Kodai: A Software Architecture and Implementation for Segmentation

Rick Rejeleene
University of Rhode Island, rrejeleene@cs.uri.edu

KODAI: A SOFTWARE ARCHITECTURE AND IMPLEMENTATION FOR SEGMENTATION

BY

RICK REJELEENE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2017

MASTER OF SCIENCE THESIS

OF

RICK REJELEENE

APPROVED:

Thesis Committee:

Major Professor:	Joan Peckham
	Lisa DiPippo
	Ruby Dholakia
	Nasser H Zawia
	DEAN OF GRADUATE COMMITTEE

UNIVERSITY OF RHODE ISLAND

2017

ABSTRACT

The purpose of this thesis is to design and implement a software architecture for segmentation models to improve revenues for a supermarket. This tool supports analysis of supermarket products and generates results to interpret consumer behavior, to give businesses deeper insights into targeted consumer markets. The software design developed is named as Kodai. Kodai is horizontally reusable and can be adapted across various industries. This software framework allows testing a hypothesis to address the problem of increasing revenues in supermarkets. Kodai has several advantages, such as analyzing and visualizing data, and as a result, businesses can make better decisions. In addition to these advantages, Kodai is open-source, which means any developer can access the code, and develop into client requirements. With the described features, it is better than other similar tools such as Gephi, a free visualization and manipulation tool.

The retail industry has grown exponentially, resulting in increasing demand for software tools to analyze consumer behavior. The analysis of consumer behavior helps businesses to stay at the forefront of market competition and provide excellent service. By focusing on consumer purchase behavior, Kodai can perform analyses, meaning it can classify consumers based on variables that capture their behavior. An example is identifying consumers who spend the most amount of money in a supermarket.

Segmentation models provide qualitative and quantitative methods to improve service for the customer and revenues for the company. These models can be used in different

fields such as finance, education and healthcare. Another important feature of Gephi is its interactive and visual modeling capabilities to help understand consumer behavior. Additionally, the software is reusable and supports the integration of future tools, following key extensibility concepts of software design.

This thesis explains the implementation of Kodai as a software architecture through segmentation models using a web-based application that implements software engineering methodology to improve revenues and consumer experience. This tool is developed to facilitate segmentation of consumer data based on purchase behavior with the goal of allowing the user to test a hypothesis to address the problem of increasing revenues in supermarkets. Most importantly, the software is reusable and can be adapted horizontally across various industries.

ACKNOWLEDGEMENTS

I would like to give sincere thanks my advisor Dr. Joan Peckham, who allowed me to work with complete independence in developing Kodai, a segmentation software on this research. I would like to give thanks to the caring community of Bridges International, a student club, who were like family to me. I wish to express my deepest appreciation to Carla White, who cared for me and taught me about family. This research would not be possible but for her. I wish to thank Romine Brooks for his constant encouragement. I would like to thank Dr. Ruby Dholakia for agreeing to review my work. I would also like to thank Dr.Lisa DiPippo for agreeing to review my work. Finally, I wish to thank Dr. Richard Vaccaro for agreeing to be my defense chair in a short possible time.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OUTLINE OF KODAI

The focus of this thesis is the design and implementation of Kodai as a software architecture for segmentation models to improve revenues for a supermarket. Kodai runs as a web-based application, using the Django Framework [1]. Web-based applications are software, that can run in any browser with client and server architecture. The Django Framework allows for the creation of fast, reusable software. The developed software is named as Kodai[a]. Kodai is open-source software, horizontally reusable and can be adapted across various industries such as education, agriculture, and finance. It allows for the testing a hypothesis to address the problem of increasing revenues in supermarkets.

Gephi [2], a visualization and manipulation software, is constrained by random access memory in a system. It takes an average of 10 minutes to import a file size of 141 megabytes. Meanwhile, Kodai takes only an average of 4 minutes to index into Elastic Search (see Section 3.10). Gephi is not able to test a hypothesis to address the problem of increasing revenues, thus Kodai is more efficient and reusable to better to test our hypothesis.

yED [3] is a powerful diagram editor that can be used to create diagrams from manually exported data for analysis and arrange large data sets by using a simple button. It provides an extensive class library for analysis, visualization graphs and network diagrams. Common tasks that can be accomplished by yED are visualization, creating and

[a] Kodai is named after famous hill-station Kodaikanal, Tamil Nadu. Kodai means umbrella in Tamil Language.

editing of graphs. Unfortunately, yED cannot meet the requirement of uploading raw data, or the testing of a hypothesis such as one related to increasing revenues through examining segments of data.

Raw Graphs [4] is an open source visualization framework with the goal of making visualization of complex data easy for everyone. It aims to provide a link between spreadsheet and vector graphics editors. Although it is an open source tool, that can handle raw file uploads, it is not able to handle file sizes above a gigabyte. It cannot help in testing of a hypothesis of improvement about revenues. Kodai is able to able to handle large file sizes, and can support testing a hypothesis of increasing revenues through examining segments of data.

Tableau [5] is a data analysis and visualization software product from Stanford University that allows users to drag and drop to analyze data. In a few clicks, users can connect data to the software and create data visualizations. Although Tableau has state of the art features such as vizQL, which translates drag and drop data into a visualization, it is not open-source. Kodai is open-source.

This is an interdisciplinary project that contributes to the disciplines of software engineering and marketing. Many organizations are obsessed with big data, as leading companies are now able to characterize people simply by observing their behavior. In a market analysis environment, the starting point is frequently the introduction of a web application to analyze different segments from collected data. Market segmentation identifies patterns of differences among groups responding to communications, products, and services [6]. The assumption is that if segments can be identified, described and reached selectively and efficiently, then an organization may increase sales and profits, and improve customer experience. Segmentation categorizes people based on a range of variables that allows for analysis of groups of people. The most frequently used variables are drawn from demographics, behaviors, or benefits sought by the customer. Kodai is empirically driven and based on segmentation models specified by the market analyst. The aim is to segment consumers based on purchase behavior and to identify customers who exhibit similar purchase behaviors.

The above is the central basis for all segmentation. Segmentation can be broadly divided into two different classes [6]: a priori segmentation and post hoc segmentation. A priori segmentation involves selecting certain groups from a population. Pre-determined segments are defined by demographics, psychographics or some readily observable behavior such as consumer spending. Post hoc segments intend to identify and classify segments based on actual market investigation and analyses of particular answers to survey questions. Because this software measures only purchase behavior, it can be classified as a priori segmentation.

Our a priori approach is implemented when the consumer data divides the market population into two or more groups. Pre-determined or a priori segmentation involves selecting categories in data based on the business requirements. By selecting certain pre-determined groups, we create segments in the dataset. In Kodai, pre-determined segments are selected based on the consumers who have spent the most, revenues of consumers, products, and coupons redeemed.

In Kodai, we do not use Post-hoc segmentation, as it is used for introducing new products in the market. The Post-hoc method tries to identify segments based on actual investigations, particularly using analysis of answers to survey questions intending to predict marketplace responses. Rather than shaping a product to serve existing consumer behavior as in a priori segmentation, posthoc segmentation is useful for introducing new products based on consumers' opinions. The following gives an outline of selecting segments in a dataset:

1.2 SEGMENT: PRODUCT SELECTION BEHAVIOR

a) Usage rates and occasions — This contains the number of times a consumer visited a supermarket and the number of products purchased.

b) A Number of different brands used regularly — Consumers selection of brands. We have observed that frequently people are loyal to the same brand.

1.3 SEGMENT: OTHER BASES

a) Stage in life cycle – Different stages include, single, married and retired.

b) Socio-economic status – Income demographics of consumers

c) Other demographics – Household size of consumers

It is a well-known maxim in business that 20% of consumers provide 80% of sales [7]. In reference to this statistics, Kodai's goal is to identify areas of growth. It identifies consumers who are increasing their purchases within a two-year period of business. Kodai accomplishes it by building an index from supermarket data through Elastic Search. Elastic Search is a software tool, that allows Kodai to send queries into built indexes [8]. These indexes provide the efficiency that other products are unable to provide (See Section 1.1)

The main work of this thesis is outlined as the following:

- Applying software architecture by building a web-based tool for segmentation, focusing on the architecture and software development methodology for the tool that supports the analysis of consumer behavior patterns.
- Implementation of the framework for the web based tool.
- Analysis of the software to test hypotheses regarding consumer buying pattern.

The remainder of this thesis is composed of three chapters

- First, we discuss background information regarding segmentation models, pricing and related work in order to achieve maximum comprehension for readers of any background.
- Next, the thesis discusses key parts of the implementation. This portion includes details about the software architecture, the software framework, the software methodology, and the technologies used to support segmentation.
- Finally, we discuss the results generated by the tool — all code is uploaded to GitHub [9], is a repository and internet hosting service. The chapter ends with concluding remarks and thoughts for future research.

List of References

[1] Holovaty, A., & Willison, S. (2008, June 4). Django. *Django at a glance.* Retrieved January 22, 2017, from https://docs.djangoproject.com/en/1.9/intro/overview/

[2] Bastian, M. (2008, July 31). Gephi. *Gephi Quick Start.* Retrieved January 22,2017, from *https://gephi.org/users/quick-start/*

[3] yWorks GmbH 2016 (2016, March 24). yWorks, Documentation. Retreived January 22, 2017, from http://www.yworks.com/products/yfiles/documentation

[4] Caviglia, G., Mauri, M., Uboldi, G., & Azzi, M. (2014, January 6). About Raw Graphs. Raw Graphs. Retrieved January 22, 2017, from http://rawgraphs.io/about/

[5] Tableau Corporation. (n.d.). Business Intelligence and Analytics | Tableau Software. Retrieved from https://www.tableau.com

[6] Struhl, S. (2013). *Market Segmentation: An Introduction and Review*. Create Space Independent Platform Publishing.

[7] Newman, M. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics,* 323-351.

[8] Banon, S. (2010, February 21). Elastic Search. *Preface.* Retrieved January 22, 2017, from https://www.elastic.co/guide/en/elasticsearch/guide/current/_preface.html

[9] Preston-Werner, T., Wanstrath, C., & Hyett, P. (2008, April). Github. *Github Help.* Retrieved January 22, 2017, from https://help.github.com/

CHAPTER 2

2.1 BUSINESS BACKGROUND READING

In this chapter, we explain the basic background material and tools necessary for the reader. Understanding the business requirements will help the reader to gain comprehension of the requirements for Kodai. Next, we discuss a high-level view of business concepts, and tools needed for Kodai. This involves segmentation models, the supermarket dataset, and software process models.

A Segmentation model is an abstract defined model created in Kodai based on patterns of differences in purchasing behavior of consumers. The Supermarket dataset that we used here contains raw data collected from a supermarket [1]. Kodai is applied and tested using this supermarket dataset. This chapter contains an outline of segmentation models as they pertain to some aspects of prices and coupons.

2.2 BUSINESS REQUIREMENTS

As this thesis is an interdisciplinary project, it requires expertise in multiple disciplines. Kodai must be able to do the following to satisfy the business requirements:

- The software should be able to quickly identify top consumers of a supermarket from any demographic.
- The software should be able to determine the households that spend the most in a supermarket.

- The software should be able to identify coupons with the highest number of consumer redemptions.
- The software should allow the developer and other users to test out hypotheses to improve revenues.
- The software should be able to run on any operating system, including Windows, Mac, Linux.
- The software should be open source and able to handle at least a gigabyte of raw data.
- The software should be extensible to other similar segmentation applications.

2.3. SEGMENTATION

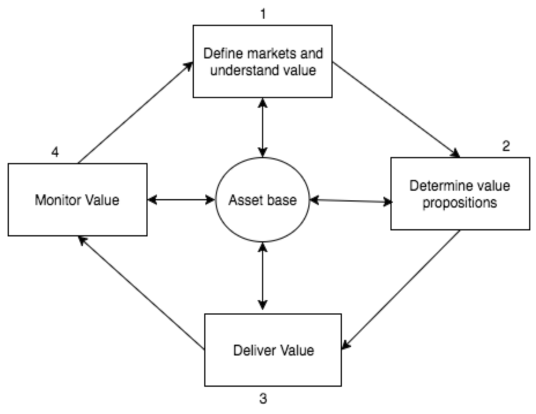

1
Define markets and understand value
2
Determine value propositions
3
Deliver Value
4
Monitor Value
Asset base

Figure 1. How marketers define segmentation [2]

Segmentation models help us to understand consumer behavior, allowing businesses to improve products and services for consumers.

In Figure 1, marketers define markets and try to understand the value of markets. Next, they determine value propositions, which can be benefits sought out by consumers. These values are delivered and monitored creating an asset base. An asset base refers to the important value created through markets. One example of an asset base could be brand recognition from consumers. For example, brands such as Apple, Microsoft, Google have a strong brand recognition that creates consumer loyalty to specific brand.

Segmentation as a concept did not appear until 1956. The most influential discussion of market segmentation appeared in an article by the former president of the American Marketing Association, Smith, addressing product strategies and the use of their application. In his paper, Wendall Smith rejected the classical economic theory of perfect competition [3]. Perfect competition is defined by market holding to conditions such as perfect information about buyers, well-defined property rights, and profit maximization of sellers. Using variety became the norm of contemporary markets. Wendall Smith said segmentation worked more efficiently than a strategy of maximizing output or simply producing as many products as possible.

Using a differentiation strategy, the manufacturer would try to make something for everybody, without an in-depth study of any particular group within the market. Meanwhile, Smith compared product differentiation strategies as trying to take a layer of a marketing pie-chart and segmenting it into slices. A truly successful organization must find segments and then create products and services fitting their needs rather than creating consumer needs or demands.

