Expertise
and insight
for the future

Aashis Rimal

Developing a Web Application on NodeJS and MongoDB using ES6 and Beyond

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

25 January 2019

Metropolia
University of Applied Sciences

Author Title Number of Pages Date	Aashis Rimal Developing a Web Application on NodeJS and MongoDB using ES6 and Beyond 36 pages 25 January 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Kimmo Sauren, Senior Lecturer

The purpose of the thesis was to study different aspects of web development using Full Stack JavaScript and to develop a prototype application based on it. Its objective was to study the usage of NodeJS, Express, MongoDB and EJS in developing a Full Stack web application. Moreover, it also focuses on studying the different versions of JavaScript and their use in NodeJS platform.

All the different versions of ECMAScript were studied in great details, especially ES6. Then, the features and implementation of JavaScript based frameworks such as NodeJS, Express and MongoDB were studied and compared to other technologies. Besides, the potential security threats to the NodeJS application and various ways to mitigate them are illustrated. Moreover, a minimum viable product of an application was developed using the mentioned stack.

As a result, an E-commerce application was developed in NodeJS and Express using the NodeJS core packages and modules as well as Express middleware. Other third-party middleware was also used in order to develop a working prototype application.

In conclusion, full-stack JavaScript is considered to be the best technology for developing modern, scalable and secure web applications. However, the inability of NodeJS to handle complex data calculations and algorithms makes it unsuitable to develop large enterprise applications.

Keywords	NodeJS, Express, MongoDB, Mongoose, EJS, security

Contents

List of Abbreviations

API	Application Programming Interface
CSRF	Cross-Site Request Forgery
CORS	Cross Origin Resource Sharing
CSS	Cascading Style Sheets
EJS	Embedded JavaScript
ES4	ECMAScript 4
ES5	ECMAScript 5
ES6	ECMAScript 6
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol over Secure Socket Layer
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LAMP	Linux Apache MySQL and PHP
LTS	Long Term Support
MVC	Model View Controller
MVP	Minimum Viable Product
PDF	Portable Document Format

PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SSL	Secure Socket Layer
SQL	Structured Query Language
TLS	Transport Layer Security
URL	Uniform Resource Locator
WWW	World Wide Web
XSS	Cross Site Scripting

1 Introduction

Web development has come a long way since the Worldwide Web (WWW) was introduced. Developers had to develop separate applications for each of the operating systems and install them locally on the computer in order to use them. Those applications were called desktop applications. In contradiction, web applications are easily accessible from the web browser irrespective of the user's operating system. Web applications have made an enormous progress since the desktop applications and static websites to the modern, user interactive and intelligent web applications. Users were only able to read the contents of the static websites. However, due to the emergence of web applications, users are able to interact with the server, listen to audio and watch videos as well as draw on the screen.

Web application development is a combination of front-end and back-end development. There are a number of programming languages and frameworks to choose from in order to develop a web application. Previously, there were no programming languages which could do both front-end as well as back-end development of an application. LAMP stack: Linux, Apache, MySQL and PHP used to be a standard for full-stack web development. PHP was mostly used for the back-end and HTML, CSS and JavaScript were the major stacks for front-end development while MySQL was used as a database. Apache acts as a web server in the Linux operating system in the LAMP stack. However, developers needed to learn more than one language, all of which used a different syntax and has a different nature, to develop a web application using the LAMP stack. Apart from this, with the emergence of NodeJS as a server-side platform, JavaScript was able to provide full-stack software development within a single programming language. In spite of NodeJS being the most common technology for back-end development, there are multiple frameworks such as Angular, React, Vue.js or Knockout.js as well as templating engines such as EJS, Handlebars or Pug that can be used for developing the front end of an application.

This thesis was carried out with the aim of studying different components and frameworks required for a full-stack application development in JavaScript. Moreover, a prototype application is developed in order to demonstrate the implementation of different forms of JavaScript being used in a single web application. The report, however, focuses more on the different components of JavaScript used to develop the application. It also compares the performance, speed and code reusability of Full Stack JavaScript application to other similar technologies.

2 JavaScript

JavaScript is a dynamic, object-oriented scripting language with first-class functions based on objects which can return a value by passing a function itself to other functions as an argument. JavaScript was first developed in 1995 by Brendan Eich in a company called Netscape to serve as a scripting language for Java [1]. Although JavaScript was primarily used for client-side scripting to perform web-based functionalities, it has evolved significantly to support the server-side scripting as well. JavaScript has now progressed as a programming language which is used for front-end and back-end of web applications, databases as well as robotics.

Full-stack JavaScript Development has several advantages over using different languages for back-end and front-end development. It is a great technological stack for developing dynamic and high performing applications. Code sharing and reusability help reduce the number of lines of code by up to 40 percent by sharing the parts of the code or even sharing the templates, libraries and models. It makes the maintenance or refactoring very easy and cost-effective. Moreover, using a common language helps to gain better team efficiency and reduce the gap between the teams since all the teams working on the application will have a better understanding of what other teams are doing. [2] In addition, developers do not need to worry about the syntactical differences as with different languages being used for the single application.

Furthermore, JavaScript increases the speed and performance of an application. PayPal moved from JAVA to using NodeJS as a server-side platform for their application and published a report on how JavaScript and NodeJS helped them increase the performance of their application and reduced the amount of code written significantly. According to the report, the JavaScript application was built twice as fast as the one written in Java with fewer people in the team. They also stated that the JavaScript application consisted of 33 percent fewer lines of code and 40 percent fewer files with both applications consisting of same functionalities. Likewise, the request per second was doubled in the JavaScript application and the average response time was minimized by 35 percent. [3]

Despite all these advantages of full stack JavaScript development, there are few disadvantages as well. NodeJS cannot handle the heavy computing, data processing, machine learning, algorithms and heavy mathematical calculations since such operations might block the incoming requests due to its asynchronous nature of programming model. In addition, NodeJS is a relatively new technology and does not have the support of a robust and strong library system yet. [2]

Stack Overflow surveyed over 1,00,000 professional software developers and students hoping to be one in future about their favorite technologies for web development in 2018.

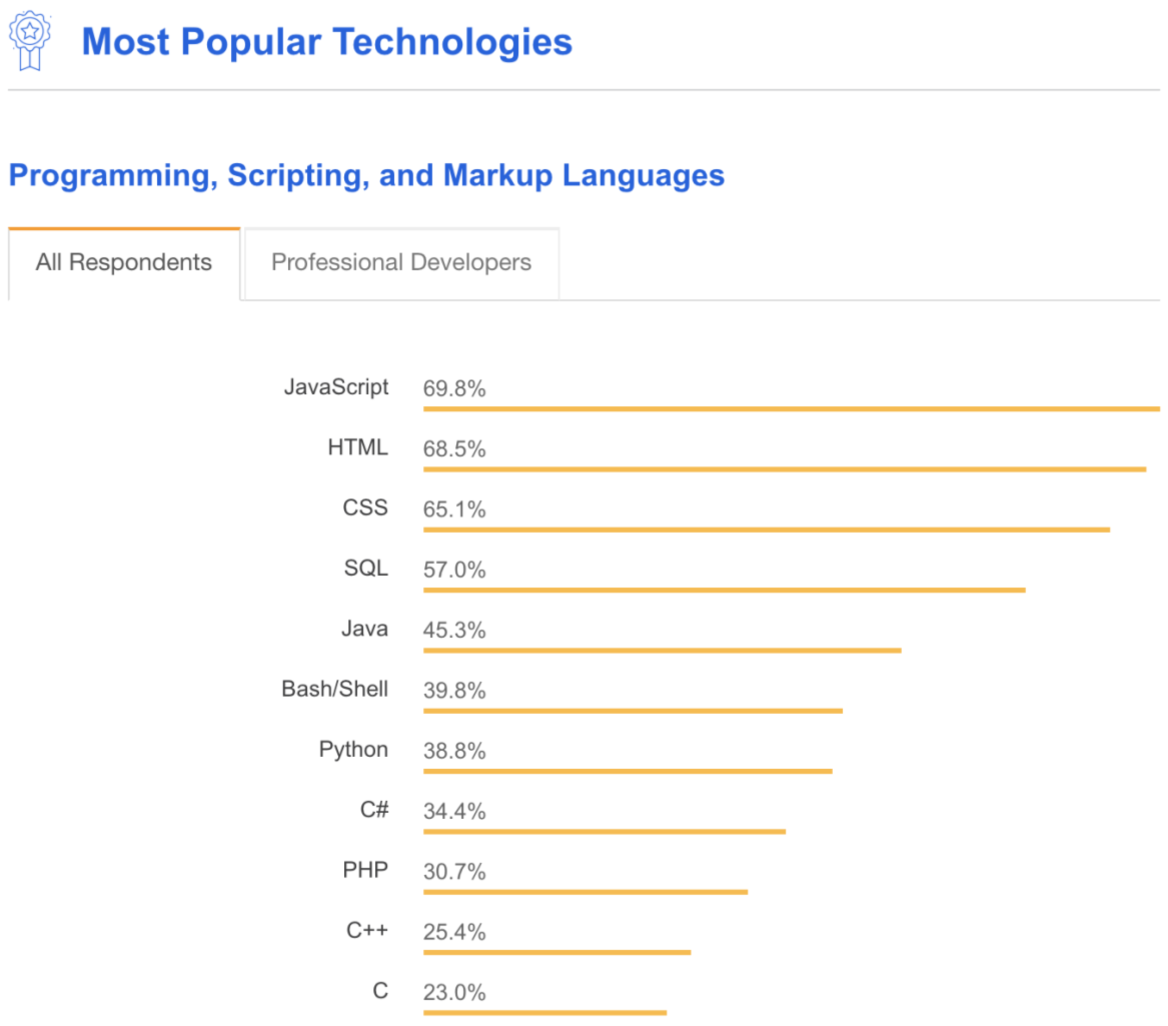


Figure 1: Stack Overflow Developer Survey Results 2018 [4]

As illustrated in Figure 1, 69.8 percentage of the total surveyed listed JavaScript as the technology they use frequently. This clearly states that JavaScript is one of the most popular languages.

2.1 Different JavaScript Versions

JavaScript was standardized by Netscape in 1996 with ECMA (Ecma International – European Association for Standardizing Information and Communication Systems), which sets the standards in IT and ICT sectors, as a host. ECMA international works to make JavaScript modern and relevant to the current technological standard [2]. Since then, different versions of ECMAScript are released of which EcmaScript6 (ES6) being the significant version. After the largest release in 2015, ECMA is committed to releasing newer versions each year. Hence, ECMA has changed the name for ECMAScript from release number to the release year. However, the number system is more popular among the JavaScript community. Hence, the number system will be used throughout this paper. The Table 1 shows a comparative list of different ECMAScript editions, their official name and the date the upgrade was published.

Table 1: ECMAScript versions and their release dates, data collected from [5]

Edition	Official Name	Date Published
ES1	ES1	June 1997
ES2	ES2	June 1998
ES3	ES3	December 1999
ES4	ES4	Abandoned in July 2008
ES5	ES5	December 2009
ES5.1	ES5.1	June 2011
ES6	ES 2015	June 2015
ES7	ES 2016	June 2016
ES8	ES 2017	June 2017
ES9	ES 2018	June 2018

Table 1 illustrates the different versions of ECMAScript with their official name and the release date. The official name for the ECMAScript versions changed starting in the sixth edition where it moved to the yearly naming convention in the hope of releasing a new version every year.

