Integration of Multiple Services using Application Programming Interface

By

Nopa Islam, Student ID: 011 151 062
Sukannya Saha, Student ID: 011 151 079
Ajmaree Sultana, Student ID: 011 151 081
Nasir Uddin Ahmed, Student ID: 011 151 086
Prity Lata Chowdhury, Student ID: 011 151 091

Submitted in partial fulfilment of the requirements
of the degree of Bachelor of Science in Computer Science and Engineering

October 17, 2019

Department of Computer Science and Engineering
United International University

Abstract

Application programming interface consists of functions, communication protocols to build software or application like building blocks. A good application programming interface makes it easier for the developers to make application faster. Nowadays programming interface is considered as a product. Service means using different modules. Module actually encapsulates code and data to implement in a particular functionality. Module is usually packaged in a single unit so that it can be easily deployed. We have studied different e-commerce platform like WooCommerce, BigCommerce, OpenCart. Shopify platform supports numerous readymade E-commerce features. Using those features in an application, with custom coding save development time and cost. As a result development cycle reduces. In the modern industry level custom coding and developing application quickly is a must. It also gives the independence of using different types of database, different payment services including local payment methods to grow and empower business quickly. Besides, these update of application and integrating new features become easier. In this project we developed an application, where we have used Shopify APIs such as Storefront API, GraphQL Admin API, Shipping and Fulfillment API, Products API, Online Store API, Customers API to make a functional e-commerce application for the user. We also developed product routes, and our systems own API to handle API requests in the server side.

Acknowledgements

We acknowledge the guide and support of **Mr. Abu Shafin Mohammad Mahdee Jameel** (Assistant Professor, Department of Computer Science and Engineering, Course Teacher of our Final Year Design Project), **Mohd Moniruzzaman** (Assistant Professor, Department of Computer Science and Engineering, Mentor of our Final Year Design Project), **Prof Dr. Salekul Islam** (Professor Head, Department Of Computer Science And Engineering), and **Dr. Swakkhar Shatabda** (Associate Professor Undergraduate Program Coordinator, Department of Computer Science and Engineering). We would like to express our gratitude to everyone who contributed to it in anyway. We owe to our family including our parents for their unconditional love and immense emotional support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The following chapter focuses on the introductory part of the project. The chapter starts with project overview, which gives a basic outline of the project.

1.1 Project Overview

An application Programming Interface (API) is a set of functions [1]. It has Communication Protocol and tools for building software. Application Programming Interfaces are called clearly defined methods, which is used for communication among various components. By following building blocks [2], a good API shorts software development time. All enterprise has multiple applications. These applications are used to keep the record of key data, business logic and business data. The web API gateway is the exposure point. It has single entry points, client specific APIS. So it makes the enterprise evolved with functions.

1.2 Motivation

In Bangladesh there are mainly two types of e-commerce application. First one is integrated system. Second one is interface. Interface works as a bridge to allow two programs to share information. Interface is less flexible and harder to correct. In the back end of a web application there are many services. These services are complex. Without API we have to build those services in real time which is costly. And it increases development time. Integrated system do not pass information between two systems over a bridge. Integrated system uses application programming interface so that the systems share same code and database. Using APIs expose the server side which means the backend and it also exposes the business logic and data. API facilitates direct access to the functionality provided by the web site. It also leverages third party efforts to add value to existing services.

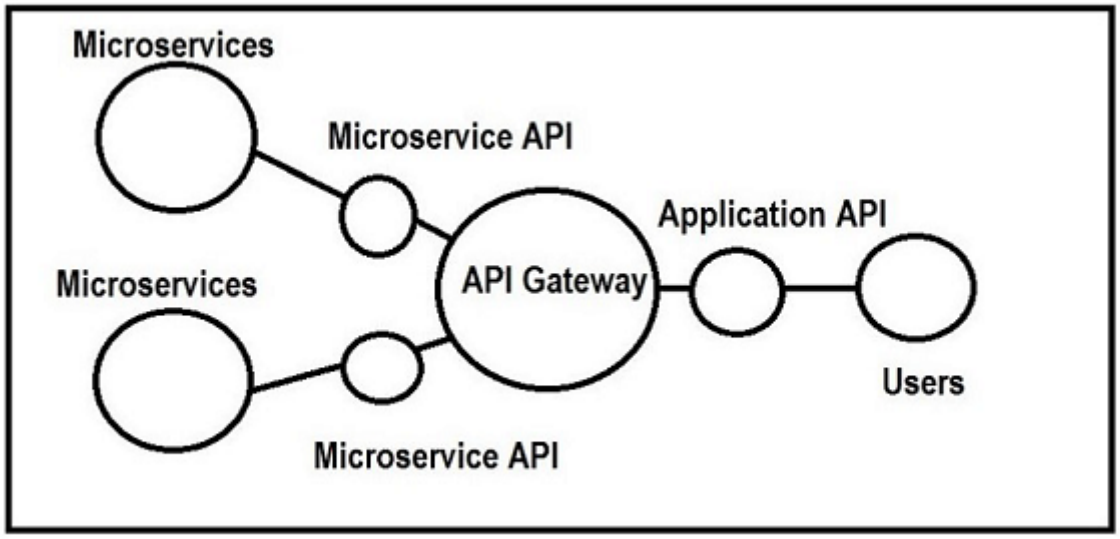

Figure 1.1: API Gateway

1.3 Objectives

Our main target is to implement multiple services by using API. Making a user friendly e-commerce application. Taking it on the world wide web so that the users can avail it. The policies for releasing APIs are private, partner and public. There will be a shop in shopify store, the shop will be connected with featured APIs. We will run our APIs in a web server. From the server and shopify we will conduct a bridge Which will have a database in the back. By calling APIs a service will run and perform functionality. API key will identify the source of request of services. Plus access token will contain security credentials.

1.4 Methodology

In this project the server client architecture is being followed. For the backend development, 6 steps have been followed. They are:
Step 1: Exposing the local environment to the internet.
Step 2: Creating and configuring application in the “Shopify Partner Dashboard”.
Step 3: Creating a Node.js project.
Step 4: Building the Application.
Step 5: Creating specific routing.
Step 6: Run the application.

1.5 Outcome

The outcome of the project is significant. Developing an e-commerce application is the outcome. User can see e-commerce products through the application. Buy products using the application. Shopify provides admin panel. So creating an SDK based application using APIs is the outcome of the project.

1.6 Organization of the Report

The report has been organized in total number of six chapters. In the first chapter Overview of the project is described. In the second chapter Background of the project is illustrated. Chapter three focuses on Project Design is being showed. Chapter four indicates Implementation and Result part. In chapter five Standards and Design constraints are illustrated. Chapter six concludes the whole project in a nutshell.

Chapter 2

Background

The following chapter describes the existing work. Focusing on the APIs related work for the project. It also describes the literature review in a proper manner.