Segmentation is defined as patterns of differences among a group's responses to communications, products, and services. Here we consider responses to purchases in a supermarket and raw data that we assume is static. The key idea is that different groups have different patterns of responses in a supermarket environment. These distinctions are inferred from analyzing the supermarket dataset, thus following a priori, empirically-driven segmentation. Segmentation is broadly classified into apriori and posthoc model. We briefly discussed these two models of segmentation in Section 1.1.

2.4 PRICE

In this chapter, we give a brief outline of price and sample price as defined in the supermarket dataset used by Kodai. In the book Pricing and Revenue Optimization [4], a historical example demonstrates the importance of price in the business. During the rule of the Dutch Republic, speculative bubbles such as "Tulipomania" caused prices of tulips to rise more than a hundredfold within 18 months. This begs the question, "What were prices exactly? And how are they defined?" We define price not as intrinsic but rather based on what consumers perceive: supply and demand. Milton Friedman in his book, Price Theory, defines prices as not determined by any one individual firm, but rather is determined by the market [5]. The supermarket dataset used by Kodai does not contain the price of products, but we can easily configure Kodai to include sample price in the dataset to determine potential impact of price change on revenues. Revenues are incomes from all units sold in the dataset.

In the dataset, we target the top 20% of consumers for increasing revenues; by increasing the prices of products that have been sold the most, we are able to gauge potential revenue increase. We proceed as follows; we define a sample price for items in our dataset. We define a sample price increase variable in our software Kodai for items in supermarket. The number of items sold in the dataset is listed under quantity. We have sample revenue, and sample increased revenue. We can then calculate potential revenue increase.

2.5 COUPON

In marketing, a coupon is an incentive or ticket that consumers can use to get financial discounts for purchasing a product. Coupons are part of sales promotions. Coupons are likely to be redeemed by price sensitive consumers, thus software using this data can segment price-sensitive consumers. We assume that buyers, who collect coupons, are more price sensitive than buyers who do not collect coupons. Therefore, from our hypothesis of price sensitive consumers, it follows that consumers who do not collect coupons would not be affected by a small increase in prices. In Kodai, we find products that have been redeemed the most, and details about consumers who have redeemed coupons.

List of References

[1] Rajkumar, V. (2011, April 22). Dunnhumby - Customer Science & Consultancy for Retailers & Manufacturers (CPGs). *Source Files | dunnhumby.* Retrieved January 22, 2017, from http://www.dunnhumby.com/sourcefiles

[2] McDonald, M., & Wilson, H. (2016). *Marketing Plans: How to prepare them, how to profit from them*. Italy: Wiley.

[3] Smith, W. (n.d.). Marketing Masters. *Product Differentiation and Market Segmentation as Alternative Marketing Strategies.* Retrieved January 22, 2017, from https://archive.ama.org/archive/ResourceLibrary/MarketingManagement/documents/9602131166.pdf

[4] Phillips, R. (2005). *Pricing and Revenue Optimization*. Stanford Business Books, Stanford University Press.

[5] Friedman, M. (1962). *Price Theory*. Chicago: Aldine Transaction.

CHAPTER 3

3.1 SOFTWARE ENGINEERING BACKGROUND READING

This chapter contains a basic outline of the software engineering process and tools involved in the development of Kodai. We begin with an outline of the supermarket dataset, and the data format used. We also describe the software used to develop Kodai including the programming languages and applications as well as the software framework used. Also, Kodai is compared to Gephi, a visualization and manipulation software to help the reader understand, how Kodai is better and meets both software and business requirements (See Section 2.2).

3.2 SOFTWARE REQUIREMENTS

Software requirements are computing pre-requisites needed for the software to run on any computer and produce needed service to the customer or user. Kodai must be able to do the following to satisfy the computer science requirements:

- The software must be reusable and applicable across many industries. This is consistent with the principles of software engineering.
- The software must be extensible for future add-ons.
- The software should import data and analyze data according to the business requirements. The section below explains our specific dataset used for Kodai to function.

3.3 SUPERMARKET DATASET

In this section, we give a thorough explanation of the supermarket dataset and describe necessary details for the reader to understand it. First, Segmentation is created from raw datasets. We define data as a collection of values of qualitative or quantitative variables, which is measured, collected, analyzed and visualized using graphs, images or other analysis tools. In the supermarket dataset, data is information about habits of consumers based on demographics, revenues, coupons. This dataset contains supermarket household level transactions.

The below dataset represents household level transactions which were collected from over two years from 2,500 households. It contains details of household purchases such as unique id, age category, homeowner, household size. Coupon data provides information about specific coupon campaigns sent to households. The data is in CSV or comma separated files (see section 3.4). These files contain the following information described below: Figure 2 shows the files and their attributes and Table 1 lists the file names.

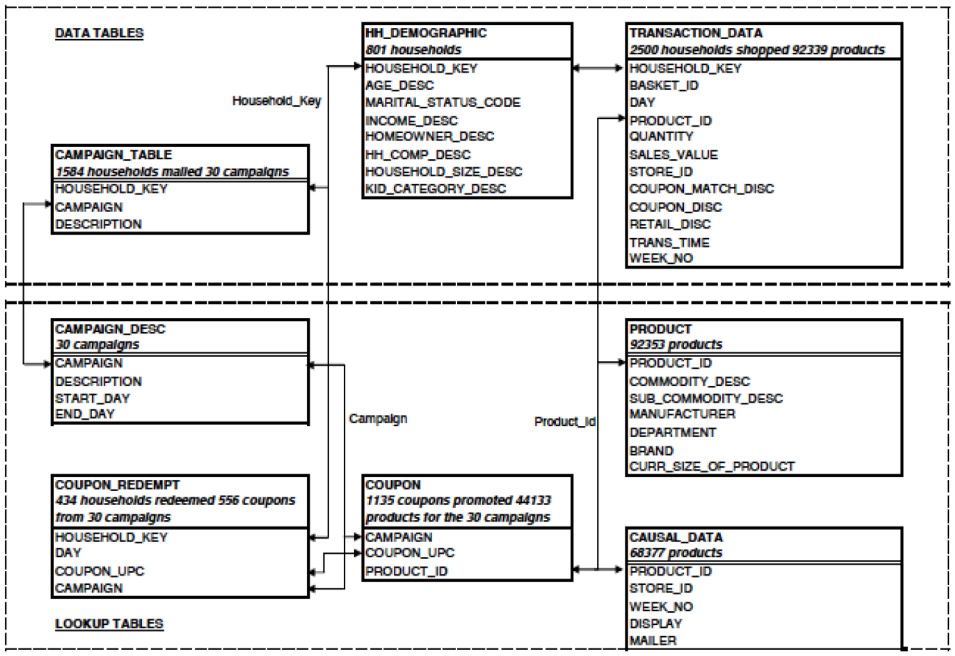


Figure 2. Supermarket dataset from households [1]

- CAMPAIGN_TABLE
- CAMPAIGN_DESCRIPTION
- COUPON_REDEMPT
- HH_DEMOGRAPHIC
- COUPON
- TRANSACTION_DATA
- PRODUCT
- CAUSAL_DATA

Table 1. Different dataset file names present household [1]

- The CAMPAIGN_TABLE file contains information about 1,584 households that received 30 campaigns via mail. A campaign occurs when the business owner decides to send coupons to select consumers. It lists the campaigns received by each household.

- The CAMPAIGN_DESCRIPTION file contains the campaign's running length of time. Any coupons received as a part of a campaign are valid within the dates contained in this table.
- The COUPON_REDEMPT file contains information about households that redeemed coupons sent by a supermarket.
- The HH_DEMOGRAPHIC FILE contains household demographics, which includes age, marital status, income, homeowner, and household size.
- The COUPON file contains all the coupons sent to customers as a part of a campaign. There are some products that are redeemable. The dataset lists different campaign names such as A, B, and C. A customer participating in Campaign A, for instance, might have received 16 coupons out of the pool.
- The TRANSACTION_Data file contains all products purchased by households within this study. It has a household key, basket ID (which identifies purchase occasion), day, product identification, quantity, sales value, and coupon match. All information pertaining to transactions are contained in this category.
- The PRODUCT file contains information on each product such as product identification, department, manufacturer, brand, and current product size.
- The CAUSAL_DATA file contains information about products that were displayed in a weekly mail or in-store display. All the above tables are organized and stored in comma separated file format.

3.4 CSV

Digital data is commonly stored in Comma Separated File (CSV). It stores tabular data in plain text [2]. Each line of the file is a data record separated by commas. All records have the same number of fields in the same order. Because most data processing applications use this format, this makes the data easy and straightforward to access & process by software applications. Kodai only uses CSV files for segmentation.

Year	Company	Car Model	Price
2004	Hindustan	Ambassador	3000.00
2008	BMW	B8	4900.00
2009	AUDI	A4	5000.00
2011	TATA	Indica	4799.00

Table 2. Comma Separated Value data format [2]

The data shown in Table 2 above can be represented in comma separated value format as shown in Table 3 below.

```
Year, Company, Car Model, Price

2004, Hindustan, Ambassador, 3000.00

2008, BMW,"B8", 4900.00

2009, AUDI," A4”, 5000.00

2011, TATA,” Indica”, 4799.00
```

Table 3. Sample example of comma separated value from table 2

Kodai takes CSV files as an input and the software process model helps the developer to take this file as an input, and process the data using Kodai.

3.5 SOFTWARE PROCESS MODEL

Every software needs thorough planning, and detailed design before a developer begins to code. Without a plan, developers cannot know the direction, and goal of their software. A software process models helps developers to systematize and plan their software development process. Software process model is an abstract representation of a software development process. It presents a description of the process from some particular perspective. We integrated two software models in developing this software, the waterfall model and an agile model. Kodai’s need for constant iterations required us to follow this hybrid agile development model.

3.5.1 WATERFALL MODEL

As software was being developed over time, each phase of software development was abstracted and systematized to build software faster and more efficiently. The waterfall model is the classical model of software engineering. The Waterfall model is a framework for software development process, where each phase of software development is developed in stages. It starts with the requirements phase and ends with maintenance. If requirements for the software are collected, this makes planning more efficient, and better quality software can be built. The Waterfall model does not allow for reflection or revision, thus there is high amount of risks and uncertainty. Due to need of iterative process in our prototyping phase, we integrated the iterative aspects of agile development. Kodai's software development process integrated waterfall and agile as hybrid agile waterfall model.

The Spiral software development model is a type of software process model. It starts with the following phases of development: identification of business requirements, design phase (involves architectural design), construction or building (involves production of actual software) and finally evaluation and risk analysis. Also, each step in the spiral process can be revisited, repeated to examine risks at each stage. It is perhaps too cumbersome for small software products, and is suited for medium to large scale software products. As Kodai is a small software framework, the spiral software process model was not chosen.

3.5.2 HYBRID - AGILE WATERFALL MODEL

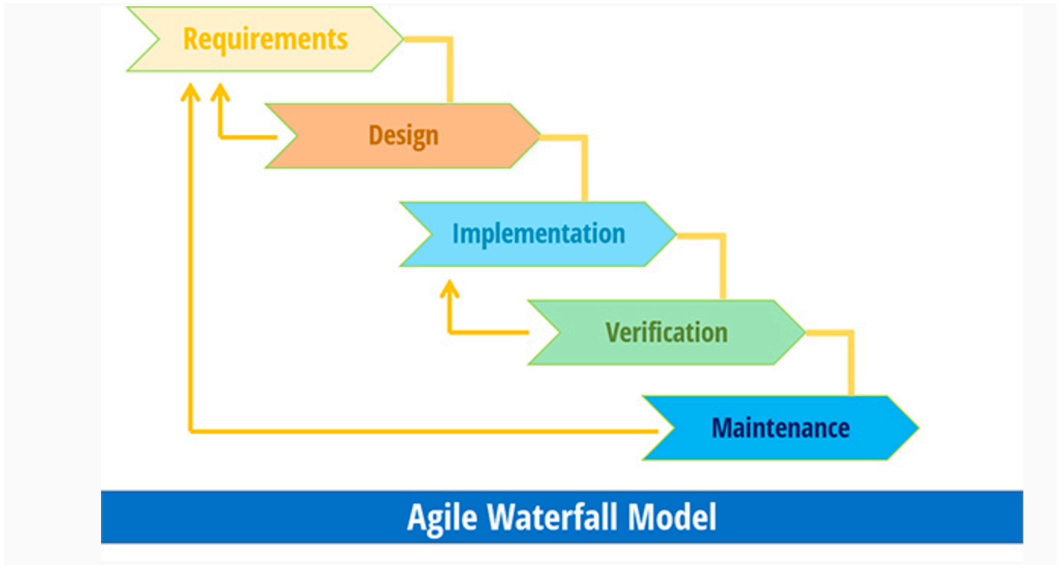


Figure 3. Phases in Hybrid Agile Waterfall model [3]

In the Hybrid Agile waterfall model, there's an iterative step at every step of waterfall methodology. We start with basic requirement identification; in Kodai, we identified that some of our product requirements need a framework to allow access to supermarket data; the user interface especially needs to be a web-based application that can be on a cloud. Next, the initial prototype consists of the development of basic requirements mentioned above in section 2.2. This includes user interfaces, with high-level functions such as an ability to view on a web browser, reusing software for different disciplines. In Kodai, we enable a given supermarket business to collect, analyze and better understand their consumers, as a result of capabilities of the software. Finally, revision and enhancing prototype feedback focus on reviewing comments to incorporate features into our new prototype, this includes functions to view the trend of consumer visits in a supermarket. Agile is ideal for new technology such as web application and flexibility for changes.

By using both the Waterfall and Agile methodology, we involve users of Kodai in the production process even before implementation. As the working model is displayed, the user gets a better understanding of the system being developed. Also, the waterfall models help to increase quality by catching possible design flaws at the testing stage.

