A Thesis/Project/Dissertation Report

on

BURGER BUILDER WEB APPLICATION

Submitted in partial fulfillment of the
requirement for the award of the degree of

B.TECH IN COMPUTER SCIENCE ENGINEERING

GALGOTIAS
UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

DR. KIRTI SHUKLA
(ASSOCIATE PROFESSOR)

Submitted By

NAVED MAHTAB KHAN -18SCSE1010065
SAJJAN SHAH – 18SCSE1180070

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"BURGER BUILDER WEB APPLICATION** in partial fulfillment of the requirements for the award of the **B.TECH** —submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 TO DECEMBER-2021**, under the supervision of **DR. KIRTI SHUKLA (ASSOCIATE PROFESSOR)**, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

NAVED MAHTAB KHAN , 18SCSE1010065

SAJJAN SHAH , 18SCSE1180070

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

DR. KIRTI SHUKLA

(ASSOCIATE PROFESSOR)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **NAVED MAHTAB KHAN | 18SCSE1010065 , SAJJAN SHAH | 18SCSE1180070**-has been held on **__________________** and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: DECEMBER , 2021

Place: Greater Noida

ABSTRACT

As we can see in various fast food companies like Burger King , Mcdonalds have fixed Burger menu on their services and this problem doesn’t satisfy the need of the today’s customer because sometimes the customer want to have a custom burger according to their liking and disliking.So to overcome this problem we came with an idea for building this application called burger Builder where customer can easily customize their burgers and can order it. In this Application the tech stack used is react.js for UI interface frontend and firebase for backend .In this application there are various ingredients such as vegetable, cheese, salad , chicken , bon. where user can add and delete the ingredients to make their custom burger, along with this price of burger is also shown and after making the custom burger user can order it by providing the details . and for designing frontend part we are using react,js a javascript main library. and for backend part we are using firebase a serverless architecture.

The Application also used proper authentication and from validation and implements routing uses React Router. Since this web application is made React , BurgerBuilder is fast single page application enriched with UI design and also have better user experience By the help of this application it solves the problems of the fast food companies where customers get benefitted by ordering custom burgers.

Table of Contents

CHAPTER-1

INTRODUCTION

A Burger builder Application built using React Js and ReduxJs. The application uses all the new features introduced in EcmaScript 6. The application is a single page application with proper components and is Mobile Responsive. The Application also used proper authentication and from validation and implements routing uses React Router.

Since this web application is made React , BurgerBuilder is fast single page application , enriched with UI design and also have better user experience

By the help of this application it solves the problems of the fast food companies where customers get benefitted by ordering custom burgers.

OBJECTIVE

Build a full stack web application – BurgerBuilder WebApplication using Node.JS, React.Js, MongoDB, EJS (Template Engine for server-side rendering) and deploying the application

• Build this app from scratch and use it in your real life; best by adding some additional advanced features to this base project.

• The main goal is to master your NodeJs and MongoDB skills and begin your full stack journey by developing this project.

• After building this app, your goal should be implementing private chat, image upload in the post section, and like, comment feature to make it more practical.

1.2 TECHNOLOGY USED

We can divide the project based on the stack used:

- HTML, CSS ,React JS Building the UI of the application
- Serving HTML dynamically and use of EJS (template engine)
- Familiarising the NodeJS environment
- ExpressJS: Framework for creating servers.
- MongoDB: Using NoSQL Database.
- Socket.IO: Building live chatting feature
- GitHub: To publish your project.
- Heroku: Deploy the full stack application
- ReactJs : Building components
- Axios : api calling integration

KEY FEATURES IN THIS APPLICATION

- **REACT JS**

  React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications

  1) **Declarative**

     React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug.

  2) **Component-Based**

     Build encapsulated components that manage their own state, then compose them to make complex UIsSince component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

  3) **Learn Once, Write Anywhere**

     We don’t make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

     React can also render on the server using Node and power mobile apps using React Native.

- **AXIOS**

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XML Http Requests.

Features

- Make XMLHttpRequests from the browser
- Make http requests from node.js
- Supports the Promise API
- Intercept request and response
- Transform request and response data
- Cancel requests
- Automatic transforms for JSON data
- Client side support for protecting against XSRF

- **CSS**

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, development of various parts of CSS specification was done synchronously, which allowed versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, CSS3. However, CSS4 has never become an official version.

- CSS is the language we use to style an HTML document.
- CSS describes how HTML elements should be displayed.

- **FIREBASE**

  Firebase has several features that make this platform essential. These features include unlimited reporting, cloud messaging, authentication and hosting, etc.

  And more features are.

  - Unlimited Reporting
  - Audience Segmentation
  - Integration with Other Services
  - Cloud Messaging
  - Authentication
  - Test Lab
  - Hosting
  - Remote Configuration
  - Dynamic Links

