

Table of Contents

1	Introduction.....	3
1.1	Existing System:.....	3
1.1.1	Examples:.....	3
1.1.2	Issues with Existing Systems:.....	4
1.2	Problem Statement.....	4
1.3	Solution.....	4
1.4	Advantage of Proposed System.....	4
2	Requirements and System Analysis.....	6
2.1	Use Case Model.....	7
2.1.1	Use Case Diagram.....	7
2.2	Use Case Brief Detail.....	8
2.2.1	Login.....	8
2.2.2	Sign up.....	8
2.2.3	Run Scraper.....	8
2.2.4	Change Category.....	8
2.2.5	Delete Account.....	9
2.3	Use Case Description Detail.....	9
2.3.1	Login.....	9
2.3.2	Sign Up.....	9
2.3.3	Run Scraper.....	10
2.3.4	Change Category.....	10
2.3.5	Delete Account.....	11
3	System Design.....	14
3.1	Phases of Data retrieval.....	14
3.1.1	Data Scraping.....	15
3.1.2	Data fetching.....	15
3.2	Sequence Diagrams.....	16
3.2.1	Sign in.....	16
3.2.2	Sign Up.....	17
3.2.3	Scrap data.....	18
3.2.4	Change Settings.....	19
3.2.5	Delete Account.....	20

4	System Implementation	22
4.1	Details of Implementation Tools and Technologies	23
4.1.1	Node.js	23
4.1.2	Express.js	24
4.1.3	Handlebars.js.....	24
4.1.4	MongoDB	24
4.1.5	Github	24
4.2	Project Screenshots	25
4.2.1	Homepage	27
4.2.2	Contact us page	28
4.2.3	Settings page	Error! Bookmark not defined.
4.2.4	Sign In page	28
4.2.5	Sign up page.....	29
5	System Testing	32
5.1	Software Quality Assurance.....	32
5.2	Software Quality Control	32
5.2.1	Black Box Testing.....	32
5.3	Test Case	32
5.3.1	Login	32
5.3.2	Signup	33
5.3.3	Run Scraper.....	33
5.3.4	Change Category.....	34
5.3.5	Delete Account.....	35
6	Conclusion	37
7	References.....	38

Chapter # 1

Introduction

1 Introduction

Kappas is a web based application system developed on Node.js that allow users to stay updated on latest sales of major brands in Pakistan. It periodically scraps data from various pre-defined websites, stores that data in database and then displays it on a Web App. The system is built as a web app on modern development stack to provide users an easy to use interface, also with high system scalability in mind.

1.1 Existing System:

There exist many different Web Apps that provide users with sale information. Most of these web apps do not show proper information of the products which are on sale. Many of these sites ask users to fill survey forms to get sale notifications and obliterate user experience with unnecessary ads. There are not many systems which show sales without the aforementioned problems.

1.1.1 Examples:

1.1.1.1 BrandsPakistan.Pk

It's a website that provide news regarding sales of many brands of Pakistan but it only displays a generic sale summary instead of in depth product sales of a specific brand. It also contains many ads.

1.1.1.2 WhatsOnSale.com

It works similar to the above example. It provides information regarding sales, deals and promotions but since it does all that it is cluttered for user to find just sales of a specific brand. And it does not take the user to the brand site.

1.1.2 Issues with Existing Systems:

- Most sites are not dynamic in a way that they require manual maintenance.
- Existing systems do not have proper categories for individual brands.
- The existing systems focus more on collective brand sale rather than individual product sales.

1.2 Problem Statement

Most of the brands do their marketing on their own websites or by television ads, etc. But there is no common platform for these sales where user can view all brands. User have to go to specific websites for the information they need. It will be very helpful for user if they can access all that information at one place.

1.3 Solution

Our proposed solution will be a system through which people can access all the information they want on brand's sale at one place. The system will have a web interface as a front end and at backend there will be a scrapper that will scrap the required brand's website and get the required data from that website to Mongo Database. From there that data will be displayed in our Web App.

The system provides following features:

- Auto update of sales
- Easier access to brand sales
- Redirect to official website for more info

1.4 Advantage of Proposed System

- Unlike common system, this application works on automated asynchronous node calls.
- Users will get to access all brand sales at one place
- Users will be able to set their favorite brands so that they see more sales from brands of their choice.

Chapter # 2

Requirements and System Analysis

2 Requirements and System Analysis

Requirements Analysis is also called requirements engineering, a process in which “what is to be done” is elicited, modeled and communicated. The descriptions of the services and constraints are the requirements for the system the process of finding, analyzing, documenting and checking these services and constraints. The first use of the term 'requirements engineering' was probably in 1979 in a TRW technical report but did not come into general use until the 1990s with the publication of an IEEE Computer Society tutorial and the establishment of a conference series on requirements engineering. In the waterfall model, requirements engineering is presented as the first phase of the development process. Later software development methods, including the Rational Unified Process, Extreme Programming and Scrum assume that requirements engineering continues through the lifetime of a system.

In the following pages I have inserted the Use Case Model for the system and detailed use case descriptions for the system.

2.1 Use Case Model

Use case model gives us information in a visual form about the system and its environment. In a quick glance it can give us the idea about how the user is interacting with the system. This generally include a UML use case diagram to show the name of use case and actors, and their relationships.

2.1.1 Use Case Diagram

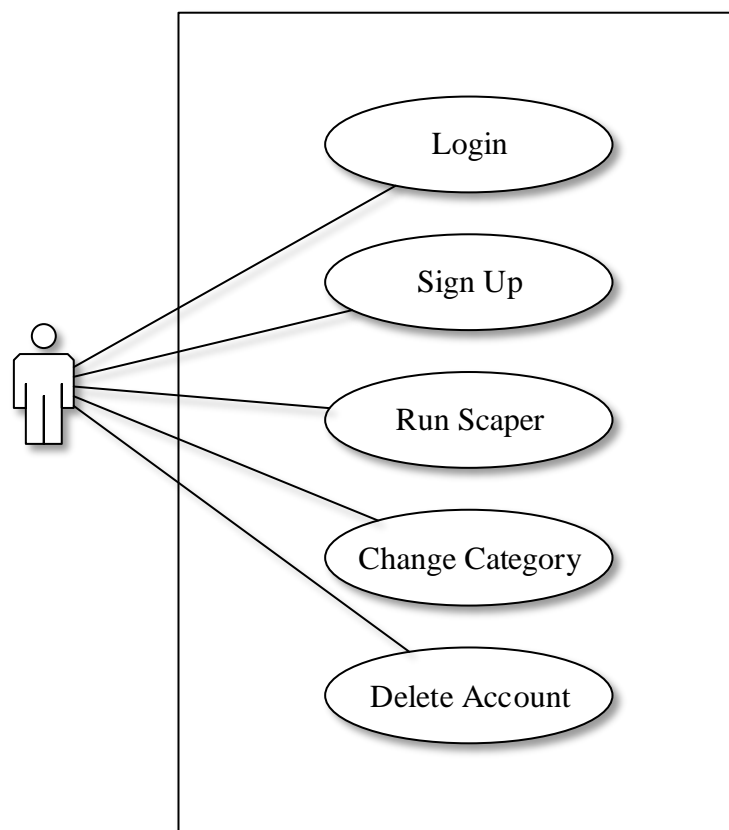


Figure 2.1.1 – Use Case diagram

2.2 Use Case Brief Detail

2.2.1 Login

Use Case ID:	UC-001
Use Case Name:	Login
Description:	This use case describes the process by which users can login
Actors:	User

2.2.2 Sign up

Use Case ID:	UC-002
Use Case Name:	Sign up
Description:	This use case describes the process by which users can Signup
Actors:	User

2.2.3 Run Scraper

Use Case ID:	UC-003
Use Case Name:	Run Scraper
Description:	This use case describes the process by which system auto scrap sales data from specified sites.
Actors:	User

2.2.4 Change Category

Use Case ID:	UC-004
Use Case Name:	Change Category
Description:	This use case describes how user can toggle category and change
Actors:	User

2.2.5 Delete Account

Use Case ID:	UC-005
Use Case Name:	Delete Account
Description:	This use case describes how user can delete their Kappas account
Actors:	User