2.1.1 Before ES6

The first version of JavaScript was released in June 1997 by ECMA International with the name ECMAScript 1. Next year in June 1998, ECMAScript 2 was released which contained the minor editorial changes to follow the ISO standard for JavaScript. However, the third edition of JavaScript, ECMAScript 3 was released in December 1999 and presented a number of new features to the language. The addition of features such as powerful regular expressions, do-while, string handling functions like concat, replace, match, split and splice using regular expressions, error handling with try/catch, narrowed error definitions, numeric output formatting and several other minor changes made the language more powerful. [6]

ECMAScript 4 (ES 4) is a version of JavaScript which never came into existence. Various new features were planned for this upgrade which includes classes, namespaces, interfaces, type definitions, getters and setters and more. The main objective for this release was to make the language appropriate to develop large-scale software and emphasize the development of ES4 libraries along with making it compatible with ES3. However, due to the disagreement between the two groups working on upgrading the language, ES4 was abandoned in July 2008. Instead, they came up with an agreement to release an incremental update ECMAScript 5. [7, p. 6]

The fifth edition of ECMAScript was published in December 2009 after ES4 was abandoned. It is an incremental update to the previous ES3.1 edition which was later renamed as ES5. Some of the new features included in ES5 are strict mode support which helps write cleaner code, different methods to search and manipulate arrays such as map, reduce, filter, indexOf and forEach properties and attributes as well as JSON. JSON is used to store and transport data as name/value pairs which is also the most popular form of storing data until the time of writing this thesis. [8]

2.1.2 ES6

ES6, also officially known as EcmaScript2015 is the name used for JavaScript version since the term ‘JavaScript’ is trademarked by Oracle. ES6, which was released in June 2015, is the major update for JavaScript after it was standardized in 1996. [2] ES6 was released with a goal to support writing complex applications, libraries as well as code generators. In addition, improving interoperability and keeping a better and simple versioning system were the main goals for ES6. As a result, the new version introduces several new features such as modules, classes, arrow functions and ES6 proxies to name a few. [7, p. 4-5]

Although ES6 was released in 2015, most of the browsers did not support new ES6 features. Hence, transpilers were used to convert the JavaScript code written in ES6 to ES5. Babel and Google Traceur were the popular transpilers used by the developers. However, all the modern browsers support new ES6 features nowadays and the use of transpilers is irrelevant. Some of the new ES6 features will be discussed below.

ES6 introduced new keywords: *let* and *const* to define a variable which are block-scoped in addition to the function-scoped *var* in ES5 [4, p. 18-19]. The *let* variables defined inside a function expression can not be accessed outside of that specific function. Unlike *let*, variables declared with *const* can not be declared again with the same name. The values defined using *const* variable are not immutable though, the value can be changed by performing JavaScript operations if the value is an object. [9]

Arrow function is a new addition to ES6 which are mainly useful for short function calls because of its abbreviated syntax using the arrow as the name reflects. It omits the need of writing the keywords *function* and *return* as well as curly braces. Not only this, the scope of keyword *this* has changed significantly. It does not create its own scope and simply inherits from the function it is in. This helps solve the ever-existing problem with *this* keyword in JavaScript. [7]

Similarly, default values can be passed to a function whenever the parameters are not provided in the function call by initializing the parameter with a default value. This function was already available in PHP and Java. In addition, rest parameter and spread operator are also added to ES6. Rest parameter is denoted with three dots followed by a named parameter (…books) which stores all the arguments of a function in an array. It allows declaring a variable number of arguments with one identifier. Spread operator can be taken as the opposite of rest parameter with the same syntax as that of rest parameter which takes an array or any iterable object and assigns to different variables. [10]

Furthermore, ES6 has introduced many other changes to the ES6 version of JavaScript. With the introduction of template literals, modules, classes, proxies, promises, destructuring, generators, getters and setters, JavaScript is now more suitable for developing advanced and dynamic applications. Moreover, the addition of other smaller features and methods boosted the performance and speed of JavaScript applications.

2.1.3 Beyond ES6

ES6 was released after a gap of four years and there were many new additions to the language. The ECMA technical committee for developing ECMAScript, TC-39, realized that releasing a new version after such a gap is unsustainable. Therefore, they decided to release the new version every year with fewer additions to the language.

ES7 has introduced three changes to the 2016 version of ECMAScript. First, a new operator with the syntax (**) is a shortcut for receiving the exponential value. It performs the same operation as of *Math.pow* but with less code. Second, *Array.prototype.includes* returns Boolean value when checking if an element exists or not in an array. Finally, the function-scoped strict mode has been changed as well. The use of *use strict* in functions with destructured parameters or function having default parameters causes a syntax error since the default parameter values could also be a function. [10]

ES8, also formally known as ECMAScript 2017, was released in June 2017. It included async functions, shared memory and atomics, trailing commas in function parameter lists and calls, string padding and added methods like *object.values*, *object.entries*, *string.prototype.padStart*, *string.prototype.padEnd* and *object.getOwnPropertyDescriptors*. [11]

ES9 or the ECMAScript 2018 is the latest version of ECMAScript which was released in June 2018. It supports asynchronous iteration of an iterable object like arrays and strings. Promises and generator functions can now be iterated using the asynchronous iterator. Similarly, ES9 has also added more regular expression features such as dotAll flag, look-behind assertions, Unicode capture groups and named capture groups. The rest and spread operators now also work for object properties. Moreover, ES9 has also added *.finally()* method in the promises. This is a callback function which executes every time when the promise is either resolved or rejected. [11]

3 NodeJS and Express

3.1 NodeJS

NodeJS is an open-source and cross-platform server-side programming platform built on Google Chrome's V8 JavaScript runtime environment which runs the JavaScript outside of the browser. It was developed by Ryan Dahl in 2009. NodeJS is a non-blocking, asynchronous and event-driven I/O system which means it does not wait for one API call to complete in order to call the next one. Instead, it executes the next event and comes back to the previous event with a callback function that was specified before. [12] The non-blocking nature of NodeJS makes it lightweight, efficient and scalable and is best for developing the real-time web applications. NodeJS creates a server on its own using the core HTTP module, listens to the incoming requests, handle those requests, validate the input data, connect to the database and return the response either by rendering an HTML page or by simply returning a JSON.

Figure 2 illustrates the single-threaded non-blocking I/O event execution of NodeJS. It creates an event loop to execute all the requests coming to the server based on the single thread. This does not block the IO execution and results in increasing the speed and performance of such applications.

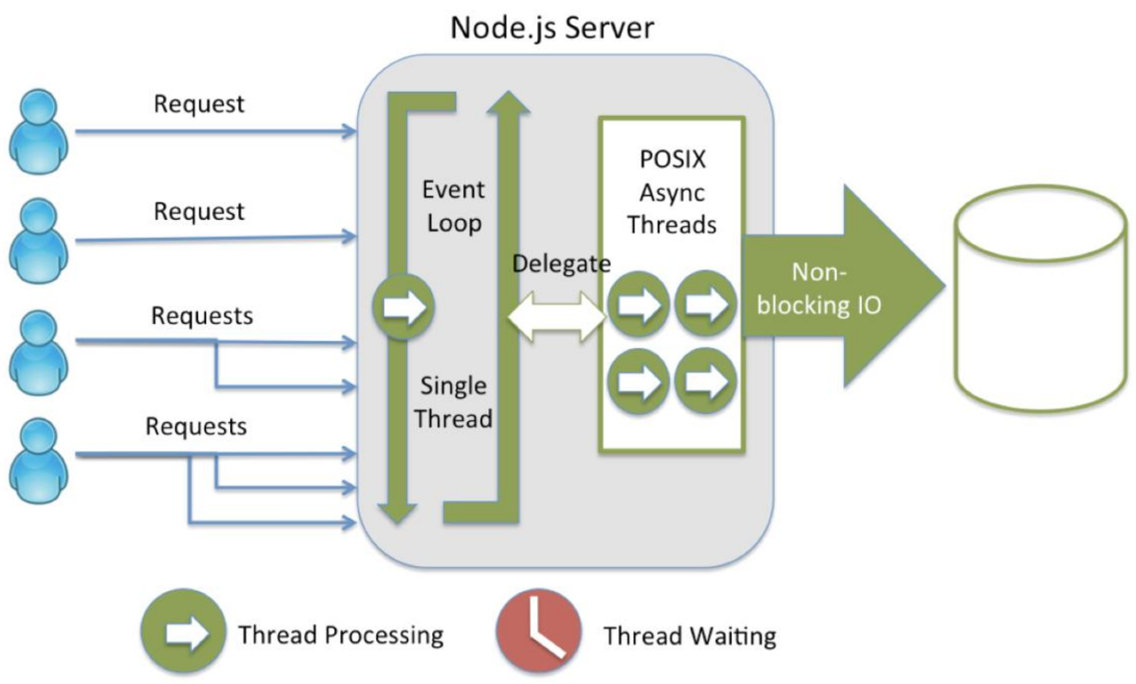


Figure 2: NodeJS single-threaded I/O execution [13]

Multiple requests are processed in the NodeJS server at the same time as shown in Figure 3. These requests are processed by a single thread and form an event loop which is delegated as the async thread to the non-blocking IO execution.

The NodeJS's performance and stability are improved regularly and the new JavaScript features from the yearly ECMAScript releases are integrated to NodeJS in its new releases.

3.2 Express

Express is a NodeJS web application framework that provides various features in addition to the core node modules that speed up the web development as well as makes it easy to develop. Express, being a minimal and flexible NodeJS framework, it provides with several third-party middlewares which extend the functionality of the express application and adds features as required. [14]

Express is a combination of middleware which is executed from top to bottom in request-response cycle. Each middleware has access to the request and the response object as well as a next function which are passed from one middleware to another. Middleware takes the request, executes the code inside, changes the request and response objects and calls the next function which activates the next middleware in the queue. An express application can have access to the application-level middleware, router-level middleware, error-handling middleware, built-in middleware and third-party middleware. The application-level middleware is bound to an app object using the app.use() and app.method() functions where 'method' is an HTTP verb that is being executed. The router-level middleware only differs from the application-level middleware since it is bound to an instance of the express router. Error-handling middleware is bound to the app object and typically takes four arguments, error, request, response and next objects. Built-in middlewares such as express.static and express.json serves the files statically and parses the incoming request to JSON respectively. Moreover, other third-party middlewares extend the functionality of the express application either by logging in the information about the incoming request or parsing the cookies. [14]

Furthermore, routers in express divide the application into several mini express applications and combine together to form an express application. Express router is composed of an HTTP verb (GET, POST, PUT, DELETE) and a path or the location of the resource. Moreover, callback functions are also specified to the routing method where a number of callback functions can be used as arguments. Such callback functions need to call the next method using the next() function so as to move on to another callback function. [14]

4 MongoDB and Mongoose

4.1 MongoDB

MongoDB is a flexible and scalable document database. The database consists of a collection of documents. It dumps the traditional method of storing data in relational databases and stores the data in a flexible way without any restrictions of the format and structure. The data is stored in BSON format, which is a binary JSON. The document can contain strings, numbers, floats and even arrays and objects. It makes the database operation faster and simpler. MongoDB embeds the other sub-documents inside the document by providing a link to that document instead of joining the collections. The MongoDB database supports various operations on the database such as querying for the documents, inserting new documents as well as editing and deleting the existing documents. [15]

The main feature of the MongoDB database is its ability to store the dynamic data. The documents in the same collection can have different properties and key-value pairs. This drops the requirement of storing similarly structured data which is compulsory in other relational databases. This feature provides more flexibility for storing the non-alike data in a single collection. Moreover, storing all the data for an entity in a single document improves the speed of the database operation as compared to the relational database. It avoids the necessity to join many tables to get the data from different rows and columns. [15]

4.2 Mongoose

Although MongoDB is a schema-less database and relies on developer’s discipline to maintain the structure of the document, a structured data is required for most applications to operate correctly. Mongoose was developed in order to solve this issue. It enforces a standard structure to all the documents in a collection using schema. It can also validate the data stored in the documents as well as allow only valid data to be saved in the database. In addition to this, mongoose provides all the features that exist in MongoDB with the addition of query building functions and business logic in the data. Moreover, it can connect the database with the server and perform similar database operations for reading, writing, updating and deleting the data. [16]

5 Project implementation

The prototype application Kinmel.com was developed to demonstrate the usage of full stack JavaScript for developing a web application. The application developed is a Minimum Viable Product (MVP) for buying and selling goods via the internet. The MVP can later be developed to cover different other aspects of an online e-commerce application.

