Enterprise Backend as a Service

(Code that writes the code)

# Project Workbook

****

**By**

**Aditya Doshatti (aditya.doshatti@sjsu.edu)**

**Darshil Kapadia (darshilpareshbhai.kapadia@sjsu.edu)**

**Devashish Nyati (devashish.nyati@sjsu.edu)**

**Maulin Bodiwala (maulin.bodiwala@sjsu.edu)**

**November 2019**

**Advisor: Gokay Saldamli (Assistant Professor, Computer Engineering, SJSU)**

# Index

* [**Literature Search, State of the Art**](#_nvy2rupigitx) **3**

[Literature Search](#_6q9a6j8fx96w) 3

[Natural Language Processing:](#_rgvc1kc62fpg) 3

[Backend as a Service:](#_2v3483wgahk)  3

[Database:](#_ttf16ixfvvbq) 4

[Prediction & Smart Insights:](#_k1t16b3re7gv) 4

[Deployment:](#_vcj13p654cnl) 5

[State-of-the-Art Summary](#_6u3nq5nkunvx) 6

[References](#_y3iz65n93k8t) 7

* [**Project Justification**](#_3znysh7) **9**
* [**Identify Baseline Approaches**](#_gyow5khfmgdk) **11**
* [**Dependencies and Deliverables**](#_739sycfnct16) **13**

[Dependencies:](#_eg640bxetauo) 13

[Deliverables:](#_auibzqkbonos) 13

* [**Project Architecture**](#_1z9cfm2p5d8p) **14**
* [**Evaluation Methodology**](#_c2vyy2106vpv) **15**
* [**System Design/Methodology**](#_m1z3ex8flgq0) **19**

[Solution Overview](#_hu6deblbwkyt) 19

[Preparation of RAML](#_uffl02c81110) 20

[Code and Docker Image generation](#_aij4tm7tq1n3) 22

[Deployment of Docker image](#_pdk4chejzzri) 23

[Pseudo code](#_2rza9h7oo6mf) 23

* [**Implementation Plan and Progress**](#_unabe7wtynb1) **25**
* [**Project Schedule**](#_4y35yk4vnlc7) **28**

# Literature Search, State of the Art

## Literature Search

#### “Natural Language Processing:“

“Natural Language Processing or NLP is a field where the interactions between humans and machines are dealt using the natural language. The purpose of NLP is to use machines to understand human languages. One of the applications of NLP is chatbots. In the research paper “Implementation of an inquisitive chatbot for database supported knowledge bases” by S Reshmi and Kannan Balakrishnan, they tell chatbot imitates humans in conversations using English or any other natural language as the input. The paper also tells how inquisitive chatbots have become a part of our system. Virtual support, electronic commerce doubt solving or tutoring, and social media uses chatbots with NLP heavily.“

#### “Backend as a Service: “

“Backend can be defined as a middleware which helps in handling the functionality of any application using APIs or SDKs. When we talk about Backend as a Service, it means that the user will have the backend made for them which can be used to interact with their frontend software. Software industry is growing everyday with tons of startup also coming forward everyday and that requires an efficient and quick development of software. Backend as a Service provides exactly that and keeps another thing in check and that is the quality of the backend which can be used as a service. In the paper “Availability Evaluation and Sensitivity Analysis of a Mobile Backend‐as‐a‐service Platform”, Costa, Igor, Jean Araujo, Jamilson Dantas, Eliomar Campos, Francisco Airton Silva, and Paulo Maciel have explained how backend as a service enables engineers to connect their application backend to cloud servers. They also explain, how backend as a service can be used to integrate with other apps.“

#### “Database:“

“Efficient management of any data can’t be done without a consistent and integrated database. For the modern IT and current science and industry maintaining such databases is an important task. For smooth functioning of backend we need to have a proper database management system which would make the interaction of code using the REST APIs fluent. In the research paper “Looking into a REST-Based Universal API for Database-as-a-Service Systems” Till Haselmann has explained that the goals of API should be to have maximum flexibility and exchangeability, for which the relational databases serves as the best source and hence we should use SQL database. To support this thought we found that in “A relational database environment for numerical simulation backend storage” Jacek Nazdrowicz also supports the thought of using SQL databases for backend.“

#### “Prediction & Smart Insights:

In one of the research papers “Forecasting Nike's sales using Facebook data” Linda Camilla Boldt has explained how Big Social Data can be used to predict real-world outcomes like the global sales of Nike Inc. Analytics and critical insights are the most important factors which can help expanding business at any level. To get greater competitive advantage companies are using machine learning (ML) and artificial intelligence (AI) to get prediction based on their past and current data and get smart recommendation from them. The most visible developments in Google’s neural network research has been the DeepMind network, the “machine that dreams.” According to Google, the company is researching “virtually all aspects of machine learning,” which will lead to exciting developments in what Google calls “classical algorithms” as well as other applications including natural language processing, speech translation, search ranking, prediction systems, and Recommendation Systems.“