2.3 Use Case Description Detail

2.3.1 Login

Use Case ID:	UC-001
Use Case Name:	Login
Description:	This use case describes the process by which users can login
Actors:	User
Pre-condition:	<ul style="list-style-type: none"> - Working internet connection - Web App is fully loaded
Post-condition:	User will be redirected to homepage
Includes:	Web App
Flow:	<ol style="list-style-type: none"> 1. User will click on login 2. User will enter credentials 3. After verification user is logged in
Exception:	N/A
Frequency of use:	Once per session

2.3.2 Sign Up

Use Case ID:	UC-002
Use Case Name:	Sign Up
Description:	This use case describes the process by which users can Signup
Actors:	User
Pre-condition:	<ul style="list-style-type: none"> - Working internet connection - Web App is fully loaded

Post-condition:	User will be redirected to homepage
Includes:	Web App
Flow:	<ol style="list-style-type: none"> 4. User will click on Sign Up 5. User will enter credentials 6. After verification user is Sign Up
Exception:	N/A
Frequency of use:	Seldom

2.3.3 Run Scraper

Use Case ID:	UC-003
Use Case Name:	Run Scraper
Description:	This use case describes the process by which system auto scrap sales data from specified sites
Actors:	User
Pre-condition:	- Working internet connection
Post-condition:	Results will be stored in database and displayed on the web app
Includes:	Web App
Flow:	<ol style="list-style-type: none"> 1. System will scrap the specified sites 2. Scraped data will be stored in database 3. Data from the mongo will be displayed on web app
Exception:	- Specified sites are unreachable
Frequency of use:	Once or twice per day

2.3.4 Change Category

Use Case ID:	UC-004
Use Case Name:	Change Category
Description:	This use case describes how user can select category and change
Actors:	User
Pre-condition:	<ul style="list-style-type: none"> - Working internet connection - Web App is fully loaded
Post-condition:	Settings will be changed as per user's need

Includes:	Web App
Flow:	<ol style="list-style-type: none"> 1. User will go to category page 2. User will select category 3. User will change category 4. Web App will refresh to reflect changes
Exception:	- N/A
Frequency of use:	Seldom

2.3.5 Delete Account

Use Case ID:	UC-005
Use Case Name:	Delete Account
Description:	This use case describes how user can delete their Kappas account
Actors:	User
Pre-condition:	<ul style="list-style-type: none"> - Working internet connection - User is logged in - Web App is fully loaded
Post-condition:	User's account will be deleted
Includes:	Web App
Flow:	<ol style="list-style-type: none"> 1. User will go to Profile page 2. User will click on Delete Account 3. Web App will request for authentication 4. User will authenticate 5. Web App will delete user account
Exception:	- MongoDB is not working
Frequency of use:	Rare

2.4 Activity Diagram

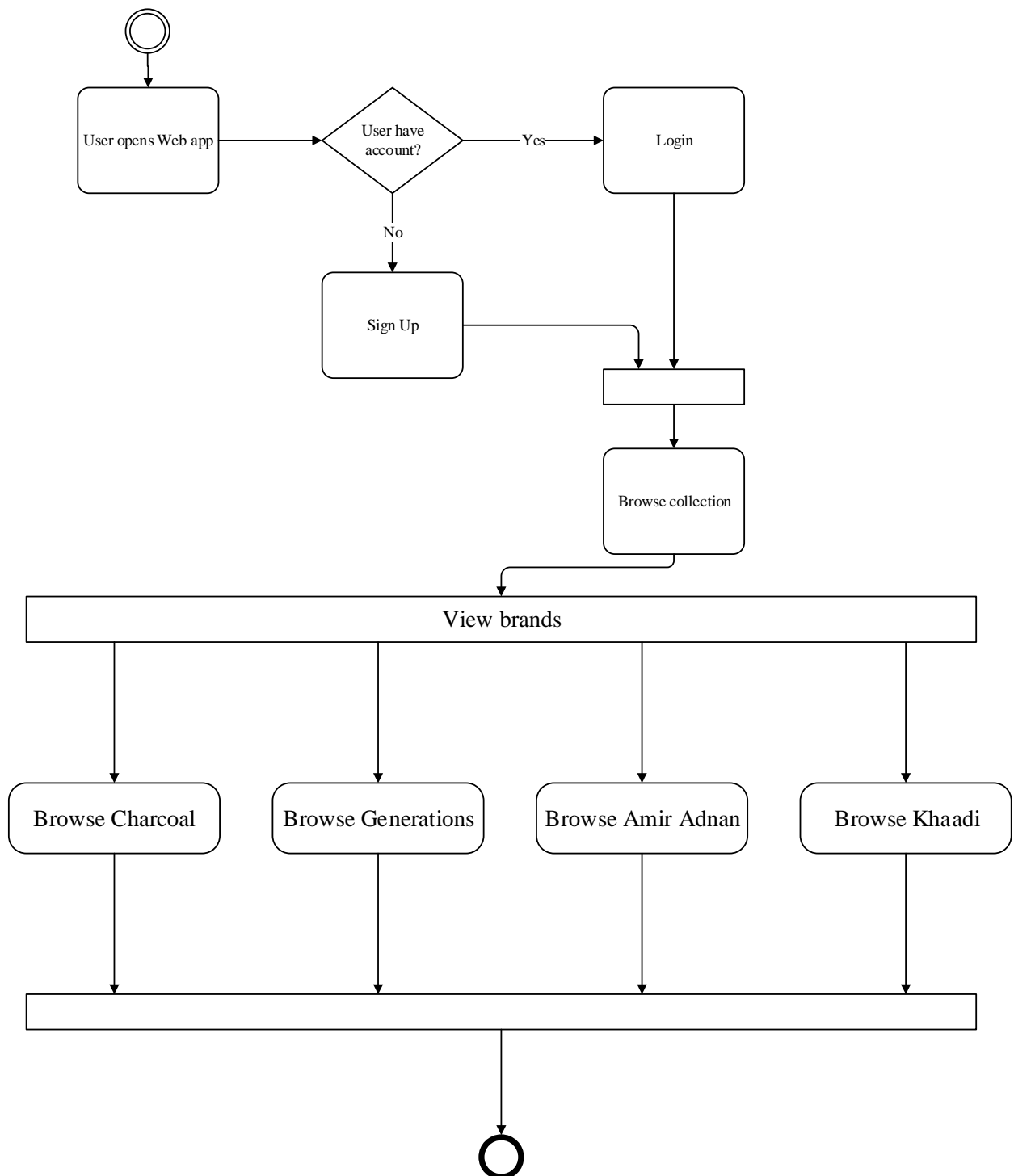


Figure 2.4 – Activity Diagram

Chapter # 3

System Design

3 System Design

I have designed our system in a way that it consists of three different parts (modules). One part is Client-Side, the next one is the Web Server and the third part is web Scraper Module. The following diagram shows different components of the system:

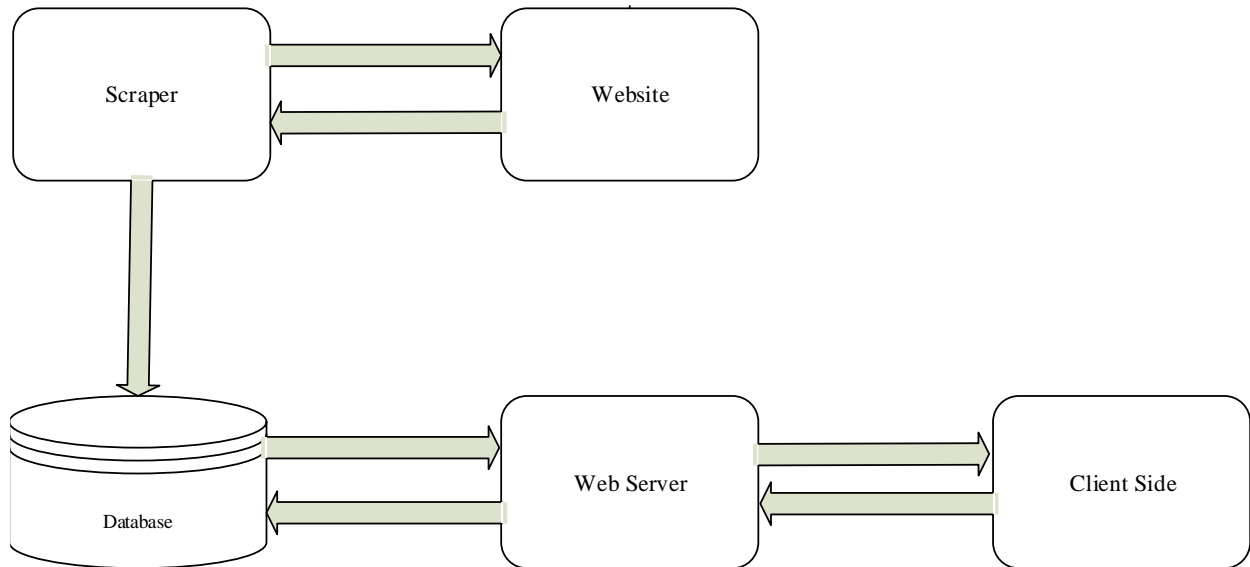


Figure 3 - System Design

Kappas works in a way that, the user will interact with the Client-side, Scraper module will let you scrap the data from various website and store it in the database. Client-side will then get the information from the database and display it to user.

Below are the phases of data analysis and different diagrams that show interactions between the user and different components of the system.

3.1 Phases of Data retrieval

I have divided our data analysis portion in four different phases, namely: data collection, feature extraction, personality assessment and occupation detection.

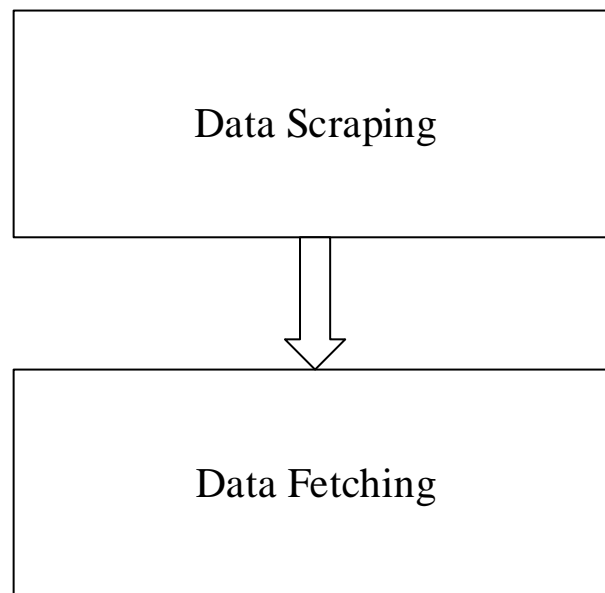


Figure 3.1– Data retrieval Phases

3.1.1 Data Scraping

The dataset I obtained will be scraped from various website such as Khaadi, Generation and Amir Adnan etc. using node.js module X-ray and storing that data in MongoDB using another node.js module mongoose. This process will run once a day.

3.1.2 Data fetching

The data will then be fetched from MongoDB and displayed on Web app using handlebars. Where user can interact with data.

3.2 Sequence Diagrams

A sequence diagram is a kind of interaction diagram that shows how processes operate with one other and in what order. It is a construct of a message Sequence Chart. A sequence diagram shows object interaction arrange in time sequence.

3.2.1 Sign in

This sequence diagram describes the process by which user can sign in to the system.

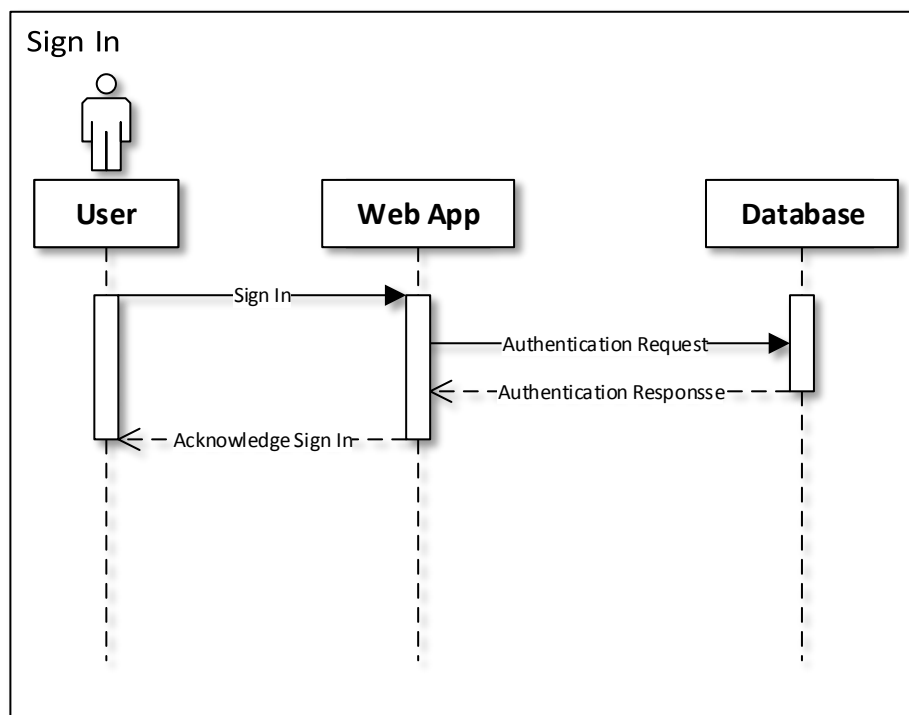


Figure 3.2.1 - Sign In

3.2.2 Sign Up

This sequence diagram describes the process by which user can sign up.

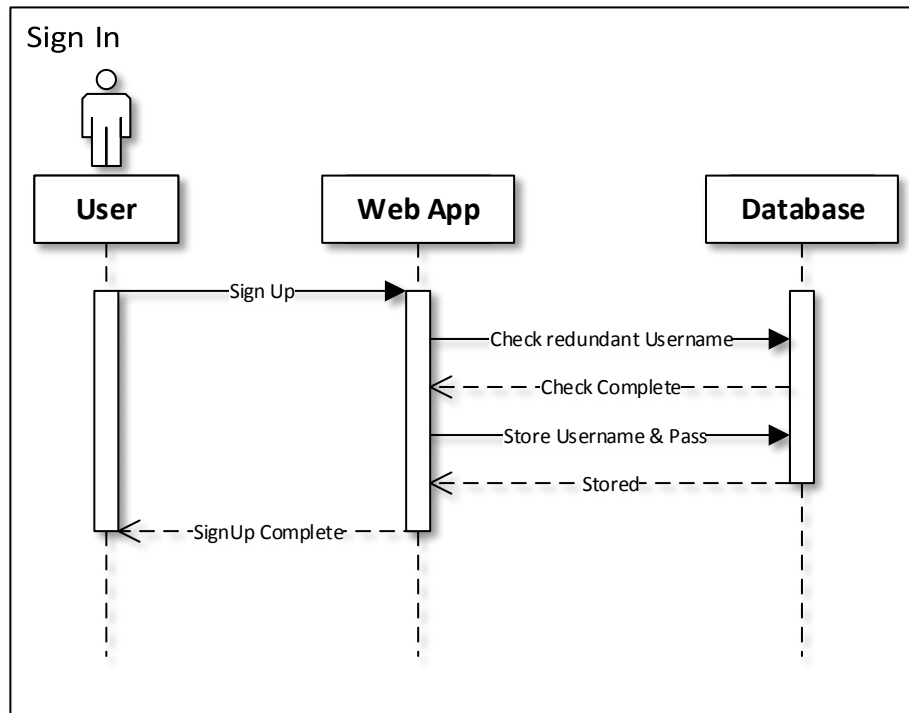


Figure 3.2.2 - Sign up

3.2.3 Scrap data

This sequence diagram describes how the Scraper will get data from websites.