3.6 GITHUB

After deciding on a software methodology, all software requires iterative updates and versions as software development takes place. In order to document and store all our code for Kodai, we need to use version control and a code hosting platform. We use Github for this purpose. GitHub [4] is a code-hosting platform for version control and collaboration. It lets software developers work together on projects from anywhere. A repository is usually used to organize a single project and contains folders, files, spread-sheets, and datasets. Kodai has all the necessary files stored in GitHub. The link to the software repository: https://github.com/ludwigwittgenstein2/supermarket_elasticsearch. We use GitHub to maintain version control, future development of Kodai, and a Web framework for developers to build efficient software.

3.7 WEB FRAMEWORK

In this section, we explain about Web framework, and the specific framework, Django used for Kodai to be developed. This allows the reader to gain sufficient knowledge in Web frameworks and Django. A Web framework is an implemented extensible abstrac-tion of software development tools required for developers to build web applications. A web framework encapsulates developers' experience from over twenty years [5]. It helps to support the development of web applications that are geared towards applications used by clients on the internet. These frameworks make it easier to reuse common HTTP operations and structure so that other developers with knowledge of a framework can quickly build and maintain the application. The common operations that can be performed with Django framework are session storage, database manipulation, security, URL routing, accessing JavaScript object notion. Session Storage and retrieval help developers store information about users' browsing activity, and later retrieve to help them identify unique users. Database manipulation helps developers to constantly update and remove data stored in a database. Security against cross-site request forgery helps to prevent common attacks to gain access to username and passwords within the web-application. URL routing lets developers quickly categorize URLs within Django. Django gives developers access to the above commonly performed operations.

3.8 DJANGO

In Django's documentation, the authors define, "Django as an open-source, high-level Python Web framework that encourages rapid development and clean, pragmatic design" [6]. Django allows us to create complex database-driven websites emphasizing reusability, plug ability of components, rapid development, and prevention of unnecessary repetition. Django is designed to help developers take an application from concept to reality in a short period of time. It includes dozens of options for handling common web development tasks such as URL configuration and security. We compare Django with FLASK, a Python web application development framework. Django has web modules built-in, thus making it faster and efficient to develop a web application development in comparison with the FLASK web development framework. We describe FLASK at the end of this section.

Django prioritizes security and helps developers avoid many common forms of cyber attacks having built-in security modules. It prevents cyber attacks such as cyber-intrusion, such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking. SQL injection is an injection technique using code to destroy the database. Cross-site request refers to applying injection attacks using code on a web application by executing malicious scripts, where an attacker can gain information. Clickjacking tricks users into redirecting into a malicious website to collect user information. Some of the busiest global sites use Django to scale to meet the heaviest traffic demands, including Pinterest and the Washington Post. Companies, organizations, and governments have used Django to build all sorts of things — from content management systems to social networks, and scientific computing platforms. Django was selected for this project because it exists to solve a real-world problem through development of powerful web applications. It also provides built-in security to prevent the above security issues.

Django arose out of the need to address inherently challenging deadlines in the news industry, which are often mere days, if not hours. As such, Django permits software developers to build large, high-quality sites far faster than earlier methods. A frequent saying among the Django community is, "If the Romans had used Django, they would have built Rome in a day" [6]. Every bit of the framework is designed with efficiency in mind. Because of all these advantages, we chose Django to build Kodai.

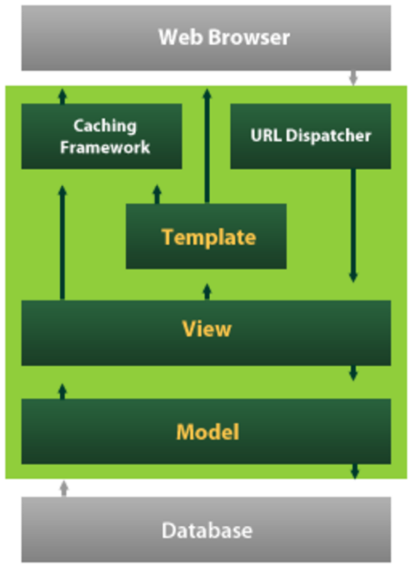


Figure 4. Architecture describing overall framework of Django [7]

Some features of the Django framework shown in Figure 4 are explained below. The URL dispatcher maps the requested URL to a view function and calls it. If caching is enabled, then the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version instead. The view function performs the requested action, which typically involves reading or writing to the database. It may include other tasks as well. The model defines the data in Python and interacts with it. Templates return HTML pages. After performing any requested tasks, the view returns an HTTP response object to the web browser

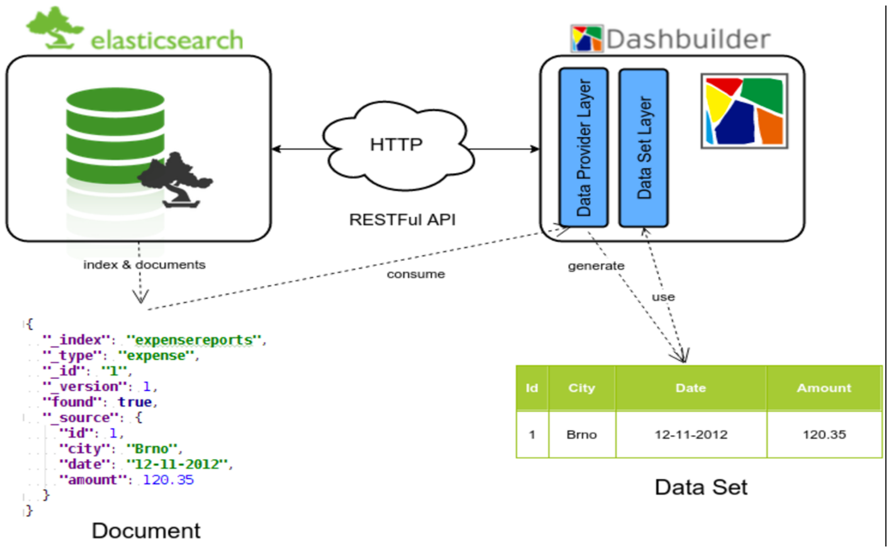
Flask is a micro-framework for Python [8]. It can be used to develop a web-based application, but it does not support Elastic Search. Django has the most active community, compared to Flask with more than 80,000 developers with blogs. The service provides a full-featured Model-View-Controller framework and could ostensibly even be used to make an extensible application. Django's REST framework generates pages to browse and execute all APIs. Thus, we can execute GETs and POSTs quickly and test it in the browser. Thus, we choose to use Django because it meets our requirements and allows us to use Elastic Search. It is written in the Python programming language.

3.9 PYTHON

Python is a high-level programming language that emphasizes code readability, meaning the syntax is closer to written English. Python also has a large and comprehensive standard library which includes an extensive documentation [9]. Python was chosen to develop Kodai because it complements the use of Django and Elastic Search. In Kodai, Python supports importing Django and Elastic Search library. Due to these reasons, Python is a more prudent choice than other programming languages such as Java, C++.

3.10 ELASTIC SEARCH

In the business requirements from Section 2.2, we understand that our software must be able to quickly find top consumers in a supermarket. In order to achieve this requirement, Kodai need tools to send queries to accomplish it. We use Elastic Search to transform raw data from supermarket to build as an index. Elastic search is a distributed search engine with a RESTful API. A distributed search engine has no central server, and query is distributed among several connected computers over a network. “RESTful API is a service that supports HTTP methods, to create, retrieve, update and delete access to service’s resources” [10]. It is used by developers to access information from a web-application. This information might include web application’s usage statistics, clicks, the number of users. Elastic search indexes raw data, and lets us perform queries and combine many types of searches, thus we use it to analyze our data to explore trends and patterns in our data. The software is distributed, which means that indices are divided into shards. A shard is a horizontal partition of data in a search engine. Related data is often stored in the same index. Below is an outline of Elastic Search's architecture:



```
{
  "_index": "expensereports",
  "_type": "expense",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "id": 1,
    "city": "Brno",
    "date": "12-11-2012",
    "amount": 120.35
  }
}
```

Id	City	Date	Amount
1	Brno	12-11-2012	120.35

Figure 5. Architecture of Elastic Search Data provider [11]

In Figure 5, Elastic Search starts from building an index connected to a dash builder using RESTful API. In Kodai, we use a head plugin as the dash builder that transforms supermarket data set. Kodai built in Django and Elastic Search allows the user to view the dataset, shown in the figure above. Some key features of the Elastic search are distributed and highly available search engine, support for more than one type of index, document oriented. Distributed search engine allows the user to search in real-time through indexes; indexes are stored set of information about data. Multiple indexes allow the developer to send real-time queries about the data. Document orientation refers to a top-down level that is stored as a JavaScript Object Notation in a unique ID; allowing Python to access our software. By using Elastic Search, we are able to build 1-gigabyte comma separated file of raw supermarket data into an easily accessible index. We use Elastic Search because it supports the Python language to send queries. In order for us to see sample results of Elastic Search, we use the Elastic Search Head Plugin.

3.11 ELASTIC SEARCH HEAD PLUGIN

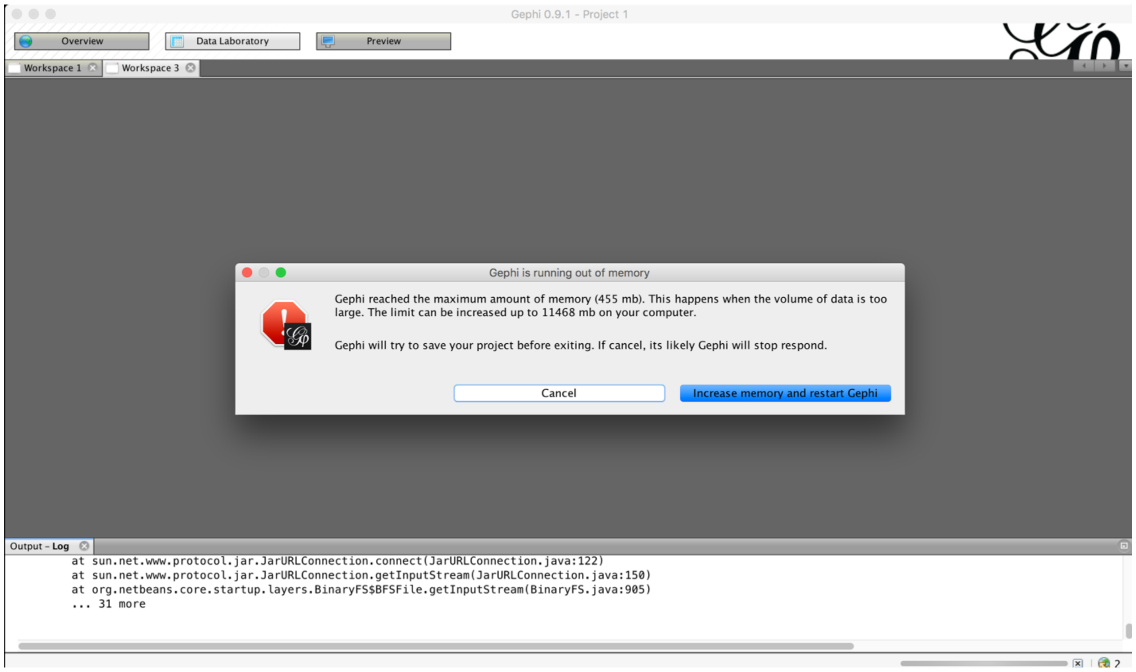
The Elastic Search head is a web frontend for browsing and interacting with indexed data [12], featuring major operations such as:

a) A Cluster Overview, which shows the index built within Elastic Search
b) Search interface, that allows us to query the cluster of indices and retrieve results in raw JavaScript Object Notation
c) Tabs that show statuses of clusters
d) An input section that allows an arbitrary call to RESTful API. RESTful API lets developers access information from the web application

We use Elastic search to index and view the data. Our main goal is to extract actionable knowledge. It helps to explore data in a short time and is capable of scaling petabytes of structured and unstructured data. By indexing, we can quickly access our data at high speed, without the need to create a database.

3.12 WHY KODAI IS BETTER

In this section, we first describe Gephi, a visualization software. Next, we compare Gephi to Kodai against the business requirements given above.



Gephi is running out of memory

Gephi reached the maximum amount of memory (455 mb). This happens when the volume of data is too large. The limit can be increased up to 11468 mb on your computer.

Gephi will try to save your project before exiting. If cancel, its likely Gephi will stop respond.

```
	at sun.net.www.protocol.jar.JarURLConnection.connect(JarURLConnection.java:122)
	at sun.net.www.protocol.jar.JarURLConnection.getInputStream(JarURLConnection.java:150)
	at org.netbeans.core.startup.layers.BinaryFS$BFSFile.getInputStream(BinaryFS.java:905)
	... 31 more
```

Figure 6 GEPHI uploading transaction file

Gephi is a visualization and manipulation software [13]. It is a tool for data analysts and scientists keen to explore and understand graphs. In addition to understanding graphs, Gephi is similar to Adobe Photoshop — but for graph data — the user can interact with representations, manipulate structures, and colors, as well as reveal hidden patterns. It can be used for exploratory data analysis and visual analytics.

In Gephi, the task of uploading data and manipulating data depends on the installation of the software. In the above Figure 6, Gephi is out of Memory while uploading transaction file. It cannot process SQL queries to manipulate data. Kodai is open-source and can run on a server without installation into a specific system. Let's consider TRANSACTION_DATA file from Figure 2; to upload this file to GEPHI, the average time is approximately 10 minutes, meanwhile, it takes approximately 4 minutes in Kodai to build it into the Elastic Search index.

