LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Developing pilot API application using NODE JS: a case study of 1UP media Oy

Bao, Vu Dao Thien

2016 Leppävaara

Laurea University of Applied Sciences
Leppävaara

Developing pilot API application using NODE JS: a case study of 1UP media Oy

Bao, Vu Dao Thien
Degree Programme in BIT
Bachelor's Thesis
September, 2016

Laurea-ammattikorkeakoulu
Leppävaara
Business Information Technology

Tiivistelmä

Bao, Vu Dao Thien

Developing pilot API application using NODE JS: a case study of 1UP media Oy

Vuosi 2016 Sivumäärä 49

Asiakaskäyttäytymisen ymmärtäminen on ensisijaisen tärkeää liiketoiminnan kannalta. Keräämällä ja analysoimalla dataa asiakkaiden käyttäytymisestä yrityksillä on mahdollisuus saada selkeämpi käsitys siitä kuinka asiakkaat reagoivat heidän tuotteisiinsa tai palveluihinsa.

1UP Media OY kehittää korkealaatuista ja edullista pilvipalvelua, jonka avulla yritykset voivat analysoida asiakaskäyttäytymistä ja sitä kautta parantaa palveluaan.

Tämä opinnäytetyö keskittyy pääasiassa pilvipalvelun kehittämisen kuvaamiseen. Kehittämistyö perustuu waterfall-kehitysprosessiin. Waterfall-kehitysprosessi on yksinkertainen ja suoraviivainen kehitysmalli, joka sopii tämän projektin tavoitteisiin. Tämän lisäksi Waterfall-mallin seuraaminen varmistaa, että projektin lopussa pilvipalvelu on hiottu ensiluokkaiseksi.

Tämä raportti kuvaa pilvipalvelun osana olevan API:n kehittämisen. Raportti sisältää lyhyen esittelyn projektissa käytetyistä teknologioista, Waterfall-kehitysprosessista ja käytetyistä kehitystyökaluista. Tämän lisäksi itse kehitystyön edistyminen on dokumentoitu waterfall-kehitysprosessia seuraten.

Opinnäytetyön lopputuotteena on pilot-versio API:sta joka täyttää asiakkaan asettamat vaatimukset. Vaatimukset keskittyvät API:n skaalautuvuuteen, tietoturvallisuuteen sekä huollettavuuteen. Testauksen perusteella voidaan todeta, että API toimii odotusten mukaisesti. Seuraavat askeleet API:n kehittämisessä ovat kapasiteetin lisääminen ja tietoturvallisuuden parantaminen.

Avainsanat: Ohjelmistokehitys, pilotti, API, waterfall-kehitysprosessi

Laurea University of Applied Sciences
Leppävaara
Business Information Technology

Abstract

Bao, Vu Dao Thien

Developing pilot API application using NODE JS: a case study of 1UP media Oy

Year 2016 Pages 49

Understanding the customer behaviour plays an important role in business. By collecting data and analysing customer behaviour, the business has a clearer awareness of the customer response to their products and services. The objective of this research work was to develop a new API application with a carefully planned schedule. The thesis was commissioned by 1UP media Oy, which is developing a high quality and affordable online system that would provide solutions for collecting and analysing reliable customer behaviour data that could be used in improving the user businesses. The thesis report briefly introduces relevant technologies, such as Node JS and Express JS framework, development methodology and development tools such as Brackets, POSTman and Azure App services, which are used in this project, and also explains the actual development process following Waterfall methodology.

In this thesis, the Waterfall development framework is used as the thesis mainly focuses on the practical development of API application. The Waterfall development framework is a classical model in software development. It is widely used by government projects and software companies. It is a process that focuses on early planning stages to avoid design flaws before development stage starts. Waterfall framework is simple and fits development schedule of the project. It also gives better visuals about development progress to project manager so that the quality of final product is ensured.

At the end of thesis, pilot version of API application is completed. It fulfils the requirements given by the company at the beginning of the project. The requirements mainly focus on scalability, maintainability and security approaches of the product. After live testing, the application has proven that it's modules work precisely and meet the expectations of 1Up media. Some recommendations also suggest that in future development, the application must be tested and improved to handle high scale data through regions and that more security strategies need to be focused on protecting the API application and its assets.

Keywords: development, pilot application, API, Waterfall development framework

Table of contents

1 Introduction 7

1.1 Company background 7

1.2 Project background 7

1.3 Objectives 8

1.4 Project scopes and limitations 8

2 Methodology 9

2.1 Software Development Life Cycle (SDLC) 9

2.2 Waterfall 10

3 Project plan 12

4 Requirements 14

5 Product Architecture Design 15

5.1.1 API structure explanation 17

5.1.2 API Call Cycle 18

5.1.3 Structuring API folder 21

6 Development 23

6.1 Technologies 23

6.1.1 Server-side Web API 23

6.1.2 Representational State Transfer (REST) API 24

6.1.3 JavaScript programming language 24

6.1.4 JavaScript best practices 24

6.1.5 Scalability 25

6.1.6 Maintainability 26

6.1.7 Software security assurance (SSA) 27

6.1.8 Node JS 27

6.1.9 Express JS framework 29

6.1.10Azure 31

6.1.11Azure App Service 32

6.2 Development tools 32

6.2.1 Jira software 32

6.2.2 Brackets Text Editor 33

6.2.3 POSTman 33

6.2.4 Git and Gitlab 34

6.2.5 Mocha Unit testing library 35

6.2.6 JSDoc 35

6.3 Development process 35

6.3.1 General setup 35

6.3.2 Creating Authentication and User management components 37

6.3.3 Creating Data collection and Device management components 38

6.3.4 Unit testing 39
6.3.5 Documenting the module 40
6.3.6 Deployment 41
7 Live Testing 41
8 Issues 43
9 Conclusion 43
References 45
Illustrations 47
Figures 48
Tables 49

1 Introduction

Understanding the customer behaviour plays an important role in business. By collecting data and analysing customer behaviour, the business has a clearer awareness about the customer response to their products and services. However, the similar products currently available on the market fail to focus on small and medium enterprises.

1UP media Oy focuses to create a high quality and affordable online system that would provide solutions for collecting and analysing reliable customer behaviour data that could be used in improving the user businesses.

1.1 Company background

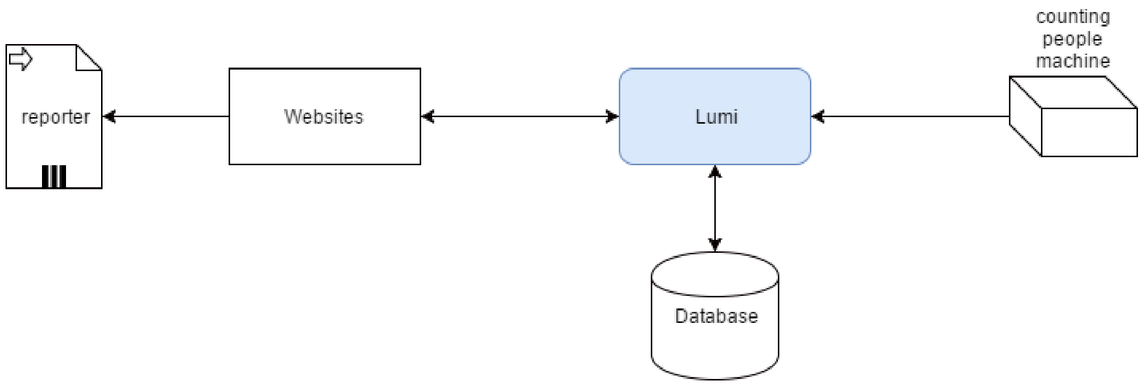
1Up Media Oy is a Finnish, start-up company located in Leppävaara, Espoo. 1Up Media Oy focuses on operating and provides software solutions in Business to Business area. The company has a very young and fresh environment. Company’s main goal is to provide high quality software to help businesses improve their services and reduce operation and production costs.

1.2 Project background

Background of this project is to create pilot version of Lumi. At the moment, Lumi has developed its core parts as mentioned earlier in Section 1, Introduction part. Within Lumi pilot version, we have developed Lumi people counting service and web application.

Lumi is cloud based platform that helps customers collect, analyse data using various services. In general, there are four main parts inside Lumi. First part is front-end side, or in other words, the part where real users or devices will access and interact with Lumi. Second part is Application Programming Interface (API) application where it handles all business logic, routing management and database connection control. Third Part is storage side, where all assets and data of entire Lumi platform are stored. Fourth part is other services to collect data.

Lumi people counting service aims to collect customer flow at a certain location. It contains physical machines to understand how many people are going in and out, at what time and location. The data is stored in Lumi platform and is analysed using other Lumi services as the final outcome.



Caption 1: Lumi platform

1.3 Objectives

The objective in this project is to provide stable pilot version of Lumi. Using the pilot, the company is able to introduce Lumi to customers and give customers better vision of how the system works.

- For customers, the pilot is very essential to make them understand how Lumi works and how it would benefit their businesses.

- For the company, by using the pilot, the company is able to introduce Lumi to customers and give them better vision of how the system works.

API application is an important part of Lumi platform. The objective of this project is to build stable API application to control the flow of data inside Lumi.

1.4 Project scopes and limitations

This thesis paper focuses on developing and structuring API from perspective of backend software. It only focuses on core business logic and explains how components in backend side are connected following industrial best practices. The final project goal within the scope of thesis paper is to build prototype version of Lumi platform. Some of the critical topics that cannot be described in detail are:

- Mass scaling solutions
- User feedbacks
- Effect in sales

The company limits the options of choosing database and development environment technology in this project.

2 Methodology

This section explains about methodologies used in the project. This section describes the Software Development Life Cycle (SDLC) as the foundation of development methodology and Waterfall Development model as the framework to develop the software.

2.1 Software Development Life Cycle (SDLC)

SDLC, or in other words, Software Development Process, is a process of producing a complete high quality software that meets customer expectations within estimated resources. SDLC consists of planning and analysing requirements, defining requirements, designing product architecture, building, testing and finally deploying the product.