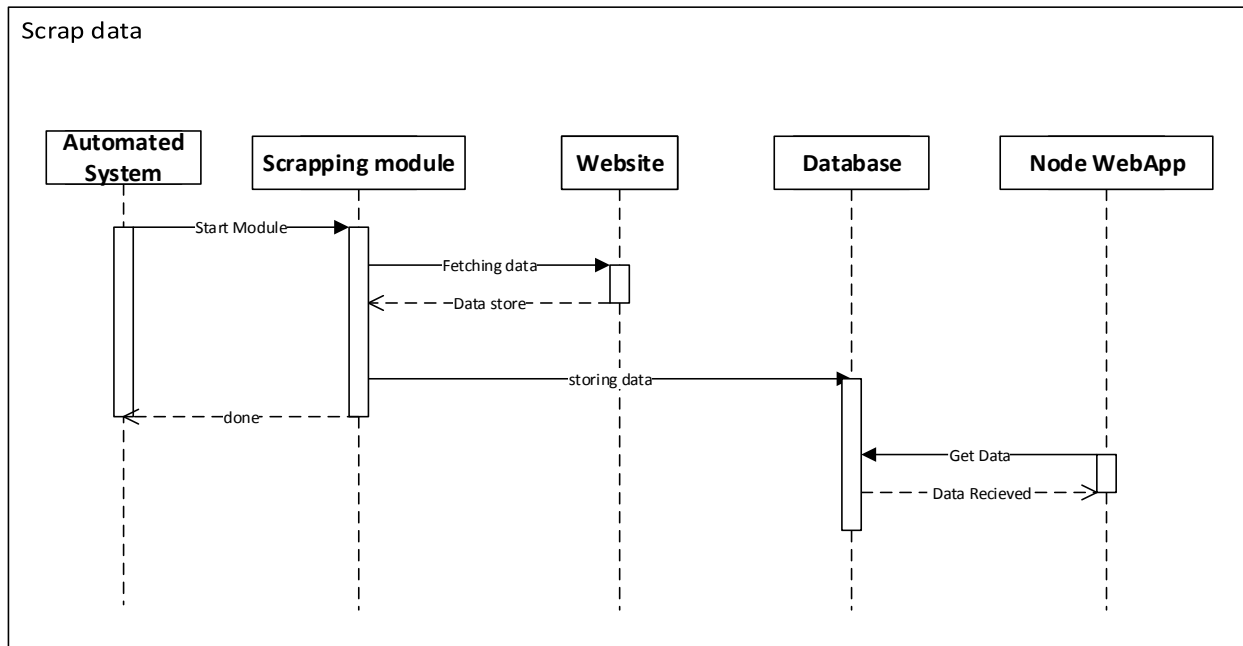


Figure 3.2.3 - Get Results

3.2.4 Change Category

This sequence diagram describes how the user is going to change their app settings.

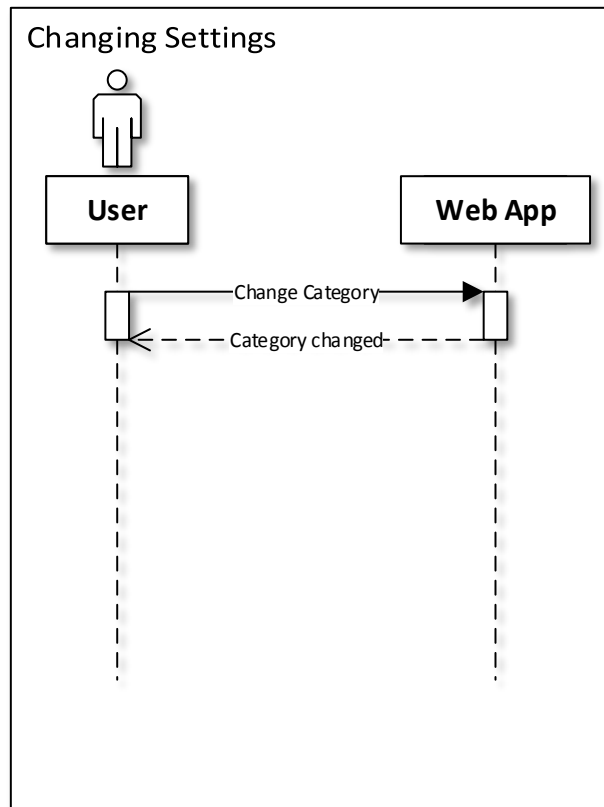


Figure 3.2.4 – Change Settings

3.2.5 Delete Account

This sequence diagram describes how the user can delete their account.

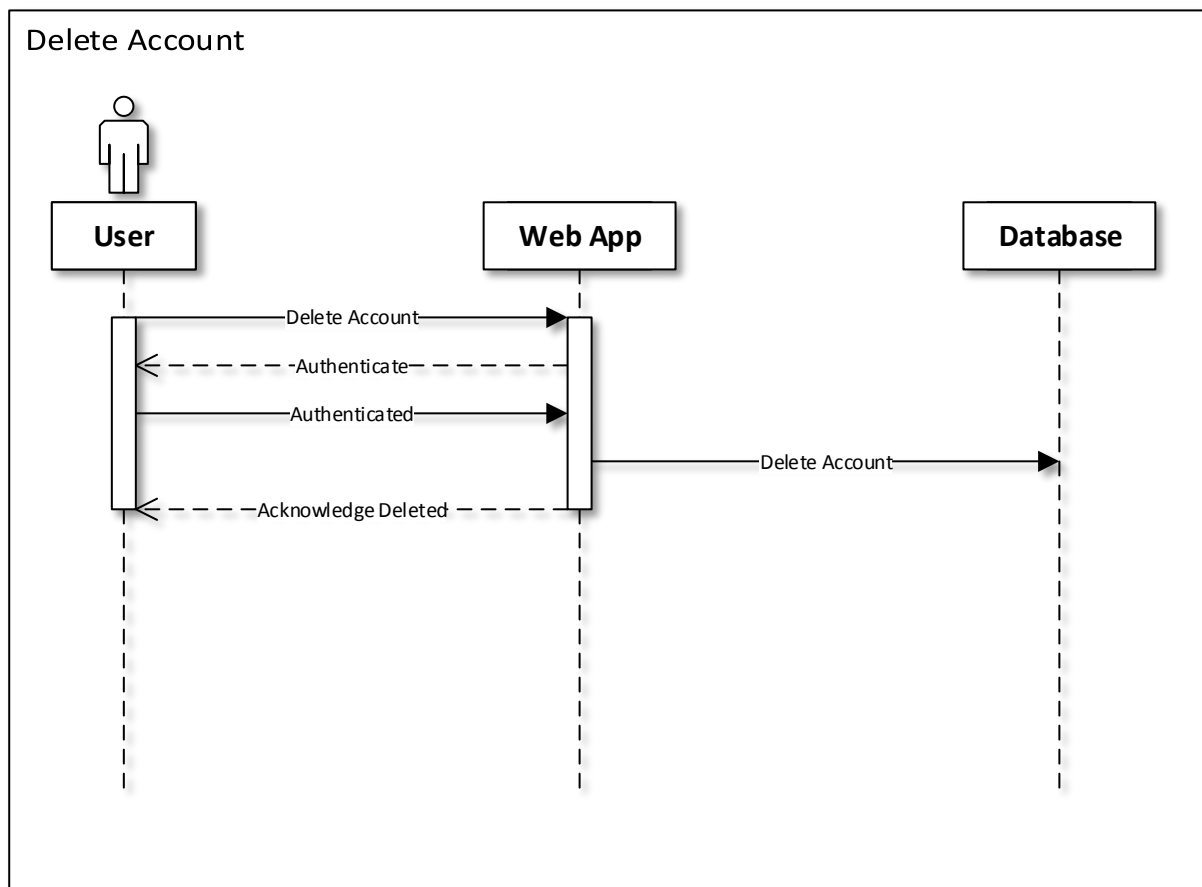


Figure 3.2.5 – Delete Account

Chapter # 4

System Implementation

4 System Implementation

I have implemented Kappas using different modern technologies and techniques. The choice of the technologies used was based on the principle of modularity and performance. I choose Handlebars.js for the client-side, Node.js for the web server backend, Express.js for the web server framework, MongoDB as the database, for Data scraping part I used Node.js module X-ray and I also used Github for code collaboration and version control. Most notable Node.js libraries that I used are: X-ray, Mongoose.

