final displayable data to the client side/presentation layer. This unit is responsible for displaying graphs and tables. It does normalization of some data fields (eg: price, size, rating etc.) for appropriately scaled, reasonable and useful graph display. The format of the UI components, fonts and colors is processed through this unit in conjunction with CSS definition file. Being a one stop functional unit for data display, it is very easy to extend this unit to be configurable especially display format. For example, users may want to display the graphs in horizontal bar or vertical bars, or may be straight lines or curved lines or a pie-chart etc.). At this point it supports the horizontal bar chart display only but it can be easily extended in future.

### 3.3.6 ETL Engine

ETL (Extract, Transform, Load) Engine is the most important functional unit that performs database operations. As explained in chapter 2, ETL is industry standard process in extracting data, transforming and loading data from multiple sources to DataMart or Data-warehouse. In AIC project, it receives data from multiple app stores, i.e. Apple’s App Store, Google Play Store and in different formats of JSON data format and Protocol buffers respectively. The key task of ETL engine is extract data from the both the stores as per the request set by Aggregator.

Both iOS App Store and Android Play store sends huge amount of data for each application searched. In Extract phase, data is collected from any/both of iOS App Store and Android Play store. IOS App store provides an API that can be used to query App store as explained earlier in chapter 2. One can use a search string or an ID. Similarly Market API, as explained in chapter 2,
can be used to query Play store with a search pattern and/or category. The data sourced from App store is in JSON format while the data sourced from Play store is in protocol buffer format.

Initially data is collected into appropriate data structures and then extracts only the required fields of the AIC project. This is especially important as iOS App Store sends several 100’s of field and data pairs. Similarly, Android Play store sends several fields 100’s of name/value pairs.

Extract phase only extracts required subset of 10-12 fields/attributes such as

- name,
- category
- version
- price
- size
- reviews
- ratings
- download URL
- icon
- screenshots

Data type: text values, images, navigations, symbols, URL links and graphs.

Data Size: need basis update of 1000’s of records.

Data source: iOS App Store, Google Play Store.

Categories: Games, Applications, Music, Books, Entertainment, Weather etc.
Later, data is cleaned to remove unnecessary and extraneous data from the above-mentioned fields. Data is also validated for correctness. All the columns and attributes are checked for any inconsistent and unacceptable formats. Also certain fields are checked for their liveliness. For eg: “download URL” is validated to be a correcting working link. Also, screenshots are checked for correct image format etc.

Once data is extracted, cleaned and validated, data transformation is performed to make all units compatible to the DataMart of the project. Eg: the size of application is converted into Bytes for both the stores. This is important because, during comparative analysis, it is important to make sure that the data being compared is in same units. Also, the images and screenshots are resized to common and fixed size. Ratings are normalized to 5 for all the applications. Similarly, price is adjusted appropriately.

After data transformation, data load phase is performed. During this phase, appropriate tables are populated with extracted and transformed data fields. Detailed DataMart architecture and schema are provided later in the chapter.

3.4 Data layer or Database architecture

Snowflake schema has been chosen as database schema architecture for the AIC project ’s Data Warehouse. In Snowflake schema, data table dimensions are normalized by removing low cardinality attributes and creating separate tables [18]. There are multiple dimension tables and single fact table as shown in Figure 10. A multidimensional data model has been implemented in the design of the AIC project Data Mart. As shown in Fig 10, a snowflake schema has been implemented.

The primary reasons for selection of snowflake schema are as following:
- Dimension tables are in normalized form thus the database is well organized and provides flexibility with defined relationship between tables. For example, per category dimension tables are created and normalized to reduce the access time, allow flexibility, and improve storage space.

- Snowflake schema improves query performance due to minimized disk storage requirements and by joining smaller lookup tables. Given the disk storage

- No data redundancy is snowflake schema making it easier to support, maintain and extend with optimized performance.

- While the queries are generally complex, the overall query performance per amount of data accessed is very high.

- Snowflake schema is especially useful for complex relationships such as many : many. The application data/fields of App store and Play store exposes such complex relationship.