CHAPTER 2

Literature Survey/Project Design

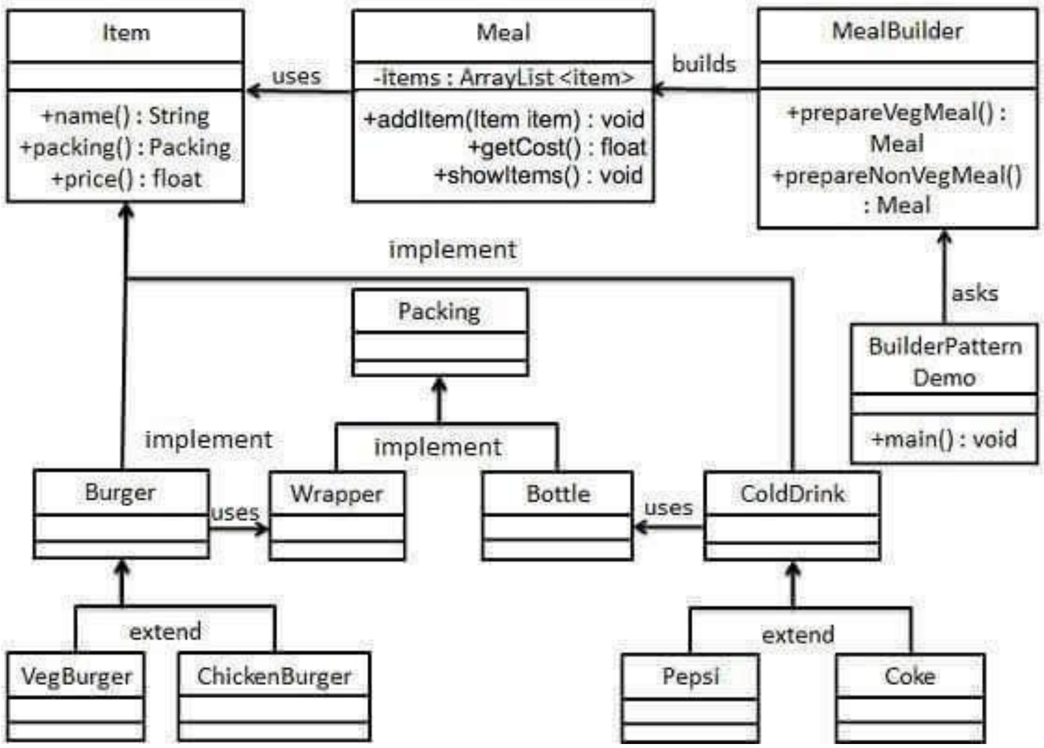
LITERATURE SURVEY

This section describes about the existing research on the Node.js as a server side language and areas where Node.js is implemented practically, along with having some general review about the same like History of Node, and some application areas of Node.js and is it efficient or not.The most important thing with a web server is its ability to handle multiple users efficiently. This has a lot to do with the programming language used to write its script. Hence performance of server-side scripting languages like PHP, Python were taken into consideration with comparison with Node.js . P. S. Bangare et al has proposed the novel secure encryption mechanism which is a combination of chaotic logistic mapping and RC4 stream cipher. Node.js is an excellent tool if you want some kind of live interaction, realtime results. It is capable of very quickly delivering data to/from a web server. Traditionally, there has always been a big problem with computers where the CPU can only do one thing at a time. It was solved long ago with multi-threading, allowing us to have multiple 'threads' on a single CPU. It switches between them all the time, and while it's pretty fast, the switching has a ton of overhead. To avoid this overhead node.js solves this problem by running in a single, event-driven thread. Rather than have a new thread get created on each request, there is one thread for every single request. When a new one comes in, it fires an event that runs some code. When you make a call to a database, for example, rather than block until it's returned, you just run a call-back function after the call is complete. Any number of call-backs can respond to any event, but only one call back function will ever be executing at any time. Everything else your program might do—like waiting for data from a file or an incoming HTTP request—is handled by Node, in parallel, behind the scenes. Your application code will never be executed at the same time as the most important thing with a web server is its ability to handle multiple users efficiently. This has a lot to do with the programming language used to write its script.

PROPOSED SOLUTION

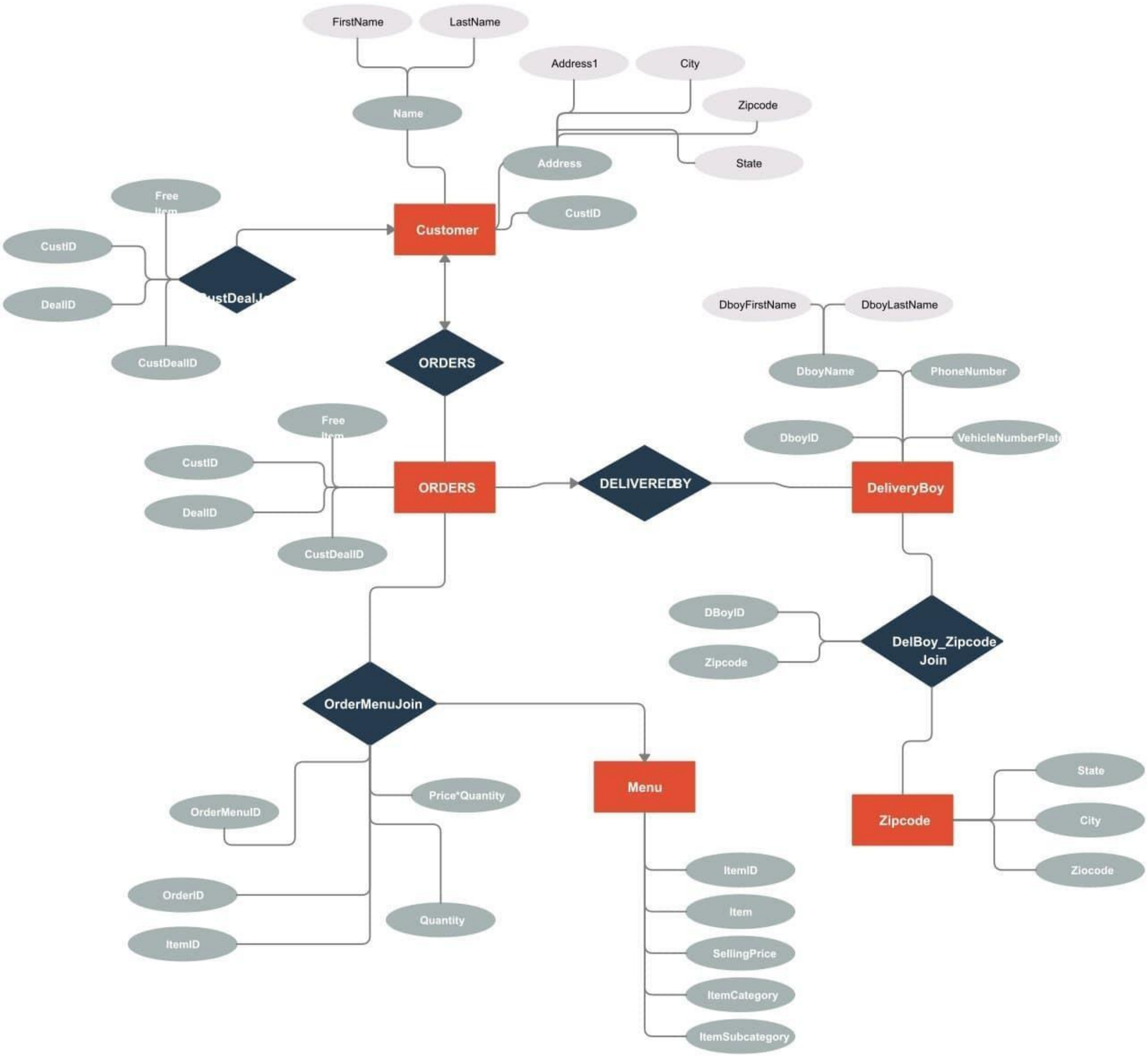
1) UML DIAGRAM

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system



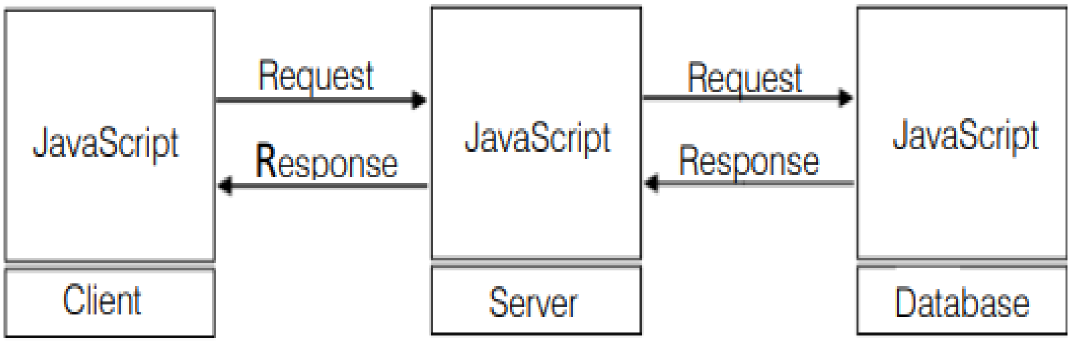
2) DATA FLOW DIAGRAM

It describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.