A developer can use the same file, and index all files in Kodai using an elastic search, as well as manipulate transaction data and view the results. Gephi would not be able to meet the requirements to find top consumers in supermarket data.

3.12.1 FLEXIBILITY

Gephi has three panes, overview, data laboratory and previews to provide an overview of data laboratory data and previews to display the results. In order for developers to take features in Gephi and implement them according to their respective requirements, they would need to uninstall the software and change features within Gephi.

In Kodai, we can create our own flexible view of results by implementing different queries using Elastic Search without going through the process of the uninstallation of software. Thus, Kodai has more flexibility than Gephi.

3.12.2 MEMORY

Gephi depends on a local computer's random access memory to run the software. To upload a comma separated file of transaction data from supermarket data into Gephi, it takes an average time of 10 minutes to import the file than an average time of 3 minutes in Kodai. With Kodai, we can store data as an index that makes queries faster to access and visualize according to the requirements. Finally, we are also able to use Kodai through Amazon Cloud or other servers.

3.12.3 REUSABLE

Gephi can be used for a variety of different data but is not reusable for a specific purpose such as calculating top revenues, top products, and top coupons renewed. With our Supermarket data, Kodai able to calculate top revenues, top products purchased, and top coupons renewed. Therefore, Kodai reusable to meet both business and software requirements. But Gephi is not able to meet these requirements.

3.12.4 SCALABLE

Gephi is constrained by a limitation on large files. Kodai could be horizontally scalable by deploying an Amazon Cloud server and creating clusters within Elastic Search, which is part of our software. Due to Gephi's failure to satisfy these requirements, Kodai provides a more suitable framework for data analysis, and visualization to researchers, and businesses.

List of References

[1] Rajkumar, V. (2011, April 22). Dunnhumby – Customer Science & Consultancy for Retailers & Manufacturers (CPGs). *Source Files | dunnhumby.* Retrieved January 22, 2017, from http://www.dunnhumby.com/sourcefiles

[2] (2005, October). Wikipedia. *Comma separated values.* Retrieved January 22, 2017, from https://en.wikipedia.org/wiki/Comma-separated_values

[3] Semantics, B. (2010, May 1). Software Development Process. *Binary Semantics.* Retrieved January 22, 2017, from http://binarysemantics.com/images/agile_waterfall_model.jpg

[4] Hello World (2017, January 22). GitHub documentation [Website]. Retrieved from https://help.github.com/

[5] Makai, M. (n.d.). Full Stack Frameworks. *Web Frameworks.* Retrieved January 22, 2017, from Makai, Full Stack Python (2017, January 22). Documentation[Website]. Retrieved from www.fullstackpython.com/web-frameworks.html

[6] Holovaty, A., & Willison, S. (2008, June 4). Django. *Django at a glance.* Retrieved January 22, 2017, from https://docs.djangoproject.com/en/1.9/intro/overview/

[7] Crott, J. (n.d.). Django. *Tardis Documentation.* Retrieved January 22, 2017, from http://mytardis.readthedocs.io/en/2.5/_images/DjangoArchitecture-JeffCroft.png

[8] Ronacher, A. (n.d.). Flask. *Welcome | Flask.* Retrieved January 22, 2017, from http://flask.pocoo.org

[9] Van Rossum, G. (2004, March 22). Wikipedia. *Python History.* Retrieved January 22, 2017, from https://en.wikipedia.org/wiki/Python_%28programming_language%29

[10] Lamos, B., Pasic, A., Wells, J., & Xueyuan, H. (2017, July 3). Azure REST API Reference. *Microsoft Azure.* Retrieved June 20, 2017, from https://docs.microsoft.com/en-us/rest/api/

[11] Martinez, R. (n.d.). The Dashbuilder Project. *Dashbuilder.* Retrieved January 22, 2017, from http://dashbuilder.blogspot.com/2015/10/uf-dashbuilder-real-time-dashboard-with.html

[12] Birch, B. (n.d.). Elastic Search Head. *Documentation.* Retrieved January 22, 2017, from http://mobz.github.io/elasticsearch-head/

[13] Bastian, M. (n.d.). Home. *Gephi.* Retrieved January 22, 2017, from https://github.com/32ephi/32ephi/wiki

CHAPTER 4

ARCHITECTURE OF KODAI

In this section, we describe the architecture of Kodai. We begin with the general description of software architecture, then we show differences between an architecture and framework. Next, we describe the behavioral aspects of Kodai. Finally, we show top coupons, one feature of Kodai, to understand the implementation.

4.1 DIFFERENCE BETWEEN ARCHITECTURE AND FRAMEWORK

Software architecture refers to the structure of software solutions needed to solve technical and operational problems. In Kodai, architecture refers to the guiding principles and code components for applying segmentation models in improving revenues. The goal of a software architecture is to build a bridge between business requirements and technical requirements. In software architectures, the structure of the system is exposed, but implementation is hidden. Architectures are built to support change in the design, Kodai is built to support future changes, reusability and development of new features [1].

Software framework refers to a set of software libraries, to address a general domain purpose such a web application. A framework is an extensible implementation that can be used to solve problems as we build an application or system. In Kodai, Django is a web-application framework to implement web based tools, and applications.

Most complex systems need a solid foundation, likewise, Kodai requires solid foundation in software architecture. Failing to consider software architecture will likely create unstable software in the long-run. A Software Architecture allows extension of software to multiple other domains. It represents an abstraction of a system, that allows mutual understanding, negotiation, communication among software stakeholders. An architecture is transferable, and reusable for future implementations [2].

4.2 ARCHITECTURAL STRUCTURE OF KODAI:

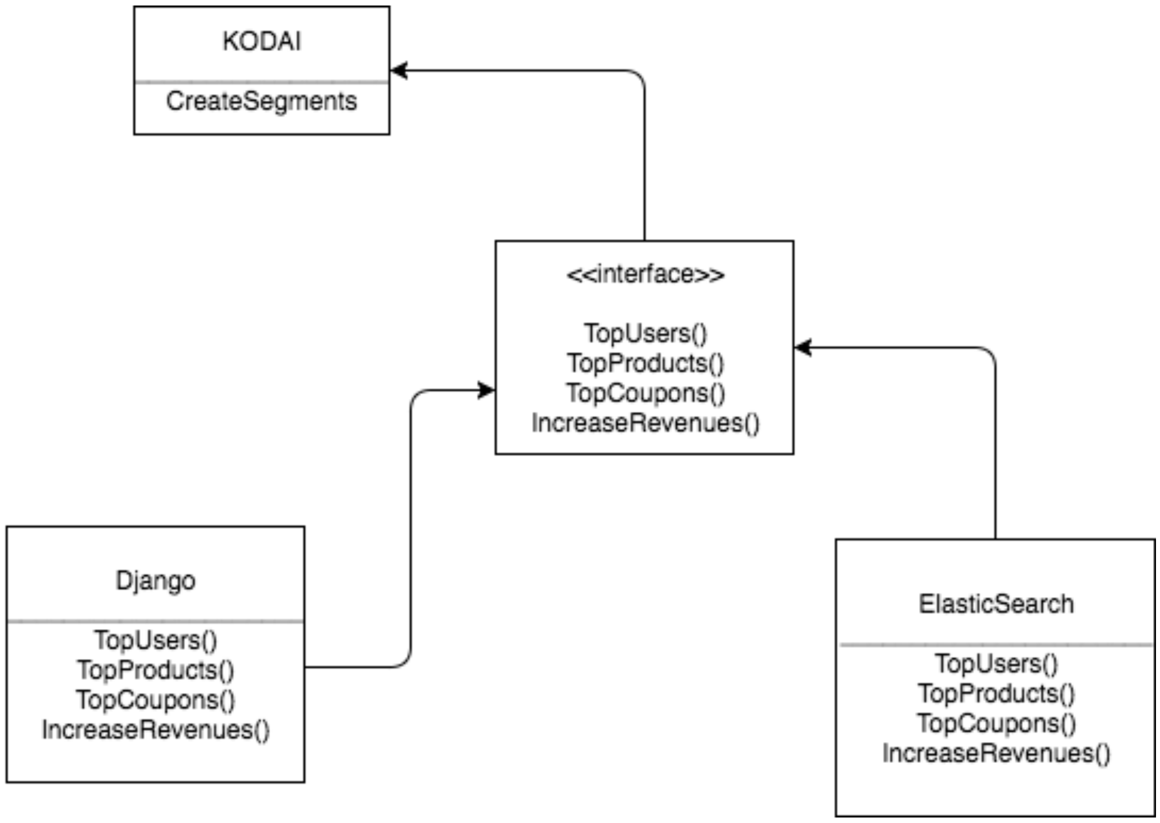


Figure 7: Software architecture and implementation for segmentation models

In Figure 7, we show an example of Kodai's software architecture. The Django module, and Elastic Search are loosely coupled, which means the modules do not interact extensively. It provides more flexibility, as Kodai can be horizontally applied to fields such as agriculture, finance and education. Kodai's architecture contains two components from Django and Elastic Search, which are used to apply segmentation models through features such as TopUsers, TopProducts, TopCoupons, IncreaseRevenues(). Each of these feature can be easily modified and used in the disciplines mentioned above. In this thesis, we focus on revenues, however if we focus on agriculture, we could use this architecture to implement a system that focuses on yield per crop instead of revenues. For example, instead of a sample revenue column, we might implement it as a yield per crop column. And, instead of sample revenue increase, we might implement it as yield per increase of crops.

4.3 A BEHAVIORAL DESCRIPTION OF KODAI:

In this section, we describe the behavioral activity within Kodai. A behavioral activity describes orchestrated, repeatable pattern of process in a software system. In the below figure, Kodai's activity diagram explains the flow of control through the structure of system.

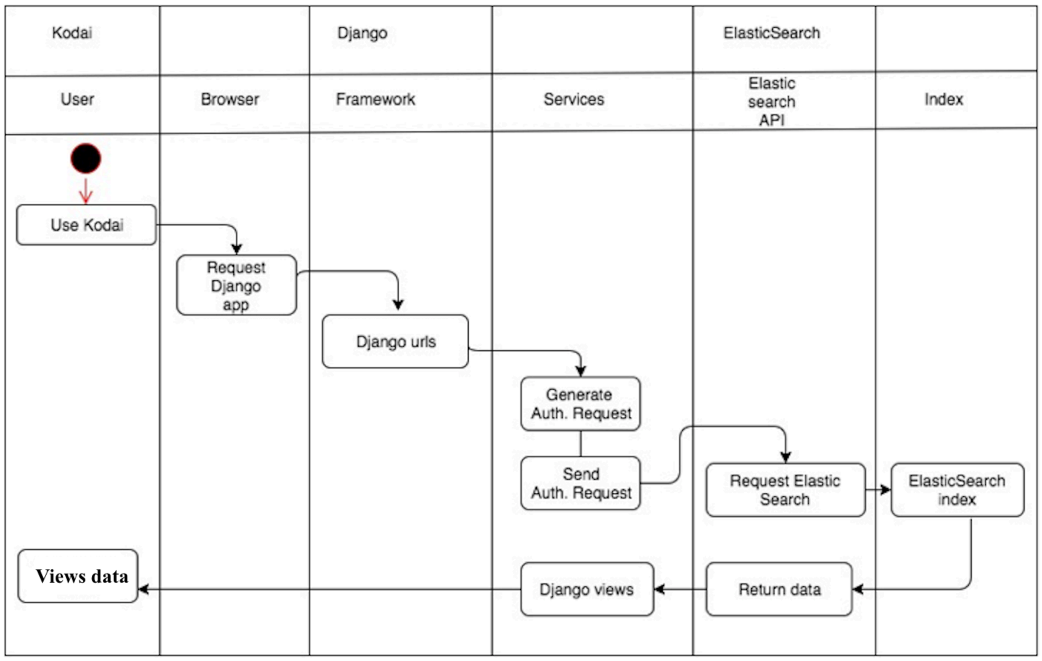


Figure 8: Kodai's Activity Diagram

In Figure 8, we show an example of the behavioral description of Kodai. This describes not only static activity but dynamic interactions between different components within Kodai [3]. In the first step, the user opens the web-application of Kodai, and clicks Create Segments. This sends a request to Django framework, within the django framework, Django handles and matches url requests. Once Django matches url requests, it generates an authorization request and sends it to Elastic Search. In Elastic Search, the request is sent to an index, and it pulls dictionary data from the index. Now, this is returned back to Django views, and displayed in the browser.

4.4 KODAI – TOP COUPONS REDEEMED

We build the implementation of the Top Coupons Redeemed feature through Kodai. In the dataset, we have coupons redeemed by consumers. A coupon is an advertisement that entitles certain benefit to the consumers when they purchase their products and redeem the coupon.

'TOP COUPONS REDEEMED' displays the top coupons redeemed by consumers in the data. This allows businesses to identify their most valuable coupon, products and to target consumers to improve revenues. We require tools such as Elastic Search and Django to run locally on our server for this feature.

As we described in Section 1.1, we use a priori segmentation based on business requirements. An a-priori segment is defined by assuming pre-conceived categorizes before looking the data. In this instance, Top Coupons Redeemed, we use usage rates and occasions of coupons in the data.

We begin by explaining an outline of how this function works. The user opens Kodai through a web browser, and clicks, 'Create Segments' menu at the navigation bar. In the navigation bar, the user clicks Top Coupons, and this sends a request using Django urls. Django lets Kodai navigate through different urls. Django requests data to Elastic Search index, and then returns data through Django views. Django views takes an HTTP Request and returns an HTTP Response to the user. Django views displays, 'Top Consumers by revenue.' This displays the total amount of units purchased by consumers in each product.