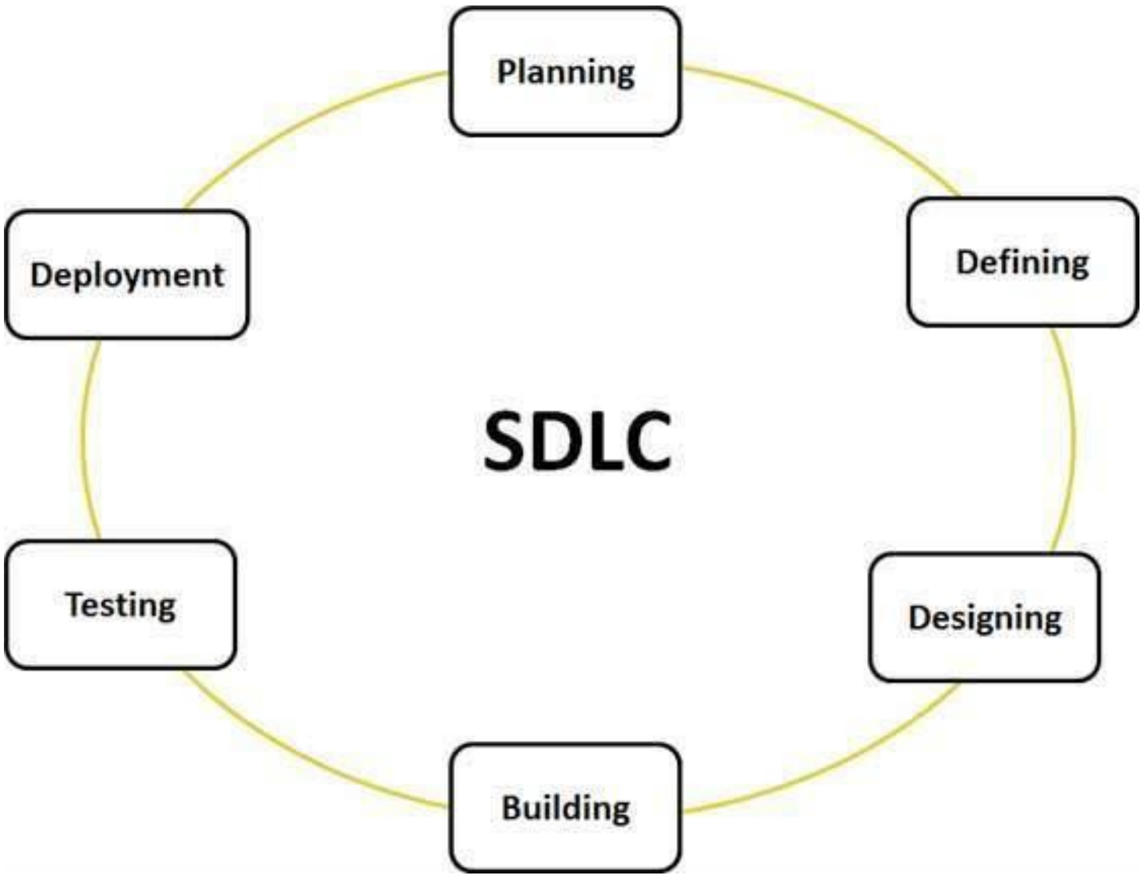


Figure 1: SDLC cycle (Tutorial points)

- Planning and requirement analysis: It is the most important step in SDLC. This is the stage where customer's and the sales team's requirements are collected and analysed. The input is used as the resource to plan foundation for project. Risk identification in the project and quality assurance requirements are also covered in this stage.

- Defining requirements: This is the stage where all requirements are collected and defined clearly. All of the product requirements are then transferred into proper documents which are called Software Requirement Specification(SRS) documents and are approved by customers or market analysts.
- Designing product architecture: Based on the SRS document, product architectures is created with detailed architecture to develop. After this stage, an architecture document which is called Design Document Specification is created. It consists of clear definition of all the architecture modules of the product. It also includes the flow of data with external modules and third party modules.
- Developing the Product: During this stage, the programming code is created according to DDS documents. Coders or developers must follow instructions given by the organization including which tools should be used and which programming language is required in the project.
- Testing the product: This is the stage when the software is tested carefully using certain scenarios to find bugs to fix them. Then it will be tested again. The process will repeat until the software reaches quality standards defined in SRS.
- Deployment and maintenance: At this stage, the organization will release the software according to the company policy and strategy. Based on the feedback, the organization will decide to officially release the product or make modifications to fit target market.

SDLC is the foundation for various software development models. Waterfall Software Development model is one of the most popular models that follow SDLC. (Tutorial points)

2.2 Waterfall

Software development is mostly guided by Agile or Waterfall development methodologies. Development methodology is a framework that describes the development process to engineering team to build a given product. (Tutorial points)

Waterfall model is a traditional, classical model in software development. It is widely used by government projects and software companies. It is a process that focuses on early planning stages to avoid design flaws before development stage starts.

Waterfall model process starts with collecting the requirements including system requirements and software requirements. The next step is building architectural design and detailed design. Then the project continues with coding, testing and maintenance. In detail, the section below explains how each step of Waterfall model works:

- System requirements: Collecting components needed for building the system. It includes hardware, software and essential equipment.
- Software requirement: List of expectations for software functionality. It also includes the study about software interactions with other parts of the system, performance and user interface requirements.
- Analysing requirements: The stage where all requirements are collected and analysed to find proper solution to fulfil the requirements.
- Architectural design: The stage of determining software framework to meet the system and software requirements. It defines major components and how they interact with each other or with other parts of the system without explaining in detail the structure of each component.
- Detailed design: Explaining briefly how each component works.
- Coding: The stage of implementing detailed design into real software product.
- Testing: Checking whether the implementation part meets the requirements and listing errors in the code.
- Maintenance: Addressing problems and handling expanding requests after the release of software.

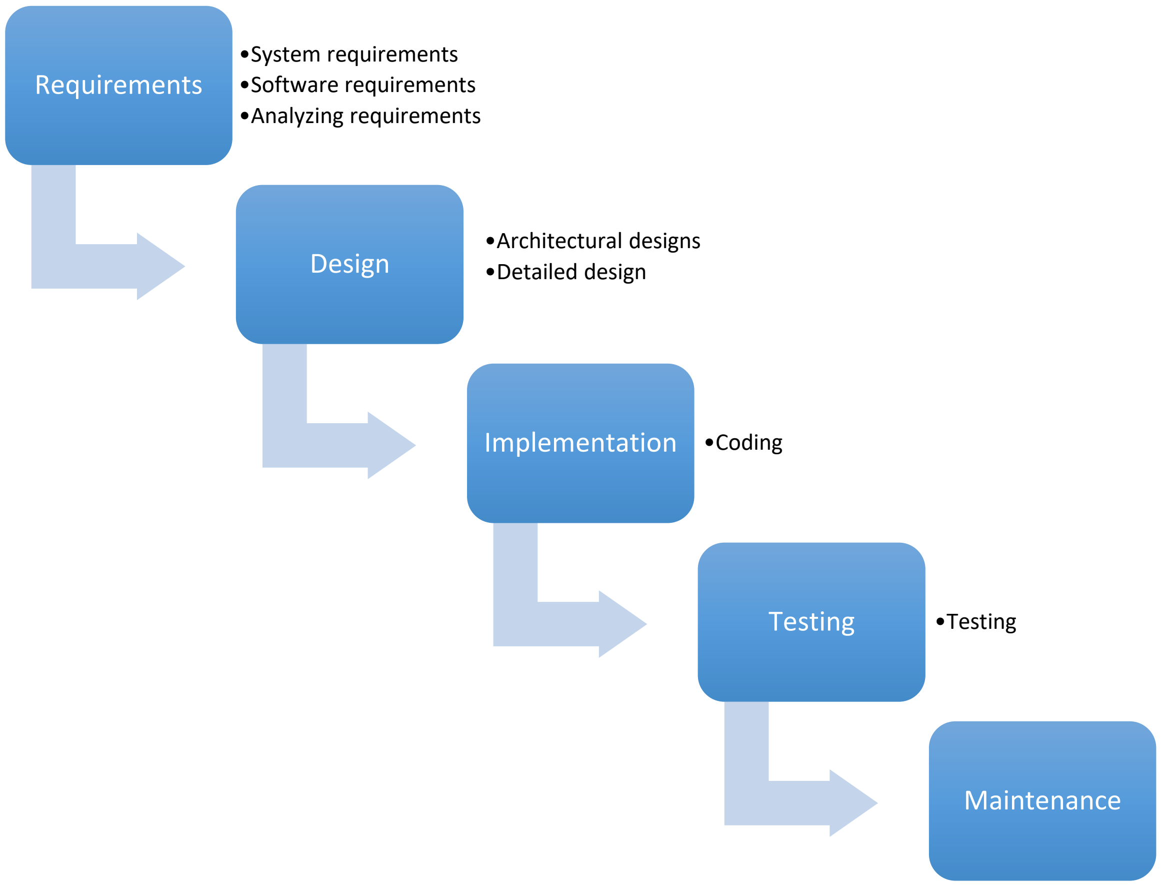


Figure 2: Waterfall model cycle (Tutorial points)

There are multiple advantages in Waterfall model. It is easy to understand and implement the concept. It is frequently used by organisations and companies. Therefore, theoretically, it is a mature framework that we can depend on in this project.

However, there are certain disadvantages of Waterfall that worth taking into consideration. One of the most disadvantage points of the methodology is that traditional waterfall model is not flexible since each step must be completed from the beginning till the end before moving to the next one and fail to be visited again (Westfall, 2008)

3 Project plan

Planning is a useful part in project. It helps both parties understand the project properly by having better documents about development process, proper documentation and tracking task, milestones and deadlines.

From the beginning of development process, after gathering customer's requirement, we analysed them carefully, arrange by priority which requirements need to be done first, when it will be completed. This process started from January to February 2016 with the contribution of customers and development teams. The final outcome of this step is well explained documentation about tasks for each team member, deadlines. This approach ensures the maintainability of the application.

From February to March 2016, we have to make clear architecture designs for the API application. It consists of all components, internal or external, in the application, how it works, with detail explanation about each component. When making architecture design, we must design it in the way that the API application is well modulized and find the best solution to protect the application by explain in detail what security approach we use for every component. The outcome of this stage is the proper architecture design documents.

Coding is the phase when we have to code functionality of components according to architecture design documents. Each developer will be assigned a certain amount of tasks and we have to fulfil requirements written in requirement analysis documents. We have to use chosen development tools and use proper name spacing, avoid dependencies among components as well. This stage started from April and the deadline was in June, 2016.

After finishing coding stage, we moved to making unit testing for each component. There was multiple level of testing, however, according to customer's requirements, we would use unit testing tools to develop unit testing scripts to ensure the stability of each module. It also helps developer check if the application fulfils the requirement of customer. During the development of business logic, we use POSTman to make quick test in order to judge the response of the API. This early testing method is good as developers do not have to write a lot of testing scripts that slow down the development process. However, it is not reliable because when the application goes bigger, when all modules has connections to each other, unit testing is more reliable as it will ensure that the API functions remains the same. All of the bugs will be reported to JIRA and fixing software issues would occur in the stage as well. The deadline for this stage is till June, 2016.

In deployment stage, we would deploy the application to the whole system to make sure if the application can work perfectly with other parts of the system. We would use Git to deploy the application to Microsoft Azure App services. This stage will be done in July, 2016.

Starting from July 2016 to end of August 2016, we would install a pilot testing machine in customer's place to test Lumi platform in general and the API application in particular to adjust the software in real case and fix potential faults during testing period.

4 Requirements

As a part of Waterfall development model, gathering requirements is an essential step to collect, analyse requirements needed to build the product. The requirements are collected by having a discussion with the company. A research was made using Internet sources to analyse and understand the requirements. There are multiple sources to collect requirements such as the system, software ones.

From customer requirement, we have to make the API so that it can connect to other parts of the software and well documented for further development in the future. Customer also requires us to use development and CRM tools that are listed in Knowledge base section.

As system requirement, the company uses Node JS as the environment for the API application and find the framework so that it must provide license as open source. the application must be able to in integrate with Microsoft Azure App services.