5.1 Project Introduction

Kinmel.com is a web application that facilitates the buying and selling of goods through the internet. It involves two sets of users, namely, admin and users. Admins are responsible for adding the products, description and price of the product to the database. Similarly, users can browse through all the products listed in the application, add the product to the cart, pay for the products bought and order the product.

After investigating the requirements of the application owner and the potential customers, the following requirements were listed for the application to be ready as an MVP.

- Users will be able to create an account for themselves.
- Users will be able to edit their account information.
- Users will be able to browse through all the products available.
- Users will be able to check the price and description of the product as well as see the picture of the product.
- Users will be able to add one or more products to the shopping cart.
- Users will be able to view products in the cart.
- Users will be able to remove products from the cart.
- Users will be able to order and pay for the products bought.

- Users will be able to download the invoice for the order.
- Admins will be able to add a product to the database.
- Admins will be able to edit a product in the database.

Based on the requirements listed above, an MVC (Model View Controller) application was designed and developed. The server side consists of the model and the controller while the views are implemented on the client side. An application designed using an MVC pattern is divided into three separate parts, a Model, a Controller and a View, all of which communicate to each other. Models are an object that contains the data of the application. The model can receive data from the controller as well as send the data to the view. The controller is the main backbone of an application which contains all the logic required for an application to operate. The actions such as adding, editing, deleting and retrieving data from models are handled in the controller. The views of an application are what the user sees and interacts with. Users provide the data through the view and the data is displayed back in the view again.

Figure 3 illustrates the scopes of model, view and controller of an MVC application and how they communicate with each other.

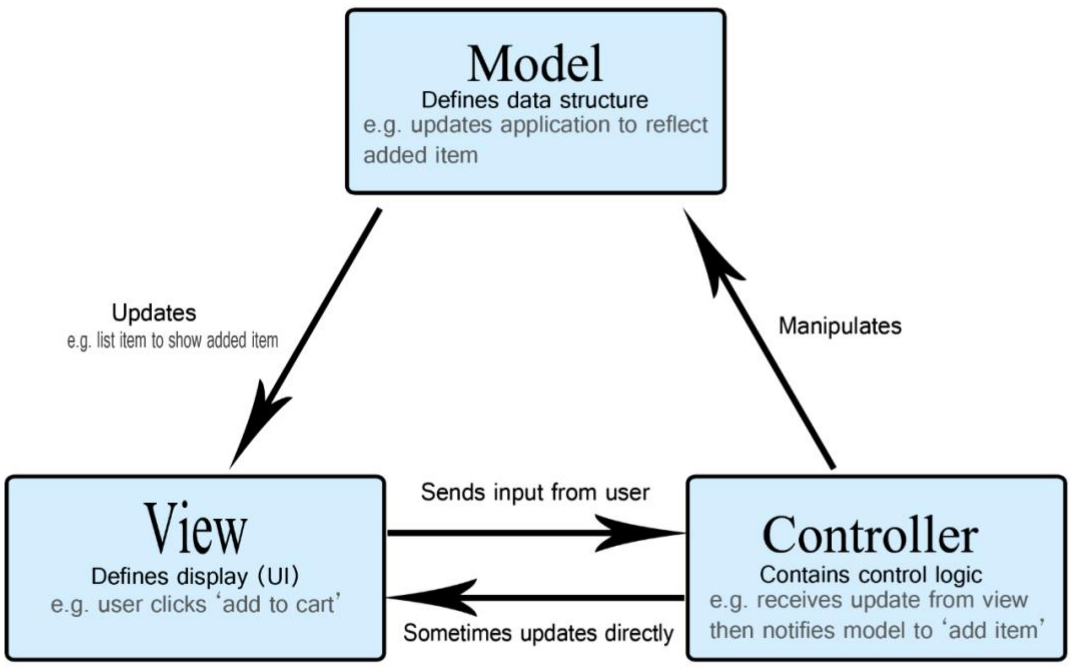

Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Figure 3: MVC architecture of an application. [17]

As shown in Figure 3, the view receives the input from the user and sends that to the controller. The controller then receives the data and modifies the model. Finally, the view gets the updated data from the model and displays the revised data in the view.

5.2 Development Environment Setup

It is vital for choosing the right tools and environment for performing any kind of work. The right combination of such tools increases the efficiency of the work, saves time, optimizes the performance as well as increase the reliability in the work done. Similarly, using the right tools and programming languages makes an application development a lot easier.

Kinmel.com is developed entirely on MacBook Pro 2017 with MacOS Mojave 10.14 as the operating system. In addition, several other software and tools were utilized in order to develop a fully functional e-commerce application.

5.2.1 Integrated Development Environment (IDE)

PhpStorm is a cross-platform, light weight and powerful IDE developed by JetBrains with built in developer tools, intelligent coding assistance, smart code navigation, reliable refactoring and easy debugging and testing [18]. The version of PhpStorm used for the development of this application is 2018.2. Additionally, a package called Node.js and NPM is installed in order in order to facilitate the development of Node application.

5.2.2 Version Control System and Hosting Service

Version control is a software tool that keeps track of changes made to a document or source code in a special database every single time the source is changed. Version control makes a group or a team working together on a project easier to collaborate with each other. There are numerous version control systems available which are Git, Sub version, Team Foundation Version Control and so on. Git, one of the most common version control system, is chosen for this project as this is a distributed version control system which emphasizes on speed, data integrity and support for distributed non-linear workflows. According to a survey conducted by Stack Overflow among the professional software developers in 2018, a whopping 88.4 percentage of respondents used Git as a version control system [19].

All the code and the code history for this application were hosted on GitHub which is a popular version control hosting service with 31 million registered users and more than 96 million repositories as of November 2018 [20]. An integrated issue tracker, branch comparison views and collaboration tools among the developers in a team are some of the main features provided by GitHub.

5.2.3 NPM, NodeJS, MongoDB, Express and EJS

NPM

Node Package Manager (NPM) is an open source package manager which helps installing several JavaScript packages available in the web. It is also used to share and distribute the code, manage application dependencies and provide and receive feedbacks with others. NPM is the default NodeJS package manager. [21]

NodeJS

The Long-Term Support (LTS) version of NodeJS was downloaded and installed from the official website of NodeJS. LTS version was chosen since it is a stable version of NodeJS with scheduled release cycle and an extended support system and security patch. The application was initialized using the command *npm init.* The name of the application, version, description of the project and the author name was specified during the initialization. A new file named ‘package.json’ was created with the information provided during the initialization.

Express

After installing NodeJS, Express framework versioned 4.16.0 was installed globally with the following command.

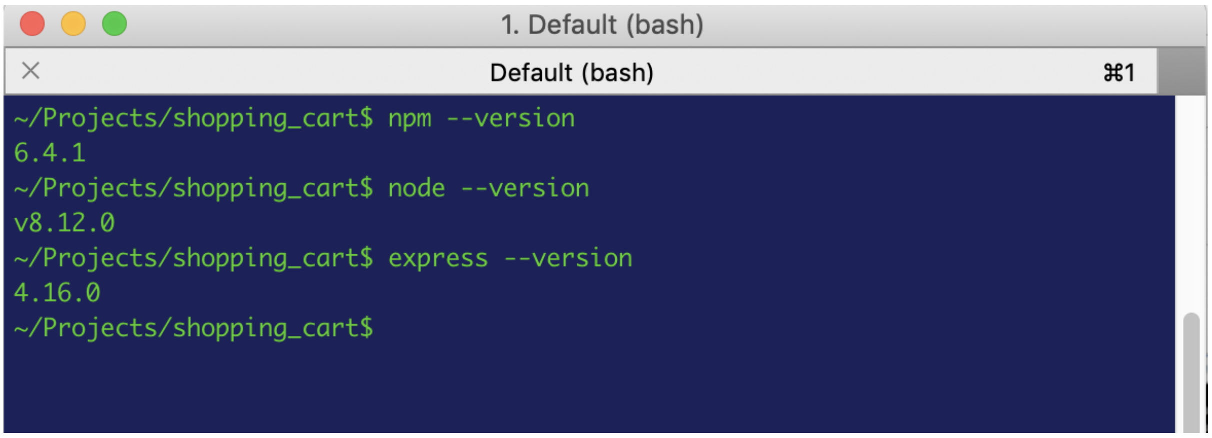
sudo npm install –express –save -g

Furthermore, an express generator was used in order to generate the standard folder structure of the application with some most commonly used middleware already included in the application. The following command was used to install the express-generator in the application.

sudo npm install express-generator -g

After this, a new express application is scaffolded with the command *express* <*shopping_cart*>. This downloads the minimal required files and folders for the application. Next, after moving into the recently generated folder <*shopping_cart*>, all the required modules already included in the package.json file are installed using the command *npm install*.

The different versions of npm, NodeJS and Express used in the development of the application is shown in Figure 4.



1. Default (bash)

Default (bash) ⌘1

```
~/Projects/shopping_cart$ npm --version
6.4.1
~/Projects/shopping_cart$ node --version
v8.12.0
~/Projects/shopping_cart$ express --version
4.16.0
~/Projects/shopping_cart$
```