List of References:

[1] Braude, Eric., Bernstein, Michael. (2016). *Software Engineering*: Modern Approach-es. Waveland Press.

[2] Sommerville, I. (2008). *Software Engineering*. Essex, England: Pearson.

[3] Bass, Len., Clements, Paul., Kazman, Rick., (2012). *Software Architecture in Prac tice*. Addison-Wesley Professional.

CHAPTER 5

IMPLEMENTATION OF KODAI

In the last chapter we described how, Kodai is a software architecture for segmentation models. In this chapter, we describe Kodai's implementations through software frameworks and tools. We begin by explaining software implementation; software development model and details of an example component in the system from an implementation perspective. At the end, we describe the process to achieve results. Software implementation explains implementation details of Kodai. It allows the software developers to evaluate the tools necessary to develop a web-application for the business requirements. Software methodology provides the guidelines for developers to successfully implement the requirements for software development. We use hybrid waterfall agile methodology to help us develop the software as explained in Section 3.5.1 and Section 3.5.2. While the implementation involves many components, we chose to explain only the 'Top Consumer' feature of Kodai, as the other components of the software were developed using a similar methodology.

5.1 SOFTWARE IMPLEMENTATION

We explain the implementation of Kodai through a high-level architecture. A high level architecture is a description of the structure of the Kodai Software [1]. It specifies how we will develop the important components from the business requirements. In addition to showing important components, it provides the platform or the system used to develop Kodai. The architecture describes how we connected the Django framework and

Elastic Search to develop a software for the business requirements. Below is a high-level architecture of our software:

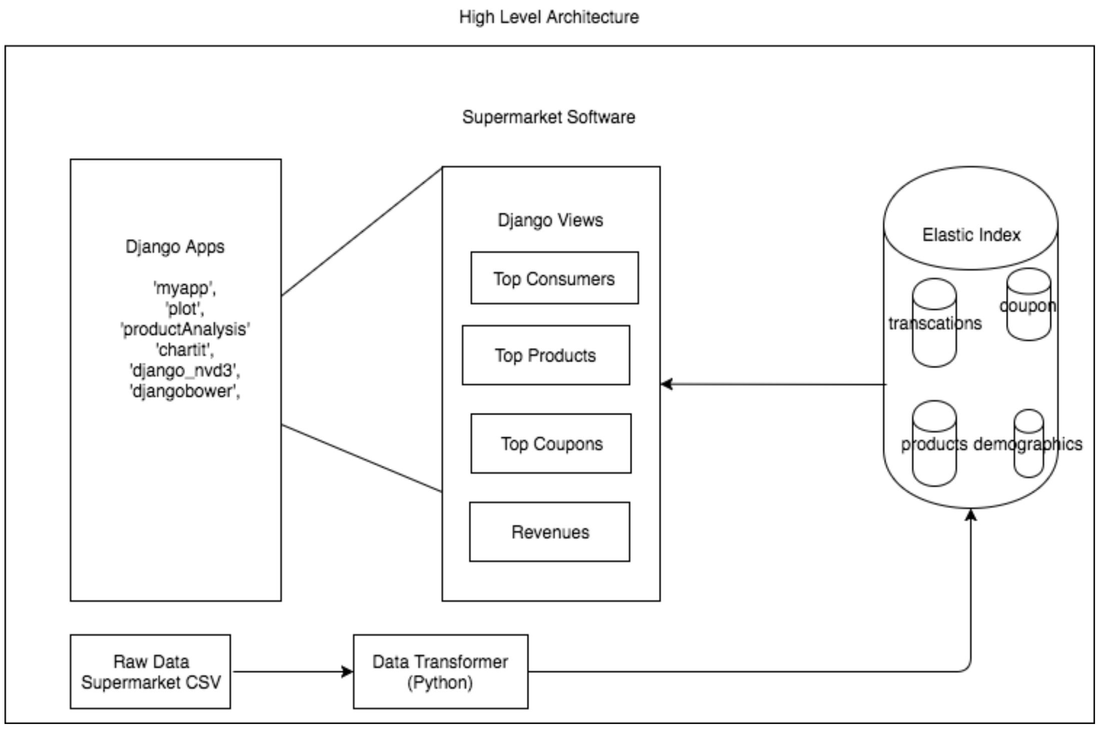


Figure 9 High-Level Architecture of Kodai

The High-Level Architecture begins with raw data, which consists of supermarket data in comma-separated-files. A Python script is used to build an indexinto the raw data. The index is sent to Elastic Search. This index allows the software developer to send queries and access data at a faster pace. Elastic Search contains the indices from supermarket data in comma-separated-file as transactions, coupons, products, demographics.

Django Views allows the software developer to capture the business requirements, and display the results. It contains, features such as ‘Top Consumers’, ‘Top Products’, ‘Top Coupons’, and allows the developer to test a hypothesis about the goal of improving revenues. The Django apps contain configuration files to load the Django framework for the software developer.

5.2 SOFTWARE FLOW DIAGRAM

The software flow diagram represents activity of actions in the software. It is a dynamic outline of activities contained in the software. It helps us to develop the important activities that the software needs to perform the business requirements. In addition to meeting the requirements, it allows the developer to understand the flow of activities in the software. Below is a high-level software flow diagram.

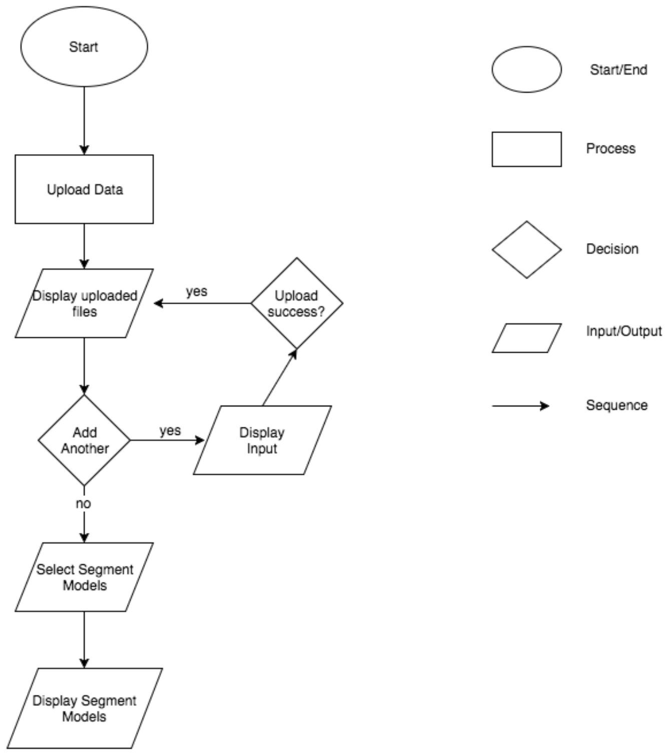
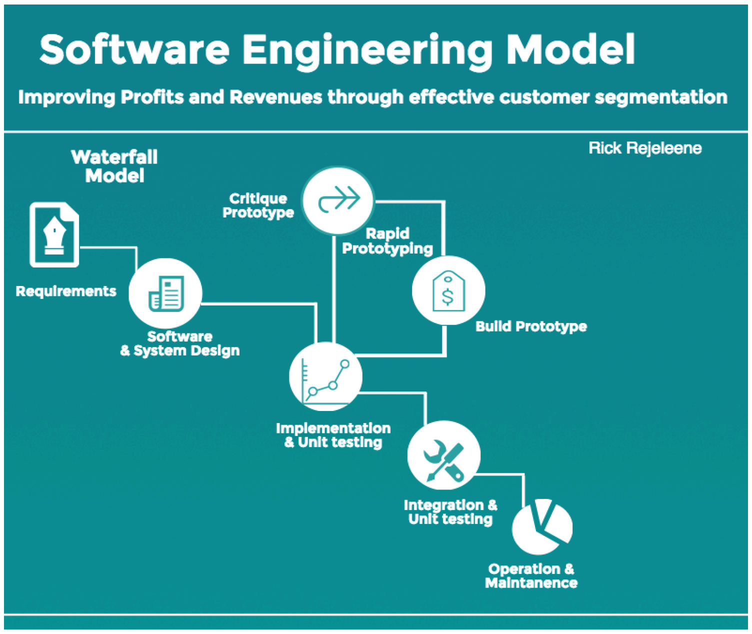


Figure 10 Software Flow of Kodai

The flow diagram begins with uploading raw data (comma-separated-files) and also constructing an index to an Elastic Search index. This index data can be viewed with an elastic search head-plugin. The software developer can upload large volume of raw data and develop an Elastic Search index. Based on the business requirements provided, a segment model is chosen to display results.

5.3 SOFTWARE ENGINEERING MODEL

In the below diagram, we describe hybrid agile waterfall steps used to develop Kodai.


Software Engineering Model
Improving Profits and Revenues through effective customer segmentation
Rick Rejeleene
Waterfall Model
Requirements
Software & System Design
Critique Prototype
Rapid Prototyping
Build Prototype
Implementation & Unit testing
Integration & Unit testing
Operation & Maintanence
Figure 11 Software Engineering model of Kodai

The software engineering model provides the steps necessary for developers to implement this software. In order for the architecture to meet the requirements, we made constant iterations. We used this process in building similar features of the software. In developing, the whole software architecture for Kodai, first we built the, “Top Users by Revenue feature” as a prototype and then iterated through various prototypes. Below we describe the steps we used to develop the features of Kodai that help the user to analysis revenues using segmentation.

5.3.1 REQUIREMENTS

In the requirement phase, we collected important business requirements to develop Kodai. As we explained in the chapter above in 2.2, The important requirements for the software are the ability to import raw data, to view data and to allow the software developer to view different segments of the data.

5.3.2 IMPLEMENTATION

In this section, we explain one feature of Kodai developed to meet the criteria of business requirements. The developed feature allows the software to display the top consumers in supermarket data. Kodai acts as an open-source data analytics framework tool that lets users from various disciples to view and understand their data. The Data Analytics framework gives users a platform to easily transform raw data into understandable categories according to their requirements. As this software is open-source, any software developer can reuse the framework and add additional features.

5.3.2.a TOP USERS BY REVENUE

As discussed in the beginning of this chapter, we here explain the implementation of the Top Consumers by Revenue feature. Since the other features are implemented similarly, this serves as a representative example. Top Consumers by Revenue displays the gross revenue generated by each consumer. This allows businesses to identify their most valuable customers to target their main profit base.

The user will need to install Django 1.9.1 and Elastic Search 2.4.4, refer Appendix for installation of Django [2]. We developed a Python script that builds an index. For example, if we wanted to build an index for a comma-separated file such as 'transaction.csv,' we would store this file as a variable within the script. The python script would access the raw data [3], sort it, and store it using the Elastic Search tool in an easily searchable format.

In the Python script, we import transaction comma-separated file in our raw data and store it as a variable to build an index. In order to view and understand this index, we use Elastic Search head-plugin [4].

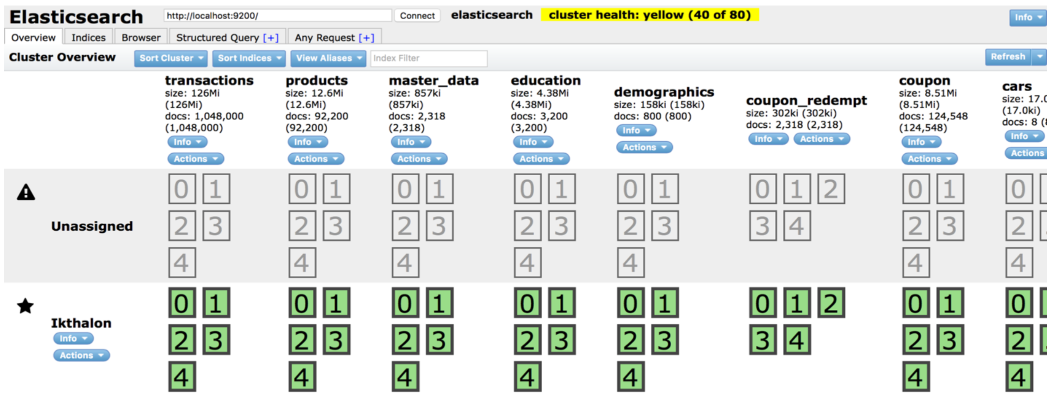
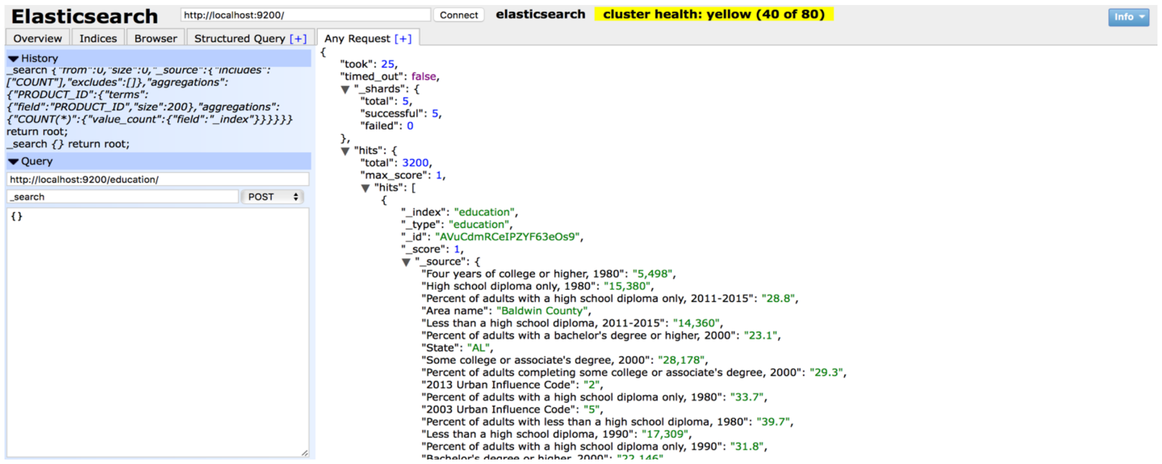


Figure 12. Elastic Search Head Plugin Graphical View

The Elastic Search head plugin gives a graphical representation of the stored index. There are five tables within the head plugin there are five tabs, each of which allows the developer to access different information about the stored index. To get a fast statistics concerning the data; we go to the 'Any Request tab', run the following command: {} and click search.