In software requirement, the application has to follow JavaScript application best practices mentioned in JavaScript best practices as one of requirements from company. According to that, the application has to fulfil scalability, maintainability and security requirements.

For scalability requirement, the application has to have these characteristics:

- Proper name spacing.
- Proper modules for every feature of the application.
- Decrease dependencies among modules

Maintainability of the application is as well a part of requirement. To fulfil that, we have to:

- Well documentation that explains clearly the application feature and structures.
- Proper unit testing modules to ensure that all features are well tested stay remained when some other components changes.
- Version control must be applied to keep track the development of the application. In case of damage, it can be reversed to previous stable version.

For security requirements, we have to apply practices to protect the application in particular and whole system in general:

- API must be well protected by multiple protection layers, stop the request sending from unknown sources.

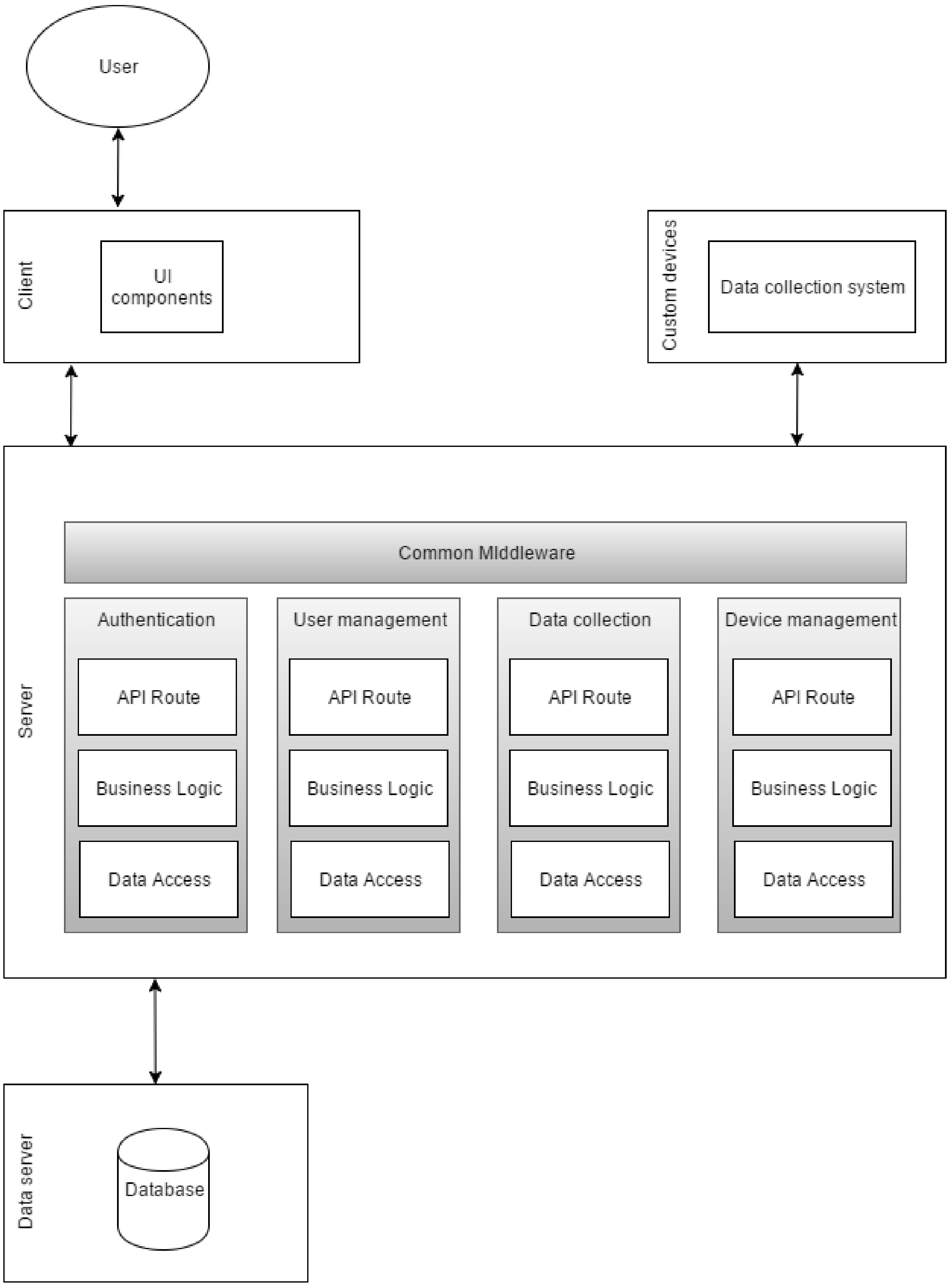
- Sensitive Data, for example, user information is essential to protect and we must use some encryption tools to make sure customer data is fully secured in our system.

5 Product Architecture Design

This section describes briefly how Lumi platform by describing:

- How functional elements are connected
- API structural explanation
- Cycle of calling API
- Structuring API folder.

The diagram bellows explains connections between parts. API plays a role as the centre of Lumi platform by controlling data flow, managing devices information and user authentications.



Caption 2: Lumi Architecture design

From client part, particularly Web Application of Lumi, the interaction between API application Web Application is handled through HTTP calls. For example, when web application requires user information, it will send HTTP request to API to get necessary data according to their request command. Developers who are responsible for the web application has to create business logics to handle response from data. In most cases, if the request is structured properly which has all requirements, the response will contain data with success status. Otherwise, API application will response error status with message which explains in detail issues of the request.

From Lumi people counting machine, data is transferred as well through HTTP call to API. The data sent to the API will be handled by logic in server and give response to the Lumi people counting machine. By this way, the machine gets the status whether the data is handled by the API or not. Developers who are responsible for the machine has to create error handler function to react to the status sending from API.

5.1.1 API structure explanation

Inside Back-end side, we divide into multiple parts called Component. Each component serves specific feature of the project. By looking at Figure x, we can see that inside every feature, it contains API route where API gateway is specifically open for that feature and is controlled inside that feature. It is the way so that Client-side can connect and interact with back-end side using HTTP requests. To table below is an example that explain how API works for User management

Feature	Request Type	Parameter	Description
Users Operations	GET	/user/{id}	Get information of specific user using ID
	POST	/user/new	Create new user by posting information of new user
	PUT	/user/{id}	Update information of existing user
	DELETE	/user/{id}	Delete user from database

Table 1: API call examples

Other components apply the same method; first part of API declare which components that API belongs to, then other parameter is structured accordingly to the requirement of every call. Any API starts with “/user” parameter is controlled in User operation component, same as “/data_collection”, “/device_management” … We also put some logic as Middleware to check the valid of every request particularly before passing request to executing phase to protect our server from any

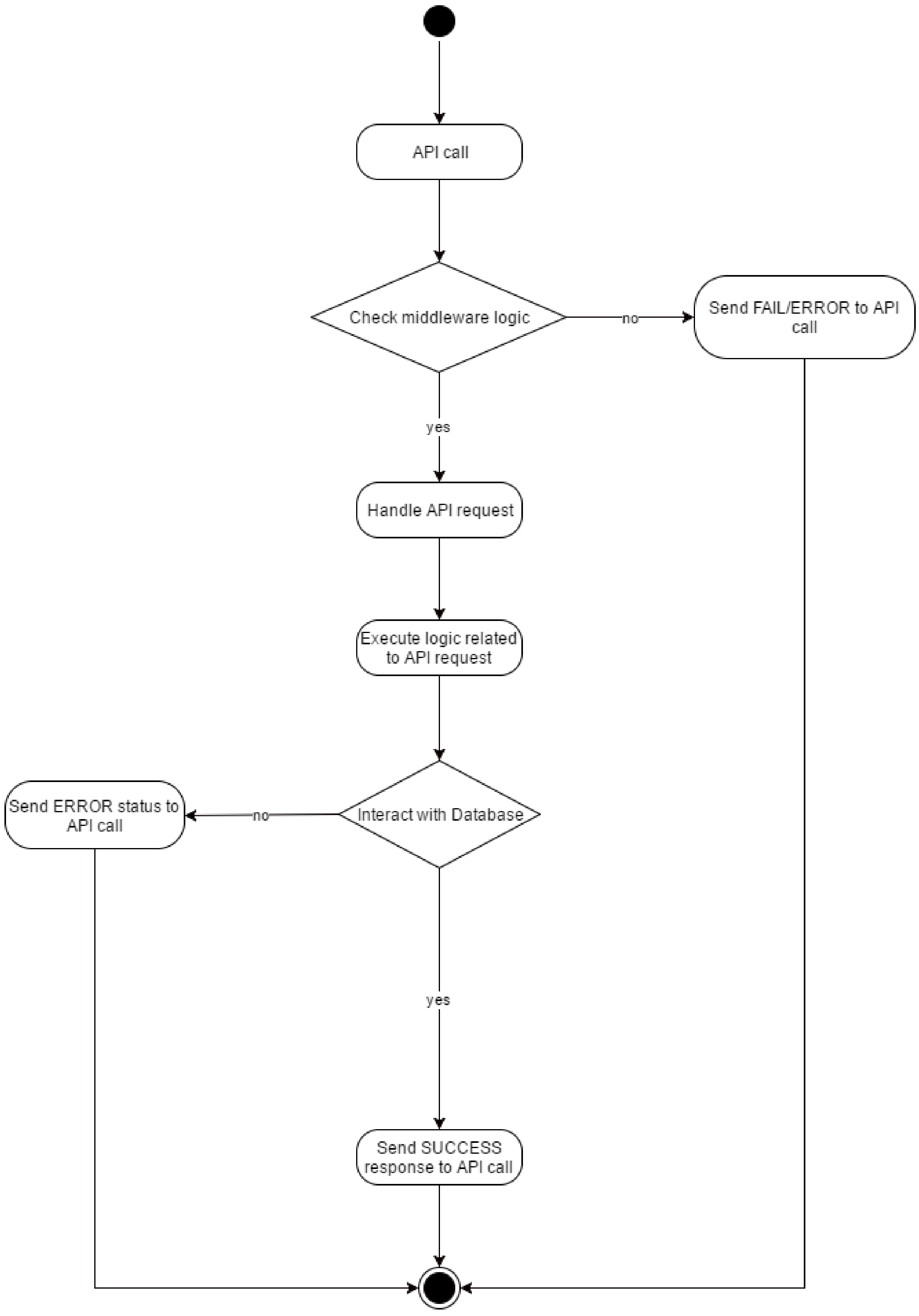
We also put Data access logic to every component. The reason why we apply that practice is explained in section 4.7, Structuring API project. From API request protocol, we handle that request separately, create business logic for every call, create access to Database and response to the request.

In Data management of every component, we create business logic related to Database activities such as create new, update, replace or delete objects in Database. We use Microsoft Azure Table storage SDK for Node JS to build business logic. We also create some custom functions to adapt SDK to project.

Data-related business logic is called in execution phase of API process. Every business logic is structured separately and tested carefully before adapting to the system.

5.1.2 API Call Cycle

The diagram explains how the process of handling basic request of API. It is typically different from request call. However, this core process is still applied. API requests are called from Client side of Lumi, Lumi Web Portal or from Lumi people counting devices.