Now with the technologies implemented the system design diagram that I have shown in chapter 3 looks like this:

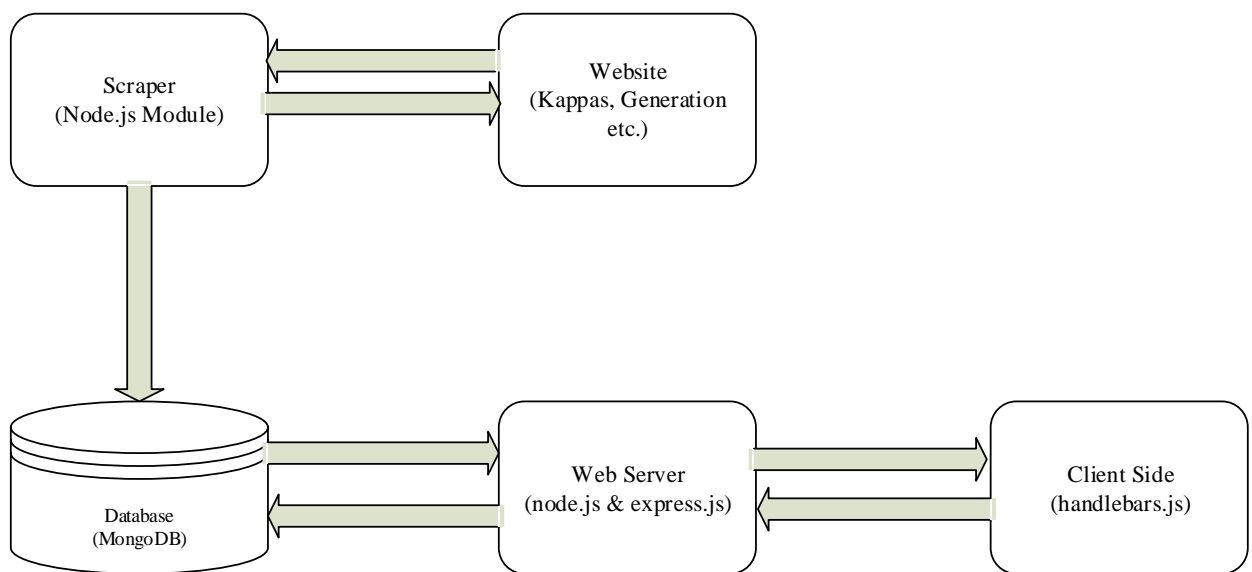


Figure. 4 - System Design

4.1 Deployment Diagram

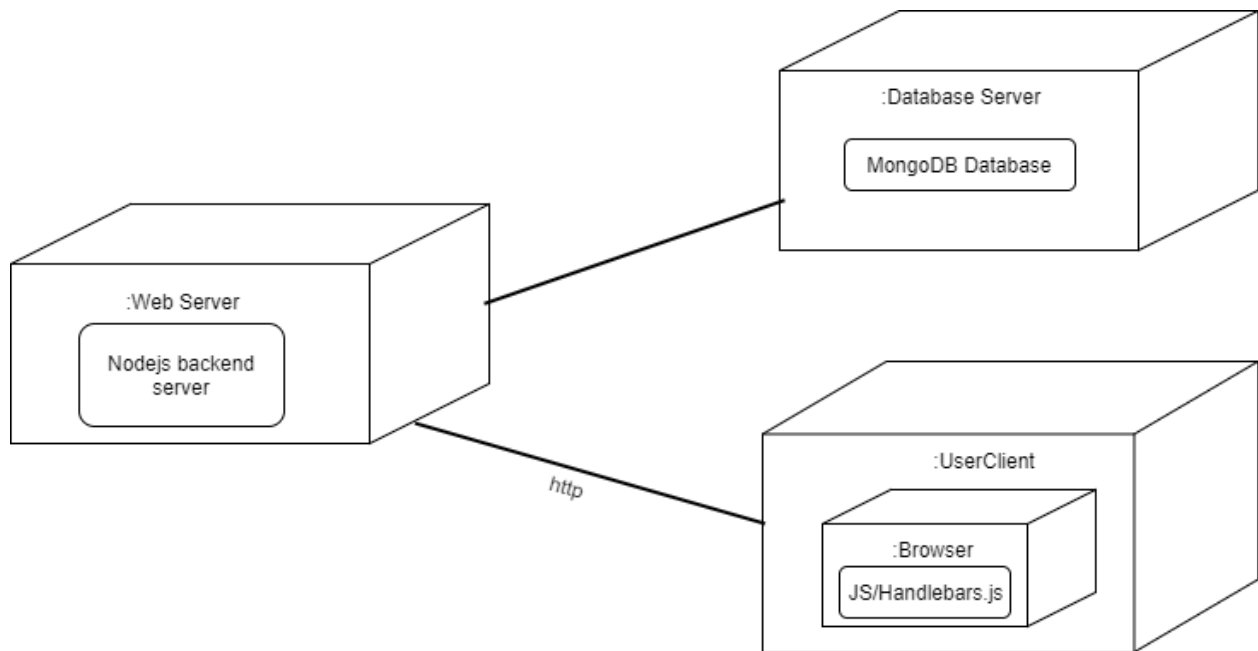


Figure 4.2 – Deployment Diagram

4.2 Details of Implementation Tools and Technologies

The details of the languages, tools and technologies I used are following:

4.2.1 Node.js

Node.js is a JavaScript runtime environment that is cross-platform, open-source and built upon Google's V8 Engine. It uses an event driven non-blocking (asynchronous) model. It is mostly written in C and C++ with some modules written in JavaScript. It is engineered to be used in high performance highly scalable application. Node.js can be used to create a wide variety of tools and application, but it mostly being used for modern web application development and APIs. It interprets JavaScript using the V8 engine and all node.js apps are programmed in JavaScript. It was built by Ryan Dahl and first released on May 27th 2009.

I choose node.js because of its modular nature and its use of JavaScript as the programming language which makes development easy because on both front-end and backend I am using the same language and a development does not need to use multiple languages to work on the whole application stack.

4.2.2 Express.js

Express is a minimalistic web application framework for Node.js. It is free and open-source. Out of the box it only provides the most basic modules and follows a modular approach so the developers can extend it as they wish using node packages. It is opinionated and allows the developers to develop the way they want to. Express is designed specifically for web application and APIs. It was first developed by TJ Holowaychuk and released on November 16th 2010.

4.2.3 Handlebars.js

Handlebars is a templating engine, it is basically an extension of the popular Mustache templating engine. It allows us to easily create dynamic pages by injecting our content dynamically from the server-side. It is a compiler that takes any HTML and Handlebars expression and compiles them to a JavaScript function. This derived JavaScript function then takes one parameter, an object and it returns a string with the HTML and the object properties' values inserted into the HTML. So, I end up with a string that has the values from the object properties inserted in the relevant places, and you insert that string as HTML on the page.