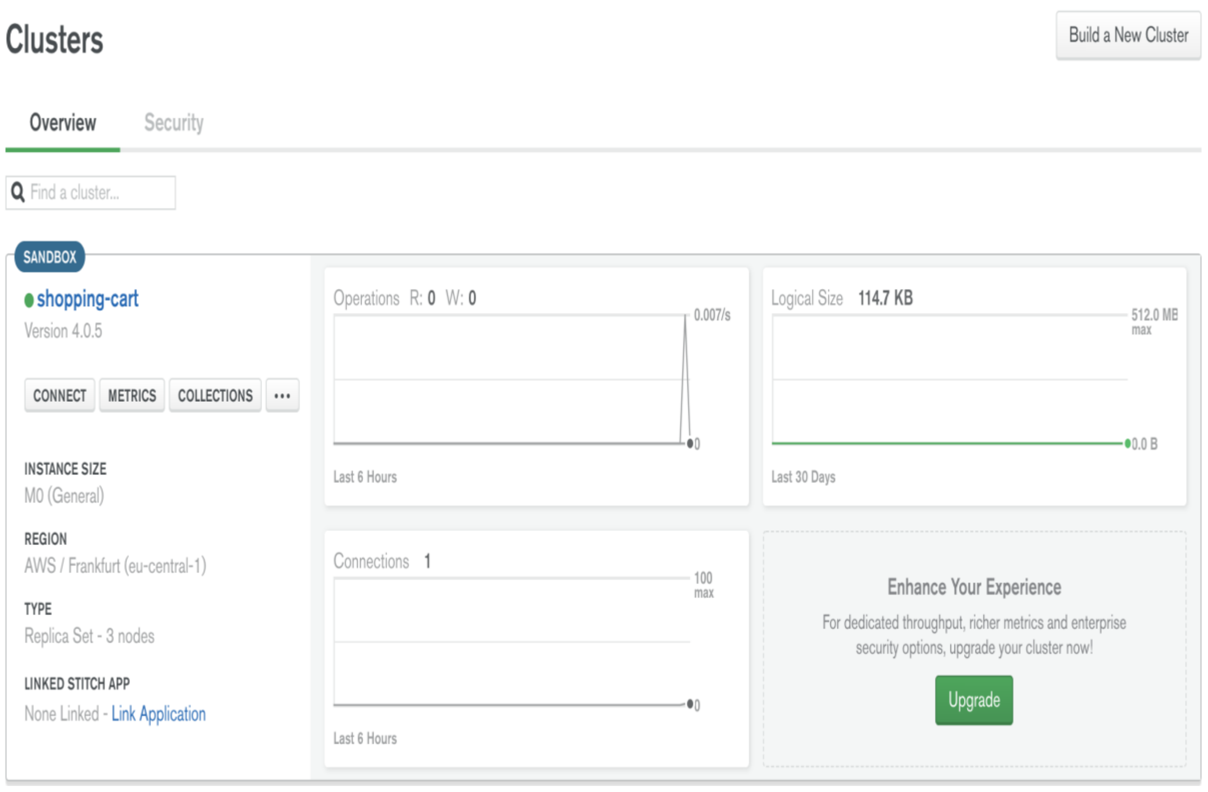
Figure 4: Terminal showing the version of NPM, NodeJS and Express

The NodeJS version used for developing the application is v8.12.0, as shown in Figure 4, which was the latest Long-Term Support (LTS) version at the time of developing this application. Moreover, the npm version 6.4.1 and express version 4.16.0 were also used in the project development.

MongoDB

MongoDB can be installed locally to the machine the application is being developed on or directly to the MongoDB Atlas as a cloud solution. An account was created in MongoDB website https://cloud.mongodb.com/user#/atlas/register/accountProfile. A new free MongoDB cluster was set up after creating a new project. Apart from this, a user with a read and write access to the database was created which was used in the node.js application to read and write the data from and to the database. Additionally, the current IP address of the machine was added to the IP whitelist so that the app can communicate with the server. Finally, MongoDB driver was installed in the application using the command 'npm install mongodb –save'.

Figure 5 shows the overview of a MongoDB cluster used for the application.



shopping-cart
Version 4.0.5

INSTANCE SIZE
M0 (General)

REGION
AWS / Frankfurt (eu-central-1)

TYPE
Replica Set - 3 nodes

LINKED STITCH APP
None Linked - Link Application

Operations R: 0 W: 0 — 0.007/s, 0

Logical Size 114.7 KB — 512.0 MB max, 0.0 B

Connections 1 — 100 max, 0

Figure 5: MongoDB cluster of the application

MongoDB cluster displays the geographical region where the server is located, read and write operations and the number of active connections on the server as well as the size of the database as illustrated in Figure 5.

Mongoose

The next step is to make use of mongoose to make the communication with the server and the database easier. The latest mongoose version available at the time of installing, version 5.3.10, was installed using the command *npm install mongoose –save* and the database connection was created between the application and the server using the URL, username and the password created while setting up the MongoDB Atlas. In addition, the required schemas and models were created which will be discussed later in chapter 7.3.

EJS

Embedded JavaScript (EJS) templating engine was selected for displaying the dynamic contents in the HTML page. Besides this, other templating engines such as Pug and Handlebars were also considered. These templates use different syntax and different set of features to output HTML content. Since EJS uses normal HTML syntax and can also implement some logic in the front-end using plain JavaScript, EJS was chosen as the templating engine for the application. EJS can be installed into the project with the command *npm install ejs –save*. Finally, it was then set as the view engine for the application.

5.2.4 Other Node Packages

Besides the core packages and tools explained in chapter 7.2.3, numerous other node packages were installed in the project to provide more features to the application. Some of them are discussed below.

Nodemon

Nodemon is a tool that used in the development environment of an application which helps automatically restart the node application after any changes that is made to the server-side code. It was installed with the command 'npm install nodemon –save -dev'. The flag '-dev' installs the dependencies only for the development environment.

Nodemailer

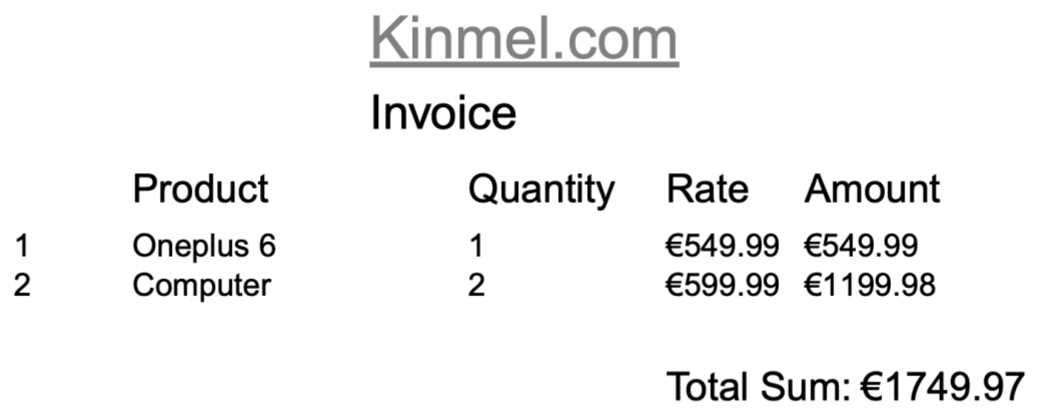
Nodemailer is a secure package that facilitates sending emails from inside a node application. It was installed with the command 'npm install –save nodemailer'. In addition to the plain text and HTML code, attachments like images and files can also be sent using the Nodemailer.

SendGrid

There are certain cases where an email has to be sent to the user. The common examples of these cases are while sending a link to the user to reset their password, notifying the successful registration to the user and so on. Creating and maintaining own email server is a very complex and expensive process. Hence, there was need of third-party email servers which could send emails securely and reliably. SendGrid is one of the email delivery engines that makes the task of sending emails easier. A free account was created on the SendGrid website which can send 100 emails per day for free. A transport plugin called 'nodemailer-sendgrid-transport' was installed with the command 'npm install nodemailer-sendgrid-transport –save'. This transporter enables the Nodemailer to send email through SendGrid's API.

PDFKit

The invoice needs to be generated while a user orders the products from the application. PDF (Portable Document Format) is the most common format for sending the invoices. Various node packages are available to print pdf in NodeJS. PDFKit generates the printable, multi-page and complex documents whenever the user wants to see or download using pure JavaScript. It was installed to the project using the command ‘npm install pdfkit –save’. The invoice was generated with the dynamic data from the user. A pdf invoice generated with the help of ‘pdfkit’ is shown in Figure 6.



Kinmel.com

Invoice

	Product	Quantity	Rate	Amount
1	Oneplus 6	1	€549.99	€549.99
2	Computer	2	€599.99	€1199.98

Total Sum: €1749.97

Figure 6: Invoice generated with PDFKit

As seen in Figure 6, an invoice is generated for the purchase made in Kinmel.com. The invoice includes hard-coded data as well as dynamic data such the name, quantity and rate of the product as well as the total amount of the products bought.

Stripe

Every e-commerce applications need some kind of payment services where their customer can pay for the goods they have purchased online. A third-party provider called stripe is integrated in the application which takes care of all the payments made to the shop. The client side of the application collects the credit card data from the customer with the help of stripe and is sent to the stripe for verification and validation. Once the credit card data is valid, stripe sends a token to the server and the amount to be paid is sent to stripe again with the same token. Stripe then charges the credit card, manages the payment and sends a response back to the server. Finally, once the successful response is received, the server clears the user's cart and creates an order. An example of a payment form by stripe is presented in Figure 7.

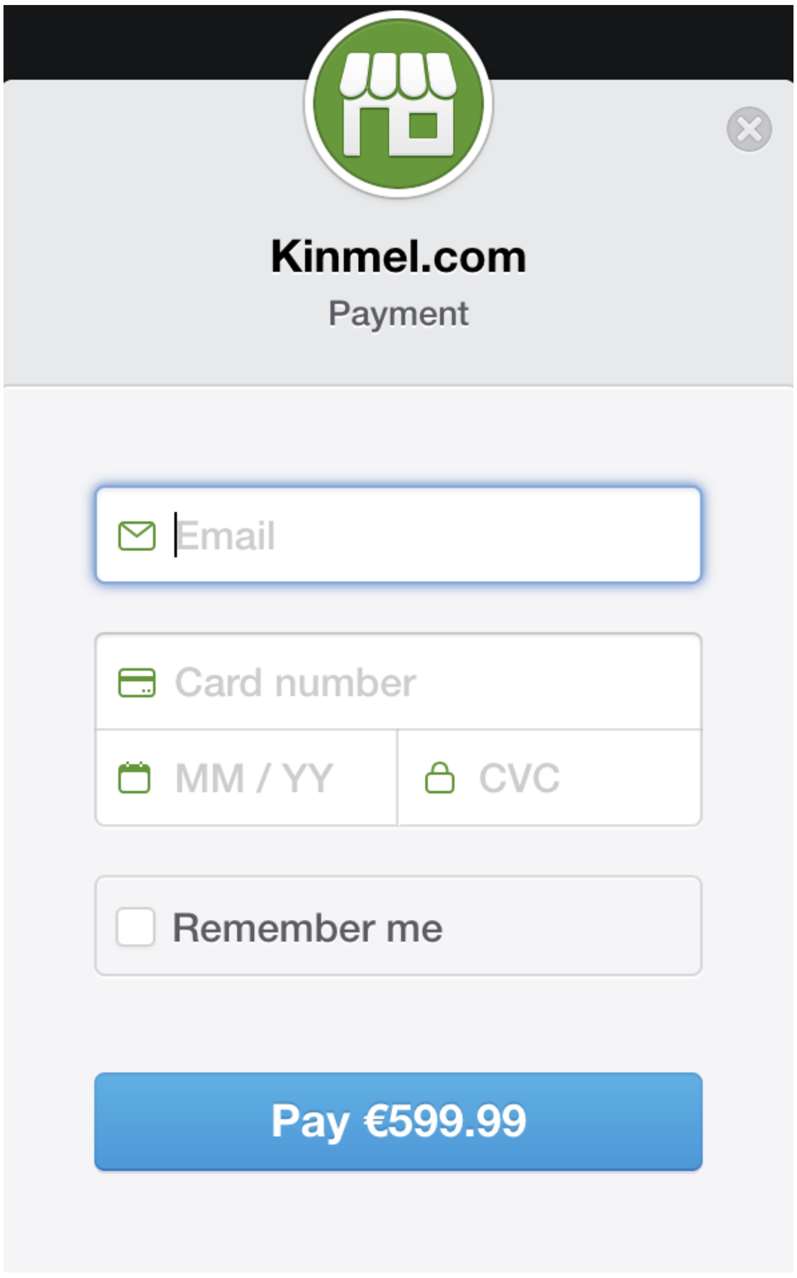
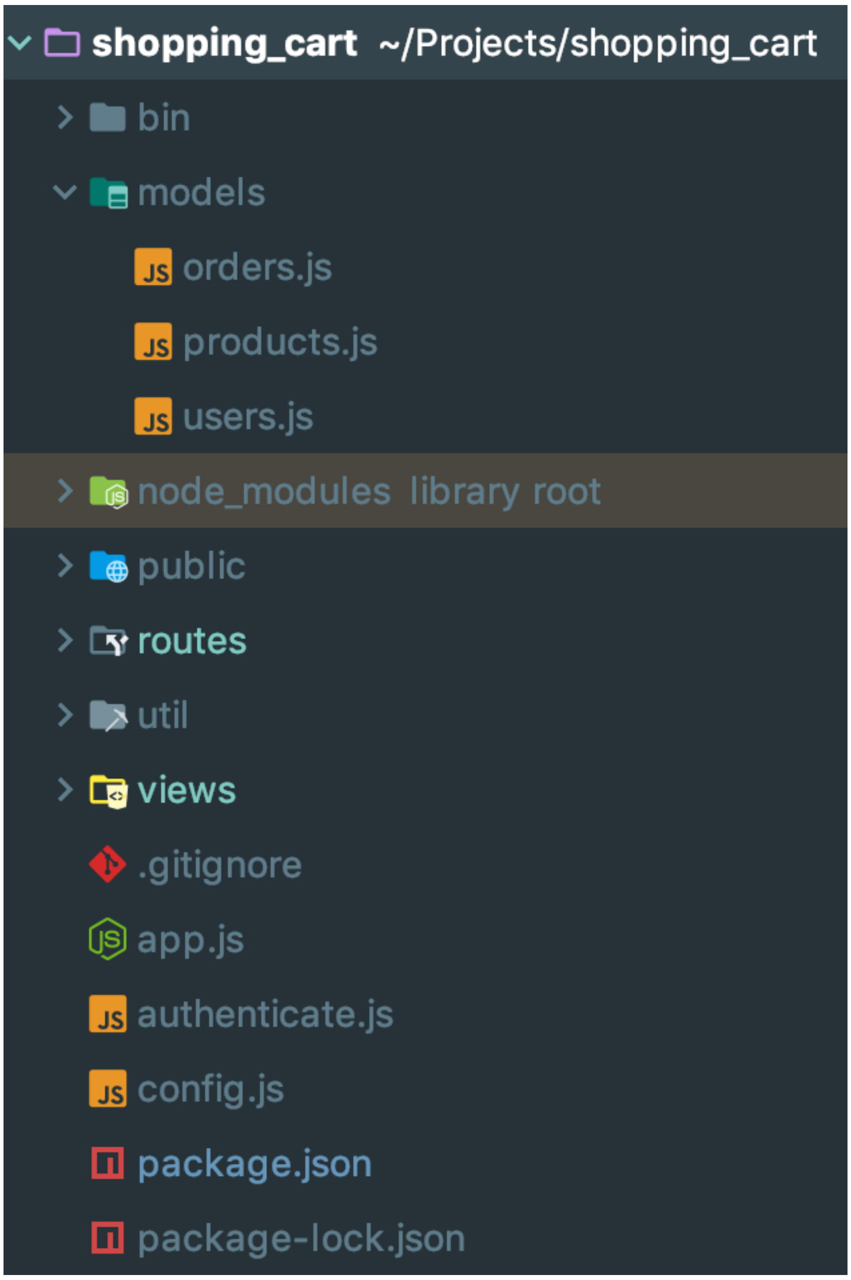