The results, which will summarize statistics such as total rows and number of instances of each category, are generated in a few milliseconds. The below picture gives us a statistic on education data.



```
_search {"from":0,"size":0,"_source":{"includes":
["COUNT"],"excludes":[]},"aggregations":
{"PRODUCT_ID":{"terms":
{"field":"PRODUCT_ID","size":200},"aggregations":
{"COUNT(*)":{"value_count":{"field":"_index"}}}}}}
return root;
_search {} return root;
```

```
http://localhost:9200/education/
_search
{}
```

```
{
   "took": 25,
   "timed_out": false,
 ▼ "_shards": {
     "total": 5,
     "successful": 5,
     "failed": 0
   },
 ▼ "hits": {
     "total": 3200,
     "max_score": 1,
   ▼ "hits": [
       {
          "_index": "education",
          "_type": "education",
          "_id": "AVuCdmRCeIPZYF63eOs9",
          "_score": 1,
        ▼ "_source": {
            "Four years of college or higher, 1980": "5,498",
            "High school diploma only, 1980": "15,380",
            "Percent of adults with a high school diploma only, 2011-2015": "28.8",
            "Area name": "Baldwin County",
            "Less than a high school diploma, 2011-2015": "14,360",
            "Percent of adults with a bachelor's degree or higher, 2000": "23.1",
            "State": "AL",
            "Some college or associate's degree, 2000": "28,178",
            "Percent of adults completing some college or associate's degree, 2000": "29.3",
            "2013 Urban Influence Code": "2",
            "Percent of adults with a high school diploma only, 1980": "33.7",
            "2003 Urban Influence Code": "5",
            "Percent of adults with less than a high school diploma, 1980": "39.7",
            "Less than a high school diploma, 1990": "17,309",
            "Percent of adults with a high school diploma only, 1990": "31.8",
            "Bachelor's degree or higher, 2000": "22,146",
```

Figure 13. Elastic Search Query in head plugin

Now that we have explained how to build an index and how to view it through a head plugin, we outline the implementation of Top Consumers by Revenue. The following files are required to find Top Consumers by Revenue: ‘transaction.csv,’ ‘product.csv,’ and ‘hh_demographic.csv.’

'products = SELECT SUM(SALES_VALUE) FROM transactions GROUP BY household_key ORDER BY SUM(SALES_VALUE) DESC LIMIT 160'

Once we have built an index from the above files, using Python we send this SQL query to the Elastic Search Index and store it as a variable.

Elastic Search receives this query from Python and sends back results in a dictionary.

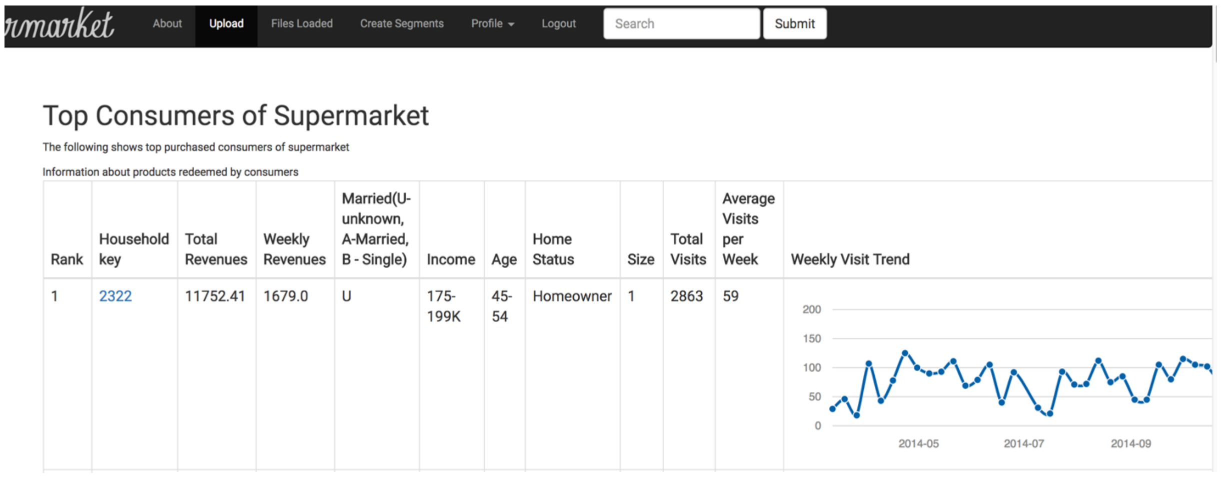
```
{
   "took":    25,
     "timed_out":    false,
  "_shards":    {
 "total":    5,
  "successful":    5,
  "failed":    0
 },
   "hits":    {
  "total":    1048000,
   "max_score":    0,
   "hits":    [ ] },
"aggregations": { "household_key":
{"doc_count_error_upper_bound":    -1,"sum_other_doc_count":
802982,"buckets":  [
{ "key":     "2322","doc_count": 2863,
"SUM(SALES_VALUE)":
{ "value": 11752.409999999993}},
 { "key":     "2459","doc_count":
3642,"SUM(SALES_VALUE)":
{ "value": 11558.11999999999}},
 { "key":     "1023","doc_count":
1242,"SUM(SALES_VALUE)":
 { "value": 11060.49}},
{ "key":     "1609","doc_count": 2815,"SUM(SALES_VALUE)":
{ "value": 10837.880000000005}},}
```

Elastic Search Dictionary

In above example of Elastic Search Dictionary, we can see how Elastic search builds a dictionary from an index. Each dictionary has key, and value pair. In order to access this dictionary, we define the key that we need for our feature in a Python script. For example: In Top Visits, we define in our for loop, the name of dictionary, 'household_key', to access household keys from products.

for product in products['aggregations']['household_key']['buckets']

After Elastic Search has received these results in the form of a dictionary, the same Python script allows us to access the values and keys. After this, the results are passed to the Django template file, which renders the results in an HTML file.



Top Consumers of Supermarket

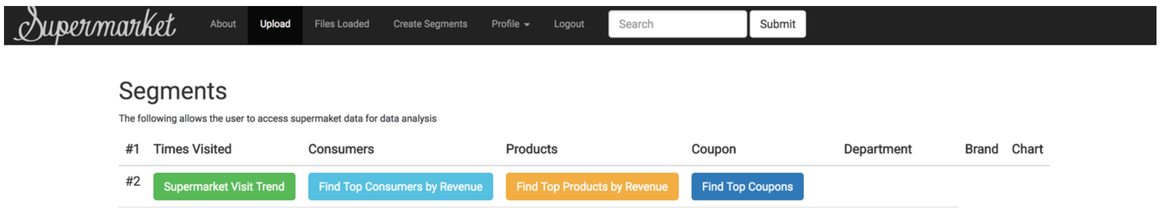
The following shows top purchased consumers of supermarket

Information about products redeemed by consumers

Rank	Household key	Total Revenues	Weekly Revenues	Married(U-unknown, A-Married, B - Single)	Income	Age	Home Status	Size	Total Visits	Average Visits per Week	Weekly Visit Trend
1	2322	11752.41	1679.0	U	175-199K	45-54	Homeowner	1	2863	59	

Figure 14 Kodai showing Top Consumers by Revenue feature

To display the results, we open web browser, ‘Google Chrome’ and type the address of local server, “127.0.0.1:8080”, which results in the following screen:



Segments

The following allows the user to access supermarket data for data analysis

#1	Times Visited	Consumers	Products	Coupon	Department	Brand	Chart
#2	Supermarket Visit Trend	Find Top Consumers by Revenue	Find Top Products by Revenue	Find Top Coupons			

Within moments our software provides the Top Consumers in Supermarket.

Hence, we have explained one component of our software as mentioned in above chapter.

Figure 15 Kodai Software showing functions

In a similar way, Kodai gives:

- Top Products by revenue
- Top Products by units sold
- Top Coupons redeemed
- Products bought by Top Consumers

- Products redeemed by Coupons
- Improve Revenues

5.4 USAGE AND APPLICATION

Our tool is simple, and easy for developers to understand and apply for various needs. It can be used in education, agriculture, real estate, financial and retail industries.

5.4.1 EDUCATION SOFTWARE

While our Software is designed to import supermarket data and allow the developer to experiment to improve revenues, it is not limited to this specific purpose. For example, it also could aid educational institutions in the admissions process. In an interview with Cynthia Bonn, the dean of Higher Education for Admission at the University of Rhode Island, the following functions of software were determined as reusable in her work:

- Finding Likely Students who would be admitted to URI
- Enabling the dean of admissions to have a software framework to work on student data collected from high schools
- Storing the collected data to track trends in admissions

5.4.2 AGRICULTURE SOFTWARE

In addition to education, our software would allow farmers to improve crop yield and overall agricultural efficiency. The following functions of the software could be used to improve productivity by:

- Providing a software framework for accessing crop data

- Storing the collected crop data to track trends
- Enabling the farmer to experiment with strategies to improve farming revenues

5.4.3 FINANCE SOFTWARE

In banking, our software could allow bankers to safely store the clients' usage data, and improve public relations with clients. This can be done through:

- Accessing client activity
- Providing a software framework to quickly access client data
- Enabling the bankers to track client behavior
- Enabling the bankers to experiment with strategies to improve banking revenues

5.4.4 SOFTWARE REUSE

Although the software was initially intended for use with analyzing supermarket data, it is a highly reusable software with significant applications in agriculture, education, and finance. The software can be used in tracking school admissions, crop production, and banking client trends, in addition to its primary purpose in supermarkets. Therefore, the software crosses domains and is horizontally reusable. As such, it meets the criteria for software reusability.

List of References

[1] Sommerville, I. (2008). *Software Engineering*. Essex, England: Pearson.

[2] Holovaty, A., & Willison, S. (2008, June 4). Django. *Django at a glance.* Retrieved January 22, 2017, from https://docs.djangoproject.com/en/1.9/intro/overview/

[3] Rajkumar, V. (2011, April 22). Dunnhumby – Customer Science & Consultancy for Retailers & Manufacturers (CPGs). *Source Files | dunnhumby.* Retrieved January 22, 2017, from http://www.dunnhumby.com/sourcefiles

[4] Birch, B. (n.d.). Elastic Search Head. *Documentation.* Retrieved January 22, 2017, from http://mobz.github.io/elasticsearch-head/

CHAPTER 6

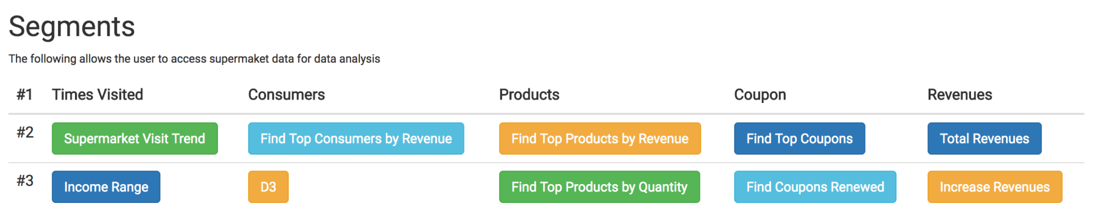
RESULTS

As we proposed, we developed Kodai as a software architecture for segmentation models to improve revenues in a supermarket applied. This allows users to upload data, and analyze methods to experiment with revenue increasing strategies. This allows the business to effectively analyze and access data through our software. We analyzed supermarket data, found top consumers, top units of products and top coupons, and hypothesized how to improve revenues with Kodai.

In addition to methods to experiment with revenue increasing strategies, our software is reusable, meeting the criteria of business requirements [1].

6.1 IMPROVEMENT OF REVENUES

The following hypothetical example helps to show how Kodai could allow for experiments to explore increasing revenue. Kodai allows a developer to easily test out this hypothetical example, by adding an additional column. It takes approximately 5 minutes to add this column to test out this hypothetical example. This is implemented in a similar way explained in 3.3.2.a 'Top Users by Revenue.' Due to lack of price in our dataset, we give a sample price of $2 for the top twenty items. We begin to add additional columns in the software that would allow experiment revenue increase. Next, we hypothesize an increase of 10 cents in each of the items. We select one product from top products by units sold feature.