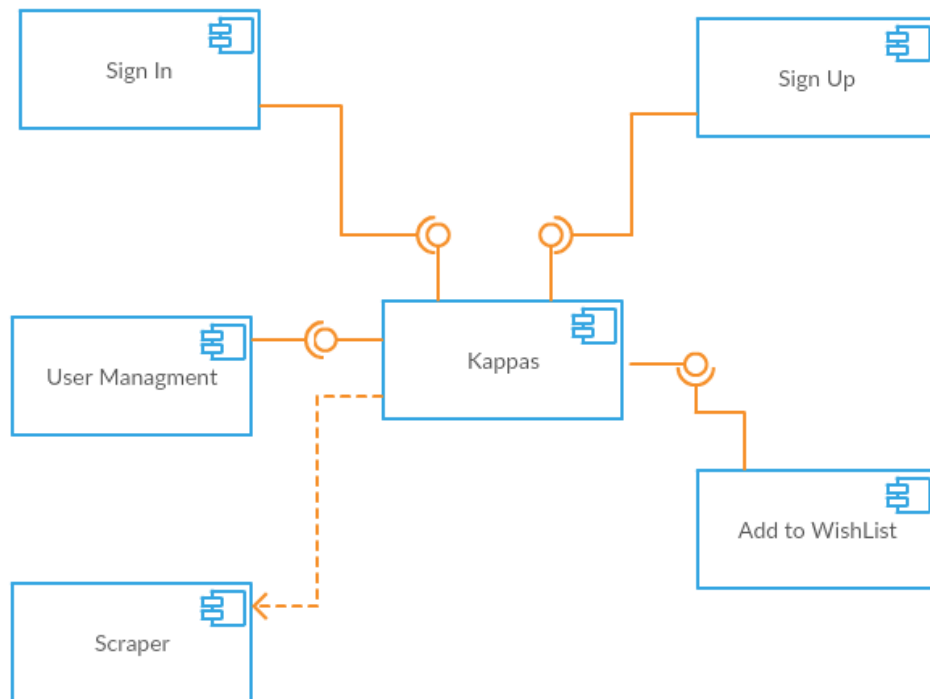
4.2.4 MongoDB

MongoDB is a Document-Oriented NoSQL Database. It is cross-platform and open-source. Mongo uses JSON like documents with schemas and stores it in the form of BSON or simply put a binary encoded format. It is also mostly written in C/C++ with some parts in javascripts and because of this low level implementation and document oriented nature it is very performant and highly scalable.

4.2.5 Github

GitHub is a web-based Git repository hosting service. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

4.3 Component diagram



4.4 Project Screenshots

Below are some of the screenshot of our project.

4.4.1 Homepage

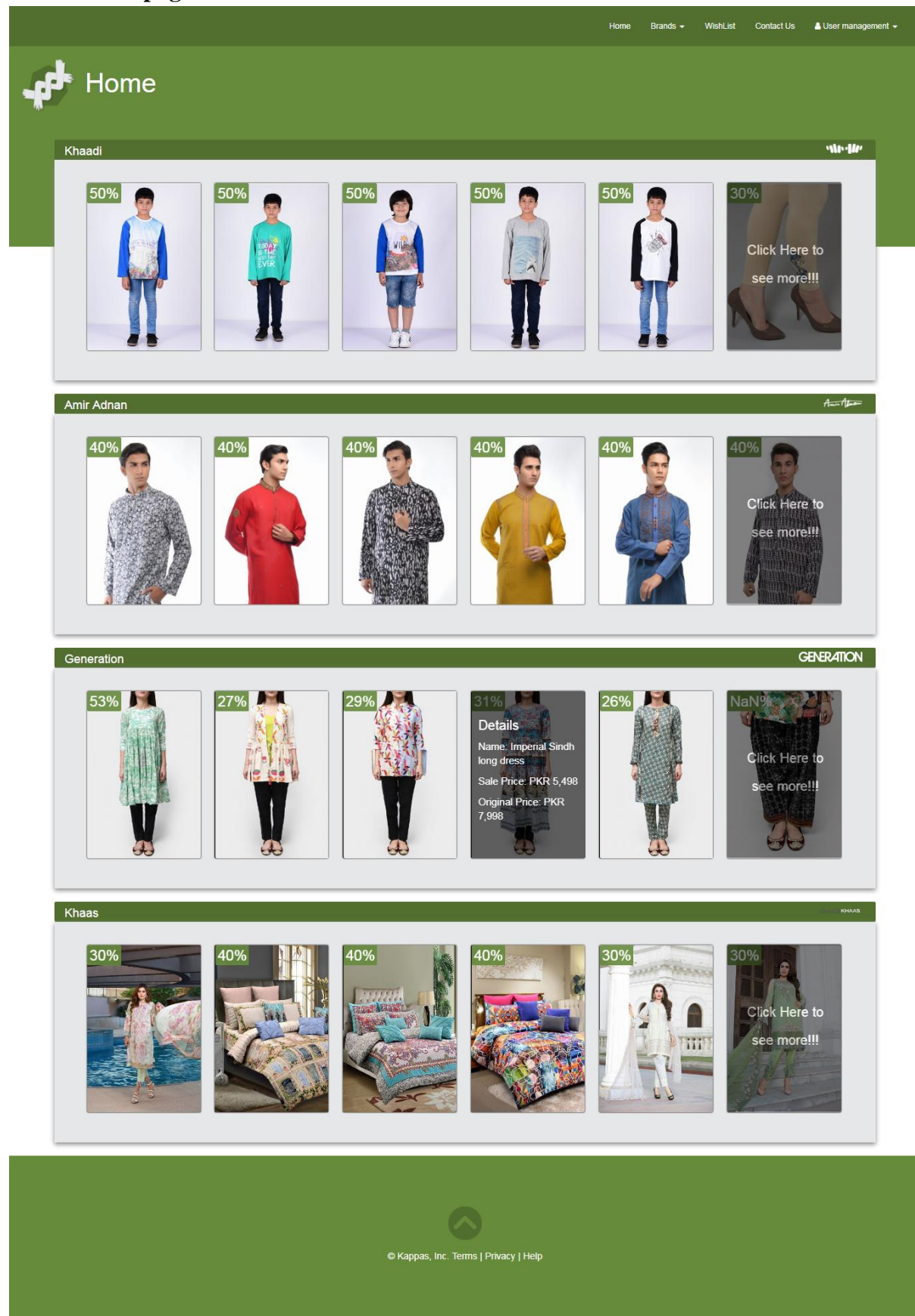
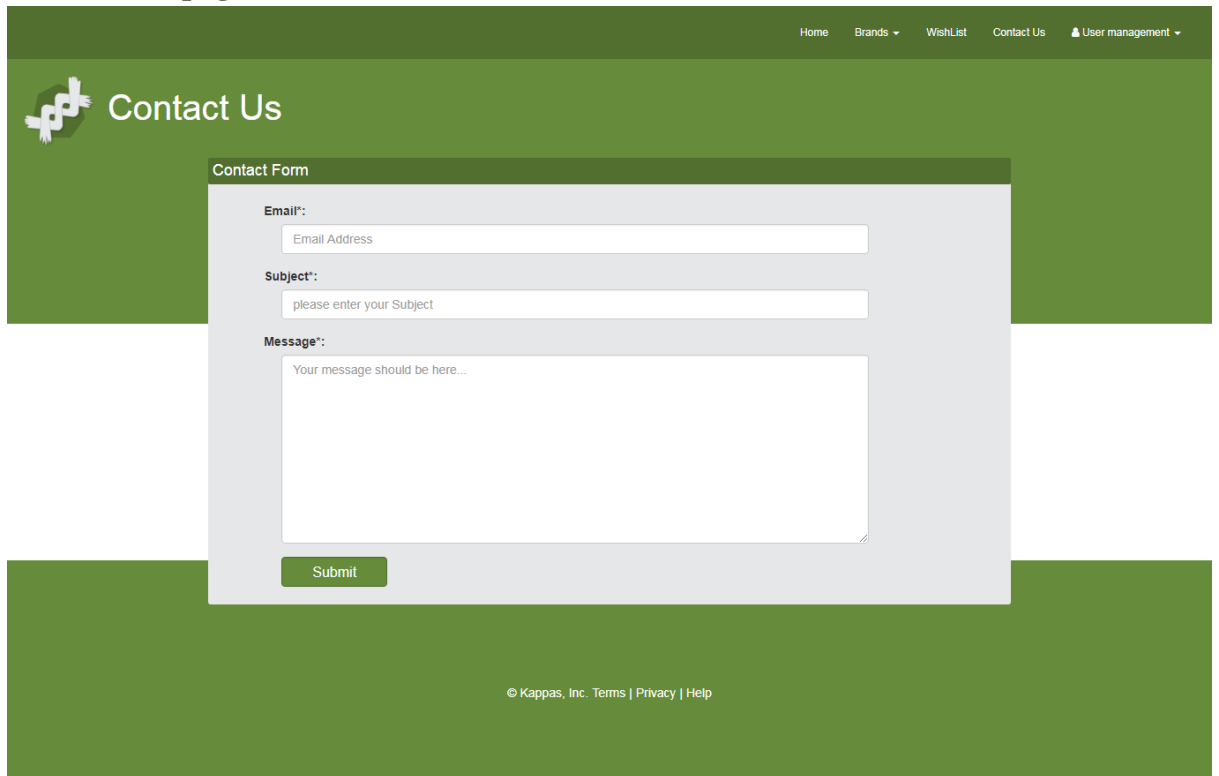


Figure 4.4.1 - Homepage

4.4.2 Contact us page



The screenshot shows a web page with a green header and footer. The header contains navigation links: Home, Brands, WishList, Contact Us, and User management. The main content area has a green background with a white contact form titled "Contact Form". The form includes fields for Email, Subject, and Message, and a Submit button. The footer contains the copyright notice: © Kappas, Inc. Terms | Privacy | Help.