API call
Check middleware logic
no
Send FAIL/ERROR to API call
yes
Handle API request
Execute logic related to API request
Interact with Database
no
Send ERROR status to API call
yes
Send SUCCESS response to API call

Caption 3: API request call cycle

From the beginning of the process, When the call is made, our Back-end system starts to carry by sending that request to Middleware part. Middleware session plays a great number of roles at the same time. One of the most common task is to check where that request comes from, is it valid by checking User authentication token. Another role of middleware is to prevent any harms caused by outsider who try to attack our system or by some mistake that possibly happen because of bad request or false data format from our client side. If the call fulfills all strict requirements from the middleware, the request will be transferred to API request handler. Otherwise, we will send response the that request with Fail or Error signal and the process ends at that point. In that case, developers need to make another call and the loop starts again until it passes the middleware.

When the request is valid, next step is to take care of the process by analysing requirement of that Request. In many case, a request can consist of multiple requirements and tasks for Back-end side. We will separate the requirement and execute business logic accordingly. API handler can be seen as the task manager for API, business logic is not stored here, it simply separate requirement and import functions from other place we call “service factory”.

When execute business logic, the interaction with Database happens at the same time. First of all, we have to check whether the data which is sent from API Request valid or not. If it is valid, we will execute Data communication logics to put new Data into Database. Then we listen to the response from Database whether it is success or not. In this step, any error response neither from Database nor from our business logic will lead to breaking the API call. We will send Error status as the Response to the Request.

After executing business logic, we will prepare to response to the Request. At this step, the Response varies depending on requirements of the Request. Normally, when API call type is “GET”, it means that the Request would like to receive some data or information. In that case, we will return the Request with Success status together with Data packages. Another case is when API call type is “POST”, “PUT”, “DELETE”, normally, we will response with single status to announce that the Request is handled and executed.

By following this process, we are able to assure that every API gateway is controlled, observed and well structured

5.1.3 Structuring API folder

Originally, Node JS project is usually setup based on roles of files. For example, all controller files will be grouped into “controllers” folder, all database service files will be saved in folder called “Database”.

However, there are many discussions about some critical disadvantages when applying this format. It is very inconvenient for developers during development period since he or she has to access multiple folders to modify single features as well as understand the flow of functions. Moreover, to import functions into files, there will be format such as require(“../../../feature.controller.js”). Finally, testing based on features will be harder since developer has to write multiple scripts to test single features. This approach was applied in other projects by customer develop team and caused so many troubles when it comes to scaling the project later.(Nemeth)

Instead, after planning with customers, we decided that we would structure project API based on features. By that structure, developers find it easier to follow functions of every feature. All files will be grouped in that feature folder. When importing files within the feature, it will be limited to *require (“./featureA.<role>”).*

It is very useful for scaling project since when new features come, developers only have to create similar folder structure and can concentrate on building business logic without having to worry about causing conflicts to other existing features.

The figure below will demonstrate how the project is structured

```
.

├── node_modules

├── Components

         ├── Feature

                  ├── feature.services.js

                  ├── feature.SQL.js

                  ├── feature.API.js

                  ├── feature.specs.js

├── Utility
```

```
          ├── Utility1
                    ├── Utility1.services.js
├── server.js
├── test
          |- feature.test.js
├── package.json
├── routing.js
├── middleware.js
├── config.js
```

“Node_modules” folder is the auto generated folder which contains all of “node package modules” which are developed by community and used in the project.

“Package.json” is considered as the config page of the project which was created automatically by node JS when the project started. It contains Name of the projects, version and author, licenses… and list of dependencies with name and versions where source codes are saved in “node_modules” folder.

“Server.js” is considered as main file in the project. When running node server, we set “server.js” as the first file to run. It declares which port the service is running on Azure table storage, opening API port for all of the services.

“Routing.js” is the file where all of the API port must go through. This file is built on Express framework. All features which need to open API port, must declare to the software in this file to activate the route.

“Middleware.js” stores general middleware business logic of entire API. Middleware plays may roles such as firewall, checking user permission, dealing with Cross-origin HTTP Request (CORS) to allow or deny any access to the API.

“Config.js” is the place where all global and specific variables are used in the project. Those variables control how the project works as well as the environment which is running it.

“Components” folder is where all of significant features are saved. Every feature has its own files which contains all of services that features need. For example, within a specific feature we called “Feature A” as it requires to have specific Restful API for that, developer will create the file called “featureA.API.js” directly to that “Feature A” folder. All functions related to Feature A’s API will be put there. When developer need to change functions specifically for Feature A, it will be much easier to track and modify. Other example is when connecting to Database, any database-related functions such as querying, updating, add new data related to Feature A will be saved in file called “featureA.SQL.js”. Managing those functions will be easier as all are put in single folder. As for testing scripts, putting it all in same feature helps developers create scenarios to test and assure that features work as expected.

“Utility” folder is the place where common services are stored to be used in other features. Until now, in utility folder, we only have some basic features which are written on top of Azure Table Storage SDK to enhance querying database. Since Utility features are re-used in multiple places, so we temp not to use it quite often as it goes against Lose coupling rule.

6 Development

The following section focuses on explaining the process of researching, planning, designing, coding, testing and deploying the API application. Within the scope of the thesis paper, we are only able to make a part of testing which is unit testing without expanding to integration testing. Deployment part explains how we can deploy the code into internal environment.

6.1 Technologies

Technology section describes concepts and tools which are used in the project. Firstly, JavaScript best practices, Node JS, Express JS framework, Azure and Unit Testing concepts are introduced. Then the next part introduces the development tools that are used in project management and software development process.

6.1.1 Server-side Web API

Hypertext Transfer Protocol(HTTP) is an application protocol to transfer, distribute hypertext between nodes containing text. It is the foundation of data communication in Web Technology. (Nanyang Technology University 2009)

Server-side web API is a programmatic interface that includes public or private endpoints under defined request-response message system. The message or data is popularly formatted as JSON or XML. The message system uses HTTP-based web server. (Profitt 2013)

6.1.2 Representational State Transfer (REST) API

REST is an architectural style of the World Wide Web that includes set of components connectors and data elements in a hypermedia system. RESTful is the extension of REST. RESTful system communicates using HTTP and uses same HTTP vers such as GET, POST, DELETE and PUT to transfer data between client and remote servers. (Rouse no date)

6.1.3 JavaScript programming language

JavaScript is an object-oriented scripting language (OOS). It is used to build cross-platform applications. JavaScript, like other Object Oriented programming (OOP) languages such as C, C# or Java, provides core sets of programming language. Those are:

- Operators
- Control structures
- Statements

JavaScript is mainly used in modern web browsers or in other words, client-side scripting. Recently, thanks to new server side platform called Node.js, JavaScript language is extended to Web server-side as well.

In Web client-side, JavaScript controls and extends HTML attributes and their Document Object Model (DOM). It can respond to events such as mouse clicks and event times among the others or connect to servers using http request calls to interact with servers. One of the most popular client-side JavaScript frameworks is Angular JS. In server-side, JavaScript is widely used on top of Node JS environment in order to interact with databases, design API, etc.

6.1.4 JavaScript best practices

For JavaScript project in general and server-side Node JS project in particular, it is very essential to make the project decouple as in complex projects, modules and components need to be separated at a certain level to maintain the project and reduce harms or any negative effects when some elements changes.

Loose coupling characteristics including:

- Performance
- Functionality
- Scalability

- Usability
- Reusability
- Maintainability
- Sustainability
- Extensibility
- Productivity

6.1.5 Scalability

First of all, the project must be built in a way that it can be scaled in the future as more services are integrated into the product. It consists of three main elements:

- Namespacing: In order to make JavaScript application scalable, using global variables as the method to form blocks and business logic in the project will help developers manage modules easily and avoid difficult maintenance. Namespacing plays an important role in complex project as it helps to reduce the risk of conflicts between modules therefore avoiding problems during the running time of the product.

- Making modules: Modules are groups of functions or services that share alike functionalities. Making modules helps developers concentrate on building specific sections of projects in order to maximize the control of each part. The practice helps handling errors easier. Isolating services using modules also supports reusability of the code. Modules must be built in independently to reduce the level of negative effects to other modules or services when some changes applied during the development.

- Coupling: Coupling in programming is the concept describing the level of interdependencies among modules inside the software project. Theoretically, low coupling is built to ensure a good structure, design and easy maintenance. To be able to build highly scalable Node JS project, loose coupling theory must be applied. A module is considered to be good when it can work independently and can be reused at any time.

By those theories, during the development period, the practice of minimizing coupling is very important. It must be well determined which feature of the project is meant for common use. Moreover, it assures that those modules must be controlled appropriately following elements mentioned earlier. Modules which are already used widely inside the project should not be changed.

6.1.6 Maintainability

This is a very essential part in developing software. Here are the specific requirements:

- Documentation: This is a necessary part of developing software process. Writing documentation is the practice of describing in detail what software does in general and what roles of each modules in particular. It is a very useful tool for developer to communicate and manage source code during development. Writing documentation properly also saves a great deal of time and effort during maintaining period when some problems may occur. At the level of development team, our documentations focus on describing the software from technical perspective called technical documentation. Proper tools are used to automate the documentation into the library for the ease of looking up created services and modules.

- Version control systems: Version control system, frequently known as Source Code Management or Revision Control System. This is a critical requirement for development team. Version control systems consists of multiple software tools that control changes in the source code efficiently. The benefit of version control is that source code will be tracked carefully so that whenever there are some bugs or problems, developers can revert the code back to the most stable stage and continue the development. It helps developers be more confident in developing new codes and avoid crashes or serious damage to the source code. Especially in complex project, version control systems create "file tree" structure that can be seen as cloning project at certain point of process. It ensures that newly developed source codes have similar environment with current project. It plays a role of a simulator where developers can test new functions freely without any harm to the original source code. It also solves one of the biggest problems that software development usually has, fragmentation of project. Since there are lots of source code version or folders which were created and are not synchronized properly. That practice can lead to huge damage to the quality of final product and is hard to fix the issues.

- Unit testing: Testing in general and unit testing in particular is important. Unit Testing brings lots of benefits to the project. One of the most advantages is finding bugs. The quality of the module will be ensured because when problem happens, developer can track and fix easier as the bug is pinpointed in testing software. More than that, it supports future development better for when developer refactors the source code, unit testing will check the accuracy of functions after the changes. Finally, unit testing does help later integration due to the fact that every part of the project has its own unit testing, later testing for the whole application will be much easier because developers have proof that which modules is running properly and does not cause any bugs. (Segue Technologies)

6.1.7 Software security assurance (SSA)

Software security assurance is the process of implementing tools and technologies to protect data, resources of the product. SSA is designed to protect the software from potential harm causing by loss, system inaccuracy and unavailability It is the process of identifying protect object, finding protection means to protect the object. It includes network access or physical access, data management and protection, environment control and human resource security as well.

6.1.8 Node JS

Node JS is an open-source, cross platform, runtime environment that supports developing server-side Application. Basic modules of Node JS are written in JavaScript and custom modules developed from the community is written in the same programming language.