Segments

The following allows the user to access supermaket data for data analysis

#1	Times Visited	Consumers	Products	Coupon	Revenues
#2	Supermarket Visit Trend	Find Top Consumers by Revenue	Find Top Products by Revenue	Find Top Coupons	Total Revenues
#3	Income Range	D3	Find Top Products by Quantity	Find Coupons Renewed	Increase Revenues

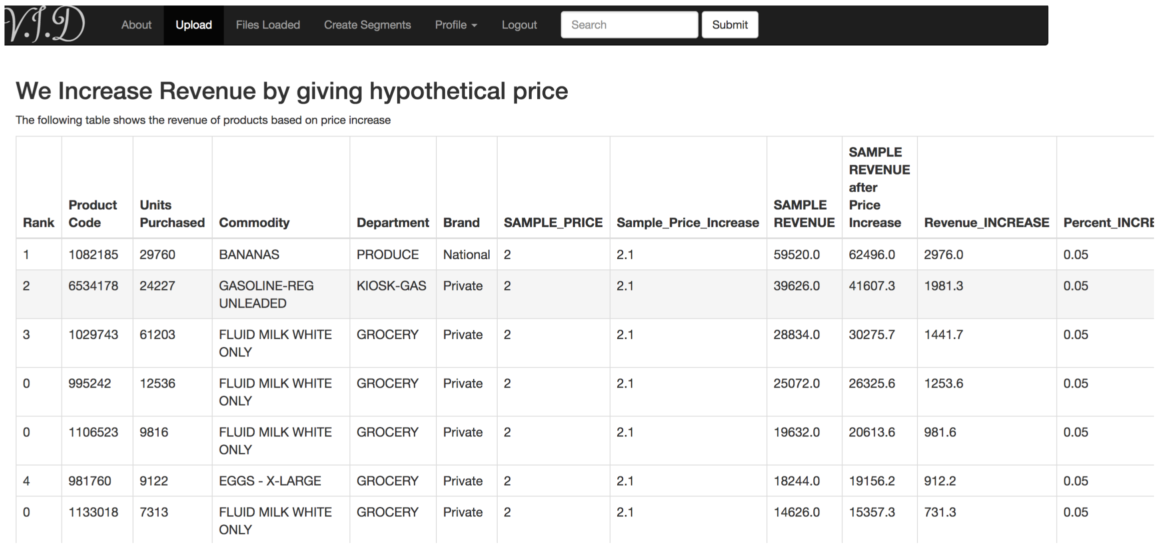
Figure 16. Segments in Kodai

Let us assume the cost of each pound of bananas is $2. In our dataset, we assign the value 2 to the variable SAMPLE_PRICE for bananas. The number of bananas sold in the dataset is 29760, and the cost of each pound of bananas that we assigned is $2, therefore the total sample revenue of the banana is $59,520.

Now, we increase the sample price of each pound of bananas by 10 cents in our data set. We assigned the value 2.10 to the sample banana price. Therefore, the total sample revenue of the banana after increasing 10 cents is $62,496.

However, if we collect purchase data over time, it might indicate increasing the cost of bananas might decrease the volume of sales. This is referred to as price elasticity. Price elasticity is the measure of relationship between change in quantity to change in price. If the developer is able to upload subsequent measurements of quantity and price, Kodai is able to measure price elasticity.

Hence, our software is able to test out this hypothesis of how to improve revenues through use of Kodai.



We Increase Revenue by giving hypothetical price

The following table shows the revenue of products based on price increase

Rank	Product Code	Units Purchased	Commodity	Department	Brand	SAMPLE_PRICE	Sample_Price_Increase	SAMPLE REVENUE	SAMPLE REVENUE after Price Increase	Revenue_INCREASE	Percent_INCRE
1	1082185	29760	BANANAS	PRODUCE	National	2	2.1	59520.0	62496.0	2976.0	0.05
2	6534178	24227	GASOLINE-REG UNLEADED	KIOSK-GAS	Private	2	2.1	39626.0	41607.3	1981.3	0.05
3	1029743	61203	FLUID MILK WHITE ONLY	GROCERY	Private	2	2.1	28834.0	30275.7	1441.7	0.05
0	995242	12536	FLUID MILK WHITE ONLY	GROCERY	Private	2	2.1	25072.0	26325.6	1253.6	0.05
0	1106523	9816	FLUID MILK WHITE ONLY	GROCERY	Private	2	2.1	19632.0	20613.6	981.6	0.05
4	981760	9122	EGGS - X-LARGE	GROCERY	Private	2	2.1	18244.0	19156.2	912.2	0.05
0	1133018	7313	FLUID MILK WHITE ONLY	GROCERY	Private	2	2.1	14626.0	15357.3	731.3	0.05

Figure 17 Kodai is able to test and add price increase column

6.2 SOFTWARE AND ARCHITECTURAL REUSABILITY

As mentioned in the previous section, Kodai is not limited to supermarket data, but it can be reused in the following fields of Education, Finance, Agriculture, new features can be easily developed, and added to our software through Django [2].

In Agriculture, Kodai can be used in analyzing crop data. In order to use Kodai, the farmer has to have crop data. This can be collected through devices such as Lidar sensors, and drones, human observation [3]. In the recent age of big data, we know that there is no lack of data available in crops. However, a platform to understand, and study the data using segmentation models is not available. In Kodai, the objective was revenues, however, in agriculture, the farmer will be able to apply segmentation models to find crop yield instead of revenues.

In Education, Kodai can be used to find out patterns of student behavior, likelihood of students passing a class. In order to use Kodai, educators ought to have student's data on behavior, and performance. In using Kodai as a platform, educators would be able to apply segmentation models to determine how they might help most students to succeed in their classes, and programs.

In Finance, Kodai can be used as a platform to find out client activity, trends and revenue improvement for banks. If the bank has data on consumers' activity, then it is possible for the bank to use Kodai. Kodai's features such as Top Consumers, Top Products can be implemented to find Top Clients by usage, Top Clients by Revenue. On using Kodai, bankers will be able to apply segmentation models to find trends, and methods to improve revenues.

For each of the applications outlined above the user would need to collect the data to be explored using Kodai, and place it into CSV format.

List of References

[1] Sommerville, I. (2008). *Software Engineering*. Essex, England: Pearson.

[2] Holovaty, A., & Willison, S. (2008, June 4). Django. *Django at a glance.* Retrieved January 22, 2017, from https://docs.djangoproject.com/en/1.9/intro/overview/

[3] Hall, David., Jellen, Mike., Neese, Marty. *Velodyne Lidar.* Retrieved October 20, 2017, from https://velodynelidar.com/docs/datasheet/LiDAR%20Comparison%20chart_Rev-A_Web.pdf

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Our work in this thesis helps to show that Kodai is a software architecture for segmentation models to improve revenues for a supermarket. It can test a hypothesis to address the problem of increasing revenues in supermarkets. We achieved this by allowing the user to upload supermarket data within the software framework, using a web-based application that implements software engineering methodology to improve revenues, allowing the developer to complete business requirements to be built into the software. The following tables explain how Kodai met both software and business requirements.

Software Requirements	**How Kodai met requirements**
Import data and analyze data according to the business requirements	Kodai is able to build an index through Elastic Search
The software must be reusable and applicable across many industries in guidance with principles of software engineering	Kodai is applicable to agriculture, finance, and education.
The software must be extensible for future add-ons and other segmentation applications	Kodai can import Machine Learning Package such as Graph Lab
The software should be able to run on any operating system, including windows, mac, Linux	Kodai can run on a cloud server. This has been shown through our implementation in Chapter 5.

Table 4 How Kodai meets software requirements

Business Requirements	**How Kodai met requirements**
The software should be able to quickly identify top consumers of a supermarket from any demographic	Top Consumers feature of Kodai shows top consumers
The software should be able to determine the households who spend the most in a supermarket	Top Revenues feature of Kodai shows households who spent the most
The software should be able to identify coupons with the highest number of consumer redemptions	Top Coupons feature of Kodai shows coupons that were redeemed the most.
The software should allow the developer and other users to test out hypotheses to improve revenues	Increase Revenue feature of Kodai enables to test out hypotheses to improve revenues
The software should be open source and able to handle at least a gigabyte of raw data	Kodai's code is available in GitHub platform for reuse. Kodai by using Elastic Search it is able to build index of more than gigabyte of raw data

Table 4 How Kodai met business requirements

Kodai has implications beyond the supermarket, enabling users to analyze and store data faster. As Kodai is open-source, any developer can freely access the code and develop it into the client requirements. In addition to open-source, our software is reusable in fields such as agriculture, finance, and education. Whether we are trying to ascertain who loves stewed tomatoes in Aisle 4 or whether Liberty University churns out admissions on the regular basis, Kodai is applicable to a multitude of real life situations.

7.2 FUTURE WORK

The future applications of Kodai could focus on the security of accessing this software, as well as features such as the implementation of user profiles and building more visualization tools. In addition to such features, Kodai can also integrate machine learning algorithms. One example of machine learning package is Graph Lab that can be integrated into Kodai for prediction of purchases, coupons, revenues [1].

7.2.1 SECURITY

Security is quality or state of being secure. In any exchange of data or transaction, there is a vulnerability for hijacking the transaction. To protect against hijacking of data or transaction, we required various security measures. In our software, uploading, and data analysis can be openly accessed. If Kodai is run on a cloud, there is high security risk of any user accessing it through the internet. In order to prevent such access, we can add secure login and access to our software. This would secure our software from unauthorized access.

7.2.2 USER PROFILE

In order to stop unauthorized access, Kodai could focus on the implementation of user-profiles. A user profile can customize individual data and stores information for each user. A user is any person who uses our software. User profiles helps to individual -ize data and content to each user. This can be implemented further through the Django framework.

7.2.3 MACHINE LEARNING ALGORITHMS

Kodai has given a platform to visualize, analyze, and find details about supermarket data. Our future work will focus on integrating Machine Learning algorithms into Kodai. This enables a program to learn through experience. A Machine learning algorithm learns by analyzing large amounts of data [2]. This allows the development of systems that can automatically adapt and customize recommendations to individual users. It can learn from data, rather from a pre-determined model.

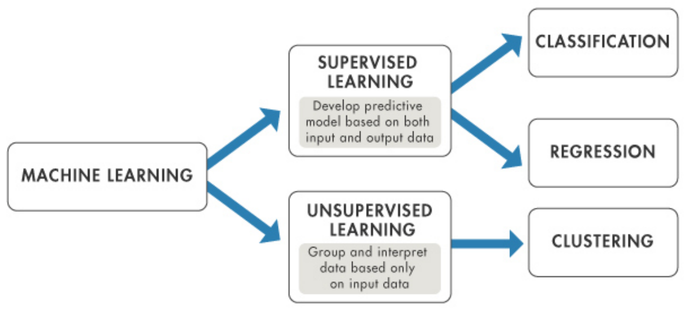

MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Figure 18: Machine Learning categories [2]

Machine Learning can be classified into supervised and unsupervised learning. In supervised learning, we have a set of input variables (x) and an output variable from which an inferred function is produced to be used on later examples. Unsupervised learning analyses sets of label data for patterns. Supervised learning focuses more on classification and regression. In Figure 18, classification techniques would predict categorical responses. In Kodai classification techniques could be applied to predict if a consumer is likely to visit supermarket. Regression involves continuous prediction, changes in weather might be correlated with consumers buying some products more than others. In the event of a snow-storm, there is a greater likelihood that customers will buy milk, bread and snow shovels. Clustering is the most common unsupervised learning. It is used for exploratory data analysis.

In Kodai, future work should focus on supervised learning to predict individualizing coupons to each consumer. Supervised learning can classify output variable based on an input dataset. This can be implemented by adding data science machine learning packages such as the GraphLab. The GraphLab is a Python package that allows developers to implement machine learning algorithms. This would help in building predictive models within Kodai.

7.2.4 DATA VISUALIZATION

In addition to predictive models built through machine learning package, to understand the results of predictive models, Kodai needs future work on data visualization. Kodai includes buying trends built on the Graphos library for Django. However, this could be extended to predictive models on purchases.

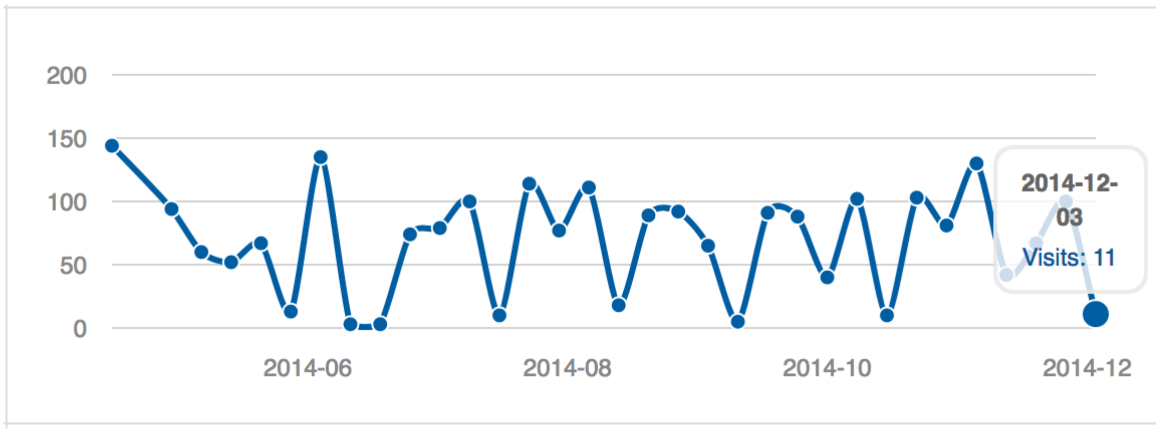


Figure 19 Kodai Weekly Visit Trend

In Figure 19, Kodai shows weekly visit trend of customers using Data Visualization. Data visualization is using graphics to communicate results of an analysis to the user. This can be implemented through data visualization packages such as D3 JavaScript library to help developers produce dynamic, interactive data visualization in web browsers.

7.2.4.1 DEVELOPING AN INTERFACE TO ANALYZE BASKETS OF PURCHASE

Kodai will be enhanced with data visualization tools to analyze baskets of purchase. This permits the identification of loss leaders. Loss leader is a pricing strategy, where a product is sold at a price below its market, to gain new customers for the product.

7.2.4.2 CAPTURING PRICE ELASTICITY OVER TIME

As we mentioned in Improvement of Revenues in Section 6.1, Kodai would be able to measure price elasticity. Price elasticity is the measure of relationship between change in quantity purchased to change in price. If the developer uploads subsequent measurements of quantity and price, Kodai will be able to measure price elasticity.