ADVANTAGES OF NODE.JS

Node.js is built from ground for the purpose of handling asynchronous I/O as it is built of JavaScript and JavaScript is built as event loop. Like the on click event for a button in client side JavaScript is and event loop. While other environments do have this feature, they have it with using third party libraries or are not built from ground for the same purpose like the Node.js and hence they are often slow, or lags and does not belongs as a standard feature to them. Similarly an edge of Node.js over others will be that it will be capable of handling multiple request while it will act like a client towards the third party services by executing only a single thread. Other languages in this regard will block the processing until the remote server responds first for their initial request as a result they will be requiring multiple threading for executions. Comparatively in Node, all what you will use is asynchronous as it will become quite hard if you are to write non-asynchronous code in it. Also Node.js will never force to buffer data before outputting while the others like Event Machine.



Why BurgerBuilder ?

As we can see in various fast food companies like Burger King , Mcdonalds have fixed Burger menu on their services and this problem doesn’t satisfy the need of the today’s customer because sometimes the customer want to have a custom burger according to their liking and disliking. So to overcome this problem we came with an idea for building this application called burgerbuilder where customer can easily customize their burgers and can order it. And one time I started using a food delivery app because I wanted to eat some really good food. Unfortunately, I ended up heading out for lunch instead, because the food that I wanted to eat was available at my choice and it really hates me a lot because the customer satisfaction is not fulfilled in this case.

Assumptions

- Adding some Cards/discounts/offers a coupon on UI helps to attract user attention.
- Good UI is the key for amazon food for a new start as their competitor doing great when it comes to UI.
- To make a brand stand out, The nutrition tracker feature is important as no one has this feature.
- Ease of payment is also a great feature that boosts the amazon food selling ratio.

BurgerBuilder effective tool:

A bespoke ‘Burger Builder’ website provides tools to allow people to design their own version with an option to share the burger online to gain public votes.

The 12 most popular burgers will be assessed by a judging panel consisting of England rugby star and 'Celebrity Masterchef' winner Phil Vickery, McDonald’s staff, an "independent expert" and a member of the public.The judging panel will choose their five favourite burgers, which will appear in all McDonald’s branches over a five week period starting in October.

Alistair Macrow, senior vice-president and chief marketing officer at McDonald’s UK, said: "Customisation and digital engagement are becoming an integral part of how consumers interact with companies and we want to continue to innovate as a brand."McDonald’s crowd-sourcing initiative comes the week after it set up its first Twitter page in the UK with a tweet promising "news, promotions and fun stuff".

The 80 different ingredients mean there are more than one million possible combinations with ingredients including guacamole, chorizo and pineapple chunks,.

The ‘Burger Builder’ tool was created by Razorfish and traffic will be driven to the site via a YouTube masthead takeover, pre-roll ads and digital display ads.

BUILDER PATTERN

It falls under the creational pattern.

This pattern is handy when construction of an object is complex (Creating object includes, creating instance + assigning values to some members) and you want to separate the construction logic from the representation so that it can be used with various representations.

The Factory pattern is more of an answer to: "What is being built?", whereas this pattern is more of an answer to: "How to built it?".

Problem Statement

You all may have heard of McDonald's Happy Meals. Let's create a code for creating a ChickenMcGrill HappyMeal.

Problems and various solutions

Problem 1

The End client has to be aware of the steps for building the final product.

A new Happy meal, say for a McVeggie, results in a substantial amount of cluttered code

(because now ChickenMcGrill will be replaced with McVeggie and Toy with HarryPotterToy).

Solution 1

Change the constructor logic of the Happy Meal as:

Problem 3

We can see for every happy meal, the construction process is the same except some attributes are affected and so those representation steps need to be separated from the construction logic.

Solution 3 Builder Pattern

The Builder pattern separates out the representation and creation from each other with the help of Builders and Directors.

Components involved in the Builder pattern are

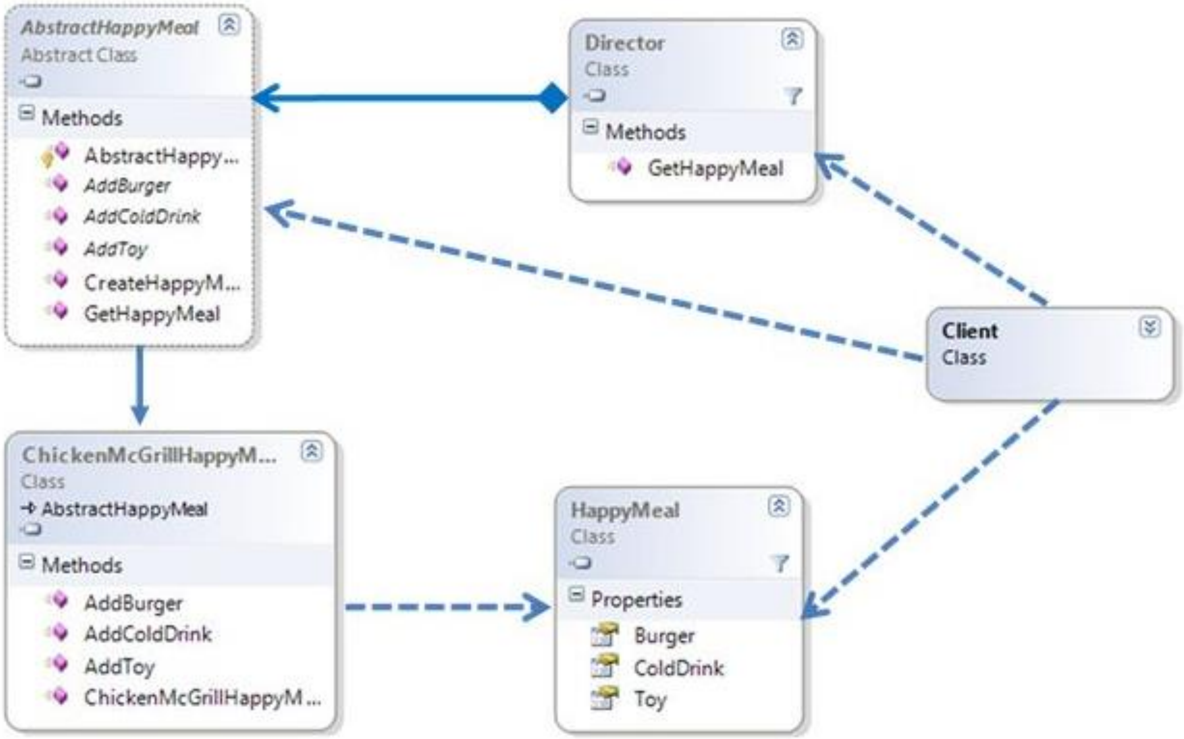
AbstractBuilder - Contains the steps required for creating final concrete object.

Builder - Constructs the Individual part of the Concrete Product implementing AbstractBuilder.

Director - Construct the complete Concrete Product using Builder.

Product - A Complex Object which is required to create.