- Given the number of fields/data in applications from iOS App Store and the Android Play store, there is very possibility of large number of dimension tables and the snowflake schema provides very good flexibility of future extension and expansion of the database.

- Each set of applications and categories have specialized attributes (non-common) that needs to be maintained and snowflake schema is good in such scenarios.
Figure 10. Snowflake schema for the database implementation

At first redundancy is removed by normalizing on “search OS”. The applications are grouped into two sets of tables. Further, each “OS” dimension is normalized on “category”. This leads to six lookup tables, each representing a specific category of applications in a specific operating system. Due to multilevel dimensional table hierarchy, a snowflake schema has been implemented. This results in simpler tables improving query performance due to minimized disk storage requirements and joining smaller lookup tables.
The AIC project data mart has one fact table and has multiple dimension tables. The fact table is “Search OS” and its dimension tables are “Android Category List” and “iPhone Category List”. Further “Android” table is normalized into six lookup tables of “Books”, “Music”, “Games”, “Business”, “Social” and “Shopping”. Similarly for “Iphone” dimension table is normalized into five lookup tables of “Books”, “Music”, “Games”, “Entertainment” and “Weather”.

As this chapter detailed the software architecture and design of the project, next chapter provides the details of the implementation.
4. IMPLEMENTATION

This chapter discusses the implementation details of the AIC project. Implementation is done using PHP, JavaScript and MySQL technologies. The application layer/server side application of the project is developed on NetBeans IDE, which has strong support from open source community especially for Java, PHP and Python based applications. The project is maintained in its own source control repository for easier development. The application is developed and tested using XAMPP stack. The database is created and maintained using SQL and MySQL workbench.

4.1 Data Mart implementation

As mentioned in the previous chapter Snowflake schema has been chosen as a database schema and to implement the data mart for the AIC project. The implementation has multiple lookup tables, which are implemented based on the dimension tables and further dimension tables are implemented based on the Fact table ‘Search_OS’. They have distinct values for the respective dimension tables. The implementation includes CRUD operations by creating, reading, updating and deleting the table according to the requirements. The following queries are used in the implementation of application data mart.

4.1.1 Connect

First, the connection establishment to MySQL is essential to send request and get response from the database.

```php
// Create connection
$conn = new mysqli($servername, $username, $password, $database_name, $post_Num);

//check connection
if($conn->connect_error){
    die("connection failed: .\$conn->connect_error");
```
4.1.2 Create

After connection is established, the databases are created using SQL language to MySQL DMBS. The tables created are Fact table, dimension tables and all the lookup tables. The following script checks the database presence; If not exists, it creates a new database.

```php
//Create database
$sql = "CREATE DATABASE IF NOT EXISTS $database_name";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
```

4.1.3 Look Up tables creation

There are 11 look up tables in the data mart, which are created using the following script. All the lookup tables have individual primary key and foreign keys. Primary keys are used as the references to the connected dimension tables. The following script check the presence of the table in the database; If does not find one, it creates new table and updates the data fields in the table. Finally the script verifies the creation and displays the result.

```sql
//create table
$sql = "CREATE TABLE IF NOT EXISTS $lookuptablename (id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, category VARCHAR(30), trackname VARCHAR(100), version VARCHAR(30),
```
release_Date DATETIME,
formatted_price VARCHAR(10),
filesize INT(10),
rating FLOAT,
url VARCHAR(200),
app_update TIMESTAMP,
app_detail_update TIMESTAMP,
image VARCHAR(600)
);

if ($conn->query($sql) === TRUE) {
    echo "Table created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

4.1.4 Insert

After creating the table the fields are filled by inserting the data in to the table from the Request object $Request. Following script INSERTS the data in to the table. The following insertion is done with the key value pair mechanism.

//insert values in table
$sql = "INSERT INTO $lookup_tablename (id, category, trackname, version, release_Date, formatted_price, filesize, rating, url, app_update, app_detail_update, image)
VALUES (($i+1), $Request.category_id, 'Books & Reference', '$Request.app_name[$i]', '', '', '', '', '', '', '', NOW(), CURRENT_TIMESTAMP(), '')WHERE NOT EXISTS (SELECT * FROM $lookup_tablename where trackname = $Request.app_name[$i])";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
echo "Error creating table: " . $conn->error;
}
The fields are updated every 24 hours by using a time stamp tag. Before updating the lookup table the aggregator unit verifies the last updated time by READ operation.