Figure 7: Stripe payment form

Figure 7 displays a payment form by Stripe which collects the credit card information from the customer. The form is sent to the stripe script already provided by stripe along with the total chargeable amount and the stripe API key provided to the application.

5.3 Application Logic

The development environment was set up and all the tools and node packages were downloaded and installed as explained in chapter 7.2. After setting up this basic structure for the application development, the process of actual coding started. The final folder and file structure of the application is as shown in Figure 8.



```
shopping_cart  ~/Projects/shopping_cart
    bin
    models
        orders.js
        products.js
        users.js
    node_modules  library root
    public
    routes
    util
    views
    .gitignore
    app.js
    authenticate.js
    config.js
    package.json
    package-lock.json
```

Figure 8: Folder structure of the application

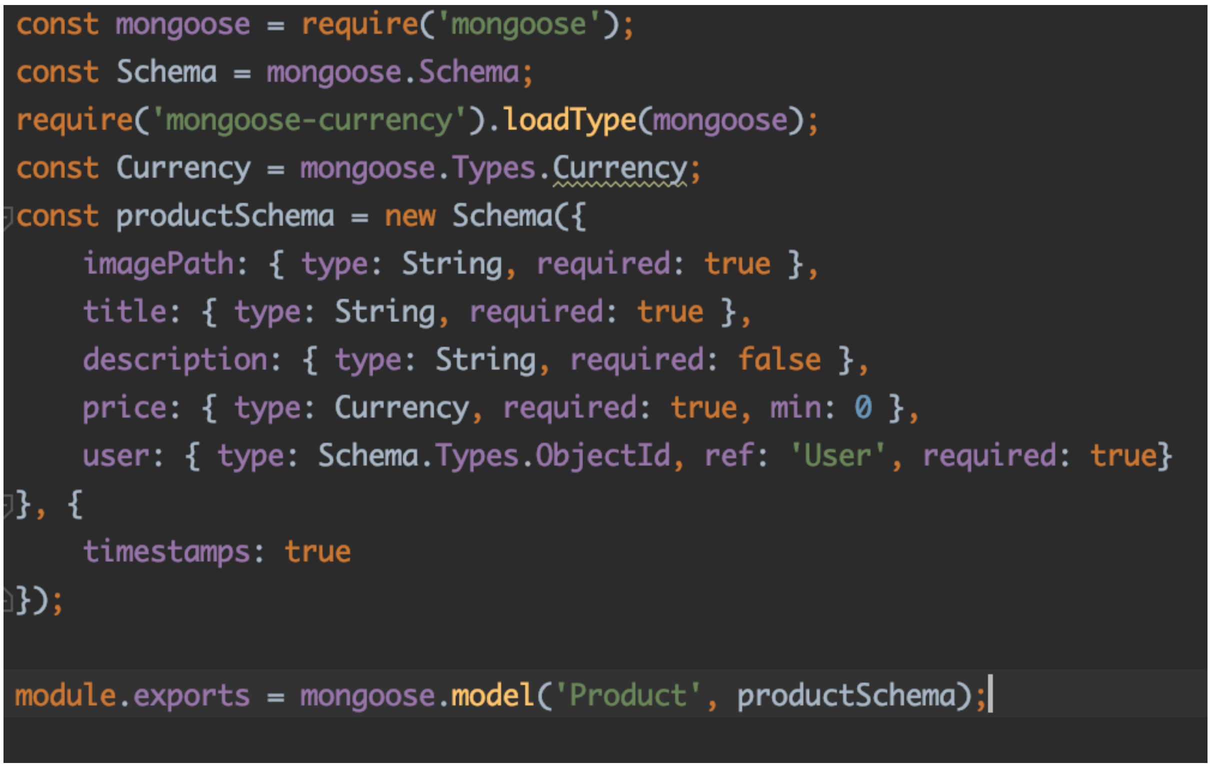
The application consists of several folders and files as shown in Figure 8 to implement the logic of the application. These files are divided into different folders so that the structure remains simple. The models, controllers and views of the application are placed in different folders which follows the MVC architecture. The 'node_modules' directory contains all the dependencies required by the server side. The 'public' folder contains all the statically served files and client-side scripts like the JavaScript, stylesheets, jQuery files as well as the images uploaded to the server. Similarly, the 'util' folder consists of the services and libraries such as code for deleting a file. The '.gitignore' file consists of all the untracked files and folders like node_modules, database configuration files and other private API keys that should not be uploaded to the version control. Moreover, ‘authenticate.js’ consists of the application logic which determines the authentication of a user while the ‘config.js’ file stores the configuration data required for the server to run. Besides, the implementation of other files and folders will be discussed below.

bin/www

The bin folder (www file) here contains the various scripts to start, stop or restart the project Moreover, it also sets up the HTTP server with the defined port or 3000 as default, listens to the server and reports errors if any. It is the main entry point to the application which first requires the app’s entry point ‘app.js’ file to handle the rest of the implementation.

Models

Models were created in order to define a standard structure for the documents being stored in a particular model. The application mainly consists of three models, namely, users, products and orders. The user model, product model and the order model store the collection of user documents, product documents and the order documents respectively. A schema for a product object is created in the product model and is exported to be used in other parts of the application as shown in Figure 9.



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
require('mongoose-currency').loadType(mongoose);
const Currency = mongoose.Types.Currency;
const productSchema = new Schema({
    imagePath: { type: String, required: true },
    title: { type: String, required: true },
    description: { type: String, required: false },
    price: { type: Currency, required: true, min: 0 },
    user: { type: Schema.Types.ObjectId, ref: 'User', required: true}
}, {
    timestamps: true
});