Node JS supports management and development of Web servers and Networking. Node JS modules are designed to integrate API design in order to reduce complexity of Web server-side application. Core modules controls file system I/O, networking, binary data, cryptography and data streams. (Caplan)

Node JS originally is written using ECMAScript 5. It supports other JavaScript alternative that can be compiled to ECMAScript 5 also such as CoffeeScript, Dart, Microsoft TypeScript. ECMAScript 6 is partly integrated and not fully supported at the time when this thesis paper is done.

The difference between Node JS and other Web server-side applications such as PHP is that it is designed to be non-blocking. Commands in Node JS is executed parallel and callback features are used to handle success or error signals.

Since Node JS uses event-driven programming, the application has a possibility to extend without using threading. It solves the problem of concurrency that harm performance of the system.

As Node JS has big community, there are a great deal of open-source libraries avaiable which are provided mainly in npm website, the most popular sources of Node custom modules.

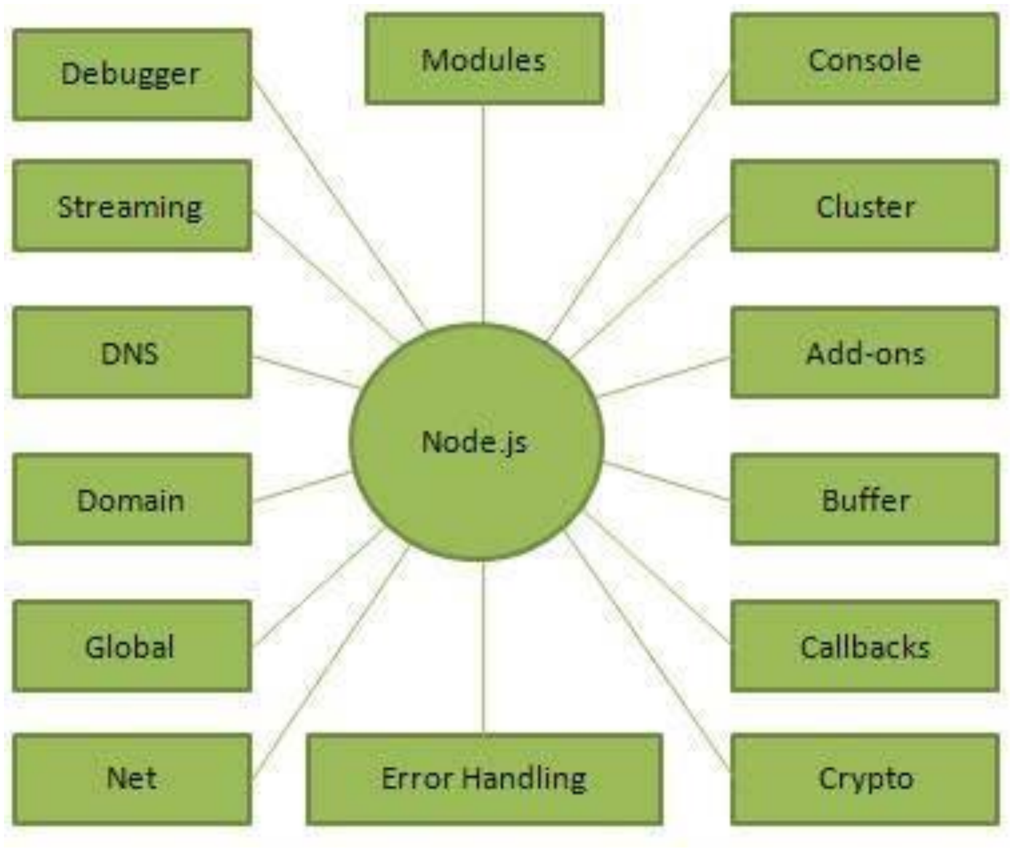


Figure 3: Node JS components (Chitra R 2016)

Node JS is non-blocking, event driven runtime. It is designed to run on a single thread event loop. It can support a great amount of concurrent connections at the same time without having to switch thread context. If there is no connection to Node JS, it will be set to sleeping mode.

However, it still remains some issues with Node JS. Since it is driven in in single-thread event loop, it is not possible to scale Node JS vertically by adding more CPU core to the server. There needs to have additional modules to handle this task. Some popular modules are cluster, pm2 and StrongLoop Process Manager.

Node Package Management (NPM) is the Node JS package manager which is installed together with Node JS. It is used to organize and manage third-party Node JS programs. More than that, NPM manages and installs code dependencies from command line. All of NPM information, installed dependencies are saved in" package.json" file in Node project.

6.1.9 Express JS framework

Framework is a structure which is built to serve the purpose of supporting or guiding in order to help developers builds projects on top of that. It plays a very important role in the project as it is the fundamental of development. Therefore, choosing the appropriate one is very necessary. (Baker)

Express JS is a web application server framework which is build on top of Node JS. It supports developing multiple type of web applications such as single-page, multiple page or hybrid types. Express JS is seen recognized as *de facto* framework for Node JS. It also means that, Express JS is a popular, unofficial standard server framework of Node JS. (Agarwal 2014)

Express JS is a minimal framework and developers can choose to add features using plugins. Normally, Express JS is well known with the stack called MEAN, the combination of MongoDB as Database, Express JS as server framework, Angular JS as frontend framework and Node JS as the environment.

Express JS supports a great number of features for developing mobile and web applications. List below describe core features of Express JS:

- It helps to create middleware to respond to HTTP Requests.
- It helps to design routing table to operate various actions based on HTTP Method (GET, POST,PUT,DELETE) and URLs. This is the core of API functioning.
- It can render HTML Pages dynamically.

Middleware is a module in Express JS framework. It has the access to all or a specific API request to the system. Middleware can be used to execute codes when the API under that middleware function is called. It can add, remove or lock variables in request function before passing to specific APIs. It can be used to end the request if the request properties fail to fulfil the requirement logic of middleware.

```
var express = require('express');
var app = express();

var check_body_input = require("body_input_verification");

function middleware(request, response, next) {
  if (check_body_input(request.body)) {
    next();
  } else {
    response.end();
  }
}

function execute(request,response) {
  console.log(request.body);
}

app.use('/example_api', middleware);
app.post('/example_api', execute);

app.listen(3000);
```

This is an example of how middleware functions works. Function “middleware” is executed before function “execute”. In this case, we assume that the middleware would like to check The format of data sent from clients or other servers using API called ‘/example_api’. If all the data can meet the requirement of logic which is stored in “check_body_input” module, then the process will move next to “execute” function. If the request fails to fulfil the requirement, middleware will send the response signal to the client to announce that the request is not appropriate. The process ends and “execute” function is not called.

However, as any framework has its own disadvantages as well. One of the biggest problem with express JS so far is built-in error handling. It causes troubles for developers when building large-scale project as losing track of Express JS middleware services.

There are numerous amounts of Express JS packages available to use. There are two packages which are used in the project will be explained below:

- Body-parser package – Express JS package middleware to handle JSON, Raw, Text and URL encoded form data from routes.
- Cookie-parser – A package to manage objects contained in cookies which is sent from Client side through API call.

As the requirement of the project is to build and develop API using Node JS technology and the framework must provide license as open source, there are several popular frameworks that can be chosen. Nonetheless, Express JS framework is chosen because of:

- Its popularity.
- Huge community support.

- Very clear documentations.

Express JS is an essential, easy to adjust and implement framework. It supports a great amount of features for developers to develop mobile and web applications. Restful API is fully integrated into Express JS framework. Moreover, it provides middleware integrations in HTTP Request responds. It also determines routing tables in order to accomplish various actions related to HTTP method and URL.

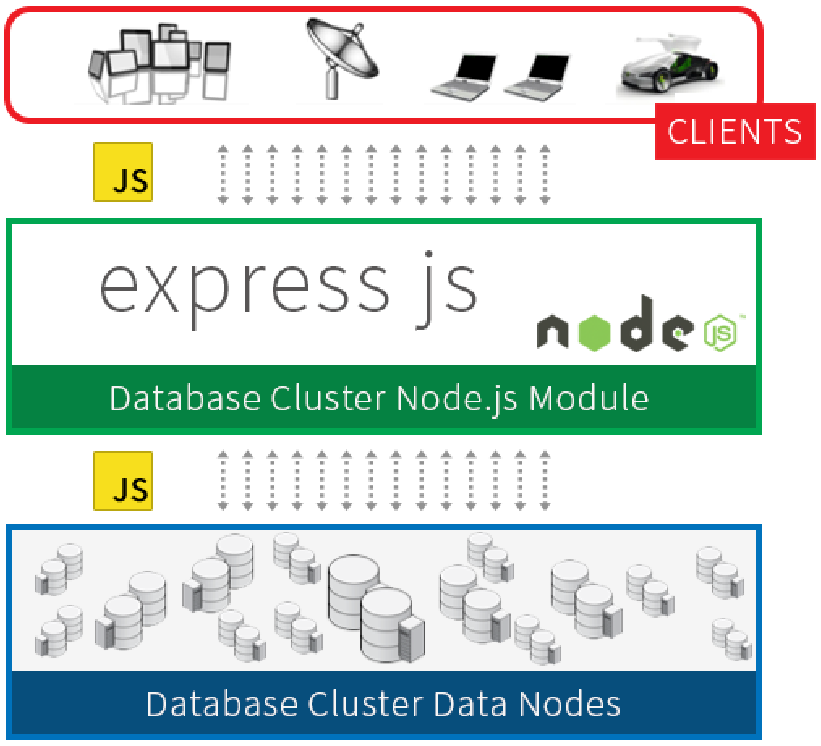


Figure 4: ExpressJS diagram (Agarwal 2014)

Due to the flexibility of Express JS framework, we can use Routing properties to structure API as modules in order to achieve Scalability requirement.

6.1.10 Azure

Azure is cloud computing platform from Microsoft to help developers develop, deploy and manage software products in Microsoft data centre. Core service of Microsoft Azure is compute services such as virtual machine, file storage and database. (Boucher 2015)

6.1.11 Azure App Service

App services is a product of Microsoft in Azure cloud system. It is a platform-as-a-service (PaaS). It manages and runs software on virtual machine. It supports web and mobile platform. It supports lots of useful tools and help deploying project easily. Developers only have to handle the logic of the software; deployment environments will be taken care by Azure App services.

Azure App services integrates Git version control system into their services. Developers just have to send source codes to App services using Git, then it will handle everything and deploy the product.

As the company is going to use Microsoft Blob storage and table storage as database and file storage in our system. Therefore, using Microsoft App Service is a good solution since the environment can be controlled under the same system.