2.1 Preliminaries

Developing an API based e-commerce application expose services to other applications [3]. Services like order management, pricing information, category content, customer profile data can be exchanged between various applications. Using APIs outside e-commerce like (Facebook Platform) growing the business [3]. It is explicitly useful to bring transactional capabilities to any user touch point, mobile, social, in store, connected devices. Internal applications for use by the business. Some e-commerce aim to monetize their content and services by offering public or se-public APIs. For example Amazon Web Services is a cloud computing product where user pay only for the service they need. PayPal offers a direct payment API for merchants who do not want to send customers to a PayPal branded interface.

2.2 Literature Review

E-commerce system can have different sort of APIs. For example product information APIs, social proofs APIs, site search APIs, personalization APIs, marketing automation APIs, and shipping APIs, price comparison APIs. As it is all about software development work, so we have studied some API integration project. Amazon affiliate is the affiliate API [4]. Affiliate API means sharing product info. Amazon affiliate API is a kind widget. By using it we can show product information of Amazon to our application. Flip-cart affiliate API, is another type of affiliate API that allows registered users to access relevant information [5]. Shipping APIs are being used by online retailers. By using shipping API retailers can automate processes of sending the parcel from sale to customer door. It provides a tracking tool. Ship-wire API it provides inventory checking, Setting up orders, updating orders [6]. UPS shipping API it provides developer kit. It has the features of

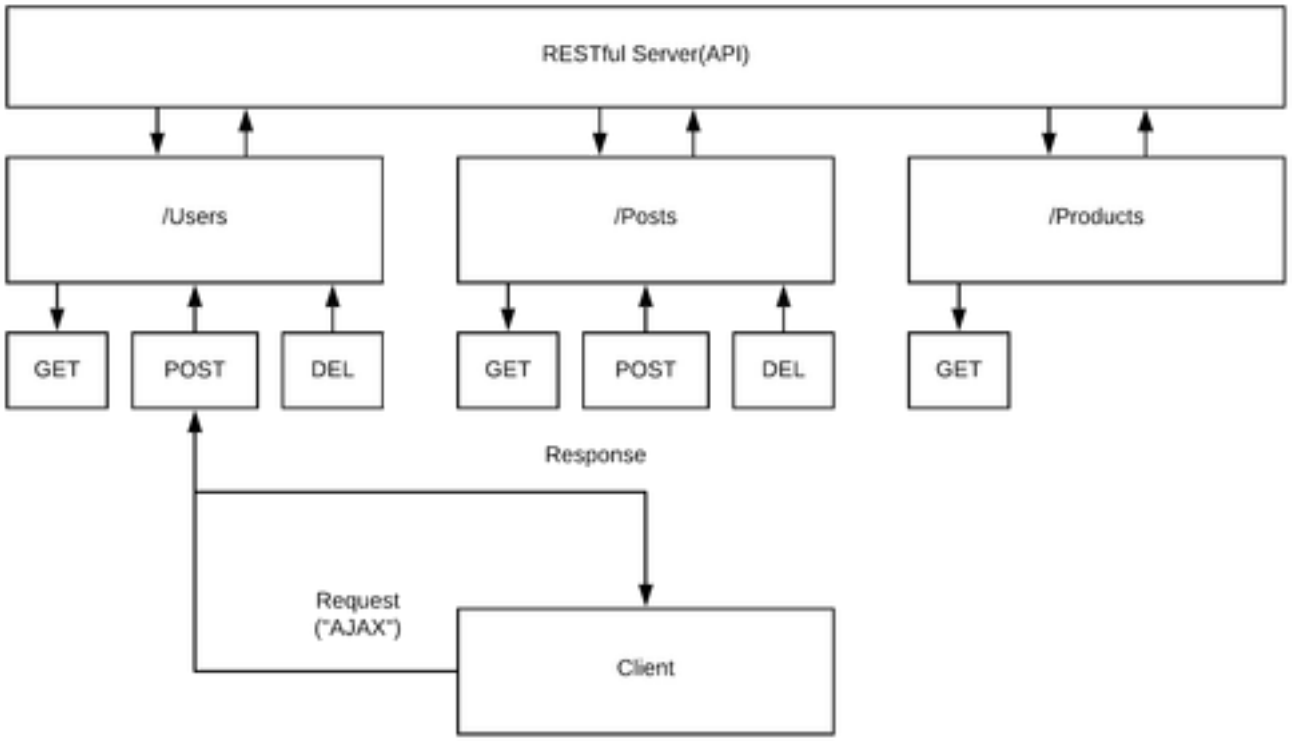

Figure 2.1: Module integration

rating, shipping, time in transit, tracking [7]. Product information API is being used to connect product catalog web-pages directly with a central global database. It depends on different e-commerce platform. Semantics3 is a product information API. It supports smooth data integration. Social proof APIs are very common. Now a days e-commerce sites are connected with social sites. Social API is a tool for retailer. They can use APIs to stream social media feeds direct. Bublebar a jewelry retailer site has integrated social media to their site to get feedback. Payment API integration is another important thing. Existing payment APIs are: Paypal API, it supports transaction in more than 100 currencies. Amazon Pay API, it is based on transaction and border based. Payza API, money transfer solutions designed for private and professional usage. For building an API based e-commerce to integrate multiple services we need a platform. The existing e-commerce platforms are WooCommerce and Shopify. In WooCommerce or Sopify we will setup a store. Shopify is a hosted e-commerce platform. It supports multi-channel integration. The best thing is it gives customize coding. WooCommerce gives limitless customization. It has specific Woo themes. The most important is powerful SEO.

In our study we found information about dynamic web services using API [8]. In this reference paper the author discussed about embodiment comprises an API gateway that identifies a plurality of software code objects for deployment and development, where the

code objects include executable code for performing subroutines. Request and response should be defined to communicate with API library, system environment. Code query also used to check errors. Third party service extension is another challenge [9]. It can optimize and minimize the web traffic in a prolific manner. It varies from device to device. For wireless network optimization is very much essential [9]. Processor monitors the input and output JSON data [10]. Input can be fixed, but in accordance with the implementation the state can change. It is also effected with endpoints. So the output changes as well. So keeping track of data using API is a must. API can provide data services by using a single path [11]. Data is relatable with the resources. Storing data using API, organization of incoming and outgoing data depends on routing [12]. Besides payload is an useful function that can be used in the back of integrated system. Workload identification is important to detect the errors in an integrated system [13]. It can be considered as a service. Based on schema, this service can be provided. API engine uses modules to verify and parse API specifications [14]. This types of engine can provide developers feedback as well. Client side SDK(Software Development Kit), backend compatibility heavily rely on specifications.