module.exports = mongoose.model('Product', productSchema);
```

Figure 9: Product Model

The product model only accepts the product properties defined in the schema as shown in Figure 9. According to this, the imagePath property is required and must be a string. Similarly, product model also contains the title, description and price of the product. The user object stores the reference of the user who created this product. Furthermore, the mongoose property ‘timestamps’ adds the created and modified date and time object to the product property. Finally, the product model is exported to be used in other parts of the application.

Routes

The ‘route’ folder contains all the server-side logic of the application. The REST APIs as well as the routing are implemented in this section. The ‘users.js’ file contains the logic and the route for all the user related activities such as displaying the register and login page, registering a user to the database and checking for the user information whenever a user tries to log in to the application. Similarly, a logout route as well as reset password route exists in this file. Similarly, other files such as orderRouter.js, cartRouter.js and productRouter.js contain all the route and logic associated with the order, cart and the product respectively.

Views

The ‘views’ folder contains the client-side of the application. It renders the HTML templates to the browser and displays the information to the client. Figure 10 provides an overall structure of the ‘views’ directory.

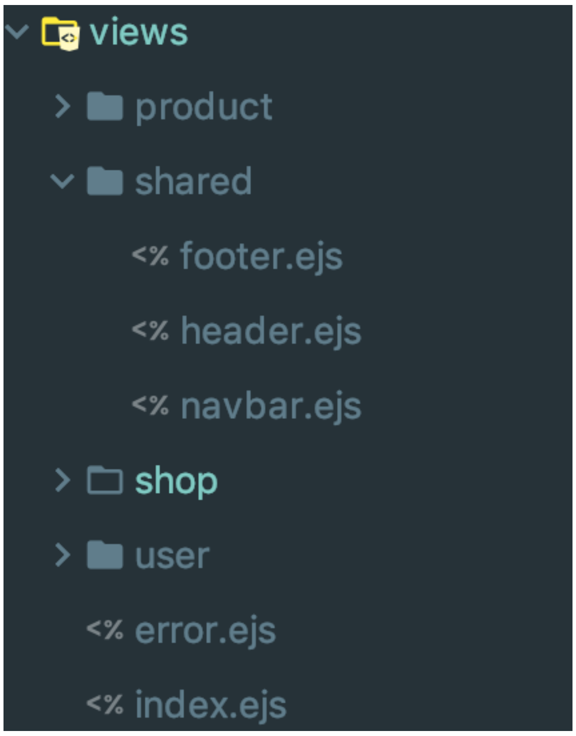
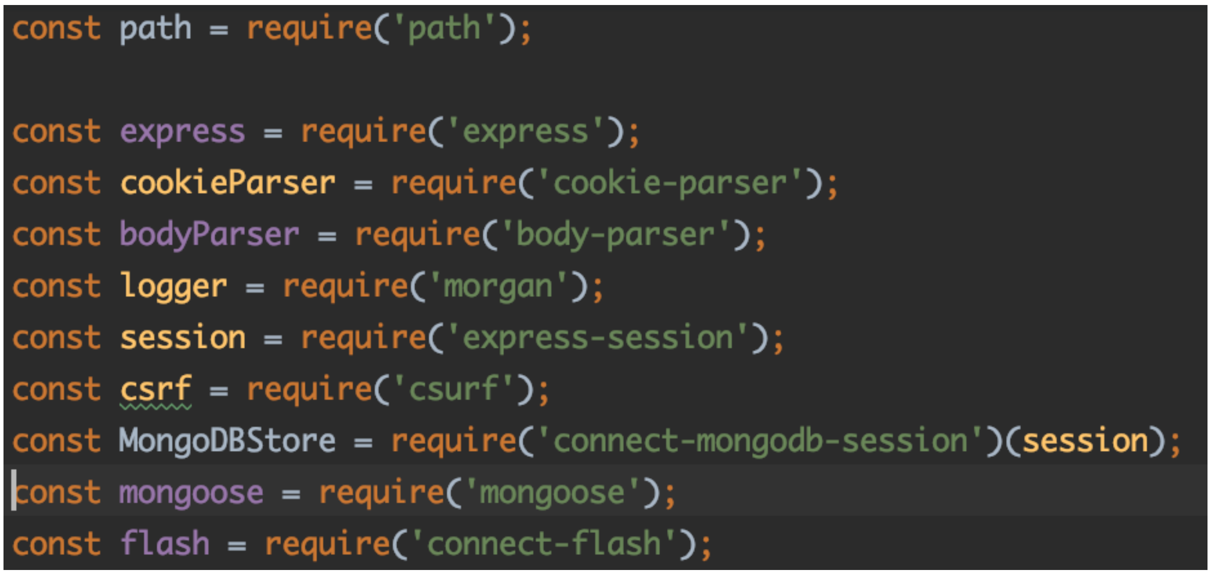


Figure 10: Views folder

The folder is sub-divided according to the page it renders as illustrated in Figure 10. The ‘index.ejs’ and ‘error.ejs’ are the default pages which displays the index and the error page of the application. In addition, the ‘shared’ folder contains the header, footer and the navigation bar which is shared across all the pages rendered. The product, shop and user directories comprise the ejs templates for displaying the corresponding HTML pages.

app.js

The ‘app.js’ file is the entry point to the express application. It imports all the required middleware, sets up the application with required settings and configuration, creates an app object and export that object from the module. A snippet from the ‘app.js’ file is provided in Figure 11.



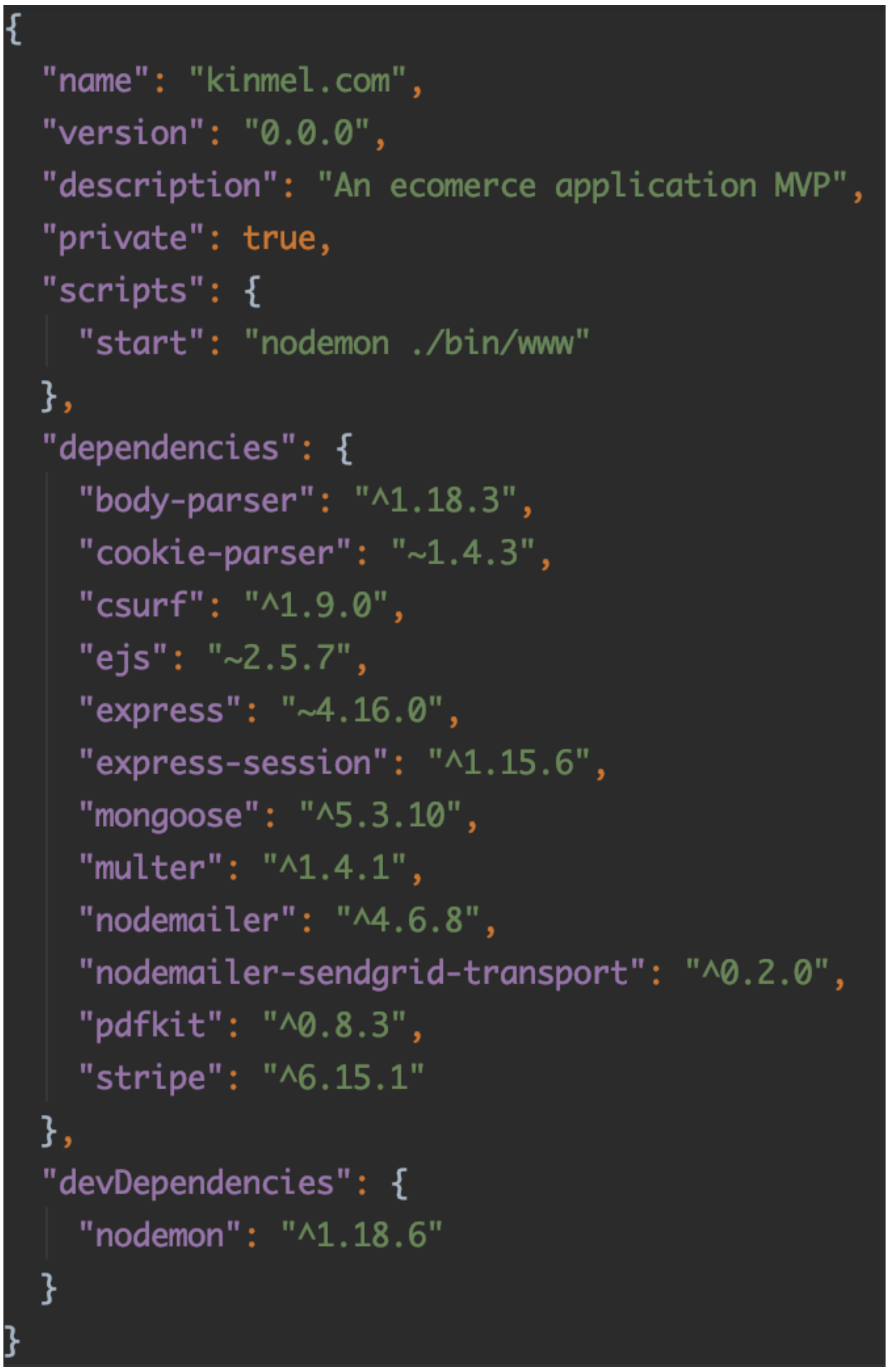
```
const path = require('path');

const express = require('express');
const cookieParser = require('cookie-parser');
const bodyParser = require('body-parser');
const logger = require('morgan');
const session = require('express-session');
const csrf = require('csurf');
const MongoDBStore = require('connect-mongodb-session')(session);
const mongoose = require('mongoose');
const flash = require('connect-flash');
```

Figure 11: Code snippet of app.js

As shown in Figure 11, several node libraries, models, routes and objects from other parts of the application which were exported using the ‘module.exports’ function is imported and used here. Moreover, a database connection was made, and an app object was created using express. This app object was used to set up the view engine or the template, add the libraries imported as middleware, add all the routes imported to define the routes for the application, handle the errors and finally export the created app object.

package.json

The ‘package.json’ file lists the detailed project information as well as the required dependencies of the project. It can contain the keywords, homepage, bugs, license, author, contributors, repository and so on in addition to the basic information such as the name, version and description. A ‘package.json’ file used in the project is presented in Figure 12.



```
{
  "name": "kinmel.com",
  "version": "0.0.0",
  "description": "An ecomerce application MVP",
  "private": true,
  "scripts": {
    "start": "nodemon ./bin/www"
  },
  "dependencies": {
    "body-parser": "^1.18.3",
    "cookie-parser": "~1.4.3",
    "csurf": "^1.9.0",
    "ejs": "~2.5.7",
    "express": "~4.16.0",
    "express-session": "^1.15.6",
    "mongoose": "^5.3.10",
    "multer": "^1.4.1",
    "nodemailer": "^4.6.8",
    "nodemailer-sendgrid-transport": "^0.2.0",
    "pdfkit": "^0.8.3",
    "stripe": "^6.15.1"
  },
  "devDependencies": {
    "nodemon": "^1.18.6"
  }
}
```

Figure 12: package.json file

As shown in Figure 12, ‘package.json’ file includes the name, description and version number of the application, dependencies required for the application and their version number as well as scripts to start or stop the application.

5.4 Security

Security is a vital element in all of the web applications. The vulnerability of web applications is increasing day by day due to the rapid increase in cyber-attacks and data breaches of individuals. Hackers use various methods such as phishing and hacking to access the login credentials as well as the credit card data. Moreover, unprotected routes provide easy access for the attackers into the application. Hence, the web applications must be tightly secured. Moreover, as e-commerce web applications include a large number of monetary transactions through the buying and selling of goods, such applications should put security as its topmost priority. A small infringement of security could lead to an enormous loss to the company as well as their customers. The following security features were enabled in the application to ensure the safety of all the stakeholders.

5.4.1 Authentication and Authorization

Authentication is the process of verifying the identity of a user accessing the application. Users can verify themselves to the server with the simple login form, online banking data, fingerprint, voice and face recognition as well as by retina scans. However, the most common process for verifying the authenticity of a user is via the login form. The user creates an account for themselves with some basic information of them such as email and types in a password which is encrypted and saved to the database. Hence, while logging in, the server verifies the information provided by the user to the information stored in the database. Once, the verification is successful, a session is created for the user which is stored in the client’s browser as a cookie and sent with every other request sent by the user. The session is destroyed once the user logs out of the application or when the browser is closed.

In contrast, authorization is the process of identifying if an authenticated user has the permission to access the services provided by the application. Kinmel.com has three levels of user roles, namely, unauthenticated users, authenticated users and admins. Unauthenticated users are able to browse all the products available, see the details of each product and register themselves in the application. Additionally, the logged in users are able to access the services such as adding a product to the cart, paying for the items in the cart and creating an order of the products bought in addition to the services accessed by unauthenticated users. Furthermore, admins are the highest in terms of the privilege provided to users. Admins are able to add the products to the database, edit them and see the orders placed by the users.

5.4.2 Filtering and sanitizing user input

The input data submitted to the server by the users should always be considered as a security threat. The users using the application are not always genuine users, there might be some who are trying to harm the server or steal confidential information from the database. Hence, the data sent by the user to the server must always be validated and sanitized. Validation can be implemented both in the client side as well as the server side. The client-side validation does not actually prevent the application from harmful attacks. Hence, the server-side validation is a must for all applications involving the user data.

Express-validator is a third-party package that validates and sanitizes the data sent by the users. It can be installed in the application using the command 'npm install –save express-validator'. The incoming data is checked in each route and the errors, if any, are collected in a constant which is sent back to the user to correct and submit the data again. Likewise, the input data can also be sanitized with the express-validator before persisting them to the database.

5.4.3 Cross-Site Request Forgery (CSRF)

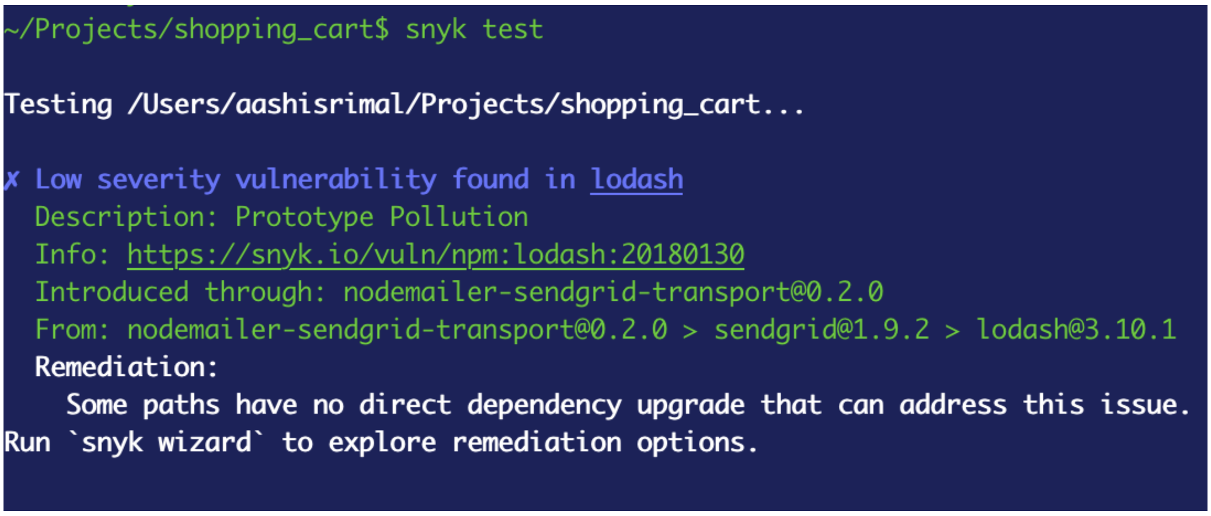
The CSRF attacks are performed by stealing sessions from a logged in user and sending requests with forged data to the real server. This type of attack cannot steal the data; however, it can manipulate and change the user's data such as email address, password or even transfer funds.

This attack pattern in mitigated in the application using a node package called 'csurf'. Csurf generates a token, called as csrf token, which is embedded in all the forms which when submitted changes the data of a user in the server. This token is then sent to the server in every request and is checked by csurf if it is a valid token. This way, even if the fake sites sent the request to the server with a stolen session, the requests lack the token and is denied.

5.4.4 Vulnerable third-party modules

NodeJS is a combination of built-in core modules and other third-party modules. Such modules are imported into the application as per the necessities to help with various functionalities that are not available in NodeJS by default. Implementing one module in the application requires a number of other modules that the module is depending on. The more third-party modules there are in the application, the more vulnerable it is to attacks and data leaks.

All these third-party packages are managed efficiently with npm. Since npm version 6, it reviews all the installed libraries automatically. Moreover, a command *npm audit* checks the malicious packages from the installed third-party libraries and lists all such libraries. Furthermore, another third-party package called ‘snyk’ makes the application more secure by checking against all the known vulnerabilities in the dependencies list. It was installed with the command ‘npm install –g snyk’ tested the project with the command ‘snyk test’. [22] A test case for the project is shown in Figure 13.