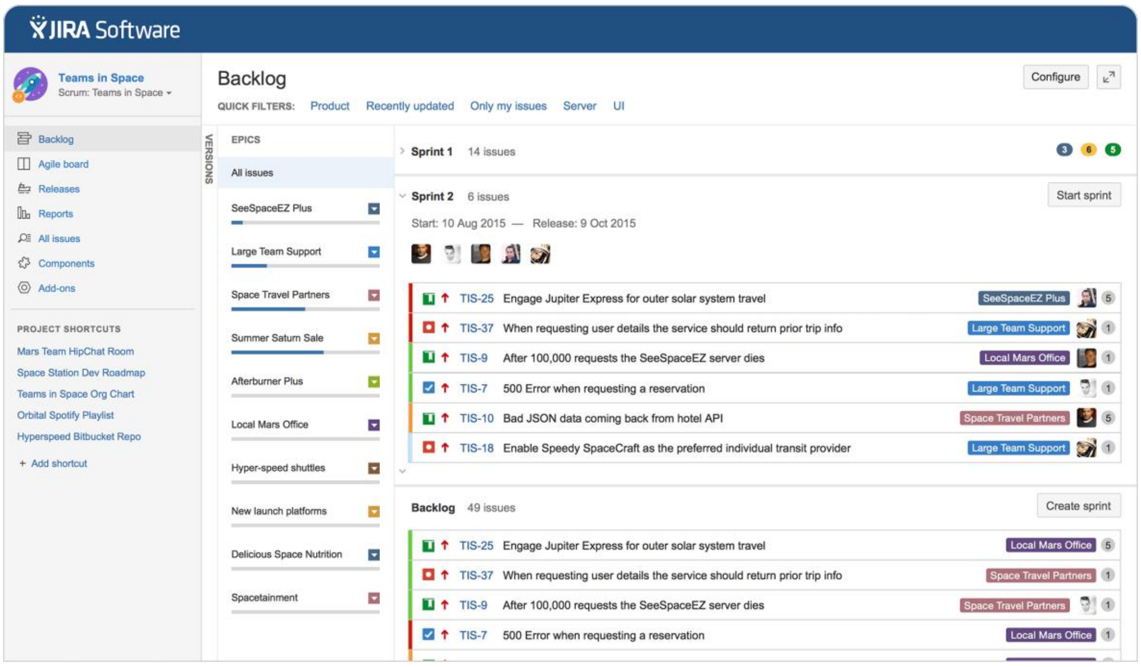
6.2 Development tools

This section describes software tools which are used during the development of application. It consists of management and development tools. Most of these tools are selected by the customer.

6.2.1 Jira software

Jira is one of the best Customer Relationship Management(CRM) tool. They provide multiple tools inside single platform which support development team and management team working together efficiently. It is built as a mini social network where colleagues can share posts, send direct message, comment or even like the post. They have task management tool which is very useful to track the process of the project. Developers can estimate and decide which task they have to finish first as well as managers have better vision how is the status of the project. (Atlassian)

Jira CRM software is being used by customer before the project starts. Therefore, they wish to use this tool in this project so that other member of the company finds it easy to manage the process of developing this application. By using Jira, customer can keep track the process of developing application and and manage deadlines as well as developing tasks.



JIRA Software

Teams in Space
Scrum: Teams in Space

Backlog
Agile board
Releases
Reports
All issues
Components
Add-ons

PROJECT SHORTCUTS
Mars Team HipChat Room
Space Station Dev Roadmap
Teams in Space Org Chart
Orbital Spotify Playlist
Hyperspeed Bitbucket Repo
+ Add shortcut

Backlog

QUICK FILTERS: Product Recently updated Only my issues Server UI

Configure

VERSIONS

EPICS
All issues
SeeSpaceEZ Plus
Large Team Support
Space Travel Partners
Summer Saturn Sale
Afterburner Plus
Local Mars Office
Hyper-speed shuttles
New launch platforms
Delicious Space Nutrition
Spacetainment

Sprint 1 14 issues 3 6 5

Sprint 2 6 issues Start sprint
Start: 10 Aug 2015 — Release: 9 Oct 2015

TIS-25	Engage Jupiter Express for outer solar system travel	SeeSpaceEZ Plus	5
TIS-37	When requesting user details the service should return prior trip info	Large Team Support	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Local Mars Office	1
TIS-7	500 Error when requesting a reservation	Large Team Support	1
TIS-10	Bad JSON data coming back from hotel API	Space Travel Partners	5
TIS-18	Enable Speedy SpaceCraft as the preferred individual transit provider	Large Team Support	1

Backlog 49 issues Create sprint

TIS-25	Engage Jupiter Express for outer solar system travel	Local Mars Office	5
TIS-37	When requesting user details the service should return prior trip info	Space Travel Partners	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Space Travel Partners	1
TIS-7	500 Error when requesting a reservation	Local Mars Office	1

Figure 5: Jira screenshot (Atlassian)

6.2.2 Brackets Text Editor

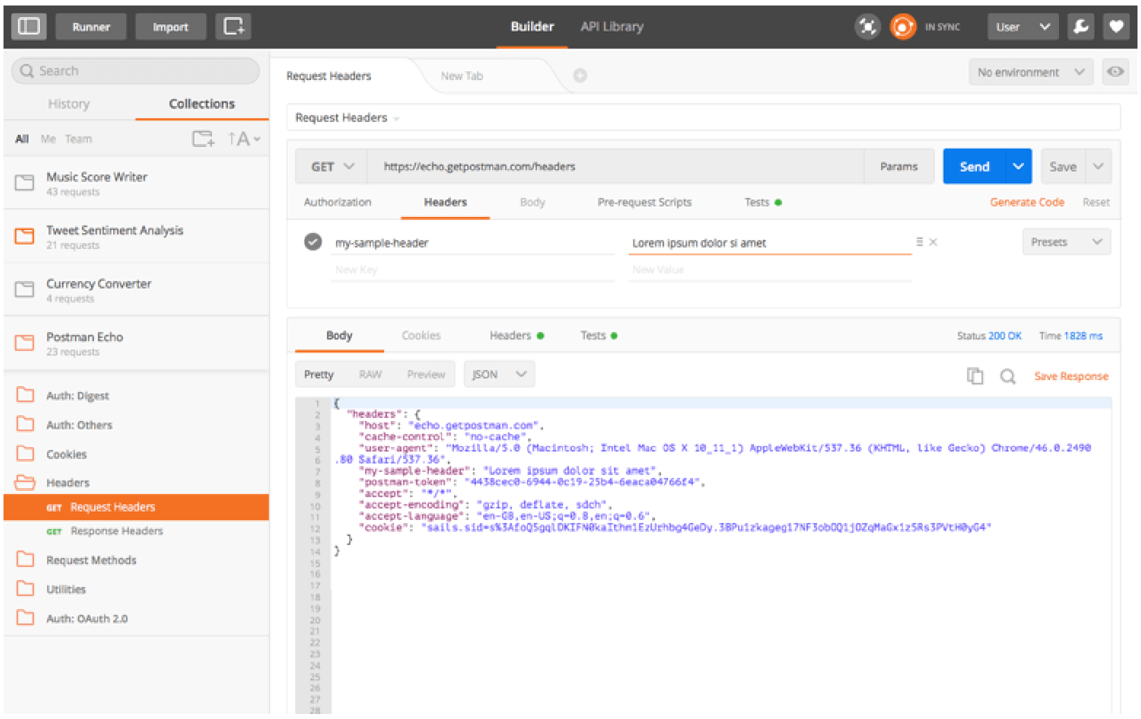
When developing the project, we are free to choose which development tool we need to improve coding performance as long as all members in development team use the same tool in order not to deal with some random errors. Since Node JS does not require compiling steps like other frameworks, we only need Text editor to program the software. There are a great number of available, free software to choose. Finally, we decided that we would choose Brackets Text Editor created by Adobe Systems.

Brackets is a free, open-source text editor mainly focus on Web development which has a great amount of supports for JavaScript programming language. It is a cross-platform software, support Mac OSX, Microsoft Windows OS and Linux based OS such as Ubuntu…(Adobe)

Brackets is also a good choice since other projects of the development team is developed using this Text editor. Other developer members gain many experiences with this text editor. We also develop some plugins and snippets to extend Bracket. Bracket also has extensions to connect to Git.

6.2.3 POSTman

POSTman is a tool that developed to support developers build API faster and better. It is used widely by big enterprises in the world. POSTman support testing the API, simulate API call and help developers collaborate with each other in a team. (POSTman)



```
{
  "headers": {
    "host": "echo.getpostman.com",
    "cache-control": "no-cache",
    "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490
.80 Safari/537.36",
    "my-sample-header": "Lorem ipsum dolor sit amet",
    "postman-token": "4438cec0-6944-0c19-25b4-6eaca04766f4",
    "accept": "*/*",
    "accept-encoding": "gzip, deflate, sdch",
    "accept-language": "en-GB,en-US;q=0.8,en;q=0.6",
    "cookie": "sails.sid=s%3AfoQ5gqlDKIFN0kaIthm1EzUrhbg4GeDy.3BPu1zkageg17NF3obOQ1jOZqMaGx1z5Rs3PVtH0yG4"
  }
}
```

Figure 6:POSTman screenshot (www.getpostman.com)

POSTman is an excellent and free tool to test API. They provide User friendly interface and members in the team can share the access to API template to test, examine or understand the format of API communication.

6.2.4 Git and Gitlab

Git is a powerful tool for develop team to collaborate in single project. Since we have a great number of developers, Git creates friendly and useful environment so that our development environment, source codes can be synchronized, merged and controlled. (Atlassian)

One of biggest advantages of Git is version control ability. It merely records the changes in files of project that implemented Git inside. It allows developers track what has been changed, how does it affect. In case of error, they have the ability to go back to stable version or where the problem did not occur. More than that, it is very useful to compare multiple version of source codes in order to understand more what causes the issues to the system, and who is in charge for that.

Until now, Git is a very popular free and open-source distributed version control system. Therefore, our team decides to use Git.

On top of Git version control, we used a software, web-based application called GitLab to manage our Git repository. GitLab is free and open source software developed by Dmitriy Zaporozhets. It supports a lot of useful features such as issue tracking and wiki of the project. Issue tracking is the feature that help developer notified if there are some problems during development and those who create the issue on GitLab can see the status of problem if it is solved. Wiki is a feature where documentation of whole project is saved. GitLab offer two packages, Community Edition and Enterprise Edition. In this project, we decided to use Community Edition as it supports features that development team needs.

6.2.5 Mocha Unit testing library

Mocha JS is BDD, open-sources testing library that support Node JS. It is a simple, scalable and fast testing suite. Mocha supports unit and integration testing. As the testing library for Node JS, it uses JavaScript syntax to build test scenarios for the object. (Hartikainen 2016)

6.2.6 JSDoc

JSDoc is a standard documentation tool for JavaScript language. It is an engine that scans all JavaScript files in a specific folder or project to create well displayed documentation. It uses special '/** */' syntax as a particular area to export contents to documentation. JSDoc has special syntaxes to define the function, modules, classes or methods.

6.3 Development process

This section explains briefly how the code is created according to Product architecture design mentioned above. We would explain general environment setup steps and component development process.