List of References

[1] Guestrin, C. (2011, June 28). Learn Turi. *GraphLab.* Retrieved April 24, 2017, from https://turi.com/learn/userguide/

[2] Moler, Cleve. (1984). *Machine Learning in MATLAB*. Retrieved October 20, 2017, from https://www.mathworks.com/help/stats/machine-learning-in-matlab.html?requestedDomain=fr.mathworks.com

CHAPTER 8

APPENDIX

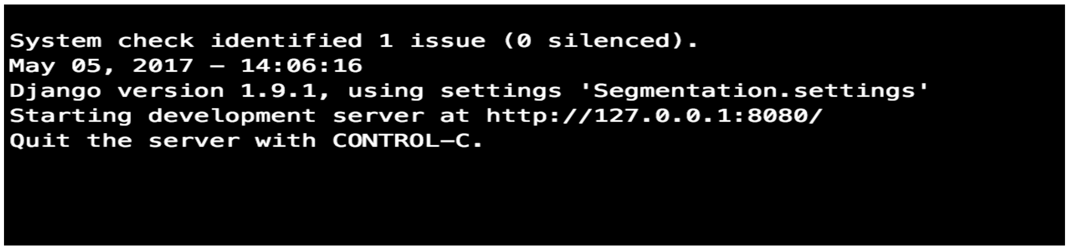
This chapter explains how to install the software tools necessary to develop a system similar to Kodai.

8.1 DJANGO INSTALLATION

The user will need to install Django 1.9.1. In order to run Django, we require the user to follow these steps:

a) Open Terminal or Command Line, and run the following:

   'Python manage.py runserver 8080'

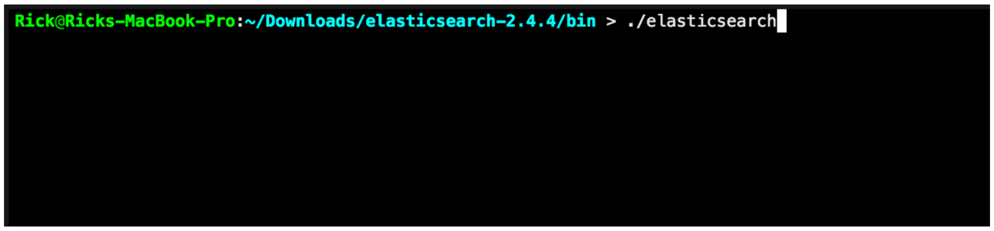


```
System check identified 1 issue (0 silenced).
May 05, 2017 - 14:06:16
Django version 1.9.1, using settings 'Segmentation.settings'
Starting development server at http://127.0.0.1:8080/
Quit the server with CONTROL-C.
```

Figure 20 User running Django Local Server

This command uses the Python language to run the Django server locally at the 8080 port. The user will need to install Elastic Search 2.4.4. We run Elastic Search through the following:

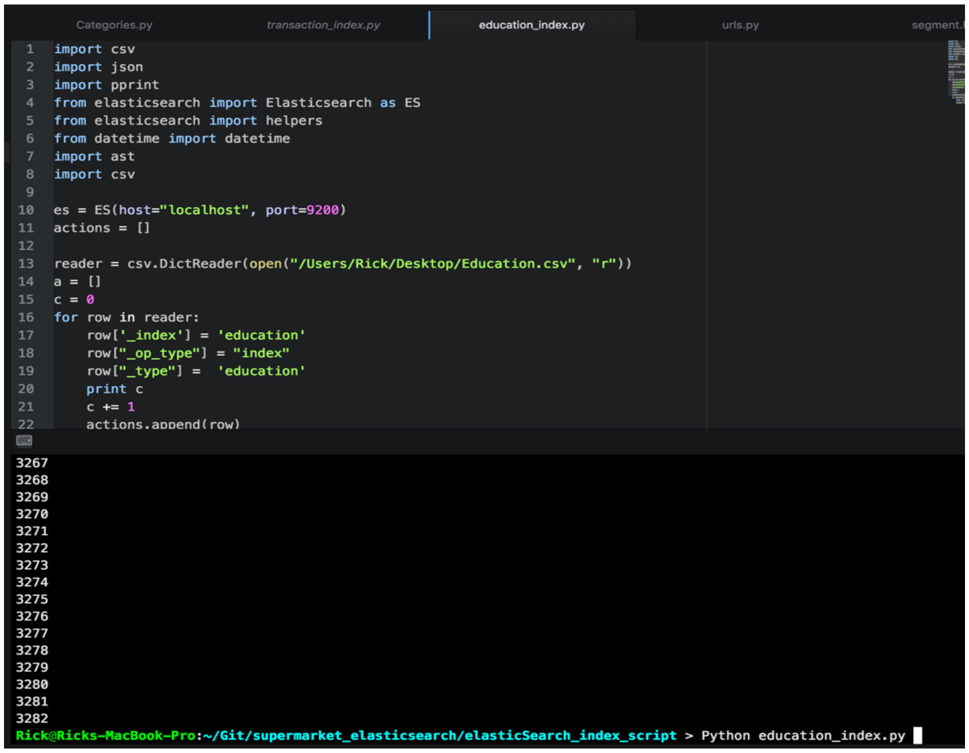
Open another Terminal or Command Line and run the following, within the bin folder of Elastic Search 2.4.4, type: './elasticsearch' to run Elastic Search.



```
Rick@Ricks-MacBook-Pro:~/Downloads/elasticsearch-2.4.4/bin > ./elasticsearch
```

Figure 21 Run Locally Elastic Search Server

Next, we must use Elastic Search to build an index that contains any necessary raw data for the Top Consumer by Revenue feature described in 3.3.2.a. This index stores raw data and makes it searchable. Each entry in the index must be assigned a type, and each type has specific properties.



```python
import csv
import json
import pprint
from elasticsearch import Elasticsearch as ES
from elasticsearch import helpers
from datetime import datetime
import ast
import csv

es = ES(host="localhost", port=9200)
actions = []

reader = csv.DictReader(open("/Users/Rick/Desktop/Education.csv", "r"))
a = []
c = 0
for row in reader:
    row['_index'] = 'education'
    row["_op_type"] = "index"
    row["_type"] =  'education'
    print c
    c += 1
    actions.append(row)
```

```
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
Rick@Ricks-MacBook-Pro:~/Git/supermarket_elasticsearch/elasticSearch_index_script > Python education_index.py
```

Figure 22 Python Index script

All the Python script used to build an index in Elastic Search is located in Github in the following link:

https://github.com/ludwigwittgenstein2/supermarket_elasticsearch/tree/master/elasticSearch_index_script

8.2 PYTHON SCRIPT FOR TOP CONSUMERS

Below is the Python script that we developed for Top Consumers:

Table 6 Python Script for top consumers

```
#=============================================================#

# Author: Rick Rejeleene

# Advisor: Joan Peckham

#!/usr/bin/Python

# Top Customers

#Arranging based on household_key not SALES_VALUE

import requests

import json

import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D


#Send SQL query to Elastic Search

products = requests.post('http://localhost:9200/_sql', data = 'SELECT SUM(SALES_VALUE) FROM
transactions GROUP BY household_key ORDER BY SUM(SALES_VALUE) DESC LIMIT 160 ').json()
#print 'name, quantity, value'


N = 10

value = []

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')

household_key_list = []

{

  "SUM(SALES_VALUE)": {
    "value": 11558.11999999999
  },
  "key": "2459",
  "doc_count": 3642
}
rank = 0
for product in products['aggregations']['household_key']['buckets']:
  household_key = product['key']
  print 'rank, household_key, value spent, Married,   Age, Home Status, Household_Size'


  values = product['SUM(SALES_VALUE)']['value']
  name_json = requests.post('http://localhost:9200/_sql', data='SELECT * FROM demographics WHERE
household_key = "'+ str(household_key)+'"').json()
  if len(name_json['hits']['hits']):
      rank += 1
      name = name_json['hits']['hits'][0]['_source']
      household_key_list.append(household_key)
print rank,'\t', household_key, '\t', '\t', val-
ues,'\t',name['MARITAL_STATUS_CODE'],'\t',name['AGE_DESC'],'\t',
name['HOMEOWNER_DESC'],'\t',name['HOUSEHOLD_SIZE_DESC']
#==================================================================#
```

BIBLIOGRAPHY

Armstrong, H. (2004, June 20). Pearson Higher Education. Marketing Research. Retrieved January 22, 2017, from http://catalogue.pearsoned.co.uk/assets/hip/images/catalog/uploads/Solch3.pdf

Armstrong, M. (2006). Advances in Economics and Econometrics: Volume 2: Theory and Applications, Ninth World Congress (Vol. 2). New York: Cambridge University Press.

Banon, S. (2010, February 21). Elastic Search. Preface. Retrieved January 22, 2017, from https://www.elastic.co/guide/en/elasticsearch/guide/current/_preface.html

Birch, B. (n.d.). Elastic Search Head. Documentation. Retrieved January 22, 2017, from http://mobz.github.io/elasticsearch-head/

Capell, K. (2008, January 29). Business Week. Tesco: 'Wal-Mart's Worst Nightmare'. Retrieved January 22, 2017, from http://shawndra.pbworks.com/f/Tesco_+'Wal-Mart's+Worst+Nightmare+-+BW.pdf

Cardoso, M., & Fonseca, J. (September 2007). Supermarket Customers segments stability. Journal of Targeting, Measurement and Analysis for Marketing, 15(4), 220-221.

Chamberlin, D., & Boyce, R. (2004, April 14). Wikipedia. History of SQL. Retrieved January 22, 2017, from https://en.wikipedia.org/wiki/SQL

Crott, J. (n.d.). Django. Tardis Documentation. Retrieved January 22, 2017, from http://mytardis.readthedocs.io/en/2.5/_images/DjangoArchitecture-JeffCroft.png

Decker, E. (2016, February 6). Central District of California. No. ED -15-0451 M. Retrieved January 22,2017, from https://assets.documentcloud.org/documents/2755241/031123088014.pdf

Donnelly, C. (2012, September 1). Role of action research in the study of small business marketing and retailer loyalty card data, National University of Ireland Maynooth, Queens Management School, University of Ulster. Retrieved January 22, 2017, from http://eprints.maynoothuniversity.ie/4091/1/CD_Action_Research.pdf

Friedman, M. (2012). Capitalism and Freedom. Chicago: University of Chicago.

Friedman, M. (1962). Price Theory. Chicago: Aldine Transaction.

Furman, J., & Simcoe, T. (2015, February 6). The White House President Barack Obama. The Economics of Big Data and Differential Pricing. Retrieved January 22, 2017, from https://obamawhitehouse.archives.gov/blog/2015/02/06/economics-big-data-and-differential-pricing

Holovaty, A., & Willison, S. (2008, June 4). Django. Django at a glance. Retrieved January 22, 2017, from https://docs.djangoproject.com/en/1.9/intro/overview/

Kadet, A. (2008, May). Smart Money, Price Profiling. Wall Street Journal Magazine

Kaplan-Moss, J. (2006, January 27). Jacobian. Why you should use Django. Retrieved January 22, 2017, from https://jacobian.org/writing/why-django/

Lipovetsky, S., Magnan, S., & Zanetti-Polzi, A. (2011). Pricing Models in Marketing Research. Scientific Research an Academic Publisher, 167-174.

Madden, M. (2014, November 12). Pew Research Center. "Public Perceptions of Privacy and Security in the Post-Snowden Era,". Retrieved January 22, 2017, from http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/

Makai, M. (n.d.). Full Stack Frameworks. Web Frameworks. Retrieved January 22, 2017, from Makai, Full Stack Python (2017, January 22). Documentation[Website]. Retrieved from www.fullstackpython.com/web-frameworks.html

Marn, M., & Rosiello, R. (n.d.). Harvard Business Review. Managing Price, Gaining Profit. Retrieved January 22, 2017, from http://web.nchu.edu.tw/~hjlee/files/Pricing_Strategy/03_Managing%20price,%20Gaining%20Profit.pdf

Martinez, R. (n.d.). The Dashbuilder Project. Dashbuilder. Retrieved January 22, 2017, from http://dashbuilder.blogspot.com/2015/10/uf-dashbuilder-real-time-dashboard-with.html

Perez, Fernando. (2010, June 5). Notebook Guide. IPython. Retrieved January 21, 2017, from ipython.org

Preston-Werner, T., Wanstrath, C., & Hyett, P. Github Help. Retrieved January 22, 2017, from https://help.github.com/

Provost, F., & Fawcett, T. (2013). Data Science for Business. CA: O'Reilly Media.

Rajkumar, V. (2011, April 22). Dunnhumby - Customer Science & Consultancy for Retailers & Manufacturers (CPGs). Source Files | dunnhumby. Retrieved January 22, 2017, from http://www.dunnhumby.com/sourcefiles

Ronacher, A. (n.d.). Flask. Welcome | Flask. Retrieved January 22, 2017, from http://flask.pocoo.org

Sommerville, I. (2008). Software Engineering. Essex, England: Pearson.

Smith, W. (n.d.). Marketing Masters. *Product Differentiation and Market Segmentation as Alternative Marketing Strategies.* Retrieved January 22, 2017, from https://archive.ama.org/archive/ResourceLibrary/MarketingManagement/documents/9602131166.pdf

Struhl, S. (2013). Market Segmentation: An Introduction and Review. CreateSpace Independent Platform Publishing.

Van Rossum, G. (2004, March 22). Wikipedia. Python History. Retrieved January 22, 2017, from https://en.wikipedia.org/wiki/Python_%28programming_language%29