Based on computing resources API can be generated [15]. Client and hosting API gateway pays a huge role on the resources. Data can be accessible from the provider. User and provider interaction is important. It requires more security [16]. Different types of encryptions can be used. API call data transaction the driven data [16]. Navigation based on API [17] plays a vital role. Using analytical tools, and navigation, a system can get users choice based on different places. It is implemented by a set of listeners. Publicly available divers users of site can be modified using custom coding [18]. Developing API based system depends on graph class [19]. It organizes the data using node, edge, components. It also helps to store data in a device as objects. API relates with user experience and interface. Determining a feature [19] and generating appropriate response is a big task. Otherwise product fails to meet the objectives. Console application [20] helps to read and write from server to application, and the reverse process. Different methods include different data format. Integrating services depends on SDK(Software Development Kit) [21]. Different platforms support different versions SDK(Software Development Kit). Universal framework can be used also [22] different layout sets interfaces in different privilege level. Cross platform application development [23] empowers generating hybrid multiple service integration. Extending REST API works [24] effects resources and also collection APIs. Integrated system supports different databases [25]. It can support different formats. APIs key performance indicates the quality level [26]. A query is been executed [26] between connection to external data source to find out hosted client performance. Service architecture depends on resources [27]. Cloud based system, server client based system [27] responses in different paradigm.

2.3 Summary

Application programming interfaces can be used to speed up development. To avail custom coding with existing features it can be used. Application programming interfaces version can vary from platform to platform. Software development kit makes it easy to build integrated system for different platforms. Modules used in integrated system can be customized. It reduces time and development cost.

Chapter 3

Project Design

The following chapter focuses on design part. Considering System Requirements Specifications. Followed by methodology and figures related to the project.

3.1 Requirement Analysis

This part of the document shows the project plan for the development of Integrating of Multiple Services Using Application Programming Interface. The plan will include, but it is not restricted to a summary of the system functionality. This document will show scheduling, delivery estimates, project risks and how those risks will be mitigated.

3.1.1 Overview

Integration of multiple services can be done in two ways, number one by following Simple Object Access Protocol (SOAP). Number two by following Representational State Transfer (REST). We aim to develop on application that will connect e-commerce platform with the APIs by following REST API method.

3.1.2 Purpose

The purpose of the SRS document is to present a detailed description of constrain of Integration of Multiple Services using API. It will explain the purpose and features of our system used here and how the system will work. What type of APIs will be used here and how this system will be operated. This document is helpful for both stakeholders and developers of the system.

3.1.3 Scope

In our system integration of e-commerce services will be cheap and less time consuming. Developer uses APIs to integrate bigger systems. Conduct a trial run of API integration. Develop a functional e-commerce application.

3.1.4 Goals

After the completion of this project we will fulfill some specific goals. Some of the goals are:
i. Developing a functional e-commerce application.
ii. Reducing development time.
iii. Using payment gateway.
iv. Connect the application with Shopify platform.

3.1.5 Overall Description

Here we will describe the overall process to provide as much as information about our project.

3.1.6 Users

There will be mainly two kinds of users of our application. Number one is customer and number two admin.

3.1.7 Functionality

It is important how our application will function. Down below we have listed the functionality.
a. Customer can buy things.
b. Customer would be provided with necessary account information.
c. Shipping, payment services can be done.
d. Easy checkout.
e. Scale up ability.
f. Promotion and discount tools.
g. Content management capabilities.
h. Social media integration.
i. Search engine optimized code.
j. Reporting tools.

3.1.8 Platform

Our project is web based, as well as it is an application. So it can be accessed by a web browser which has internet connection, and also by using android app it can be accessed.

3.1.9 Development Responsibility

For developing the application we are responsible for the creation of interface, server setup, managing server tools, managing the content of the e-commerce store and support.

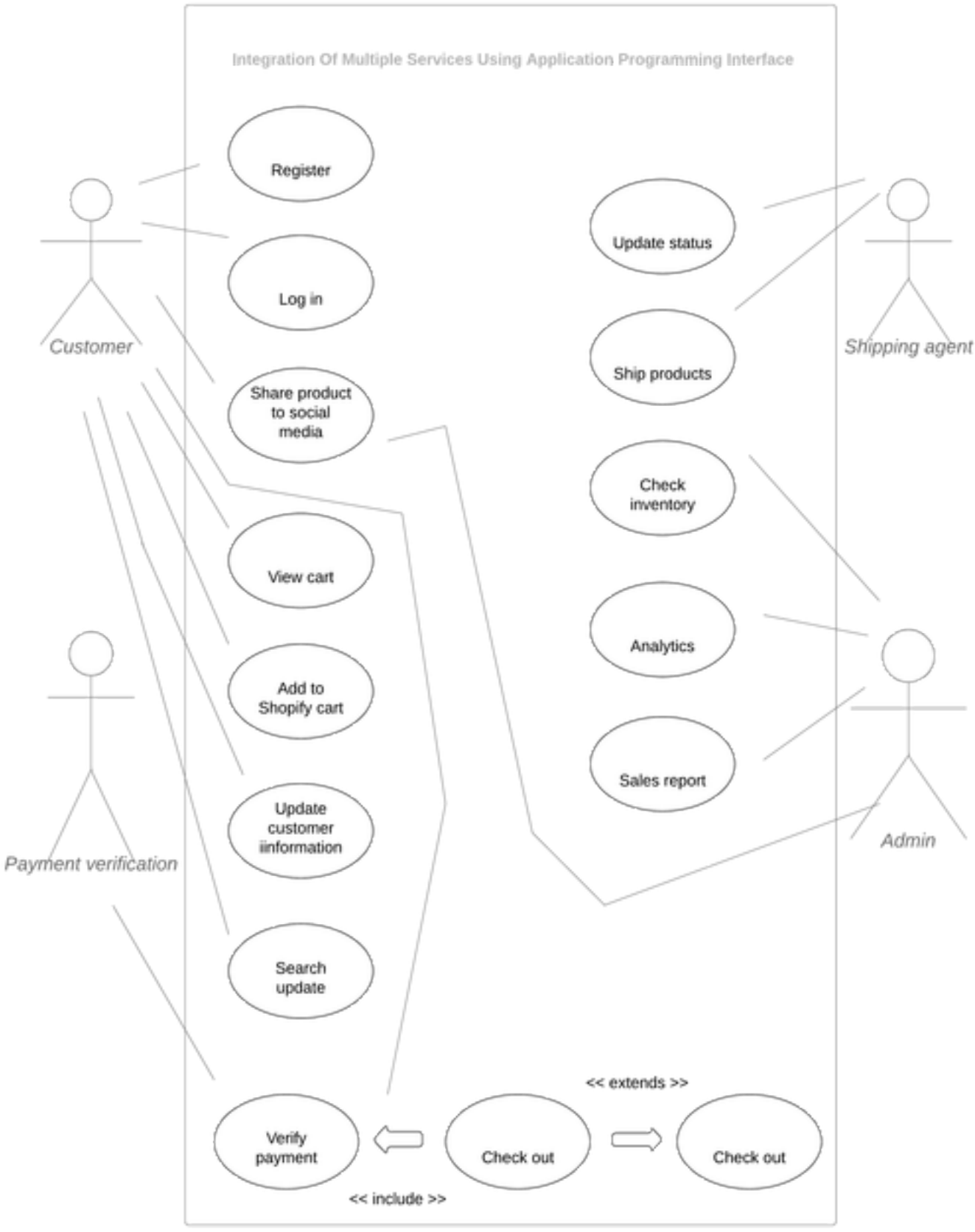

Figure 3.1: Use case diagram of SRS

3.1.10 Functional Requirements

Functional requirements describe the behavior of the system. Our web application would be easy to use user would find it comfortable. Products will be categorized. All the basic user functions would be shown at the homepage. Step By Step description: Customers uses the application Use Case diagram given help us to visualize the full process.

a. A customer can register/log in.
b. A customer can update his/her information.
c. A customer can add products to cart.
d. A customer can share product to the social media.
e. Admin can post to social media.
f. Admin can share to social media.
g. Admin can check the inventory.
h. Shipping agent would be notified.