#### “Deployment:“

“There’s an old dev saying that goes something like, “Always develop in an environment that is the same as your deployment environment.” Therefore, things like Virtual Environments exist. This is sage advice. “AppScale is a free and open-source software framework for running Google App Engine applications. The main goal of AppScale is to permit developers to have appliance portability. It is a cloud computing platform (marketed as platform as a service), supporting Xen, Kernel-based Virtual Machine (KVM), Google Compute Engine, Amazon EC2, Rackspace, OpenStack, Cloud Stack, and Eucalyptus. It is developed and maintained by AppScale Systems, based in Santa Barbara, California. AppScale was initially funded by Google, IBM, the NSF, and NIH.“”

“AppScale chains the capability to host several App Engine applications with the ability to exchange distributed datastores such as Accumulo, HBase, Hyper table, and Apache Cassandra. It has held up for Python, Go, PHP, and Java applications by implementing scalable services such as the datastore, memcache, blobs tore, users API, and channel API .”

## State-of-the-Art Summary

“**ZenQuery and apiOmat**“

“When it comes to providing APIs without writing a trace of code, we have several technologies to take inspiration from. ZenQuery provides API instantly given an SQL Query to it. It is basically a JAVA application which is hosted on user’s server and thus providing security of their data. apiOmat does have extra features like it supports the specification user’s data and also provide SDKs for mobile platform for Android and iOS. The only problem here is that the code for the API which is being generated by these technologies stays with them and is not provided to the users. Our project aims to provide the code for the exact same APIs.“

**Util-RAML-Code-Generator**

“Util-RAML-Code-Generator generates a code based on RAML specification. It needs RAML specification for the API i.e services which needs to be generated and the RAML specification for entities i.e the objects which will be manipulated by the services. This gives us a good base to start generating API at a very basic level like the CRUD operations. The challenge over here is that there is no way we can integrate the database with which we want the APIs to communicate.This is one feature which is missed dearly while using this software.“

## References

1. Costa, Igor, Jean Araujo, Jamilson Dantas, Eliomar Campos, Francisco Airton Silva, and Paulo Maciel. "Availability Evaluation and Sensitivity Analysis of a Mobile Backend‐as‐a‐service Platform." *Quality and Reliability Engineering International* 32.7 (2016): 2191-205.
2. Reshmi, S., and Kannan Balakrishnan. "Implementation of an Inquisitive Chatbot for Database Supported Knowledge Bases." *Sādhanā* 41.10 (2016): 1173-178.

URL: <https://link.springer.com/article/10.1007/s12046-016-0544-1>

1. Gropengießer, Francis, and Kai-Uwe Sattler. "Database Backend as a Service: Automatic Generation, Deployment, and Management of Database Backends for Mobile Applications." *Datenbank-Spektrum* 14.2 (2014): 85-95.

URL: <https://doi.org/10.1007/s13222-014-0157-y>

1. T. Haselmann, G. Thies and G. Vossen, "Looking into a REST-Based Universal API for Database-as-a-Service Systems," 2010 IEEE 12th Conference on Commerce and Enterprise Computing, Shanghai, 2010. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5708388&isnumber=5708385>
2. J. Nazdrowicz, "A relational database environment for numerical simulation backend storage," *2015 22nd International Conference Mixed Design of Integrated Circuits & Systems (MIXDES)*, Torun, 2015. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7208595&isnumber=7208464>
3. Prajak Chertchom, Shigeaki Tanimoto, Hayato Ohba, Tsutomu Kohnosu, Toru Kobayashi, Hiroyuki Sato, Atsushi Kanai, Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, vol. 721, pp. 107, 2018.
4. S. Asur, B. Huberman, "Predicting the future with social media", Web Intelligence and Intelligent Agent Technology (WI-IAT) 2010 IEEE/WIC/ACM International Conference on, vol. 1, pp. 492-499, 2010.
5. “Various Techniques in Deployment of Application in Cloud – Public Cloud” N. Usha1, V. Bhavana1, Vinay and Hegde3, 1Lecturer, Dept. of ISE, SJBIT, Bangalore, Karnataka, India. Lecturer, Dept. of ECE, Amrita Vishwa Vidyapeetham University, Bangalore, Karnataka, India. URL:<https://www.researchgate.net/publication/281639121_Various_Techniques_in_Deploymnt_of_Application_in_Cloud_-_Public_Cloud>
6. Blog.v soft consulting.com. (2019). *Swagger (OAS) vs. RAML - Which is Better for Building APIs?*. [online] Available at: <https://blog.vsoftconsulting.com/blog/is-raml-or-swagger-better-for-building-apis> [Accessed 2 Oct. 2019].
7. Zenqry.com. (2019). *ZenQuery - Enterprise Backend as a Service*. [online] Available at: <https://www.zenqry.com/> [Accessed 2 Oct. 2019].
8. GitHub Paysera. (2019). *paysera/util-raml-code-generator*. [online] Available at: <https://github.com/paysera/util-raml-code-generator> [Accessed 2 Oct. 2019].