CLASS DIAGRAM

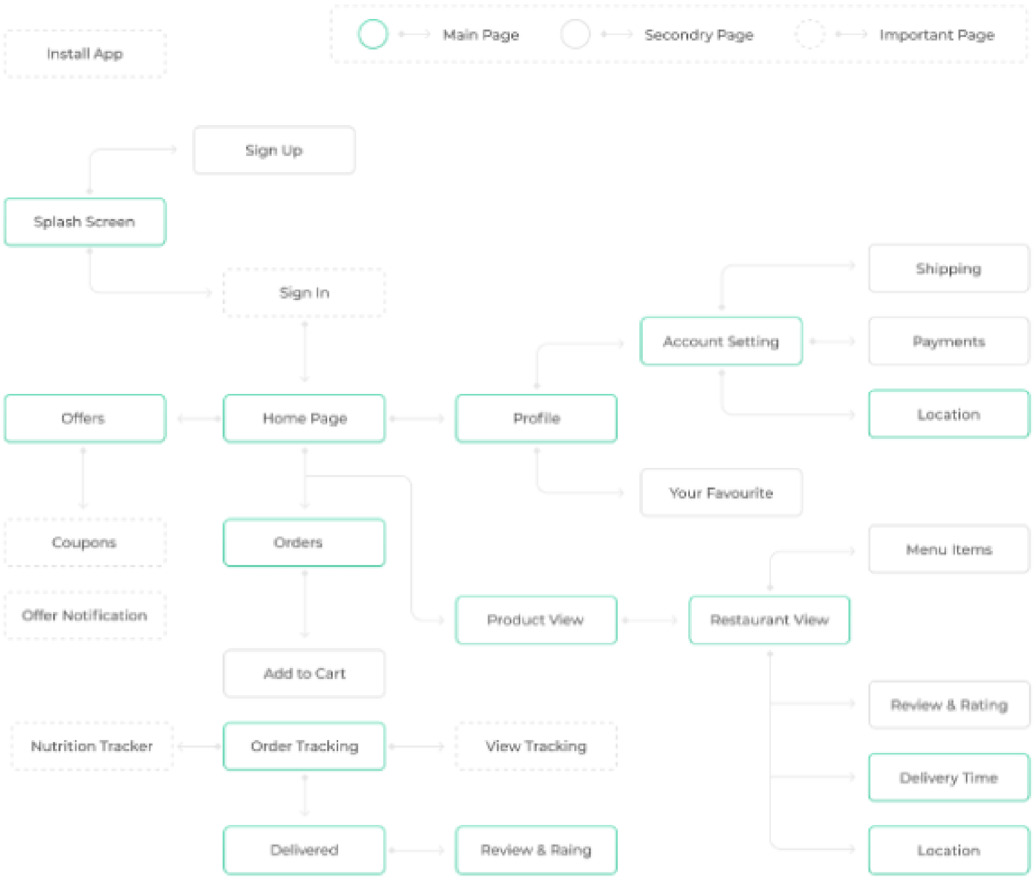


Patterns are made for solving the business problems. If any pattern does not solve the problem then it would be better you revamp it as per your requirements.

Even in some scenario, 2 or more patterns are used together.

Please download both of the attached source codes for a complete demonstration of the pattern.
The second sample is all about implementing the builder pattern in ASP.Net and you will also find how two patterns (Factory and builder) work together.

USER FLOW


Main Page
Secondry Page
Important Page
Install App
Sign Up
Splash Screen
Sign In
Shipping
Account Setting
Payments
Offers
Home Page
Profile
Location
Your Favourite
Coupons
Orders
Menu Items
Offer Notification
Product View
Restaurant View
Add to Cart
Review & Rating
Nutrition Tracker
Order Tracking
View Tracking
Delivery Time
Delivered
Review & Raing
Location

Process

To design a food ordering app, it’s vital to think over the variety of steps and clear navigation that will enable users to quickly make and get the order under diverse circumstancesIts extended functionality allows users to order a traditional burger from the menu or customize any option for themselves adding or removing the ingredients.

They feature all the flow of choice and customization of a burger as well as the screens for delivery or picking up an order.

Burger Card: Variety of Choices

Choose the Burger

The menu shows actual positions and special offers that the restaurant or service has at the moment. Users may see item photos and basic data on ingredients and weight. Color accents highlight price and calls-to-action for quick scanning. What’s more, the system of filters at the top of the screen enables a user to customize the search and find a needed position faster.

Having chosen a specific position and moving to the product screen, users see the big product photo, core information about pricing and weight and the CTA button enabling to add the position to the cart. It makes the visual presentation emotional and catchy, immediately sets the association with the burger. Also, the screen looks clean – it isn’t overloaded with details about ingredients which isn’t interesting for users who buy this position regularly or don’t care about the details and want to make an order asap.

Customize the Burger

Those who do care about the ingredients or want to customize their burger, use the tab “Ingredients” in the bottom part of the screen. Just pulling it up, they open the tab and check the contents of the burger, organized along categories such as Vegetables, Meat, Sauce, Topping, etc. To make the visual performance of the list effective and appealing, the interface features photos of all ingredients. At any stage of interaction with this screen, users may save the item to favorites just tapping the heart icon in the top right corner. The cart icon features one more important UX affordance: a yellow dot on it gives a quick prompt that it isn’t empty.

As for the color palette in the app, the designer played with the contrast of backgrounds: interactions zones aimed at reading copy, observing and manipulating positions in the lists are presented on the light background to provide a high level of readability. Still, photo content and tab bar apply a dark background that supports the visual performance, makes the graphics look stylish and elegant. Also, the designer paid deep attention to building balanced and scannable visual hierarchy to make interactions quick and screen scanning easy. Hungry people are definitely not the audience which will want to devote much time and effort to learn how the app works – everything has to be clear in short seconds.

The flow of interactions on customizing the order looks like the following animation.

Create a Burger

A typical food ordering app for restaurants and cafes usually features the meals of the fixed formulation. Tasty Burger app pushes the limits: it suggests users not only customize the existing offers by adding or removing ingredients but also create their own burger from the ingredients in stock. Adding ingredients, users can see how the price is changing relatively.

Here’s the animated flow of interactions on creating a burger in the app.

Order and delivery.Having decided on the order, a user is offered two ways to get is: delivered at the particular address or picked it up from the restaurant. Also, a variety of payment methods is presented.

If a user chooses to pick the order, the map helps to choose the best location and shows the route. Color contrast makes the screen clear while carefully selected fonts provide a high level of readability.

Here's the interaction flow for the food app design regarding the order and delivery solutions.

Slight and unobtrusive animation applied to transitions and microinteractions made the food ordering app design lively and delicious.

Landing Page

Landing page for the presented app is based on the attractive and catchy animation: fresh tasty burger creates a mouthwatering effect to immediately set the theme and emotional appeal.

Tasty Burger app design process was an exciting creative challenge for the Tubik design team: it was a cool attempt to broaden the horizons of food delivery applications with extended functionality, strong usability, and finger-licking appearance.