Home Brands WishList Contact Us User management

Contact Us

Contact Form

Email*:
Email Address

Subject*:
please enter your Subject

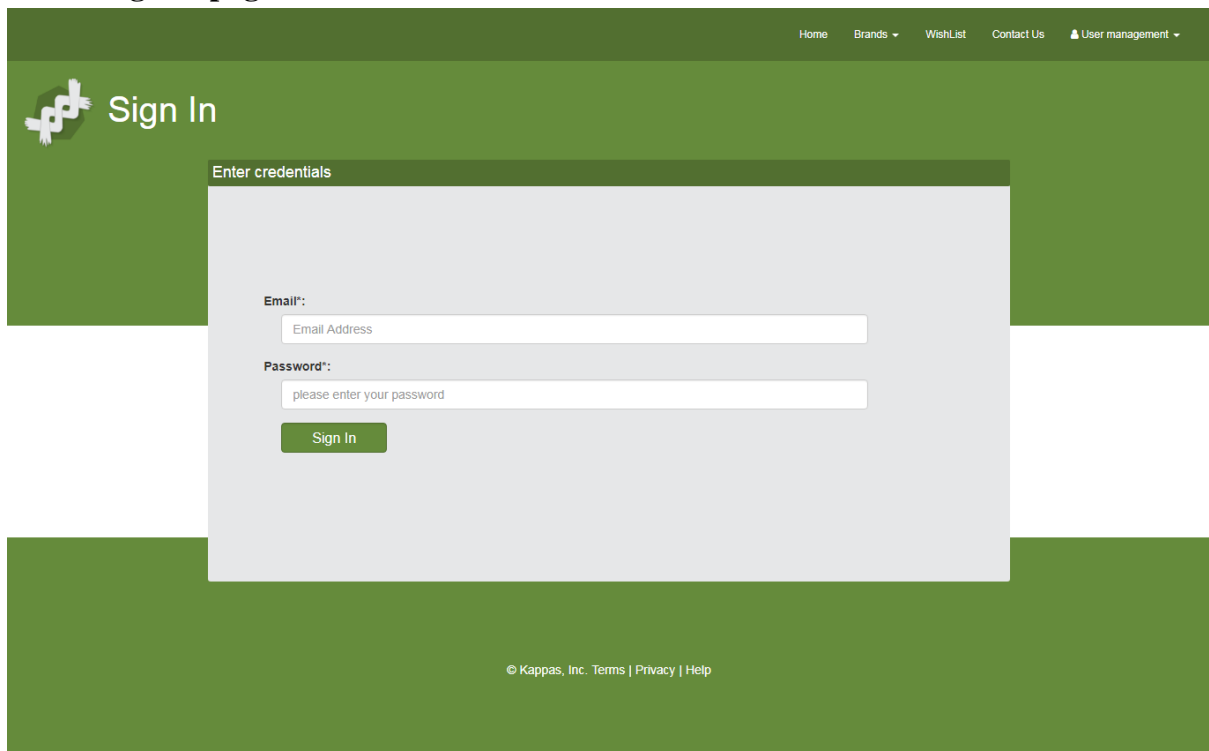
Message*:
Your message should be here...

Submit

© Kappas, Inc. Terms | Privacy | Help

Figure 4.4.2 – Contact us Page

4.4.3 Sign In page



The screenshot shows a web page with a green header and footer. The header contains navigation links: Home, Brands, WishList, Contact Us, and User management. The main content area has a green background with a white sign-in form titled "Enter credentials". The form includes fields for Email and Password, and a Sign In button. The footer contains the copyright notice: © Kappas, Inc. Terms | Privacy | Help.

Home Brands WishList Contact Us User management

Sign In

Enter credentials

Email*:
Email Address

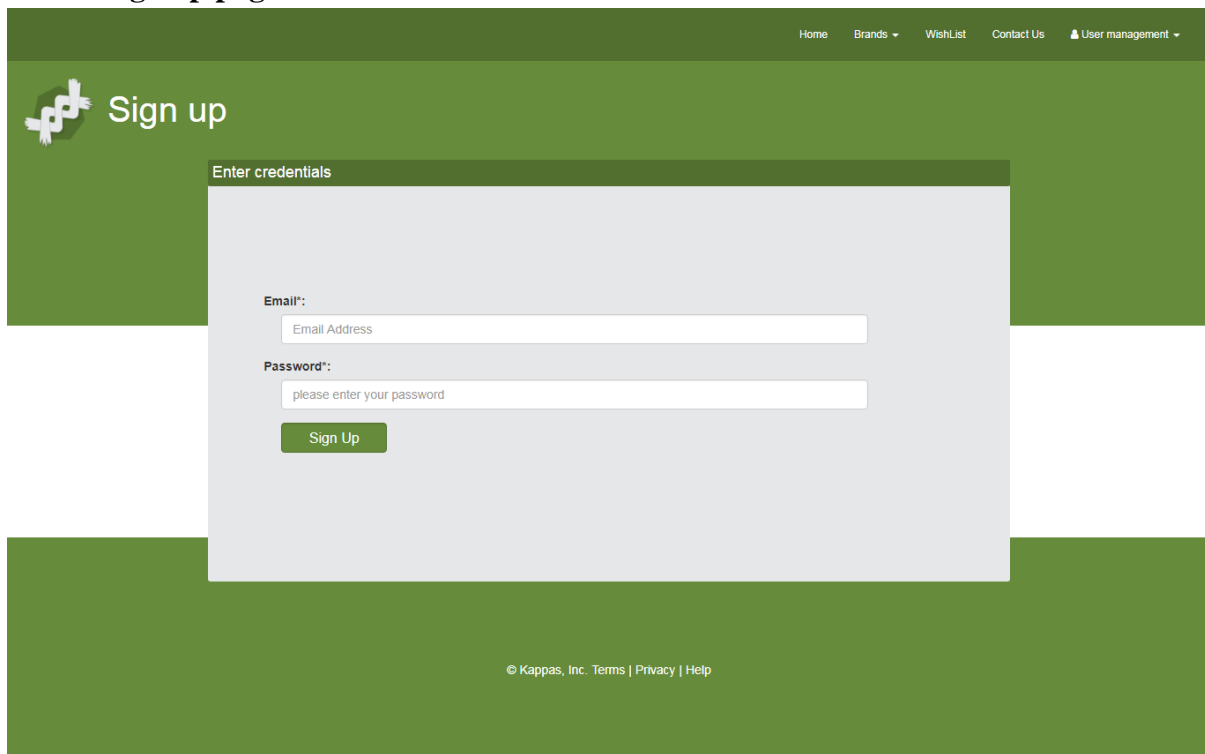
Password*:
please enter your password

Sign In

© Kappas, Inc. Terms | Privacy | Help

Figure 4.4.3 – Sign in page

4.4.4 Sign up page



The screenshot displays a web application's sign-up interface. At the top, a dark green navigation bar contains links for Home, Brands, WishList, Contact Us, and User management. Below this, a green header section features a logo and the text 'Sign up'. The main content area is a light gray box titled 'Enter credentials' which contains two input fields: 'Email*' with a placeholder 'Email Address' and 'Password*' with a placeholder 'please enter your password'. A green 'Sign Up' button is positioned below the password field. At the bottom of the page, a green footer bar contains the copyright notice '© Kappas, Inc. Terms | Privacy | Help'.