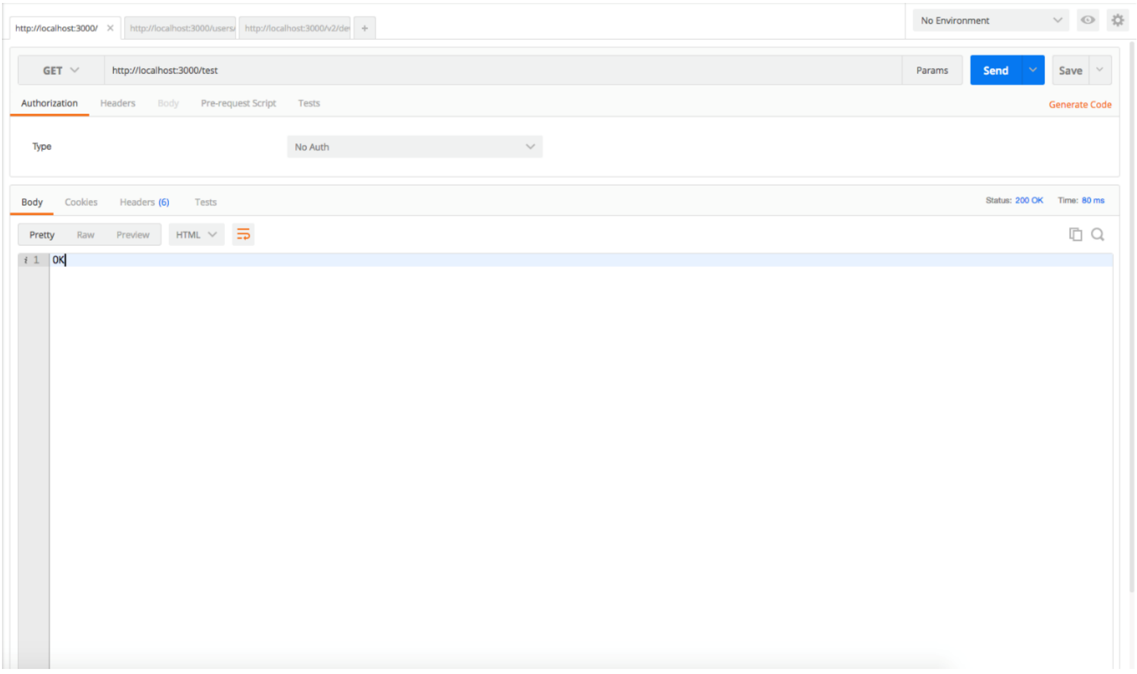
6.3.1 General setup

After planning and research steps finish, we start to create environment for the software. Using Azure Cloud platform, we setup App services for API platform and Data storage. Then we create Git connection to App Services and clone one version to store, develop and test locally.

First step is to setup project structure. As mentioned earlier, we build project structure according to section “Structuring API project” in this thesis paper. Then we install necessary modules and framework using Node Package Manager. We installed Express JS, Azure Data Storage SDK- Node JS version.

After setting up environment successfully, next step is to create structure for the software. Using Brackets Text Editor, we created “server.js” file as the main file for whole project. In that file, we declare listening port on server. We setup the server so that it will listen to Port 9000 on local machine and default port 80 on real Server. We declared installed modules mentioned before. After that, we setup simple API port called “/test” with default response status using “GET” type. Express JS plays the role as framework to control API. When API is called, it will return Success status without having to go through Middleware since it is not setup yet.

We used Postman tool to simulate API call. We tested by calling “http://localhost:9000/test” URL. It returned Success status on POSTman. Then we go to conclusion that software is setup correctly. Then we pushed our source code to App service using Git. Then we restart the services and tried the same method but used different URL, we use “http://<name_of_services>.azurewebsites.net/test”. It returns similar result as in local machine. App services environment works perfectly and can handle our API software.



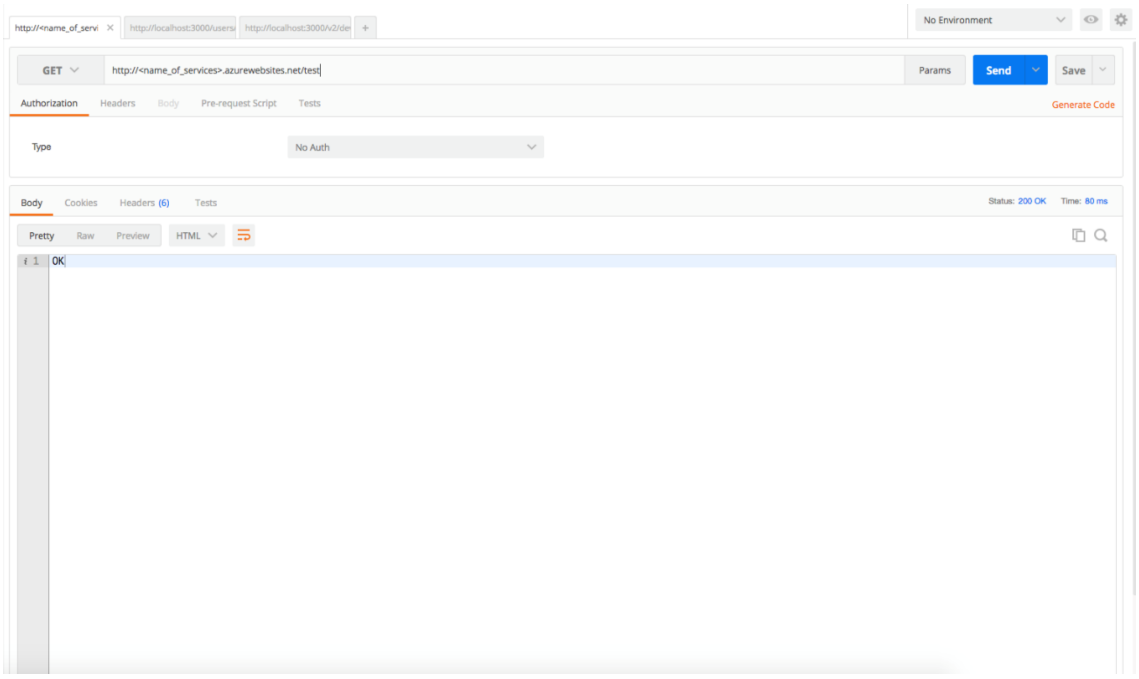
GET http://localhost:3000/test

Type: No Auth

Status: 200 OK Time: 80 ms

```
OK
```

Caption 4: POSTman test API locally



GET http://<name_of_services>.azurewebsites.net/test

Type: No Auth

Status: 200 OK Time: 80 ms

```
OK
```

Caption 5: POSTman test API on cloud

6.3.2 Creating Authentication and User management components

We start to create core components such as Middleware, User and Authentication components. We save all in folder named "Components". From the beginning of the process, we create User component where user can create user ID and password and it will be saved in the Database. User component can return information of existing user as well. Then we create Authentication component. It plays the role as controlling authentication of software. When user login, it does not use User API but Authentication API instead. We will check if user exists with correct user ID and password. If it is valid, we return the API call with access token. This token is controlled by API and has expiration time for security purposes. That access token works with very strict strategy. It will send an encoded string as response and that string contains information developers would like to transfer.

When receiving the token returned from authentication API, Client side will save it in properties of requests in Cookies of browser. By using cookies, we ensure that hackers fail to use any harmful JavaScript or any regular attack to steal access token and from now on, whenever any API is called from that Client side, it will contain that access token. After access token flow is setup, we create middleware to check if that access token is valid. We check expiration of token, decode and allow API process to continue if it passes all requirements and has valid information. Since any call from Client Side contains this Access token and must go through Middleware, our API is protected as MVP version require.

At this stage, we also create some logics to prevent attacks from clients. Middleware will stop any requests coming from the client where it detects unusual high requests such as few thousands requests per second. The client address will be put to “black list” and unable to connect to API application. This practice can prevent some attacks that try to reach the API to its maximum bandwidth to break the system.

Any user information saved in database is encrypted in order to protect user’s sensitive data from hackers and other employees in the company. We also set level for users. With this approach, each user at certain level can access to limited data set in that level. We can ensure that not all information is exposed to even high level users. The only way to get decrypted data is using proper API link with valid authentication token at certain level for user.

6.3.3 Creating Data collection and Device management components

We moved to develop Data collections and Device Management components. These parts have strong connection to each other. This can be seen as foundation for custom devices which will be integrated in the future. API gateways related to these components are for definite under the observe of Middleware mentioned earlier. When building Data collections, we plan to use it as the gateway for data coming from devices. It also serves as Real-time data collection. Whenever an event occurs in device that create data, it will send directly to this Data collection API and inside business logic, we create methods to announce data changes and send new data at the same time to Lumi Web portal as well as communication to Database. We also send Log activity of devices, backup data if the connection is lost.

Device management component is also a very important one. It is the gateway where Web portal can control the device remotely. Users can enable or disable devices status, receive to ban data from devices. This component is under development and will be finished in the near future.

In the future, whenever a custom device is connected to our platform, we will use those components as foundation to control, monitor the connection. By using this strategy, we are confident that scalability is achievable and the speed of developing new components is faster since all fundamental needs are completed.

6.3.4 Unit testing

When we finish coding part, next step is to setup and implement Unit testing for each component. To setup Unit testing tool, there are two parts need to be installed. First of all, we installed Mocha environment using NPM tool. Then we install project-based Mocha module to our project. We created folder called "/test". This is the place where we save all of the test scripts built using Mocha Unit testing JavaScript syntax.

There are two main parts that we have to test. First, we write script that describe how each component works. Going through possible scenarios and find if every of each passes the test without causing any error. Then we test components together by using more complex Unit testing scripts to ensure that those components work perfectly with each other.

To make unit testing for API application, we use POSTman during the development. For example, after finishing GET API request logic for getting specific user information from database through "/get_user_information" API. We use POSTman to make a request call to server to in order to check whether the response of the API meet the requirement of the coder or not. After that test, we move to writing testing script using Mocha library. Here is a simple example

```
describe("Get user information", function() {
 it("Return user information based on user ID", function(done) {
              chai.request(server)
              .get('/get_user_information?user_id=12345', function(response){

              res.should.have.status(200);
              res.should,be.a('object');
              done();
              })
   });
 });
});
```

By using mocha test, we create a scenario by calling API GET request and check response result. It is very convenient because it ensures the stability of the API, in the future, when we need to change some modules or business logics, we just have to run test commands to check whether new changes affect negatively to API result.

6.3.5 Documenting the module

Documentation is very essential during development process. To achieve proper JavaScript documentation, we use JSDoc and its syntax to write description to the code file. First we define the module as the parent. Then to document specific function, in this case is get_device_data function, we write that detailed description above the function.