How we brought BurgerBuilder Design System to Mobile

Like many of today’s leading brands, from Apple and J.Crew to AirBnB and Warby Parker, Burger King has embraced a design-led approach that looks at visual design as one of the best ways to tell their story and communicate who they are and what they’re about. And they aren’t leaving anything to chance. Every interaction with Burger King is infused with their custom design language—everything from the website, TV ads, and in-store menus, to (you guessed it) their mobile application. As Raphael Abreu, Head of Design at Restaurant Brands International (parent company of Burger King) put it recently:

Design is one of the most essential tools we have for communicating who we are and what we value, and it plays a vital role in creating desire for our food and maximizing guests’ experience.

The way that Burger King and other design-led brands create this consistency is through the use of design systems. Design systems offer a set of visual standards and specifications that help companies establish a unique brand identity that is consistently experienced across all channels—both physical and virtual. In many ways, the design system is the brand.

For web developers building frontend experiences across mobile, desktop, and web, design systems have become part of the common vocabulary. Although a design system can mean many things - from pure design specifications to actual, working UI components - it often boils down to a shared UI library that can be used and reused across any web project.

Now, this is easy enough to do with the web, with its vast array of CSS customizations and tailor-made UI components. But what about mobile? As any UX, UI, or frontend dev team will quickly discover, it turns out that bringing their design system to mobile is not so simple.

So how does Burger King do it? Before I answer that, let’s run down the options for customizing your UI in a native mobile app, based on which platform or SDK you’re building with, and then dive into how they make it happen.

Native iOS and Android

Android and iOS offer native software development kits (SDKs) that provide a rich library of frontend UI components, animations, and gestures. The catch is, they’re not your UI components. A big limitation of the native SDKs is that your ability to customize the UI components - and the overall experience - is limited to whatever Apple and Google are willing to support. You have to use their pre-built library and customization options. If you have a highly custom design language, with very specific fonts, colors, or design patterns, you might not be able to replicate those patterns for your mobile applications. Want your button component to look and behave just so? Better hope the native UI toolkits support it. Otherwise, you’re out of luck.And if you have an existing web-based UI library that you’d like to repurpose, you definitely won’t be able to use these components in your native app if you’re building with the Android and iOS SDKs exclusively. This means you’ll be rebuilding everything from scratch; which effectively negates the value of a shared library of reusable components—at least as it concerns mobile.

React Native, Flutter, and Xamarin: Limitations to Native UI Toolkits
So what about cross-platform solutions like React Native, Flutter, or Xamarin?

React Native and Xamarin both use the same native UI toolkits. Although you write your app logic using JavaScript or C#, you evoke the native UI components during runtime. React Native and Xamarin will often tout this deep native integration as a benefit, but these options leave you in the same boat as if you were building with the native UI toolkits. Any customizations you want to make are at the whim of Apple and Google.

Flutter is somewhat unique, because they’ve built their own UI library and graphical rendering engine—but the same limitations apply. You won’t be able to bring in your existing design system or web-based UI library to Flutter. Instead, you’ll have to recreate each component, and all customizations must be supported by Flutter. In the end, teams who choose to build with the native toolkits or popular cross-platform solutions (like React Native, Flutter, or Xamarin) will find themselves recreating their design system for mobile, often without fully matching the UX and design specifications that their corporate brand and design systems would prefer to use.

Capacitor: The Cross-Platform Solution for Design Systems

Solving this challenge was one of our goals when we built Capacitor and Stencil—two open source projects that, when combined, provide a powerful solution for teams who want to bring their existing UI library to mobile.

First, Capacitor is a cross-platform native runtime that provides fully native mobile experiences that can be deployed natively on iOS, Android, and desktop platforms. You can also run Capacitor apps on a traditional web browser or a mobile-ready Progressive Web App, all from the same codebase.

The best thing about Capacitor? The UI layer is completely web-based, using the same open web standards, libraries, and frameworks that power your existing design system.

Let bring thing in practice

Let’s say you have an existing design system based on React UI components. If you build your mobile experience with Capacitor, you can use your existing UI library even when deploying a native mobile app. Of course, to create the best experience, you’ll need to account for mobile styling, navigation, and platform-specific UI guidelines specific to iOS and Android, but fundamentally your UI library will run fine on Capacitor right out of the box.

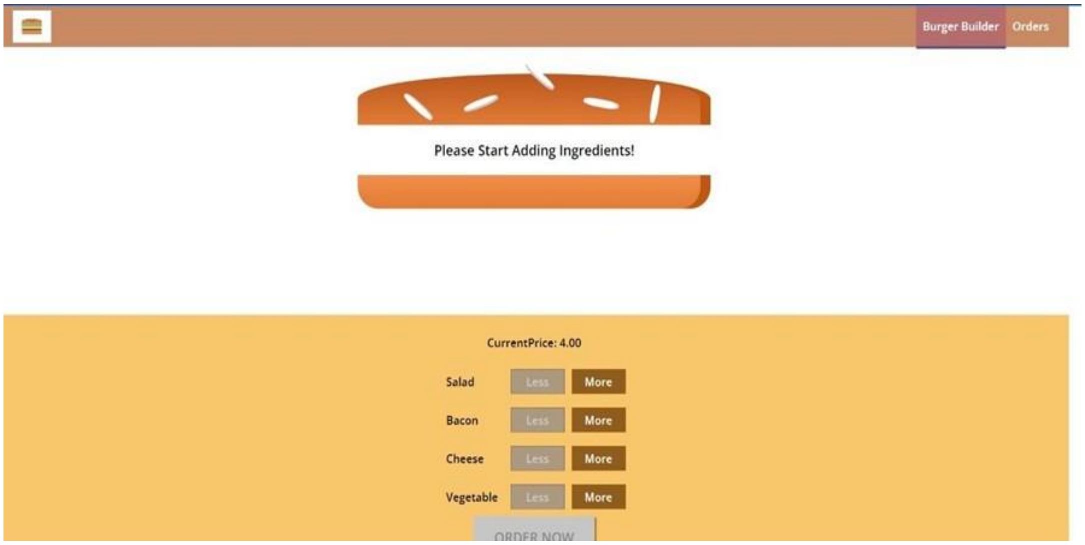
Burger King’s Approach

In fact, that’s exactly what Burger King did. The team at Burger King chose Capacitor because it allowed them to bring their existing UI library to mobile in a really unique way. If you check out their iOS and Android mobile app experiences, you’ll note that, while they look and feel totally native to the device, you’re not using the stock iOS or Android components. Their design team wants to make sure that the way you experience the BK menu, for example, is consistent whether you’re on their website, inside the restaurant, or on their mobile app. Capacitor allows them to do just that, by persisting their web-based UI components across web and mobile.

Chapter 3

Functionality / Working of the Project

BASIC UI OF THE APPLICATION



Please Start Adding Ingredients!