//check the timer for update
date_default_timezone_set('America/Los_Angeles');
    $today = date('Y-m-d h:i:s a');
    $sql = "SELECT app_update FROM $tablename where id = 1";
    $result = mysqli_query($conn, $sql);
    while($row = mysqli_fetch_assoc($result))
    {
        $seconds_count = (strtotime($row['app_update']) - strtotime($today));
        $sql=mysqli_query($conn, "SELECT trackname as name FROM $tablename where id = 1");
        $data=mysqli_fetch_assoc($sql);
        if((($seconds_count >= 86400) || ($seconds_count <= -86400))||($data['name'] == NULL))
        {
            {//Needs update, aggregator generates API query to fetch data from stores}
        }Else{  //Data is still fresh, no need to update
    }
If the updated date is expired on bases of 24 hours then the aggregator in the application will update the database using the following UPDATE operation. Table update is performed based on the Request object $Request.

//update table
$sql = "UPDATE $lookup_tablename SET version = '$Request.app_key_value[version]', release_date = '$Request.app_key_value[releaseDate]', formatted_price = '$Request.app_key_value[formattedPrice]', filesize = "$conn->error;
4.1.5 Iphone Category Dimension

The following script describes the CREATION of Iphone_Category_list dimension table to maintain the normalized OS specific data with multiple categories.

//create Iphone category table
$sql = "CREATE TABLE IF NOT EXISTS $Iphone_category_list ( 
  Iphone_Category_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, 
  Book_id int FOREIGN KEY REFERENCES Iphone_Books(id), 
  Games_id int FOREIGN KEY REFERENCES Iphone_Games(id), 
  Music_id int FOREIGN KEY REFERENCES Iphone_Music(id), 
  Entertainment_id int FOREIGN KEY REFERENCES Iphone_Entertainment(id), 
  Weather_id int FOREIGN KEY REFERENCES Iphone_Weather(id), 
  OS_name VARCHAR(30) 
)"; 
if ($conn->query($sql) === TRUE) {
    echo "Table created successfully";
} else {
    echo "Error creating table: " . $conn->error;
The following is the INSERT script to populate the created Iphone_Category_list dimension table based on the Request object. This table contains list of chosen categories from the Iphone OS and this further refers to five lookup tables.

```
//insert in to table Iphone_category_list
$sql = "INSERT INTO $Iphone_category_list (OS_name)($Request.OS_name)"
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully"
} else {
    echo "Error inserting record: " . $conn->error;
}
```

### 4.1.6 Android Category Dimension

The following is the CREATE script for the Android_Category_list dimension table to maintain the normalized OS data with multiple categories.

```
//create Android category table
$sql = "CREATE TABLE IF NOT EXISTS $Android_category_list_list (Android_Category_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, Book_id int FOREIGN KEY REFERENCES Android_Books(id), Games_id int FOREIGN KEY REFERENCES Android_Games(id), Music_id int FOREIGN KEY REFERENCES Android_Music(id), Social_id int FOREIGN KEY REFERENCES Android_Social(id), Business_id int FOREIGN KEY REFERENCES Android_Business(id), Shopping_id int FOREIGN KEY REFERENCES Android_Shopping(id), OS_name VARCHAR(30)"
if ($conn->query($sql) === TRUE) {
```
echo "Table created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

The following is the INSERT script to populate the created Android_Category_list dimension table based on the Request object. This table contains list of chosen categories from the Android OS and this further refers to six lookup tables.

//insert in to table Android_category_list
$sql = "INSERT INTO $Android_category_list (OS_name) ($Request.OS_name)"
    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error inserting record: " . $conn->error;
    }

4.1.7 Fact Table

The Fact Table is created by joining all dimension tables. The following CREATE script is for Fact table creation with the table name “Search_OS” joining with dimension tables of Iphone_category_list and Android_category_list. Finally the script verifies the creation of the table and let the developer know about the success or failure of the table creation.