3.1.11 Requirements Specifications

Let us discuss the non-functional requirements of our product. System Properties : The system properties are listed below.

a. Integration of multiple services runs on the internet.
b. Runs on a Linux server.
c. HTML/CSS, JAVA Script based UI design.
d. Node. Js based functioning system. And Java as well.
e. Android version : 4.4 minimum. API level 19.

Storage Requirements : The storage requirements are

a. The application will run on digital ocean server. Where
i. Memory : 1GB
ii. Virtual CPU : 1 Unit
iii. SSD Disk : 25 GB
iv. Transfer : 1 GB
v. Price : 5 dollar per month

b. Accessibility : Here we are stating someone can access this application.
i. The system is accessible from mobile/desktop/laptop devices.
ii. User privilege and access management guidelines will be added in the documentation.

Availability : The system will be available to specify users as stated below.

i. Any user can get access to this application 24/7.
ii. Minimum amount of downtime will be taken during maintenance period.

3.1.12 Backend Programming Language

For backend development we have various languages but we have chosen Node. Js because it uses asynchronous programming. It allows multiple things to happen at the same time.

Single threaded, non-blocking application run outside browser. Java is used for the app side.

3.1.13 Server

For running the application we have chosen digital ocean platform. In case of pricing Digital Ocean is cheaper than Google cloud. And Digital Ocean is offering features more that Google cloud.

Digital Ocean	**Google Cloud Platform**
SSD	Cloud infrastructure
Global Image Transfer	Data analytics
DNS management	Managed Computing
Private networking	Scalable Environment
99.99 percent up line	Layered security
Floating IPs	
Team accounts	
Multiple data center Locations	
Tier-1 bandwidth	
Powerful hex core machines	

Table 3.1: Comparison between Digital Ocean and Google cloud platform.

3.1.14 Droplet

We would want to work a droplet. Alternative of droplet can be local host but droplet gives virtual server environment which can be accessed from worldwide at any time.

3.1.15 API Type

It provides REST admin API reference which is one of the features of our project. It also gives APIs option for our app. Using APIs, we have two options web API and rest API. We want to choose rest API. The reason why we want to choose rest API over web API

Rest API	**Web API**
Hyper media User Interface	Machine to machine communication
Independent evolution of C and S	Coordinated evolution of C and S
Uniform interface	Custom interface
Code on demand	Mobility

Table 3.2: Comparison between Rest API and Web API.

3.1.16 Database Selection

In developing the application we will need a database. The options are mongodb and mysql. We want to choose mongodb. The reason we want to choose mongodb over mysql.

MongoDB	**MySQL**
Represents data as JSON document	Represents data in tables as rows
Uses object querying	Uses structured query
Does not need to define schema	Need to define schema

Table 3.3: Comparison between mongodb and mysql.

3.1.17 License

For developing the project we are using various software and platform. The licenses are given below.

Npm License	The Artistic License 2. 0 Copyright (c) 2000-2006, The Perl Foundation.
Node.js	MIT License.
Putty Gen	MIT License.
Notepad++	General Public License V2
ngrok	Apache License, Version 2.0
mongo-express	MIT License

Table 3.4: Licenses of used tools.

3.1.18 Our License

MIT License (Open Source)

Permissions	**Conditions**	**Limitations**
Commercial use	License	Liability
Distribution	Copy right notice	Warranty
Modification		
Private use		

Table 3.5: License of developed project.

3.1.19 Work Division

Since its a group project and web based system with huge work, so we have divided the work load among our group members in the following way :

Ajmaree Sulatna:

i. Project proposal
ii. Study Materials
iii. Working with JSON
iv. Report writing
v. Implementing Shopify Store
vi. Features Implementation

Prity Lata Chowdhury:

i. Project proposal

ii. Study Materials
iii. Working with JSON
iv. Report writing
v. Implementing Shopify Store
vi. Features Implementation
vii. Prepare report on Latex

Nopa Islam:
i. Project proposal
ii. Preparing slides
iii. Study Materials
iv. Working with JSON
v. Report writing
vi. Implementing Shopify Store
vii. Features Implementation

Sukannya Saha:
i. Project proposal
ii. Preparing slides
iii. Study Materials
iv. Working with JSON
v. Report writing
vi. Implementing Shopify Store
vii. Features Implementation

Nasir Uddin Ahmed:
i. Project proposal
ii. Study Materials
iii. Working with JSON
iv. Report writing
v. Implementing Shopify Store
vi. Record project video
vii. Edit Video and upload
viii. Server Setup
ix. Features Implementation

3.2 Methodology and Design

This part is describing the working process of the project. The diagram below shows the workflow of the project.
This is the process diagram of the project. We highly depend on REST APIs. As we