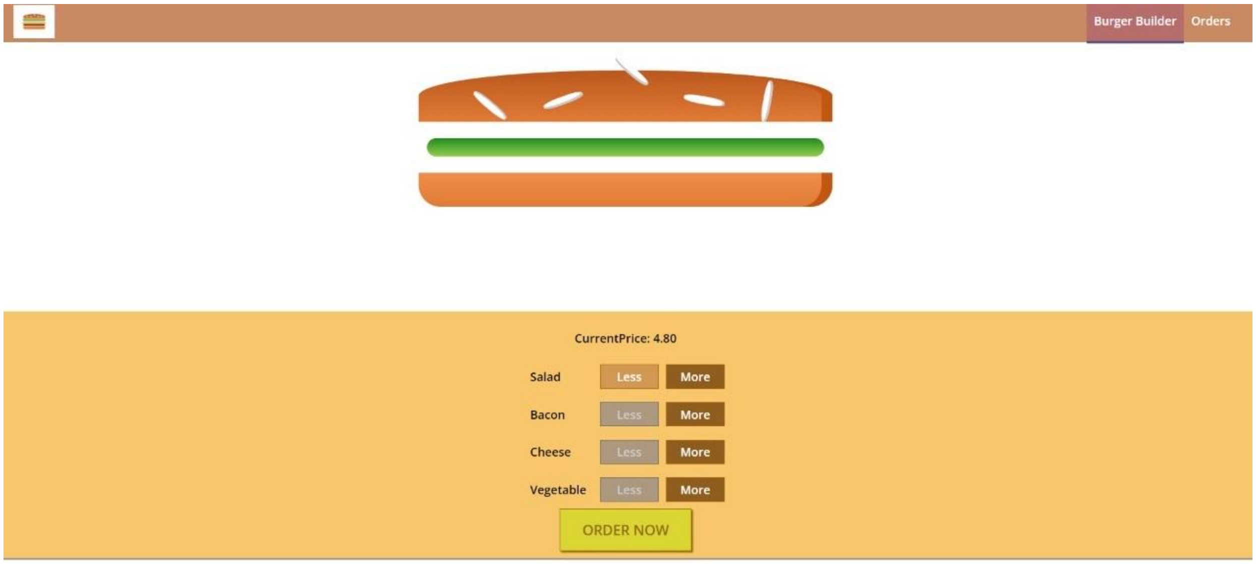
CurrentPrice: 4.00

DESCRIPTION

As you can see on the top-left corner there is a burger icon and in top-right side there are two routers BurgerBuilder and Orders in which on clicking BurgerBuilder we navigate to the main page of application and on clicking orders we can navigate to the list of ordered burgers.

And the main part comes in center where we can actually build our custom burger and in which top most and lower most bread are static and ingredients to be added inside it are dynamic

UI WITH FUNCTIONAL COMPONENTS

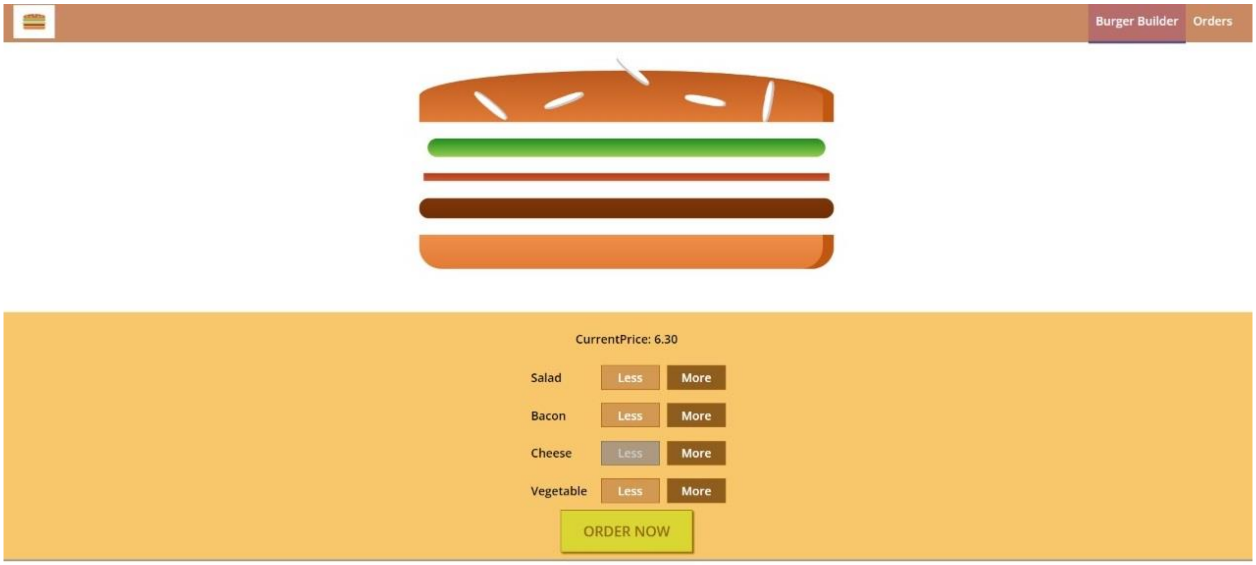


DESCRIPTION

Initiallty in our web application only user can see the ui parts of the application but now it has the features that user can interact with the web page . Like you can see in UI parts when we click on the button like less or more components get added between bread .

This is possible through add event listeners and events that is used in jsx elements

BUTTON FUNCTIONALITY



DESCRIPTION

In this section button plays an role in this web application to manage the features embedded in this application .

These buttons are not only components but it has the featues of providing interaction to the user.

Similarly we have order summary where we can see the orders details .

APPROACH

1. Technology choice

1.1. Mobile App

We knew from day one that The Burger Collective wanted to launch both iOS and Android apps at the same time. We have considered few options - native apps, React Native or Ionic. Early stage startups are always limited by funding so we needed to deliver the best possible experience with minimal costs. Another consideration was rapid development and on the fly deployments which can bypass lengthy app store reviews.

We have considered Ionic for a while. Mostly because of getting the web app as well as Android/iOS mobile app almost for free. But from our experience Ionic apps fall behind native apps in animation speed and in displaying rich content properly. For this reason it wasn't appropriate for a user facing app.

Building both apps natively was too costly and it wouldn't give us the advantage of rapid development and deployment as Javascript based apps would. So the winner was React Native. We had delivered few React Native apps in the past and were confident that it will check all the boxes for what The React Native framework is now mature enough to deliver great apps with native-like experience and the tool set around it allows better-than-native development with hot reloading and extremely fast releases which ensure all new features are rolled out immediately to all users.

1.2. Restaurant POS App

Restaurants run their separate app which can read a user QR code and associate orders to their accounts. It's an important piece of the puzzle in TBC business as users can redeem their vouchers, get member’s discount and collect points for purchasing and reviewing burgers. Although the purpose of the restaurant app is quite different than consumer facing app and the app is distributed via private enterprise channel, we decided to go with React Native for the same reasons as with the user app.