```
~/Projects/shopping_cart$ snyk test

Testing /Users/aashisrimal/Projects/shopping_cart...

✗ Low severity vulnerability found in lodash
  Description: Prototype Pollution
  Info: https://snyk.io/vuln/npm:lodash:20180130
  Introduced through: nodemailer-sendgrid-transport@0.2.0
  From: nodemailer-sendgrid-transport@0.2.0 > sendgrid@1.9.2 > lodash@3.10.1
  Remediation:
    Some paths have no direct dependency upgrade that can address this issue.
Run `snyk wizard` to explore remediation options.
```

Figure 13: Testing the project with snyk

The command ‘snyk test’ was run in the command-line tool which listed all the vulnerabilities in the external dependencies of the application as shown in Figure 13. The project found a low-risk vulnerability in the ‘nodemailer-sendgrid-transport’ dependency. It also provides the remediation strategies by running another command ‘snyk wizard’.

5.4.5 HTTP Header Injection

Injecting malicious code in the HTTP header may affect the browser’s built-in security mechanisms such as the same-origin-policy or XSS (Cross-Site Scripting) filter. This can lead to the exposure of sensitive information such as CSRF token. Moreover, the attacker can exploit the XSS vulnerabilities by setting up cookies. For example, the attacker might input some code which deactivates the same-origin-policy of browser and activates the CORS

(Cross-Origin Resource Sharing), the sensitive information might be stolen using JavaScript code. [23]

HTTP header injection can be prevented in the NodeJS application with a third-party package called Helmet. Helmet is a collection of small middlewares that set HTTP header to the response. The different modules in the library can be activated in the application which sets the Content Security Policy, controls browser DNS prefetching, prevents clickjacking, removes X-Powered-By header, disables the client-side caching and so on. [22]

5.4.6 Cookies and Secure Session Handling

Attackers will be able to gain access to an application by stealing the session from logged in users. The sessions and the cookies used in the application must always be secure so that the attackers will not be able to steal them from a client's browser. The session can be stolen mainly by session hijacking or session fixation. In session hijacking, the attacker tries to steal or predict a valid session id and use that to gain unauthorized access to the web server. The common methods for hijacking a session are predicting the valid session token, sniffing the session token, client-side attacks such as XSS and harmful JavaScript codes as well as by intercepting the communication between the client and the server. [24] However, in session fixation, the attacker already has access to a valid session which is then injected to the victim's browser. The injection can be done by sending the session token in the URL or by sending as a hidden form field or by setting the session id as a cookie in the victim's browser. When a victim logs in to the application, the web server does not create a new session id and assigns the one sent by attacker as victim's session id. [25] With this session id, the attacker can log in to the application as a victim and perform harmful activities.

Express-session middleware is used in the application to handle the session and the cookies. It was installed in the application with the command 'npm install –save express-session'. In order to secure a cookie, a signed cookie with a hard to guess secret needs to be used. Furthermore, various other options can be set while initializing the session which ensures the safety of cookies. The options 'secure' and 'httpOnly', if set to true, sends the cookies only over the secure HTTPS connection. Likewise, the domain and the path options compare the domain of the cookie to that of the server and send the cookie to the request only if both of them matches. Similarly, cookies can be set to expire after a certain period of time which minimizes the risk of being misused. [22]

5.4.7 Using Transport Layer Security (TLS)

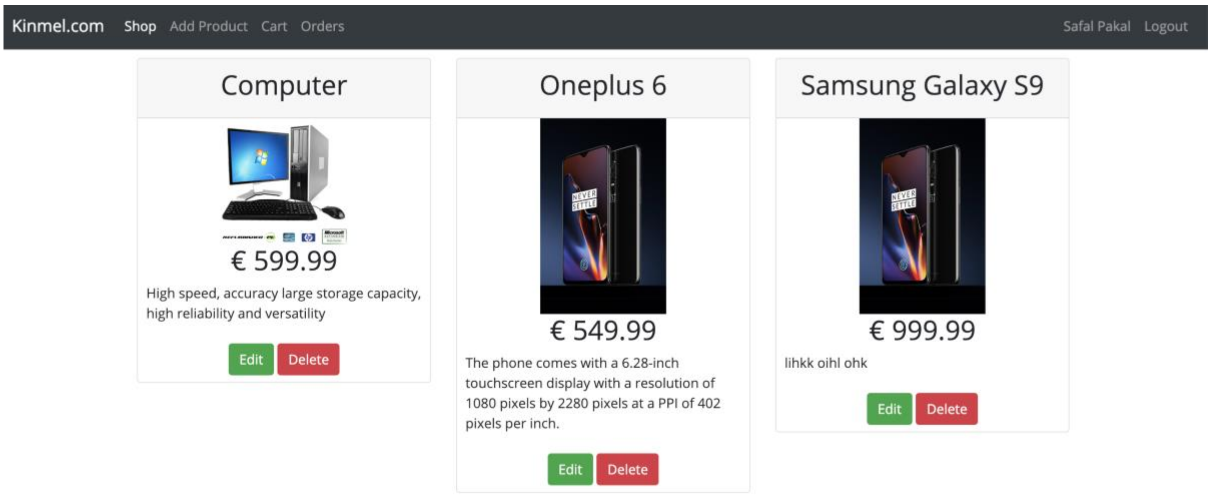
Transport Layer Protocol (TLS) is a cryptographic protocol that secures all the communication between a browser and a server. It is the newer version of Secure Sockets Layer (SSL). This protocol prevents the stealing of data by encrypting the data even before being sent from the client's browser to the web server. A TLS certificate can easily be obtained free of cost from 'Let's Encrypt'. 'Let's Encrypt' is a free and open certificate authority that provides the digital certificate to enable HTTPS connection for websites. [22]

5.4.8 MongoDB Atlas IP Whitelist and User Authentication

MongoDB Atlas provides a secure way of storing the data in the cloud. There are various security measures taken to minimize the risk of database attack. Each cluster in the MongoDB Atlas has users with specific roles. These user needs to be configured while connecting the database from the server. Only the authenticated user with the read and write access to the database is able to perform read, write, update and delete operations in the database. In addition to this, Atlas only allows the connection to the database from the IP addresses that are listed in the project's whitelist. The IP address of the machine the project is deployed on must be added so as to connect successfully to the database. [26]

6 Results and Discussion

The MVP prototype application for demonstrating the use of Full Stack JavaScript was developed successfully. NodeJS along with Express and several other node libraries were used for developing the backend of the application while EJS was used as a templating engine. A NoSQL database, MongoDB was used for storing the data as documents.



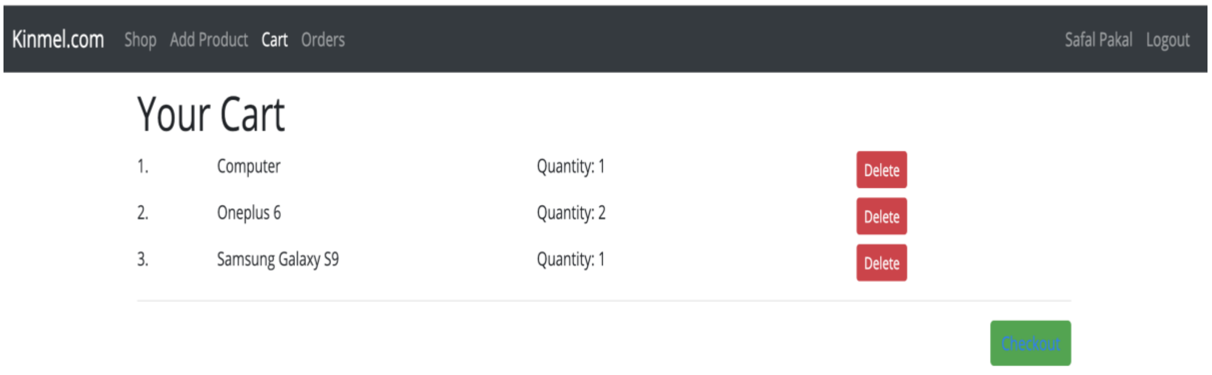
Computer
€ 599.99
High speed, accuracy large storage capacity, high reliability and versatility

Oneplus 6
€ 549.99
The phone comes with a 6.28-inch touchscreen display with a resolution of 1080 pixels by 2280 pixels at a PPI of 402 pixels per inch.

Samsung Galaxy S9
€ 999.99
lihkk oihl ohk

Figure 13: Landing page of the application

The landing page of the application for the admin users is shown in Figure 13. It has a navigation bar to navigate to different pages of the application. However, the ‘Add Product’, ‘Edit’ and ‘Delete’ buttons are only shown to admin users. Furthermore, each individual product is linked to an edit and delete button which the admins can use to ‘Edit’ and ‘Delete’ the respective product. The product details page is accessed when clicking on the product which also has a button to add the product to the cart. A basic view of the cart page is shown in Figure 14.

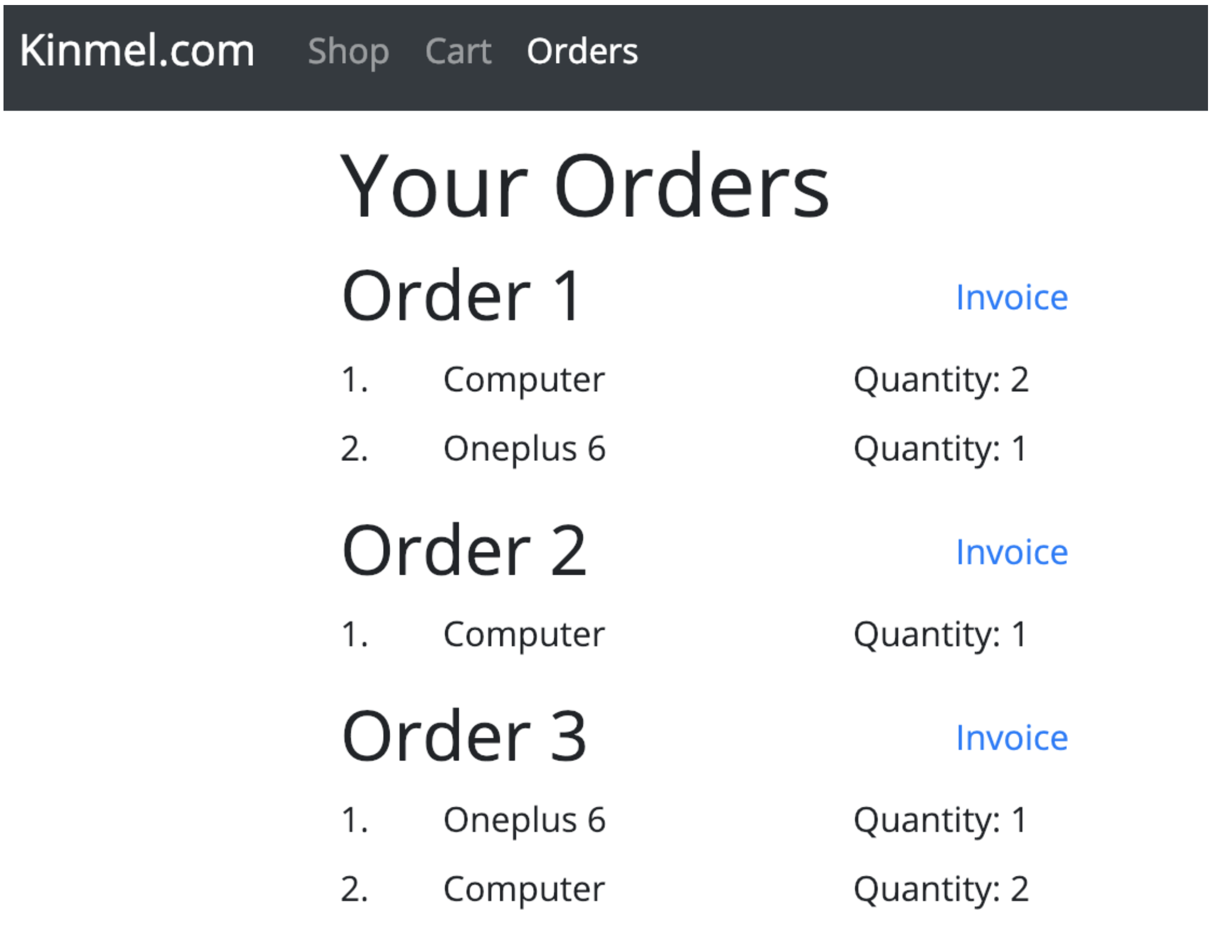


Your Cart

1.	Computer	Quantity: 1
2.	Oneplus 6	Quantity: 2
3.	Samsung Galaxy S9	Quantity: 1

Figure 14: Cart page

The ‘cart’ page displays the products that a user has added to the cart as displayed in Figure 14. Moreover, the user can delete the product from the cart with the delete button. The ‘Checkout’ button opens a form which lets the user input the credit card information and pay for the order. After the payment is done, the user is redirected to the ‘order’ page and the ‘order’ page is displayed as shown in Figure 15.



Your Orders

Order 1 Invoice

1. Computer Quantity: 2
2. Oneplus 6 Quantity: 1

Order 2 Invoice

1. Computer Quantity: 1

Order 3 Invoice

1. Oneplus 6 Quantity: 1
2. Computer Quantity: 2

Figure 15: Orders page

As shown in Figure 15, the ‘order’ page is where all the orders placed by the logged in user are displayed. The page also consists of a link to download the pdf invoice on each order.

Although MEAN stack and MERN stack are considered the most buzzing stack in the full stack JavaScript web development, this application’s client side is developed using a templating engine called EJS so as to make the application ready as a minimum viable product. However, the front end will be developed with React in upcoming releases.