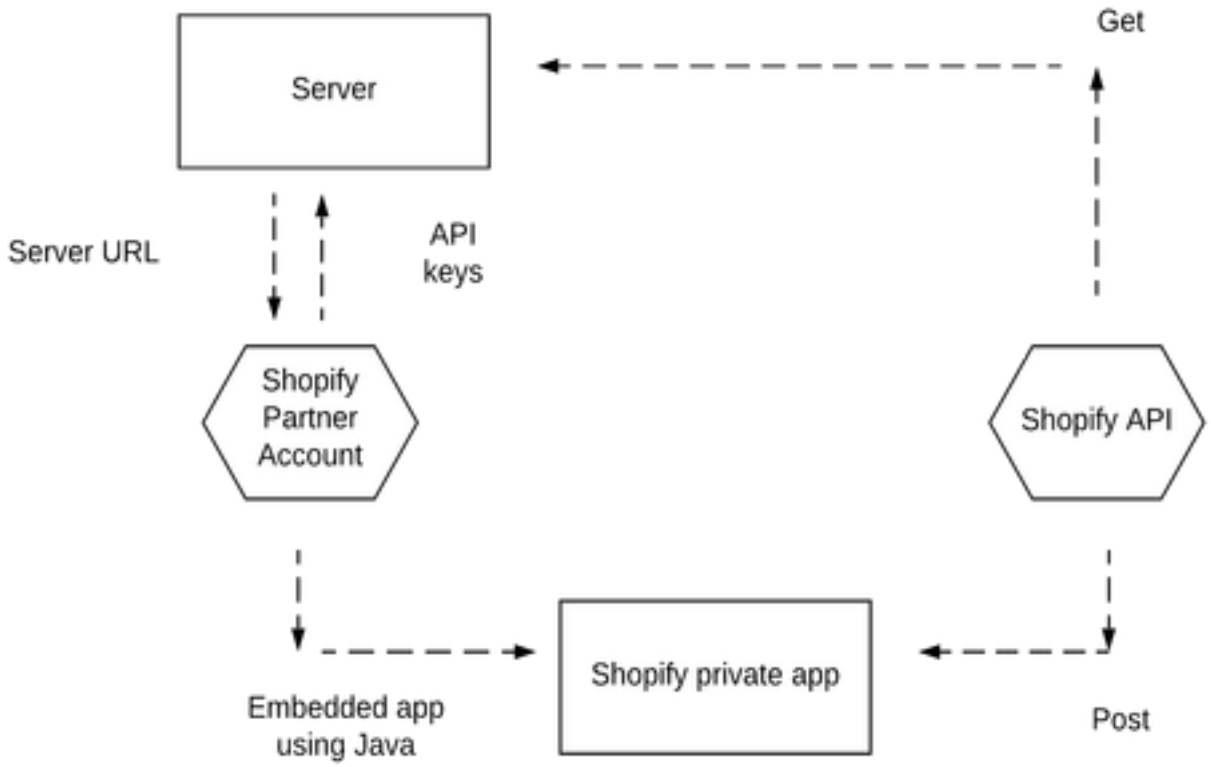

Figure 3.2: Design Of The Project

are using Shopify APIs. Shopify uses it's internal server. In that internal server it stores all Store related information. By using Shopify API, we simply "GET" them. Then the API back end code parses them, and "POST" the information in the Private Application. Shopify Partner account is connected with the Application and also with the server. Because from the partner account Store admin can manage the store. Partner account dashboard provides the Shop URL (Custom Domain), and API keys to the Private app side using embedded Java application (middleware). For handling APIs outside Shopify, and providing our applications API, we have Digital Ocean server.

In Shopify partner account, the project has a online store. In the store there are 15 Shopify supported features.
The features are
a. Product, b. Location, c. Theme, d. Payment, e. Domain, f. Shipping, g. Orders, h. Online Checkout, i. Taxes, j. Customers, k. Analytics, l. Productivity Tools, j. Application For The Store, k. Dropshipping, l. Marketing and Discounts.

These features are integrated in an application. So far product, theme, domain, online checkout, customers, application for the store, marketing and discounts are integrated.

For integrating these features, six types of Shopify supported APIs are used. They are a. Storefront API, b. GraphQL Admin API, c. Shipping and Fulfillment API, d. Products API, e. Online Store API, f. Customers API.

a. **Storefront API :** Basically it is being used for three purposes, they are
i. Fetching a single product information.
ii. Creating unique checkout experience.
iii. Create a new customer or modify existing ones.
b. **GraphQL Admin API :** It actually makes the functionality of the shopify admin.
c. **Shipping and Fulfillment API :** It works at the checkout option. Helps to install application as a store's fulfillment service.
d. **Products API :** It manipulates store's catalog. It uses collection APIs.
e. **Online Store API :** It updates the storefront.
f. **Customers API :** Helps to manage customers data.
There are two types of Shopify application. Public and Private. Public application does not support custom coding. Private application supports custom coding.

3.3 Summary

This chapter of the report expressed the design and features. Using different Shopify APIs, the project is being integrated. For third party payment, it requires external server to handle the API requests.Shopify uses internal database. Having the access token, api secret key the developed application connected with the store. Shopify supports readymade modules, so it saves the time of development and cost. The analytical tools inside the store can be used freely.

Chapter 4

Implementation and Results

The following chapter focuses on implementation part. After implementation, what are the criteria of evaluation that is also discussed.

4.1 Environment Setup

In this section we will discuss about the implementation process in detail. We are developing an e-commerce application on Shopify platform using APIs. Because it reduces time, and development cost. Readymade modules can be used directly. If needed it can also be customized. The first thing that we need is to create an account. For that we need Shopify partner account. It charges 29 dollars per month. There are different plans based on subscription. But we choose this 29 dollars per month plans because it supports online store including built in website version, and blogs, unlimited product can be added, two staff accounts can be created, anytime support, different sales channel, manual order creation, discount codes, free SSL certificate, fraud analysis for payment, integrating third party modules.

After going to the Shopify partner account we need to open the account. Providing necessary information and payment details we can avail the account. After opening the account we can customize the online store. It actually gives the admin access. There we can find out Shopify features. Going to the products section we can see all products, transfers, inventory, collections. Under customers, we can see registered customers details. Under analytics dashboard the store's information can be seen in graph chart format. Marketing option helps to increase the sales. Inside sales channel there are options like domain, managing blogs, navigation. We need to set a domain. For domain setup we need to buy a public domain. Shopify by default provides a domain. But by adding a customize domain the store development can get a new advantage. While connecting with the external server, public domain gives authentication freedom. For domain we visited namecheap.com. Domain price varies in accordance with domain name type. We bought a domain named "karakari.com" spending 5 dollars. After buying the domain, we need

to set it with the Shopify online store. After going to the domain add section, we need to put the domain. Shopify would suggest automatically the DNS naming server. We need to configure that. It would take around 8 hours to get verified. Then comes the connection validity test. If the connection is right, we can access a dummy website of the store. But the store is not completed yet. Now we have to manage a private application for the store. For that we need to create private application folder in Shopify store. We need to configure the Shopify private application option. Setting up a developer, putting credentials. This will provide API secret key, password for the custom coded application that we want to develop. It also gives data read and write permission. Now we need to fix the routes. It is an alternate process. If we have public domain then we would not need to use routing in such a way. As we develop the application in local environment, so we need to fix the routes. For that we used ngrok. It is mainly a tunneling service. By setting up tunneling service our local development can work as a server. At first we downloaded the ngrok. Then opening the software, from the command prompt we set up port number 3000 of our local machine as the server. Now we need to configure it in the Shopify dashboard where private application is being located. For that we need to set our application name, application URL. Here for local development environment we set the URL that ngrok has been generated. Later on we alter it with the public domain. We also need to set whitelisted redirection API. Whitelisted API does the transaction between Shopify store and local environment. Which means how the response will be generated and forwarded to the client that is being defined. After that we will be able to start the integration. But before that we need to test the applications outline. So for that we used a tool named postman. Which simply checks this API calling. We also checked from the browser that the application is responding by running a simple hello world string.

In any application there are always to sides. First one is backend, second one is the app side. We are using Nodejs as our backend. So we need to verify Nodejs in the server. We have digital ocean server. So far we have configured the Shopify platform. Now it is time to configure the digital ocean server. We will need help of PuttyGen, Wincsp to set NodeJs environment inside the server. For local development we need to install just the NodeJs package. For back up support of local environment we will need Npm. On the other hand for code version controlling over the server side Git is a must. So we created a digital ocean server admin first. Then used command lines to set Nodejs environment. For data breach protection we used SSH key encryption. But before that we installed Ubuntu operating system inside the digital ocean server. Then we build the Nodejs environment, which actually a run time environment for the application. To secure the server we need to disable root. To handle the load we need a load balancer. But in this case we will also need a process manager. PM2 is a good solution. It works as a process manager at the same time it is a built in load balancer. It allows to alive the application. Besides PM2 helps to load the application without downtime. Now we need to create a droplet. Droplet works as a remote virtual machine. It can be created based on shared resources

or dedicated resources. Droplet supports multiple operating systems. As our server runs on linux, so we need to use Ubuntu as the droplet's operating systems.