POS App

1.3. Back-end and admin UI

We had to deliver 3 important bits on the back-end:
API which will handle requests from mobile and POS app - many API calls were geolocation heavy, we needed to make sure API will scale with growing user base. Benchmark was set to support up to 5000 concurrent users.
Admin UI - TBC team needed to execute various operation tasks without developers, such as approving reviews and new restaurants, removing inappropriate content etc.Admin section for restaurants where they can update their details, working hours and menu.
First important step was choosing a database. After initial data modelling we found that data doesn't need to be too heavily relational and we didn't need to support atomic transactions on multiple tables. Also, apart from basic data queries we needed a database which worked really well with geolocation queries. Hence, we decided to use MongoDB as our database. Another reason was that with MongoDB we could use excellent admin UI / CMS keystone.js which saved TBC a lot of time and money compared to custom solution we would have to build. Keystone.js is built on top of express.js which is de-facto standard for api development in Node.js; another

problem solved. For a restaurant admin section we decided to build a small custom react.js app as this required more restrictive approach and custom UI elements to make restaurant details updates as easy as possible.

2. Application Hosting, deployment, monitoring

Startups in early stages usually can’t afford on-premise servers and staff to service them. And it would be foolish to even think about this choice when you don’t know how big your initial user load will be and how fast you will grow. So here the choice was quite obvious to select a cloud provider which will cope with flexible startup needs.

We chose AWS as we have the best experience with their services, especially Elastic Beanstalk. It allows us to streamline deployment process and with a little setup, it connects well with CloudWatch where we can monitor application load and get alerts if anything goes wrong. Elastic Beanstalk main advantage comes when you need auto-scaling of application servers. With just 2 boxes to set, you get fully working load balanced fleet of instances which will grow with user load. You need to set up to how many instances you want to spin and most importantly, when you want to do so. EB gives you multiple options, the most common is to set a threshold, when CPU utilization reaches some specific value. We recommend to load test every app with most common usage scenario to find out where this threshold is. This can significantly improve autoscaling efficiency and lower costs.

3. Hot releases with CodePush

We live in times where change is an essential part of our lives. This puts a huge pressure on development teams, mainly to make development cycles shorter and to be more responsive to incessant changes in requirements. We were able to accommodate many new changes in application UX and to introduce hotfixes in matter of hours or days based on our member’s feedback. We also managed to push out new versions immediately. This flexibility wouldn’t be possible without employing a CodePush service which allows to deploy React Native or Cordova applications instantly without going through Apple App store or Google Play store approval process. This approach helped us to connect with our members and quickly react on how they use the app.

4. Analytics

4.1. MixPanel

Every decision made during design & development process needs to be based on hard data and insight about in-app user behaviour. Otherwise, you are just guessing and this might lead to decreased potential ROI and suboptimal use of time & resources on development. In our case, TBC team decided to employ MixPanel service to fully understand user’s journey through the app. The MixPanel provided TBC team with necessary information to make the right design decisions, test user behaviour with A/B testing, engage with users through segmented push notifications and provide analytical data for our executive & marketing teams.

From developer’s perspective, the MixPanel implementation requires adding event trackers into various places in client's apps, depending on what they want to monitor. The MixPanel React Native wrapper is available on GitHub which makes integration with the app rather easy.

4.2. AppsFlyer

AppsFlyer service is a mobile attribution & marketing platform which helps marketers to understand effectiveness of marketing campaigns. TBC marketers were able to see ROI for different campaigns (a number of new app installations as a result of a specific campaign - e.g. on Facebook) and adjust their decisions from what they learned through AppsFlyer.

As with MixPanel, React Native wrapper component for AppsFlyer is available on GitHub.

5. Testing

5.1. Server side-testing

When it comes to testing the server-side code, we prefer the maturity of Mocha framework together with Chai assertion library. In NodeVision, we use BitBucket as version control system which comes with so called, Pipelines. You can achieve Continuous Delivery with this tool quickly and without cumbersome configuration. Every commit into BitBucket triggers automated tests, which provide immediate feedback about potential issues in your code base.

5.2. Client side-testing

We adopted more traditional user testing with a prepared set of user testing scenarios. Before every release, our testing team carries out a full regression test on reference devices to ensure full functionality of the app. We are fully aware that this situation is not optimal as we can cover only limited number of devices (this is problematic mainly for Android ecosystem). Moving forward, our plan is to switch to more automated testing with AWS Device Farm. This would allow us to cover numerous phones from different manufactures and to automate test with test scripts.

6. Load testing the app

As mentioned above we needed to ensure the app can handle ~5000 concurrent users. We also needed to test that auto-scaling will spin new instances at the right time. We used https://artillery.io/ framework to load test production servers. With a chosen scenario of the most likely app user journey, we let Artillery to add more and more users and we then monitored the output. We managed to get up to 6 t2.large instances, handling all load with very fast 36ms average request latency.

Load testing

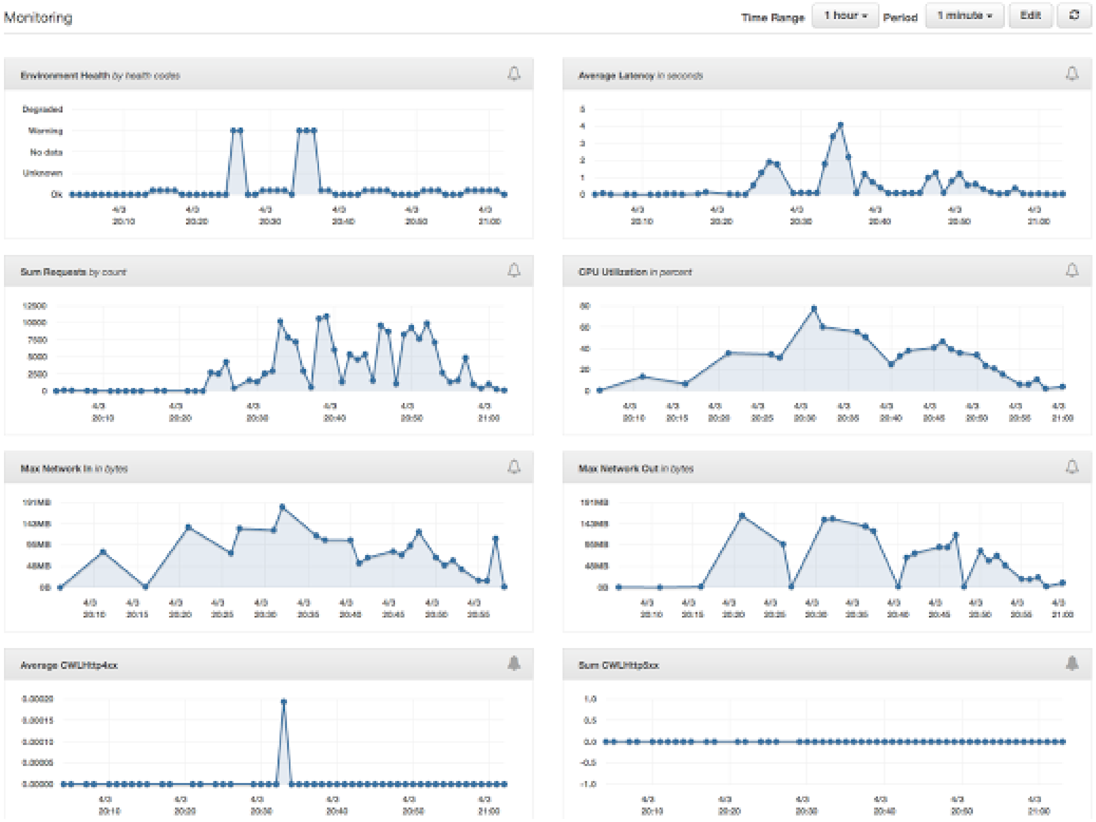
7. Crash Reporting & Logging & Monitoring

7.1. Client Side

There are many products and tools for app monitoring and your choice might be affected by multiple factors such as, cost, supported platform, range of analytics, details about crash events, etc.