//create Fact table
$sql = "CREATE TABLE IF NOT EXISTS $Search_OS ()
LEFT JOIN Iphone_category_list i ON i.OS_Name = $Request.OS_Name
LEFT JOIN Android_category_list a ON a.OS_Name = $Request.OS_Name GROUP BY Iphone_category_id, Android_category_id"
    if ($conn->query($sql) === TRUE) {
        echo "Table created successfully";
    } else {
        echo "Error creating table: " . $conn->error;
    }
The Fact table is created by joining all the dimension tables (Iphone_Category_list table, Android_Category_list table) and grouping them by dimensions (Iphone_category_id, Android_category_id).

After the fact table is created, it should be altered to add the foreign key indices and constraints to the dimension tables. The following is the ALTER table script:

```sql
$sql = "ALTER TABLE 'OS' 'Search_OS'
ADD INDEX 'Iphone_id' ('Iphone_Category_id' ASC),
ADD INDEX 'Android_id' ('Android_Category_id' ASC)"
if ($conn->query($sql) === TRUE) {
    echo "Table Altered successfully";
} else {
    echo "Error Altering table: " . $conn->error;
}
$sql = "ALTER TABLE 'OS' 'Search_OS'
ADD CONSTRAINT 'Iphone_Category_list'
    FOREIGN KEY ('Iphone_Category_id')
    REFERENCES 'OS'. 'Iphone_Category_list'('Iphone_Category_id')
    ON DELETE NO ACTION,
    ON UPDATE NO ACTION,
ADD CONSTRAINT 'Android_Category_list'
    FOREIGN KEY ('Android_Category_id')
    REFERENCES 'OS'. 'Android_Category_list'('Android_Category_id')
    ON DELETE NO ACTION";
```
4.2 Implementation of Application User Interface

The user interface for the AIC project is designed specifically for the two types of users:

1. Consumer user: Those who needs top rated applications from a favorite store to evaluate and get details of the application. These users may also want to see the feature difference between various applications and also may want to evaluate the OS (iOS or android) for their next phone purchase. They may also want to download an application for their device.

2. Developer user: Developer community is more interested in what type applications are out there in the market and how one can introduce better application. AIC project may be useful to do market trend analysis, application analytics, advertising, brand protection and search results etc. The application may be helpful in learning and understanding the data modeling, dynamic content update, UI & web development etc.

For a better idea on how the user interface looks like, the following sections includes all the different screens possible in the application. The implementation of the application is also explained as well.

All the presentation layer pages are implemented in HTML with support of PHP, JavaScript and JQuery for complex operations. Charts are developed using JS_chart API. The UI component style definitions are passed through a CSS file.
4.2.1 Welcome Page

Figure 11 shows the welcome page where user can understand the application overview and potential usage scenarios of the application. User can navigate to the different pages like Project Overview, Help, Feedback, Contact us etc.

![Welcome Page](image)

Figure 11. Welcome page

4.2.1a Project Overview

Figure 12 illustrates the project overview and also explains the usefulness of the project for developer by describing briefly about Data mart design, data processing (Extract, Transform, Load), performance and Dynamic UI content.
Welcome to the Project overview

There are two types of users of this project, (1) a consumer user who needs top rated applications from a favorite store. (2) Developer user who may be hopeful of learning and understanding Data modelling, dynamic content update, UI & web development e.t.c.

Figure 12. Project overview page

4.2.2 Help Page

Figure 13 explains the flow of the AIC project and help information for navigating through the application. It also has several use case reports.

The following scenarios can be used as help tutorial for the one-stop mobile application store project.

(1) To get the details of an IOS application.
   a. From home page, go to Iphone tab.
   b. Choose category of interest for list of top applications (updated daily) from store. Click on category of interest. Eg: click “Books and Reference”
   c. Select any of the top applications of the day from drop down list and click submit. Eg: Select “The Holy Bible” and click submit.
   d. Now one can find the table of information specific to the application.