The full stack JavaScript proved to be one of the best stacks to develop an application. The separation of model, views and controllers in the application made it easy to implement the business logic to the application. Moreover, there was no need to switch between different programming languages and knowledge of JavaScript was enough to develop the full application. Furthermore. MongoDB uses JSON format to store the documents which is a preferred format for JavaScript developers as it is easy to transform the data and work with it. Similarly, the documents can be stored in the cloud even in the development environment which minimizes the efforts to change the database server form development to production during the deployment of the application. Apart from this, NodeJS, Express and MongoDB is already a proven technology and is being used by small companies to larger tech giants; hence, it is easy to get solutions from the developer community in case of any difficulties.

Despite all these advantages, there are a few disadvantages as well that needs to be discussed. First, it was somehow difficult to manage the database as MongoDB is based on document object model completely different from the traditional relational databases that were being used. It does not provide the same level of functionalities and flexibility as the relational databases do. There were high chances of the same data being repeated in different collections as one document contains other embedded documents. Sometimes, the decision to store just the reference to an object id or the full document had to be made, for example, while storing an order, full product object had to be embedded in the order object since the changes in the product object affected the orders made before the changes.

Second, handling of the payment using the Stripe service was troublesome in a project which uses csurf to protect from the CSRF attacks. As Stripe already comes with the CSRF protection, the csrf token generated by the project was causing an error in processing the request. Thus, the payment route involving Stripe had to be taken out of the controller directory and was placed in ‘app.js’ file before the csurf middleware was implemented. This somehow affected the MVC architecture of the application.

7 Conclusion

The main goal of the thesis was to study and report different aspects of full stack JavaScript frameworks and develop a prototype application based on it. As the author was new to the JavaScript frameworks, although not completely new to the JavaScript programming language, a significant amount of time was invested in learning different aspects of the frameworks such as NodeJS, Express and MongoDB. The advantages as well as the disadvantages of developing a web application using these technologies were reviewed and compared with other similar technologies. Moreover, the speed and performance of the application were also acknowledged.

The experience gained during the thesis proved that JavaScript is the best language for developing web applications as it provides all the components needed for an application development such as NodeJS for the server side, Angular, React and others for the client side as well as MongoDB as a database. The NodeJS application comes with great speed and fast development cycle. Moreover, it is best suitable for developing high-performance and scalable web applications. The presence of a huge pool of NodeJS libraries and packages helps to extend the services provided by NodeJS without much concern. In addition, the new JavaScript features are being added to the language every year and a release of new ECMAScript versions scheduled every year, there is no doubt that JavaScript is the technology of the future.

Nevertheless, full-stack JavaScript is not the best solution for developing all kinds of applications. NodeJS is unable to handle heavy data processing and computation in the backend. Hence, high tech enterprises which use machine learning or algorithms cannot benefit from this technology. Despite this, the involvement of JavaScript developers' community and its open source nature, NodeJS is only going to evolve and flourish from here.

References

1 MDN Web Docs. What is JavaScript? Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [Accessed 4 December 2018].

2 Alexsoft. The Good and the Bad of JavaScript Full Stack Development. Available from: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-javascript-full-stack-development [Accessed 24 December 2018].

3 Harrell J. Node.js at PayPal. 22 November 2013. Available from: https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal [Accessed 24 December 2018].

4 Stack Overflow. Developer Survey Results 2018. Available from: https://insights.stackoverflow.com/survey/2018#technology [Accessed 23 December 2018].

5 ECMA International. ECMAScript 2018 Language Specification. June 2018. Available from: https://www.ecma-international.org/ecma-262/9.0/index.html [Accessed 26 December 2018].

6 ECMA. ECMAScript Language Specification. 24 March 2000. Available from: https://www-archive.mozilla.org/js/language/E262-3.pdf [Accessed 15 December 2018].

7 Rauschmayer A. Speaking JavaScript: An In-Depth Guide for Programmers. Chapter 5 Standardization: ECMAScript. March 2014. Available from: http://speakingjs.com/es5/ch05.html [Accessed 4 December 2018].

8 Rauschmayer A. Exploring ES6: Upgrade to the Next Version of JavaScript. About ECMAScript 6. 2016.

9 MDN Web Docs. Const. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const [Accessed 25 December 2018].

10 ECMA International. ECMAScript Language Specification: 5.1 Edition. Available from: http://www.ecma-international.org/ecma-262/5.1/ECMA-262.pdf [Accessed 15 December 2018].

11 Zakas Nicholas C. Understanding ECMAScript 6. Functions. Available from: https://leanpub.com/understandinges6/read#leanpub-auto-functions [Accessed 26 December 2018].

12 NodeJS. About Node.js. Available from: https://nodejs.org/en/. [Accessed 27 December 2018].

13 Roth I. What makes NodeJS faster than Java? 30 January 2014. Available from: https://strongloop.com/strongblog/node-js-is-faster-than-java/ [Accessed 27 December 2018].

14 Express. Express: Node,js web application framework. Available from: https://expressjs.com/ [Accessed 27 December 2018].

15 MongoDB. MongoDB Architecture Guide, MongoDB 4.0. June 2018.

16 Mongoose. Mongoose ODM v5.4.5. Available from: https://mongoosejs.com/ [Accessed 28 December 2018].

17 MDN Web Docs. MVC Architecture. Available from: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture [Accessed 03 January 2019].

18 JetBrains. PhpStorm The Lightning-Smart PHP IDE. Available from: https://www.jetbrains.com/phpstorm [Accessed 07 January 2019].

19 Stack Overflow. Developer Survey Results 2018. Available from: https://insights.stackoverflow.com/survey/2018#work-version-control [Accessed 07 January 2019].

20 GitHub. About. Available from: https://github.com/about [Accessed 07 January 2019].

21 NPM. Available from: https://www.npmjs.com/ [Accessed 07 January 2019].

22 Express. Production Best Practices: Security. Available from: http://expressjs.com/en/advanced/best-practice-security.html [Accessed 15 January 2019].

23 Netsparker. CRLF Injection and HTTP Response Splitting Vulnerability. Available from: https://www.netsparker.com/blog/web-security/crlf-http-header/ [Accessed 15 January 2019].

24 OWASP. Session Hijacking Attack. Available from: https://www.owasp.org/index.php/Session_hijacking_attack [Accessed 24 January 2019].

25 OWASP. Session Fixation. Available from: https://www.owasp.org/index.php/Session_fixation [Accessed 24 January 2019].

26 MongoDB. Security Features and Setup. Available from: https://docs.atlas.mongodb.com/setup-cluster-security/ [Accessed 15 January 2019].