In NodeVision, our number one choice is Crashlytics which is a part of a bigger platform - Fabric. It gives you every possible information for any crash which has occurred in your app. You can extend crash reporting with your custom metadata which could help you to duplicate an issue in your dev environment (e.g. user id, screen, user action etc.).

Another great advantage of Crashlytics over the other tools is that it’s free, even for enterprise sized project. A React Native wrapper for Crashlytics is available as an open source project on GitHub which allows to push costs of implementation down.



7.2. Server Side

7.2.1. Code execution and health monitoring - Loggly

For server side logging and request monitoring, we used Loggly services. It helped us to detect any anomalies in our server code execution and to monitor response times. It provides detailed error reporting and allows to monitor key resources and metrics. Basically, Loggly gives you everything your team needs to quickly react if something goes wrong or even before that.

7.2.2. AWS

The AWS comes with extensive resource monitoring which could be extended by custom metrics and alarms. This allowed us to keep a close eye on server resources and proactively react during high workload periods. Generally, the AWS tools don’t provide a deeper insight into potential code issues. This requires different tools which are integrated into your code base (e.g. Loggly).

As mentioned before, the AWS metrics are an essential part of auto-scaling management. It allows to optimize a number of server instances based on the current load. We were able to configure AWS EB instance in a way that any excessive workload automatically increased a number of server instances. E.g when our marketing team ran a promo campaign which resulted in increased number of requests.

CHAPTER 4

RESULTS AND DISCUSSION

The study got the finding about the implementation of Node.js. Below is discussed the implementations positive findings as a result of literature review and the survey.

- The Node.js have made Full Stack Developers' job a dream come true. In absence of Node.js it was hard for a developer to learn several different languages and environments to manage the complete system at server side and client side.
- Organizations and developers can now with the invent of Node.js build highly load bearable and faster applications and by using Single Page Applications (SPA) now the server calls are reduced and the applications are more user friendly and faster.
- Node.js made it easy to achieve high load operations like graphic processing and Internet GIS very quicker, and it can be reliably used in every field where the files sizes are high or the network bandwidth is highly consumed. Node.js will make such operations faster and with less need of bandwidth.
- Community like its feature that the same language is also being used at server side while JavaScript is always been used at client side for ages

CHAPTER 5

Conclusion and Future Scope

Node.js have some challenges in context of its use in the community as well as its adoption by the developers and organizations over the existing programming technologies. No doubt that Node.js have great benefits, it have also some challenges to the community. One such challenge is the ability of misuse of the widely used language by developers. One enthusiast have made a backdoor software using Node.js on Raspberry Zero. It can create backdoors in the target computer and their network even if the computer is password locked. Although there are solutions from such backdoors but some seems impractical like totally blocking the USB ports, and closing the web browser every time the user leaves the computer. And other options are not implemented by majority and mostly might not be aware of it like using secure layer on ones websites (https), and enabling secure flags on the cookies which common users might not know about it.

- There is a plus point but as understood from the survey conducted that the community feel it hard to learn JavaScript for Node.js
- Also the developers having knowledge of other programming languages have complications in adopting Node.js. Even the setting up of server for their programming work is not an obstacle. This is as concluded from the survey results.
- Another plus points were event-Driven Programming, Non-Blocking I/O, and asynchronous feature. But according the survey results is that the features like event-driven programming, NonBlocking I/O, and Asynchronous processing is a hindrance.

REFERENCES

1. "The benefits of web-based applications," [Online]. Available: http://www.magicwebsolutions.co.uk/blog /thebenefits-of-web-basedapplications.htm. [Accessed 25 November 2016].

2. Web Application Basics, Pearson Higher Education.

3. F. Bridge, "What Types of Developers Are There?," tree house, 24 June 2016. [Online]. Available: http://blog.teamtreehouse.com/what-types-of developerare-there. [Accessed 25 November 2016].

4. M. Wales, "Front-End vs Back-End vs Full Stack Developers," Udacity, 08 December 2014. [Online]. Available:http://blog.udacity.com/2014/12/front-end vsback-end-vs-full-stack-web-developers.html. [Accessed 25 November 2016].

5. J. Long, "I Don't Speak Your Language: Frontend vs. Backend," tree house, 25 September 2012. [Online]. Available: http:// blog.teamtreehouse.com/ i-dont-speakyour-language-frontend-vs-backend. [Accessed 25 November 2016].

6. A. Mardan, "PHP vs. Node.js," Programming Weblog, [Online]. Available: http://webapplog.com/ php-vs-node-js/. [Accessed 28 January 2016].

7. J. Kaplan-Moss, "Quora," [Online]. Available: https://www.quora.com/What-are-the-benefitsof developing-in-Node-js-versus-Python. [Accessed 29 June 2016].

8. "Node.js Tutorial," tutorials point, [Online]. Available: https://www.tutorialspoint.com/nodejs/index.htm. [Accessed 25 November 2016].
9. N. Chhetri, "A Comparative Analysis of Node.js (ServerSide JavaScript)," Culminating Projects in Computer Science and Information Technology., p. 5, 2016.

10. R. R. McCune, "Node.js Paradigms and Bench marks," 2011.

11. Node.js, "Home page of Node.js," Joyent, 2016. [Online].Available:https://nodejs.org/en/. [Accessed 01 May 2016].

12. G. Developers, "Chrome V8 | Google Developers," Google, [Online]. Available: https:// developers. google.com/v8/. [Accessed 27 May 2016].

13. eventmachine, "GitHub, Inc," [Online]. Available: https://github.com/eventmachine/eventmachine.[Ac -cessed 30 June 2016].

14. Twisted Matrix Labs, "Twisted Matrix Labs," [Online]. Available: http://twistedmatrix.com/trac/. [Accessed 30 June 2016].

15. The Apache Software Foundation, "Apache MINA," [Online]. Available: http://mina.apache.org/. [Acce ssed 30 June 2016].

16. The Apache Software Foundation, "Apache MINA," [Online]. Available: http:// mina.apache.org/ async webproject/index.html. [Accessed 30 June 2016].

17. Ryan Dahl: Original Node.js presentation. [Film]. Youtube, 2012.

18. "How Loading Time Effects Your Bottom Line," Kissmetrics Blog, [Online]. Available: https://blog. kissmetrics.com/loading-time/. [Accessed 25 November 2016].

19. C. Buckler, "Site Point Smack Down: PHP vs Node.js," Site Point, [Online]. Available: http://www. sitepoint.com/sitepoint-smackdown-php-vs-node js/. [Accessed 28 January 2016].