Figure 13. Help page

4.2.3 Feedback Page

Users can provide their comments/feedback about the application through the feedback page as shown in Figure 14.
4.2.4 Contact Us

Figure 15 details the contact information.

Name: Harini Chilamantula
California State University, Sacramento.
Zip code: 95816
Email: harinichilamant@csus.edu

4.2.5 Index Page

Index page is the main page of the AIC project and it has five tabs to navigate. Depending on user choice, the slide switch gets activated and the selected tab is displayed.
The slider mechanism allows rich display format. It is implemented in JQuery with CSS style sheet.

4.2.6 Iphone tab

Figure 16 illustrates how to choose a particular category from the list of categories of App Store.
4.2.7 Iphone App Select

Figure 17 illustrate how the page displays the top most 100 applications in a dropdown list box and allow user to select one of them to explore the details. The list is updated every 24 hours to get the latest top applications. After selecting the application user has to click the submit button.

![Iphone Application selection page](image)

**Figure 17. Iphone Application selection page**

4.2.8 Iphone App Result

After submitting Iphone App Select, it triggers the AIC project in getting the requested data. Figure 18 shows how it displays the data fetched by the tool.

![Iphone Application results page](image)

**Figure 18. Iphone Application results page**
4.2.9 Android Tab

Figure 19 describes about the Android Play store and allows user to choose a category from the list of categories.

4.2.10 Android App Select

Figure 20 shows how the page displays the top most 100 applications in a dropdown list box and allow user to select one of them to explore the details. This list is updated every 24 hours to get the latest top applications. After selecting the application user has to click the submit button to navigate in to details page.
4.2.11 Android App Result

After submitting Android App Select, it triggers the AIC project in getting the requested data. Figure 21 shows how the page displays the data fetched by the tool.

4.2.12 Compare Tab

Figure 22 shows the compare tab which is important where user can compare any number of applications from any of the store and from any categories. Once the category is selected from the drop down list, the appropriate App Selection becomes active.
After selecting the category and app, user has to click the appropriate “add to list” to add it to the compare list. The above steps can be repeated as many times as the user wants to add any number of applications. Once done adding all the needed applications to the compare list, clicking submit button will get comparison result as shown in Figure 23.
4.2.13 IPhone Vs Android Result

Figure 24 shows the comparison tables of application details and graphical charts for some attributes.
4.2.14 Chart Analysis

Some attributes are also analyzed using graphs as shown in Figure 25.
Figure 25. Comp results page with comparison graphs

4.2.15 Case Study Tab

Figure 26 shows a Case Study of comparing applications from multiple stores/categories scenario.
Figure 26. Use Case study of compare.

4.2.16 App Store Data Mart

Figure 27 shows the details of the schema used in designing the Datamart of the project.
As this project provided the details of the implementation, the next chapter provides some use case scenarios.
5. USE CASE STUDY

This chapter describes different type of users of the project and so the different perspectives. One way of describing the use case scenario is UML. The use case diagrams describe the actors and actions for the application. It is a standard diagram within specification of the UML.

The three different use case models are that provides different perspectives are

1. Consumer use case
2. Developer use case
3. System use case

5.1 Consumer Use Case

In this model the actor is a consumer and actions are “Searching Top Apps”, “App Compare”, “App details” as shown in Figure 28. This model represents the majority of the use case scenarios.

Figure 28. Consumer User perspective UML
5.2 Developer Use Case

In this model the actor is a Developer and the actions are “App Discovery”, “App Analytics”, “App Details”, “App Compare”, “Chart Analysis” as shown in Figure 29. These use cases are very important for the developer to stay up to date with information of the latest and greatest market applications, the consumer interest and the market trends etc.

![Developer User UML Diagram](image_url)

Figure 29. Developer User UML
5.3 System Design UML

Figure 30, System design UML provides the system perspective. The System here is the actor and the actions are “Decode Request”, “Comp Analysis”, “Fetch Updates”, “Data Extraction”, “Data Transformation”, ”Data Load”, ”Render”.

5.4 Use Case Report

The following use case reports details various comparisons can be done in the AIC project.