Figure 4.4.4 – Sign up page

4.4.5 WishList

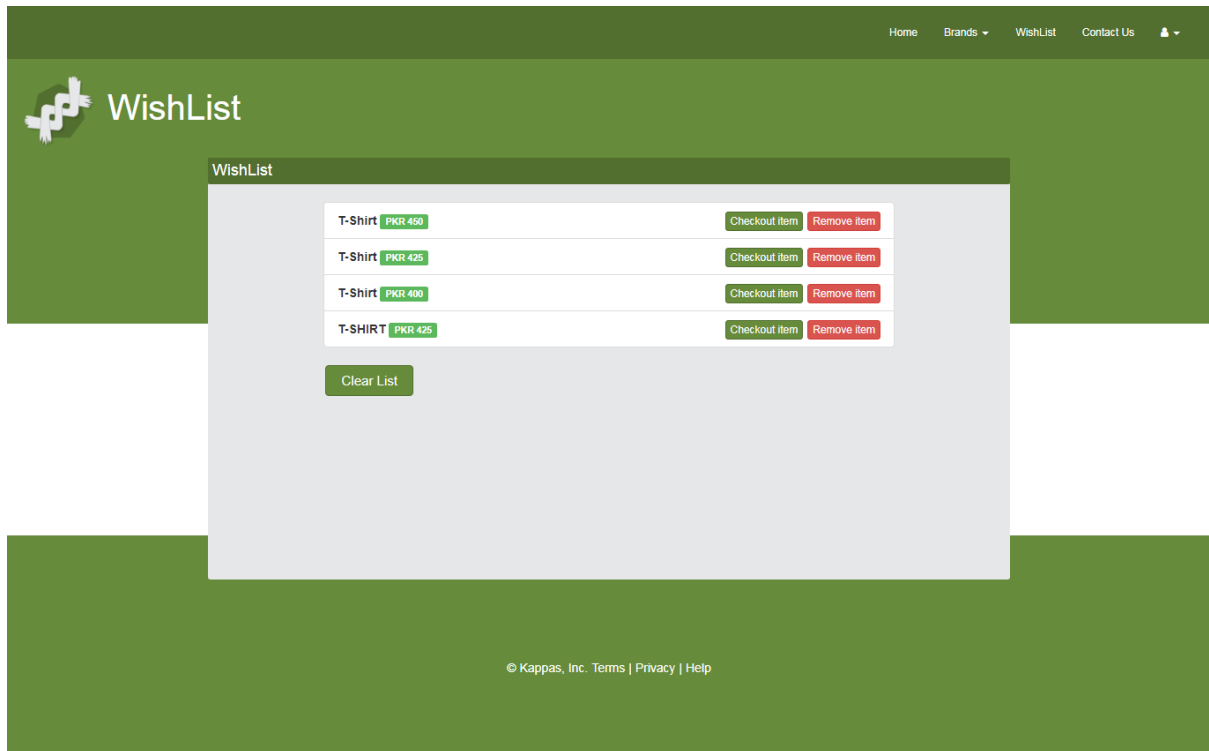


Figure 4.4.5 – WishList page

Chapter # 5

System Testing

5 System Testing

5.1 Software Quality Assurance

A set of activities designed to evaluate the process by which products are developed or manufactured.

5.2 Software Quality Control

Software Quality Control is the function that checks whether the software project follows its standards processes, and procedures, and that the project produces the desired internal and external (deliverable) products i.e. output.

5.2.1 Black Box Testing

Black box testing is also known as specification-based testing. Black box testing refers to test activities using specification-based testing methods and criteria to discover program errors based on program requirements and product specifications.

The major testing focuses:

1. Specification-based function errors
2. Specification-based component/system behavior errors
3. Specification-based performance errors
4. User-oriented usage errors
5. Black box interface errors

5.3 Test Case

Following are the Test Cases for our project (Kappas):

5.3.1 Login

TC1: Login	
Test Case ID:	TC-001
Wrote By:	Haseeb-ur-Rehman
Test Type:	Black box testing
Product Name:	Kappas
Test Item:	Web App
Documented Date:	07/10/2017
Test Suite:	1a
Version Number:	1.0

Test case description:	This use case describes the process by which users can login
Operation procedure:	<ol style="list-style-type: none"> 1) Go to homepage of Kappas 2) User will click on login 3) User will enter credentials 4) After verification user is logged in
Pre-conditions:	<ul style="list-style-type: none"> - Working internet connection - Web App is fully loaded
Post-conditions:	User will be redirected to homepage
Required test scripts:	No

5.3.2 Signup

TC2: Sign Up	
Test Case ID:	TC-002
Wrote By:	Haseeb-ur-Rehman
Test Type:	Black box testing
Product Name:	Kappas
Test Item:	Web App
Documented Date:	07/10/2017
Test Suite:	1a
Version Number:	1.0
Test case description:	This use case describes the process by which users can signup
Operation procedure:	<ol style="list-style-type: none"> 1) Go to homepage of Kappas 2) User will click on Sign Up 3) User will enter credentials 4) After verification user is signed up
Pre-conditions:	<ul style="list-style-type: none"> - Working internet connection - Web App is fully loaded
Post-conditions:	User will be redirected to homepage
Required test scripts:	No

5.3.3 Run Scraper

TC3: Run Scraper

Test Case ID:	TC-003
Wrote By:	Haseeb-ur-Rehman
Test Type:	Black box testing
Product Name:	Kappas
Test Item:	Web App, MongoDB, Node Modules
Documented Date:	07/10/2017
Test Suite:	1a
Version Number:	1.0
Test case description:	This use case describes the process by which system auto scrap sales data from specified sites
Operation procedure:	<ol style="list-style-type: none"> 1) System will scrap the specified sites 2) Scraped data will be stored in database 3) Data from the mongo will be displayed on web app
Pre-conditions:	- Working internet connection
Post-conditions:	Results will be stored in database and displayed on the web app
Required test scripts:	No

5.3.4 Change Category

TC4: Change Category	
Test Case ID:	TC-004
Wrote By:	Haseeb-ur-Rehman
Test Type:	Black box testing
Product Name:	Kappas
Test Item:	Web App, MongoDB
Documented Date:	07/10/2017
Test Suite:	1a
Version Number:	1.0
Test case description:	This use case describes how user can toggle preference and change
Operation procedure:	<ol style="list-style-type: none"> 1) User will go to Category page 2) User will select Category 3) Web App will refresh to reflect changes
Pre-conditions:	- Working internet connection

	<ul style="list-style-type: none"> - User is logged in - Web App is fully loaded
Post-conditions:	Category will be changed as per user's need
Required test scripts:	No

5.3.5 Delete Account

TC5: Delete Account	
Test Case ID:	TC-005
Wrote By:	Haseeb-ur-Rehman
Test Type:	Black box testing
Product Name:	Kappas
Test Item:	Web App, MongoDB
Documented Date:	07/10/2017
Test Suite:	1a
Version Number:	1.0
Test case description:	This use case describes how user can delete their Kappas account
Operation procedure:	<ol style="list-style-type: none"> 1) User will go to profile page 2) User will click on Delete Account 3) Web App will request for authentication 4) User will authenticate 5) Web App will delete user account
Pre-conditions:	<ul style="list-style-type: none"> - Working internet connection - User is logged in - Web App is fully loaded
Post-conditions:	User's account will be deleted
Required test scripts:	No

Chapter # 6

Conclusion

6 Conclusion

In this project I have developed a system in the form of a web application that can show product sales from various brands in Pakistan. Whereas previous works focused on general brand sales instead of focusing on products. It also notifies the users based on their preference of brands. The system will scrap data from various known brands such as Khaadi, Amir Adnan etc. Using node.js and mongoose it will store that scrapped content to mongodb and will asynchronously refresh this data. Also by using handlebars.js the data will be retrieved from mongodb to web app. This system will help users by finding all sales in one place instead of going to different websites and searching for specific products/sales.

7 References

1. <https://www.udemy.com/the-complete-nodejs-developer-course-2/>
2. <https://www.udemy.com/the-complete-web-developer-course-2/>
3. <https://www.udemy.com/the-complete-react-web-app-developer-course>