# Project Justification

“The purpose of this project is to create a web application, that can be used to create backends, databases, host them on cloud servers and connect them to machine learning services. Small organizations or start up ideas face problems in creating back-end API’s and hosting the server on cloud platforms. Automated generation of back-end API’s can save a lot of time. It can benefit not only IT experts but non ITpeople also. It can help back-end developers, front-end developers, data scientists and non technical people as follows:“

1. “**Back-end Developers:** It can help back-end developers to create databases, quickly create API’s by auto generation, modify the API’s if needed and quickly host the server on some cloud platform.“
2. “**Front-end Developers:** It can save a lot of time for front-end developers who needs backend API’s and databases hosted on a cloud platform.“
3. “**Data Scientists:** It can help data scientists and machine learning engineers as they can quickly save data into databases using API’s, add on the API’s for their purpose.“
4. “**Non Technical People:** It can be a life saver for non technical people as they can create databases using this, create and host a server with RESTful API’s, connect with a machine learning service. Later they can even use the code to extend the functionalities.“

“The need for fast development of applications is rising. Companies constantly urge developers to create more quickly applications and other business software without sacrificing quality. It is very important to increase the development speed. By using the application created under this project, a user can rapidly develop databases, backend servers, host them on cloud services and use AWS machine learning services. They can talk to create their applications using chatbot. Thus the project is important for non technical people as well as software engineers.“

# Identify Baseline Approaches

“An API, when explaining to a layman can simply be explained as a code which will allow two different software to work and communicate with each other. As the software industry is growing more and more microservices being produced, APIs have played a huge role. Consider a situation where a microservice has been developed but to use it, you need to download it rather than having an API which one can call to get the desired operation. Obviously that microservice is doomed to fail. But as heavily as we are relying on APIs to help us out, it is interesting to know that there is almost no industry standard for creating APIs as well as documenting it. We need to have common vocabulary for various software to communicate. A proper definition of designing an API is half the work done. There are various languages out there which deals with this and the most popular ones are RAML(RESTful API Modelling Language) and Swagger. Various characteristics need to be analyzed like the code reusability, the support-base, the allowance of design patterns and the ease of use. RAML does have an edge over swagger when it comes to supporting the entire API’s lifecycle and is considered to be the fastest framework to work with. We are planning to use RAML for defining our APIs.“

“RAML is an approach to create backend API’s. It is used by Util-RAML-Code-Generator, which is an open source package used to create API’s. It takes input specification in RAML format and creates code. We plan to provide RAML specification for the API i.e services which needs to be generated and the RAML specification for entities i.e the objects which will be manipulated by the services. We also plan to add database connection by providing the endpoint in the RAML specification. In our scenario, the user will provide the requirements to create the database like table names, the fields, and the operations. Our application will create a database and will also dynamically create RAML specification files for tables(means entity) which will have fields to describe that entity and also for the operations(means services) which will finally be translated to APIs.“

# Dependencies and Deliverables

#### “Dependencies:“

“According to our plan, we plan to have our code running on our server and we would take input from user for where user wants it database to be hosted and provide a docker image of the APIs. So we are assuming that user must have a servers for this purpose.

The other concern is right now we are planning to work with relational database specifically SQL if the user wants the database to be NoSQL or any other database we will have to make changes and reconsider the complete plan for the project.“

“Our code which would be interacting with the user through chat bot or voice commands would be needing the ML services provided by Amazon, So our code is directly dependent on the ML services provided by Amazon. If we try to use our own ML code that will increase the scope of project significantly, so we decided to use AWS ML services.“

#### “Deliverables:“

“The one main deliverable is the prototype of the project, Where user would interact with either our website or chatbot or through voice commands to give the requirements as well as the details related to servers. Then our code would create the database with proper schema according to the requirement and generate a docker image which would have the code for accessing this database through REST APIs with basic operations. The docker container would be running on the API server if details are provided otherwise user would have a link to the published docker image which he can download and spawn a docker container from it.“