5.4.1 IOS App Comparison

[Step 1] OS selection: Choose iOS App related fields on the left part of the page
[Step 2] **Category Selection**: Select “Books and Reference” from the list of available categories.


[Step 4] **Add App to be filtered**: Click the Button named “Add to iOS list”

[Step 5] **Repeat Apps Selection**: Select an app “Wattpad” from the list of available Applications.

[Step 6] **Add App to be filtered**: Click the Button named “Add to iOS list”

[Step 7] **Repeat Category Selection**: Select “Entertainment” from the list of available categories.
[Step 8] Repeat Apps Selection: Select an app “Netflix” from the list of available Applications.

[Step 9] Add App to be filtered: Click the Button named “Add to iOS list”

[Step 10] Verify app selection: Check the “Selected Apps” list box to verify if the selected apps are added to this list.

[Step 8] Submit: Click the “Submit” button to be navigated to results page.

[Step 9] Verify the list results: Verify the selected apps are listed in a tabular result list with the corresponding app related information like version, release date, file size, rating and download URL are listed.
[Step 10] **Verify download link:** When the user clicks a download link, user should be redirected to the appropriate market place. Selecting the download link in first row must redirect user to app location in app store.

![Details of all the selected Application table](image)

[Step 11] **Verify accuracy information:** Verify the app information listed in the Search application is accurate as listed in the market place.
[Step 12] Rating Comparison: The rating of each application is captured in a graphical format for user to do a side by side comparison of apps belonging to the selected category and OS.

[Step 13] Size Comparison: The size of each application is captured in a graphical format for user to do a side-by-side comparison of apps belonging to the selected category and OS.
5.4.2 Android App Comparison

[Step 1] **OS selection**: Choose Android App related fields on the right part of the page

[Step 2] **Category Selection**: Select “Books and Reference” from the list of available categories.

**Android**

Category selection: [Books & Reference]

[Step 3] **Apps Selection**: Select an app “Audible for Android” from the list of available Applications.

App selection: [Audible for Android]

[Step 4] **Add App to be filtered**: Click the Button named “Add to Android list”

[Step 5] **Repeat Apps Selection**: Select an app “Wattpad – Free Books & Stories” from the list of available Applications.
[Step 6] **Add App to be filtered**: Click the Button named “Add to Android list”

[Step 7] **Repeat Category Selection**: Select “Business” from the list of available categories.

**App selection**: [Wattpad - Free Books & Stories]

[Step 8] **Repeat Apps Selection**: Select an app “Business Motivation Quotes” from the list of available applications.

**App selection**: [Business Motivation Quotes]

[Step 6] **Add App to be filtered**: Click the Button named “Add to Android list”

[Step 7] **Verify app selection**: Check the “Selected Apps” list box to verify if the selected apps are added to this list.
[Step 8] Submit: Click the “Submit” button to be navigated to results page.

[Step 9] Verify the list results: Verify the selected apps are listed in a tabular result list with the corresponding app related info like version, release date, file size, rating and download url are listed.

[Step 10] Verify download link: When the user clicks a download link, user should be redirected to the appropriate market place. Selecting the download link in first row must redirect user to app location in app store.
[Step 11] **Verify accuracy information**: User can verify the app information listed in the Search application is accurate as listed in the market place using the URL.

[Step 12] **Rating Comparison**: The rating of each application is captured in a graphical format for user to do a side-by-side comparison of apps belonging to the selected category and OS.

[Step 13] **Size Comparison**: The size of each application is captured in a graphical format for user to do a side-by-side comparison of apps belonging to the selected category and OS.
5.4.3 IOS Apps Vs Android Apps

A user can also compare different application from different categories and different OS.

[Step 1] Category Selection: Select “Books and Reference“ from the list of available categories from both the OS.

[Step 3] Apps Selection: Select an app “Wattpad” from the list of available Applications.

[Step 4] Add App to be filtered: Click the Buttons named “Add to iOS list” and “Add to Android list“.
[Step 5] Repeat Category Selection: Select “Games” from the list of available categories.

[Step 6] Repeat Apps Selection: Select an app “Solitaire” from the list of available Applications.