Now we need to create a NodeJs project. We need to install all the packages. We have used express package, nodemon package, promise package, request-promise package, morgan package, Shopify API package. All these packages have different roles. Express is a framework, it enables us to develop a mobile application. Which is the outcome of the project. Nodemon monitors the changes in the source code. Promise, request-promise does the API requests handling. Morgan works as a middleware logger. Shopify API node does all the request handling. It simply work as objects. It returns Shopify instances according to the API call is being made.

Now we need database. We choose MongoDB as the database. We choose MongoDB atlas version. Which supports cloud as well. We choose it because it is free. It works with collection. After selecting version we need to integrate the database with moongose package manage in the code. For that MongoDB gives additional code. It allows to create modules route to handle requests. Operations like Crud (Create, read, update, delete) are supported by the moongose package. Now we need to setup our code editor. We are using Java. And for mobile application we need to setup Android Studio. After that we need to create a project. There we need to import coding packages according to the Shopify API documentation [28]. We would need Storefront API, GraphQL API. Http cache policy API, mutation graph call API, Query graph call API, Retry handler API, Payment token API, Products API, Collection API. In the application at first we need to integrate the Shopify private application's API key, server's public key. Then the application can communicate in both ways. Storefront API, basically it is being used for three purposes, they are fetching a single product information, creating unique checkout experience, create a new customer or modify existing ones. GraphQL admin API actually makes the functionality of the Shopify admin. It works with node, edges, fields technique. Shipping and fulfillment API works at the checkout option. Helps to install application as a store's fulfillment service. Products API manipulates store's catalog. It uses collection APIs. Online Store API updates the storefront. Customers API helps to manage customers data. GraphQL does mainly the query. It uses single request to fetch data.

Working with payload is essential. Payload is handled as object. It comes from server. We can also send payload to the server. It is mainly used to handle complex requests. Metasploit framework supports payload. It takes each API of Shopify and check the endpoints. Then it throws the request to the server. It has built in networking protocols. Besides we found the alternatives of corresponding APIs. We developed our own APIs to connect our system with other platforms. It includes products and orders route.

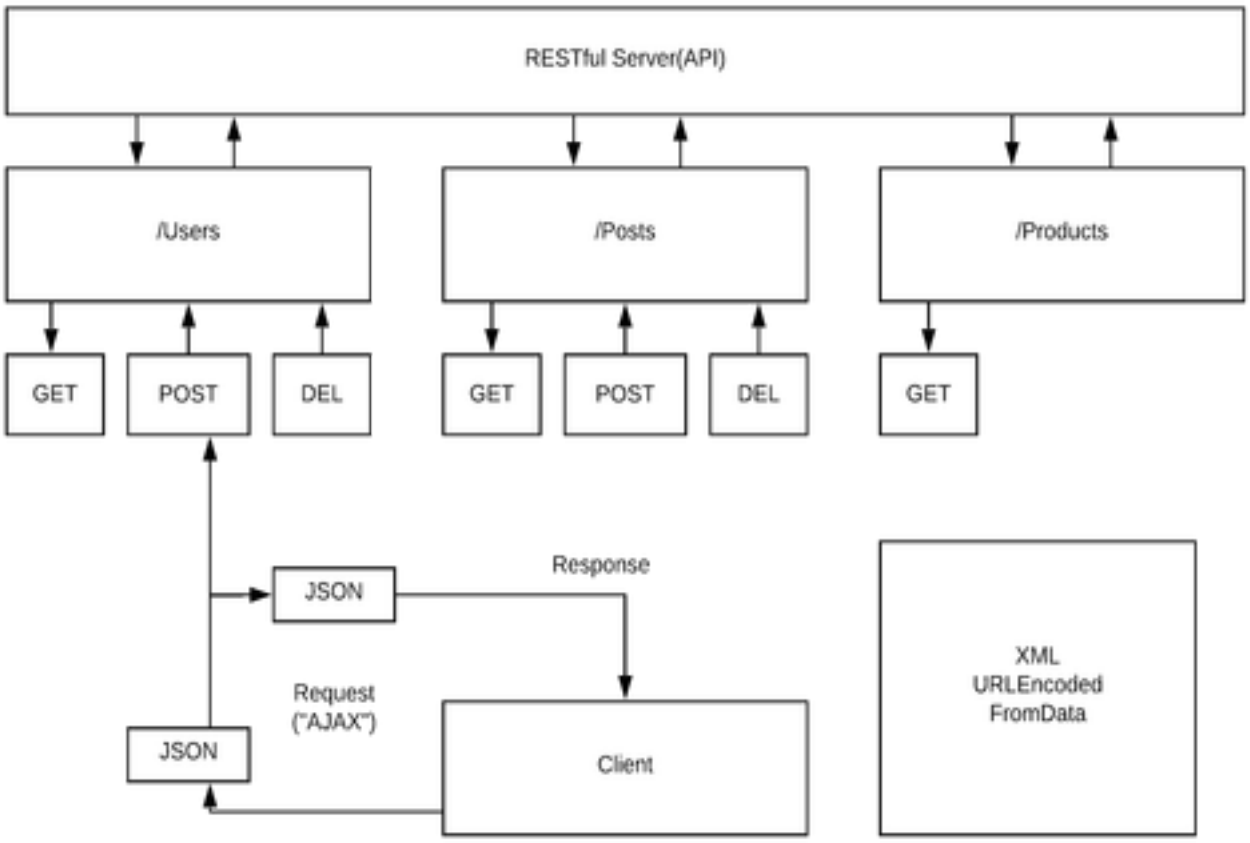

Figure 4.1: Data exchange between modules and client

4.2 Evaluation

We can divide evaluation in some major steps, they are introduction, tasks to be tested, user groups, methods, results and decisions. Conclusion.

So for introduction Application programming interface consists of functions, communication protocols to build software or application like building blocks. A good application programming interface makes it easier for the developers to make application faster. In the modern world application programming interface is considered as a product. Services means using different modules. Module actually encapsulates code and data to implement in a particular functionality. Module is usually packaged in a single unit so that it can be easily deployed. We have studied different e-commerce platform like WooCommerce, BigCommerce, OpenCart. Shopify platform supports numerous readymade e-commerce features. Using those features in an application, with custom coding saves development time and cost. As a result development cycle reduces. In the modern industry level custom coding and developing application quickly is a must. It also gives the independence of using different types of database, different payment services including local payment methods to grow and empower business quickly. Besides update of application and integrating new features become easier. In this project we developed an application, where we have used Shopify APIs such as Storefront API, GraphQL Admin API, Shipping and Fulfillment API, Products API, Online Store API, Customers API to make a functional e-commerce application for the user. We also developed product routes, and our systems own API to handle API requests in the server side.