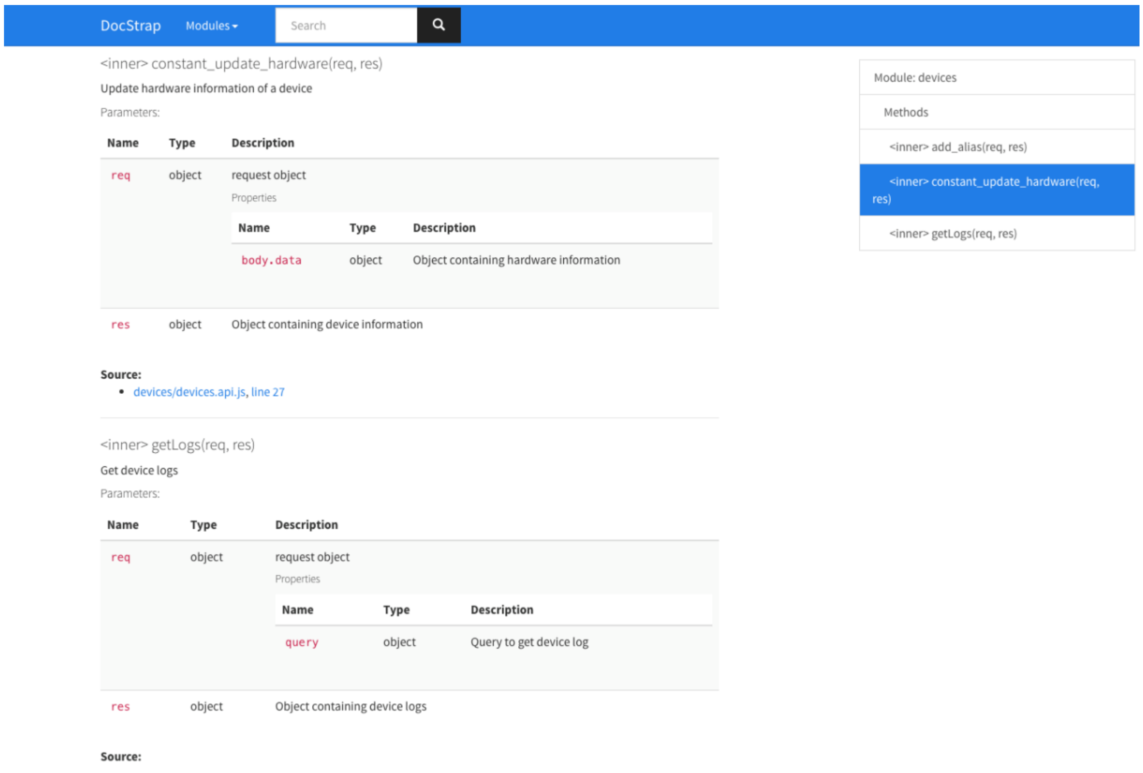
```
/** @module device */

var device = module.exports = {};

  /**
   * Get data of specific device using its ID
   *
   * @param  {String} device_id unique ID of device
   * @param  {Function} callback function to handle data of device from Database
   * @param  {Object} callback.device_data Data of device

   */

function get_device_data (device_id, callback) {
                  table.getData(device_id, function(device_data){
                  callback(device_data)
                  }

}
```

After writing detailed description of functions and modules to the file, I run the command line in Terminal

```
/path/to/API-application jsdoc –o /path/to/output/documentation
```

JSDoc will generate the documentation as an HTML page to show all of my documentation as a library to search in the future.

Here is the screenshot of documentation generated by JSDoc



<inner> constant_update_hardware(req, res)

Update hardware information of a device

Parameters:

Name	Type	Description
req	object	request object Properties Name / Type / Description body.data / object / Object containing hardware information
res	object	Object containing device information

Source:
- devices/devices.api.js, line 27

<inner> getLogs(req, res)

Get device logs

Parameters:

Name	Type	Description
req	object	request object Properties Name / Type / Description query / object / Query to get device log
res	object	Object containing device logs

Source:

Module: devices
- Methods
  - <inner> add_alias(req, res)
  - <inner> constant_update_hardware(req, res)
  - <inner> getLogs(req, res)

Caption 6:Documentation library

6.3.6 Deployment

When we finish testing the application. The next step is to push it to Lumi system. We use Git integration in Azure App services and create Git remote to Azure. Then we pull existing codes from Git and check if there are any conflicts with other parts of Lumi. After fixing possible conflicts displayed briefly in the log report from Git, we push API application to Azure and then we used POSTman to test the API function using real URL instead of “localhost”.

7 Live Testing

After deploying the API to Azure App services, we made live testing in real environment. Our customer locates in Helsinki city. Test period starts from June 2016 and finish in the end of August 2016. During testing period, the device was used in 50 days due to customer’s policy of opening hour. We installed one machine and setup the software to connect to API application. There are two connections between the machine and API.

First is device registration and installation, the machine could connect to API through the Internet. The machine synchronizes information with the device. All information from the machine was successfully sent to API and the API saved data to Database.

Second is sending live data from the device to Database. When the event from machine occurs, it will automatically send newest data to Database and API will handle the task of sending event signal to platform Website to announce that new data has been saved. After few weeks of testing, the machine has sent multiple data and all of it was successfully saved to Database through API. We could query data based on their requirements such as:

- Location based data
- Time based data
- Type of data (people going in or out)

Web portal can connect to API application to generate multiple reports based on customer demands as the final customer.

From the pilot device, everyday it sends from 100 to 200 messages depending on how many people going through the camera connecting to the device.

During live testing, there were some unexpected issues. On the first day, after finishing installation, the machine could connect to API and start sending data as well as synchronized information. From the office, remote controls can be sent to the devices. However, the day after that there were no data sent to API. Running unit testing in API application showed that the application run stable without any errors or bugs. Using POSTman, API on cloud was tested and run stable as expected. At the end of the day, the problem was found. The software which is responsible for counting people crashed due to memory leak and the processor of the device was reached to 110%. Therefore, the software failed to send data to API application.

In July, 2016, there were 3 days that the API did not receive any data sending from devices. The connection to the device was interrupted. The possibility of Internet connection was predicted. However, pinging to the router returned successful signal. Then after long investigation, the customer reported that they had to turn off the electricity in the area where device located due to some electricity issues.

We had Internet issues from the client machine in August. Fortunately, it was already predicted so we had backup plan for that issue in the machine. When collecting the machine back, after connecting the machine to Internet, all backup files were transferred to API and all the data was safe. Lumi API could handle that task very well by transferring large amount of backup data from the machine the API application without any problem.

8 Issues

During development process, we faced not so many problems since we had prepared well before development phase and we have made some projects using similar technologies with smaller scale. However, there still remains some critical problems that we had to solve.

One of the biggest issues we faced is the difference between local development and how source codes execute in Azure App services. Once we had the problem with authentication access token. As discussed earlier, assuming that user logins using valid user ID and password, it the Request pass the middleware. We return valid access token to Client side and access token is saved to cookies section on browser. It works as we described, in local machine. Unfortunately, when we deploy to App services, whole process was broken. Thanks to testing system, we were sure that all of our parts work correctly. However, after testing whole process, we noticed that even though Client side was saved in cookies, it fails to attach that token to all request call. Therefore, middleware stops the API request at that point. Then we realized that in local machine, everything is run under URL "localhost". Therefore, automatically, cookies from Client machine is accepted. However, when running in real environment, API server and Client side run in two different hosts. We had to make a research and thanks to experience we get from other projects, we came to conclusion that was Cross-origin HTTP Request(CORS) problem. There are some restrictions about what to send in HTTP Request as standard Internet Security. We have to declare to API server that IP from Client side is trusted sources and open cookies session in HTTP requests. After that, access token works perfectly and problem is solved. Although we are able to solve the problem, we have to make a lot of tests in order to ensure that our solution does not harm the stability of whole system and attackers do not use that gateway as the way to attack the system.

9 Conclusion

After evaluation test, the pilot version of API application has successfully developed, fulfilled all given requirements when the project started and its performance was ensured after test cases during development phase live testing. Nonetheless, pilot version has some problems that need to be improved or paid more attentions in the future.

API application must be able to handle big scale data transport. It has to support requests from more than 1000 devices at the same time without causing crashes or any other issues. In the future, clustering systems with multiple servers need to be implemented. It must be able to scale horizontally to multiple regions to support better to other customers in those areas by reducing traffic distances and make the system run faster.

In term of security, there should be more attack prevention strategies applied to avoid attacks from hackers or other individuals or organizations trying to harm the system like DDOS attacks or unauthorized access to database.

In short, the pilot version of API application fulfilled its requirements. However, there must be more works in future developments to build fully functional and stable product to be used in mass scale.

References

Adobe. No date. Accessed July 2016. http://brackets.io/

Agarwal, R. August 2014. Why use ExpressJS over NodeJS for Server-Side Development?. Accessed July 2016. http://www.algoworks.com/blog/why-use-expressjs-over-nodejs-for-server-side-coding/

Atlassian. No date. Accessed July 2016. https://www.atlassian.com/software/jira

Atlassian. No date. What is Git. Accessed July 2016. https://www.atlassian.com/git/tutorials/what-is-git/

Atlassian. No date. What is version control. Accessed July 2016. https://www.atlassian.com/git/tutorials/what-is-version-control/benefits-of-version-control

Baker, M. No date. What is a Software Framework? And why should you like 'em?. Accessed July 2016. http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em

Boucher, R Jr. June 2015. Introducing Microsoft Azure. Accessed July 2016. https://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/

Capan, T. No date. Why The Hell Would I Use Node.js? A Case-by-Case Tutorial. Accessed July 2016. https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Chitra R. August 2016. Node.js And React.js. Accessed July 2016. https://www.linkedin.com/pulse/nodejs-reactjs-chitra-r

Crockford, D. JavaScript: The World's Most Misunderstood Programming Language. Accessed July 2016. http://www.crockford.com/JavaScript/JavaScript.html

Croll, A. 2010. Namespacing in JavaScript. Accessed July 2016. https://JavaScriptweblog.wordpress.com/2010/12/07/namespacing-in-JavaScript/

Hartikainen, J. January 2016. Unit Test Your JavaScript Using Mocha and Chai. https://www.sitepoint.com/unit-test-javascript-mocha-chai/

Margaret, R. No date. RESTful API. Accessed July 2016. http://searchsoa.techtarget.com/definition/REST

Mozilla Foundation. July 2016. JavaScript Guide. Accessed July 2016. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction

Nanyang Technology University. 2009. Accessed July 2016. https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Nemeth, G. 2016. Node Hero - Node.js Project Structure Tutorial. Accessed July 2016. https://blog.risingstack.com/node-hero-node-js-project-structure-tutorial/

Osmanim, A. September 2011. Essential JavaScript Namespacing Patterns. Accessed July 2016. https://addyosmani.com/blog/essential-js-namespacing/

POSTman. No date. Accessed August 2016. https://www.getpostman.com/

Proffitt, B. 2013. What APIs are and why they are important. Accessed July 2016. http://readwrite.com/2013/09/19/api-defined/

Segue Technologies. October 2014. The Benefits of Unit Testing. Accessed July 2016. http://www.seguetech.com/the-benefits-of-unit-testing/

Stefanov & Stoyan. 2010. JavaScript Patterns. O'Reilly Media, Inc.

Tutorial points. No date. SDLC – Overview. Accessed August 2016. https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

Tutorial points. No date. SDLC – Waterfall model. Accessed August 2016. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

Illustrations

Caption 1: Lumi platform........8
Caption 2: Lumi Architecture design........16
Caption 3: API request call cycle........19
Caption 4: POSTman test API locally........36
Caption 5: POSTman test API on cloud........37
Caption 6:Documentation library........41

Figures

Figure 1: SDLC cycle (Tutorial points) 9
Figure 2: Waterfall model cycle (Tutorial points) 12
Figure 3: Node JS components (Chitra R 2016) 28
Figure 4: ExpressJS diagram (Agarwal 2014) 31
Figure 5: Jira screenshot (Atlassian) 33
Figure 6:POSTman screenshot (www.getpostman.com) 34

Tables

Table 1: API call examples 17