[Step 7] Add App to be filtered: Click the Buttons named “Add to iOS list” and “Add to Android list”.

[Step 8] Verify app selection: Check the “Selected Apps” list box to verify if the selected apps are added to this list.

[Step 9] Submit: Click the “Submit” button to be navigated to results page.
[Step 10] **Verify the list results:** Verify the selected apps are listed in a tabular result list with the corresponding app related info like version, release date, file size, rating and download url are listed.

![Tabular result list](image)

[Step 10] **Verify download link:** When the user clicks a download link, user should be redirected to the appropriate market place. Selecting the download link in first row must redirect user to app location in app store

![Download link](image)

[Step 11] **Verify accuracy information:** user can verify the app information listed in the Search application is accurate as listed in the market place using the URL.

[Step 12] **Rating Comparison:** The rating of each application is captured in a graphical format for user to do a side-by-side comparison of apps belonging to the selected category and OS
[Step 13] Size Comparison: The size of each application is captured in a graphical format for user to do a side-by-side comparison of apps belonging to the selected category and OS.

Now that use case reports have been detailed, this report is summarized in next chapter.
6. CONCLUSION

6.1 Conclusion
While the hardware and OS played important role in success of the smartphone, the most key aspect of the success of the mobile revolution is App availability through secure and reliable application stores (Apple’s App Store and Google’s Play Store). As with any technology revolution, there occurs an inflection point at some point due to which the momentum saturates. This usually happens if the consumers do not realize any added benefits with technology upgrades or the industry doesn’t put diligent effort in educating the consumer about the benefits of upgrades. It is concerning that we are closer to that inflection point in smartphone due to the number of applications being more than 3 million and it is a daunting task for the consumers to pick few applications of interest from such large database.

This project tries to solve some of the above problems by providing capabilities such as (a) continuous (every 24 hours) update of top ranking applications in each store and category, (b) App discovery and details, (c) App search results, (d) App analytics, (e) Market trend analysis, (f) AIC project for all the reputed stores etc. Further the project optimizes the database performance by using innovative techniques of query consolidation and time based data fetch and update.

Another key goal of the project is to showcase and demonstrate the capabilities of Data-Warehousing when the data is being sourced from multiple sources and formats. During this modern age computing, the amount of data is huge. Efficient data extraction, transformation and loading is required for data warehousing. This project demonstrates the ETL process capabilities to extract both JSON and google protocol buffer data formats from App Store and Play Store respectively. Such capabilities of data warehousing is not limited to App stores but can be used in
wide variety of day-to-day life. Eg: stock market analysis, ground water analysis, commodity price analysis, census and social demographic analysis etc.

Various technologies has been employed to realize the presentation layer, application layer and the DataMart design such as PHP, HTML, JavaScript, JQuery, SQL, MySQL, XAMPP, NetBeans etc.

6.2 Future Work

The AIC project demonstrates capabilities of data warehousing and how it helps in analytics. There is huge scope of expansion in the future such as adding more stores (like windows, tizen etc.) or more analytic capabilities (like historic app ranking capabilities) or over the air download etc. Further new analytic features such as pie-chart, curve chart, s-curves, moving averages can be implemented in comparative analysis. New features such as personalized accounts, personal app stores could be expanded.

The potential expansion is not just limited to mobile application related features but there are wide varieties of real life applications where this type of project can be employed. Examples of such applications are petroleum reserve analysis at various off shore/on shore sites, stock market analysis and trend, census and demographics analysis, scientific and space applications where the source data tends to be huge and complicated.

With so much potential in data warehousing, we are hopeful that there is continued research in this domain to further improve the capabilities. Example of such research could be specialized systems with database specific features implemented in hardware, or OS improvements for data operations, or advanced compression techniques to minimize the database size and bandwidth etc.
Finally, I am fascinated how the Data warehousing projects helps in improving the life of human kind and hoping that others are inspired as well.
REFERENCES

[1] Wikipedia. PHP, available on


[3] jQuery. What is jQuery, available on


[7] MySQL. What is MySQL, available on