There are some tasks that is to be tested to evaluate. They are application testing, testing the server, testing device compatibility, testing the server, testing the APIs. All these things are tested to identify the systems performance and reliability. User group is a big issue. The user of the application are small e-commerce business person, customers. Methods show the process. In our case . We highly depend on REST APIs. As we are using Shopify APIs. Shopify uses it's internal server. In that internal server it stores all Store related information. Bu using Shopify API, we simply "GET" them. Then the API back end code parses them , and "POST" the information in the Private Application. Shopify Partner account is connected with the Application and Also with the server. Because from the partner account Store admin can manage the store. Partner account dashboard provides the Shop URL (Custom Domain), and API keys to the Private app side using embedded Java application (middleware). For handling APIs outside Shopify , and providing our applications API, we have Digital Ocean server.

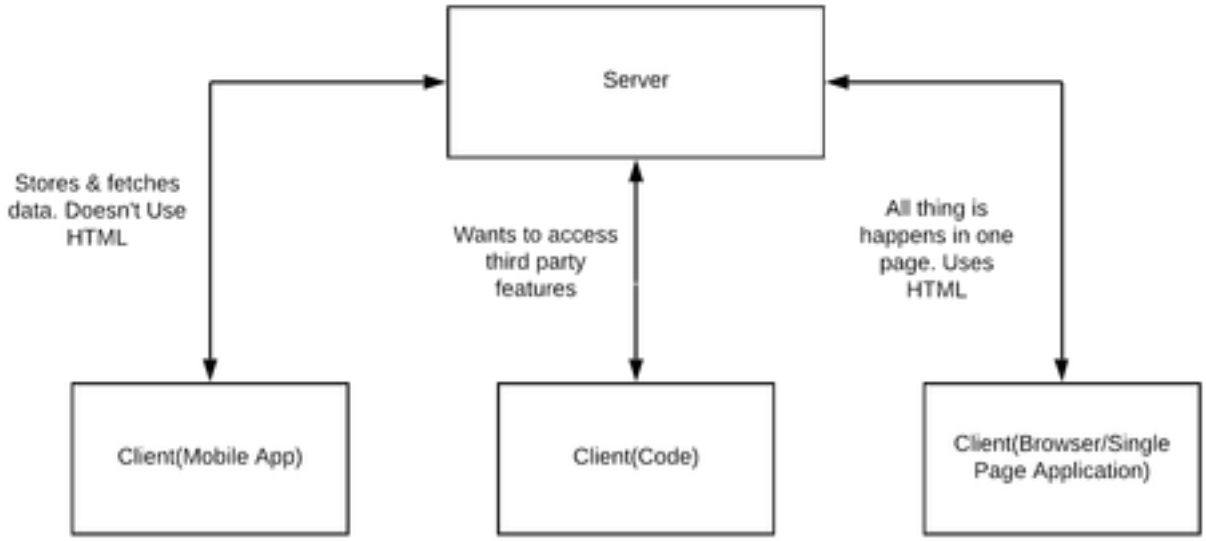

Figure 4.2: Architectures

4.3 Results and Discussion

Systems evaluation is a bigger part. It identifies the performance of a system. Before integration of the APIs in the system, we tested the APIs, by using postman. Each and every APIs were tested according to the Shopify API documentation. Then we tested the server by using HTTP test, Simple HTTP, Simple HTTPS, DNS, IMAP etc. We tested the MongoDB databse by putting data from local machine. The application that we built tested in different devices. It ran successfully. In the Shopify admin panel we tested products information. Manually tested the input output pattern, it worked. We followed six basic steps they are
Step 1: Exposing the local environment to the internet.
Step 2: Creating and configuring application in the "Shopify Partner Dashboard".
Step 3: Creating a Node.js project.
Step 4: Building the Application.
Step 5: Creating specific routing.
Step 6: Run the application
All of those steps worked correctly.

Integration of multiple services can be done in two ways, number one by following Simple Object Access Protocol (SOAP). Number two by following Representational State Transfer (REST). We aim to develop on application that will connect e-commerce platform with the APIs by following REST API method. In our case REST APIs worked along side with GraphQL. Integrated system uses application programming interface so that the systems share same code and database. Using APIs expose the server side which means

the backend and it also exposes the business logic and data. API facilitates direct access to the functionality provided by the web site. It also leverages third party efforts to add value to existing services.

4.4 Summary

In this chapter we discussed the implementation process. Which described what we have done, how we have done. The methods implementation is challenging. It has so many dependencies in the backend. Besides the integrated system has large scale up factors, which was handled by PM2 technique. Building an functional, marketable e-commerce application was the perquisite of this chapter. All the features of Shopify can not be integrated. But we tried to implement the most frequent and demandable features.

Chapter 5

Standards and Design Constraints

Finding out the standards and constraints and putting them in tabular format is the main target of this chapter.

5.1 Compliance with the Standards

In short, we have to maintain certain laws, policies, terms and regulations. In this modern world of available information fetching legally and permitted code, function, data transparency is highly followed. It is also a part and parcel of engineering ethics.

5.1.1 Technological Standard

Programming Languages	**JavaScript, Node. Js**
MarkUp Languages	Latex, HTML
Package Manager	Npm
Software	Shopify Tools, Ngrok, Putty
Server	Digital Ocean

Table 5.1: Technological Standard.

5.1.2 Ethical Standard

PCI Compliance: PCI Compliance is mandatory for all businesses and organizations that accept collect or process credit cards, including the University of Waterloo.

ISO/TC 321: Standardization in the field of transaction assurance and upstream/downstream directly related processes in e-commerce
i. The assurance of transaction process in e-commerce.
ii. The protection of online consumer rights including both prevention of online disputes and resolution.

IEEE/ACM code of ethics: It is maintained for professional purpose. Following coding standard to set benchmark.

Trademarks Act: In the competitive and constantly changing digital marketplace, a trademark is a very valuable asset. It differentiates your business and the quality of your products from those of your competitors. Your trademark carries your reputation with it, and reinforces long-term relationships with your buyers.

5.1.3 Safety Standard

We would work with accidental inputs. We have to ensure security of the software so that there would be no data lick. If needed we will ensure encryption of data.
a. IEEE STD-1228
b. Software assurance (SwA).

5.1.4 Hardware Standards

Our system follows hardware standards. It includes influencing factors, network related issues, layout for application, maintenance. With the proliferation of standards it becomes important to determine which to adopt or adhere to.

5.2 Design Constraints

5.2.1 Political Constraint

Political constraint shows different issues. It varies from country to country. Different country have different e-commerce application rules. It includes taxes, permission issues. We have built an integrated system. For this we are using different functionalities. None of them are related with political issues. So we can say we have not found any political constraints.

5.2.2 Sustainability

We will develop the project considering accidental and special cases, so we can ensure the design will be sustainable. It will be easy of use and helpful in the e-commerce industry.

5.2.3 Technological Constraint

It is a web based project. So internet connection is required. Wireless or wired internet connection is required. Besides it also runs on pocket data.

5.2.4 Manufacturability Constraint

The system need to be easily scalable and easily up-gradable. Cost is also a bigger issue. In our case we are using Shopify 29 dollars per month plan. It is quite cheap. Besides the droplet would cost around 5 dollars monthly. So within a limited budget we are trying to manufacture the application.

5.3 Summary

Chapter five describes the standards and constraints of the project. The standards and constraints those are related to the project highlighted here. Different standards and constraints can be found. In our case we found technological constraint as the bigger issue. On the other hand ethical standards were the main focus, as we were using others APIs.

Chapter 6

Conclusion

This chapter concludes the project in a nutshell. Summary part describes the brief history in short. Limitations showed the bounds of the project. Future work focuses further improvement of the project.

6.1 Summary

This chapter summarizes the whole part of our capstone project, the procedures and experiences. Our main aim was to develop a functional e-commerce system by using different features of Shopify platform. Earlier we studied different e-commerce platforms such as Woo commerce, Bigcommerce, Opencart. We found Shopify very helpful because of it has readymade features and existing resources and tutorials. We have integrated multiple services, added custom coding, and finally prepared a functional e-commerce system, using Shopify's public APIs. We have fulfilled the requirements, those we have stated in our project objectives.

6.2 Limitation

We are working on Shopify. Our system's performance heavily, depends on Shopify. For any reason if Shopify fails, then our system may fail as well. Besides, Shopify have different API versions. Shopify also have different SDK versions. So updating the system time to time is another limitation. If parameters of any API is changed then the system may fail. These are some of the limitations.

6.3 Future Work

We have developed an online store using Shopify. Later we used different APIs such as Storefront API, GraphQL Admin API, Shipping and Fulfillment API, Products API, Online Store API, Customers API to make the application work. Considering future works, we propose to integrate local online payment solutions like bKash and NexusPay

in our application. Global online payment solutions like PayPal, AliPay, GooglePay etc can also be added.

References

[1] Shyue Ping Ong, Shreyas Cholia, Anubhav Jain, Miriam Brafman, Dan Gunter, Gerbrand Ceder, and Kristin A Persson. The materials application programming interface (api): A simple, flexible and efficient api for materials data based on representational state transfer (rest) principles. *Computational Materials Science*, 97:209–215, 2015.

[2] Dana Petcu, Ciprian Craciun, and Massimiliano Rak. Towards a cross platform cloud api. In *1st International Conference on Cloud Computing and Services Science*, pages 166–169, 2011.

[3] Jens Dietrich. A rule-based system for ecommerce applications. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 455–463. Springer, 2004.

[4] Greg Helmstetter and Pamela Metivier. *Affiliate Selling: Building revenue on the web*. John Wiley, 2000.

[5] D Delano Ross Jr, Daniel D Ross, Joseph R Michaels, William R May, and Richard A Anderson. Affiliate commerce system and method, September 30 2003. US Patent 6,629,135.

[6] Khuong Le-Nguyen and Yue Guo. Choosing e-commerce strategies: a case study of ebay. vn partnership. *Journal of Information Technology Teaching Cases*, 6(1):1–14, 2016.

[7] Damon Williams. The paypal api. *Pro PayPal E-Commerce*, pages 105–155, 2007.

[8] Patrice Mahiddini. Dynamic application programming interface publication for providing web services, May 16 2017. US Patent 9,652,314.

[9] Ross Bott. Methods and systems for providing application programming interfaces and application programming interface extensions to third party applications for optimizing and minimizing application traffic, February 22 2018. US Patent App. 14/474,248.

[10] Todd E Kaplinger, Victor S Moore, and Wendi L Nusbickel. Self-documentation for representational state transfer (rest) application programming interface (api), May 1 2018. US Patent 9,959,363.

[11] Salvatore DeSimone, Edgar J St Pierre, Frederick A Crable, Vinodh Ravindran, Won T Cho, and Puneet B Lal. Method, apparatus, and system for managing data storage with an application programming interface, December 18 2018. US Patent App. 10/157,124.

[12] Robert C Streijl. Enhanced network congestion application programming interface, March 5 2019. US Patent App. 10/225,761.

[13] Charles D Garrett and Christopher W Fraser. Bottleneck detector application programming interface, January 9 2018. US Patent 9,864,676.

[14] Ken Elkabany. Interface description language for application programming interfaces, August 27 2019. US Patent App. 10/394,552.

[15] Ryan Paul Green. Generating an application programming interface, March 29 2018. US Patent App. 15/279,007.

[16] Robert Dykes. Self-adaptive application programming interface level security monitoring, July 11 2019. US Patent App. 16/234,104.

[17] Matthew James Way, Chee Yu, Steve Ayers, Patrick Greaney Willett, and Eli Schleifer. Autonomous vehicle application programming interface and communications systems and methods, August 1 2019. US Patent App. 15/918,588.

[18] Jingming Li, Nianping Li, Kereshmeh Afsari, Jinqing Peng, Zhibin Wu, and Haijiao Cui. Integration of building information modeling and web service application programming interface for assessing building surroundings in early design stages. *Building and Environment*, 153:91–100, 2019.

[19] Patrick D Quillen, Antonio C Ionita, Duncan Po, and Christine Tobler. Graph class application programming interfaces (apis), July 9 2019. US Patent App. 14/984,942.

[20] Filip Eliá and Filip Nguyen. Console application through web service, May 28 2019. US Patent App. 10/303,531.

[21] Srikanth Nalluri, Dattatraya Kulkarni, Raja Sinha, Venkatasubrahmanyam Krishnapur, Kaushal Kumar Dhruw, and Kamlesh Halder. Mobile application management, January 24 2019. US Patent App. 16/138,904.

[22] Florian Stephan. Universal application framework for streamlined frontend development of user interface applications, April 25 2019. US Patent App. 16/168,425.

[23] Vipul Kaushik, Kamali Gupta, and Deepali Gupta. React native application development. *International Journal of Advanced Studies of Scientific Research*, 4(1), 2019.

[24] Timothy Demulder and Wim De Waegeneer. Extending representational state transfer application program interface (rest api) functionality, May 21 2019. US Patent App. 10/298,656.

[25] Sangeeta Gupta. Unstructured data analysis with passphrase-based rest api nosql for big data in cloud. In *Innovations in Computer Science and Engineering*, pages 457–463. Springer, 2019.

[26] Aida Rikovic Tabak, Ciprian Mocanu, Andrei Gabur, Adrianus Augustinus Mathijssen, and Georgi Ivanov. External data collection for rest api based performance analytics, April 4 2019. US Patent App. 15/787,409.

[27] Olivier Colle, William James Staples, Carlos Aguilar Mares, Samuel Lenz Banina, Karandeep Singh Anand, Kyle Werner, and Gautam Thapar. Application service architecture, April 30 2019. US Patent App. 10/277,582.

[28] Michael Larkin. *Shopify Application Development*. Packt Publishing Ltd, 2014.