# Project Architecture

“The project architecture can be divided into 3 parts. The first part is the User interaction part. Here we would have a website wherein user would input data in forms or user can use the chatbot to give his inputs. We also plan to use dialogue flow to get inputs via voice commands. After collecting the input in the second part we have the EBaaS server, Where our code would process the inputs and generate the database with the schema on the user provided database server and generate the code for APIs for the same and publish a docker image with the code for the code. We plan to use the machine learning services provided by AWS to suggest the data points which user should consider adding to their schema to make it more good. Also use the ML services for natural language processing required for chatbot/ voice command to understand the user input and get the requirements. The output would be a published docker image which the user can download and spawn a container out of it on its API server. The container will also have the code so the user can get into the container and customize the code if he wishes to do so.“

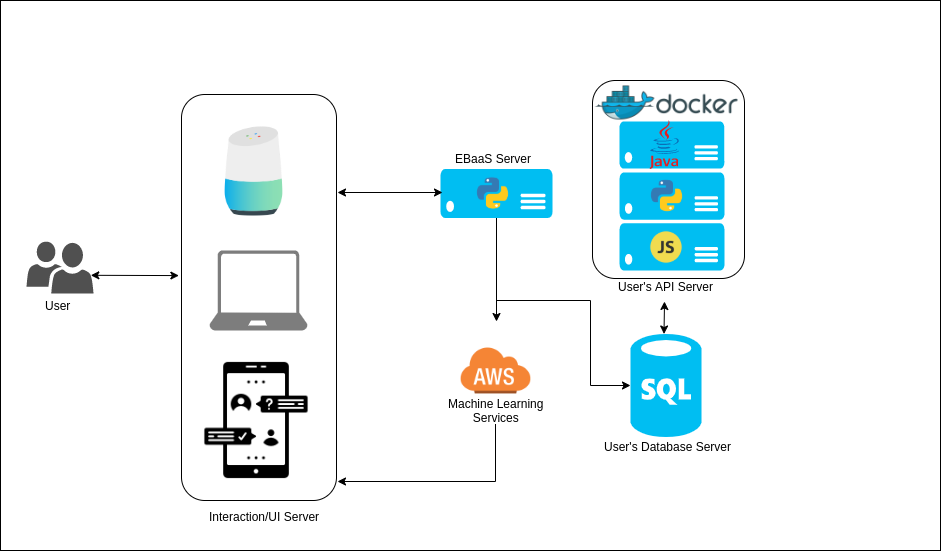


Fig1. Architecture of the Project

# Evaluation Methodology

As described in chapter 3, we are going to use RAML, i.e. Restful API Modeling Language. RAML is a way to deal with backend API's. We are going to use Util-RAML-Code-Generator, which is an open source bundle used to make API's. It takes input particular in RAML configuration and makes code and documentation. We intend to give RAML particular to the API i.e services which need to be created and the RAML specifications i.e the items which will be controlled by the service. We additionally plan to include database association by giving the endpoint in the RAML specifications. In our situation, the client will give the prerequisites to make the database like table names and the fields. Our application will make a database and will likewise progressively make RAML particular records for tables(means substance) which will have fields to depict that element and furthermore for the operations(means administrations) which will at last be meant APIs.

Steps to install and use Util-RAML-Code generator:

1. Clone <https://github.com/paysera/util-raml-code-generator> to your local machine.
2. Install composer by running the following lines on mac. Go to <https://getcomposer.org/download/> for more details.

|  |
| --- |
| * 1. php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"   2. php -r "if (hash\_file('sha384', 'composer-setup.php') === 'a5c698ffe4b8e849a443b120cd5ba38043260d5c4023dbf93e1558871f1f07f58274fc6f4c93bcfd858c6bd0775cd8d1') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"   3. php composer-setup.php   4. php -r "unlink('composer-setup.php');" |

1. run php composer.phar install.
2. Create config.json files with the location where the files are to be stored.

|  |
| --- |
| {  "SomeApiName": {  "raml\_file": "path/to/api.raml",  "clients": {  "javascript": {  "repository": "path/to/location",  "library\_name": "@vendor/some-api-client",  "client\_name": "SomeApiClient"  }  }  }  } |

Testbed-Setup and Result

We evaluated the performance of Util-RAML-Code generator by generating basic RESTful API service on a table called Person. We developed a simple form input which asks the user to provide the table name and upto 4 columns consisting in that particular table. Based on that, we created a dynamic RAML files related to Person table which will then be provided to the generator to verify the RESTful API code being generated.

Fig 6.1 shows the input which describes that the table to be made is Person with id, Name, Address and StudentID. When the user submits the form inputs, our web component processes that and produces the related RAML files. Fig 6.2 shows a Person.raml file being generated.

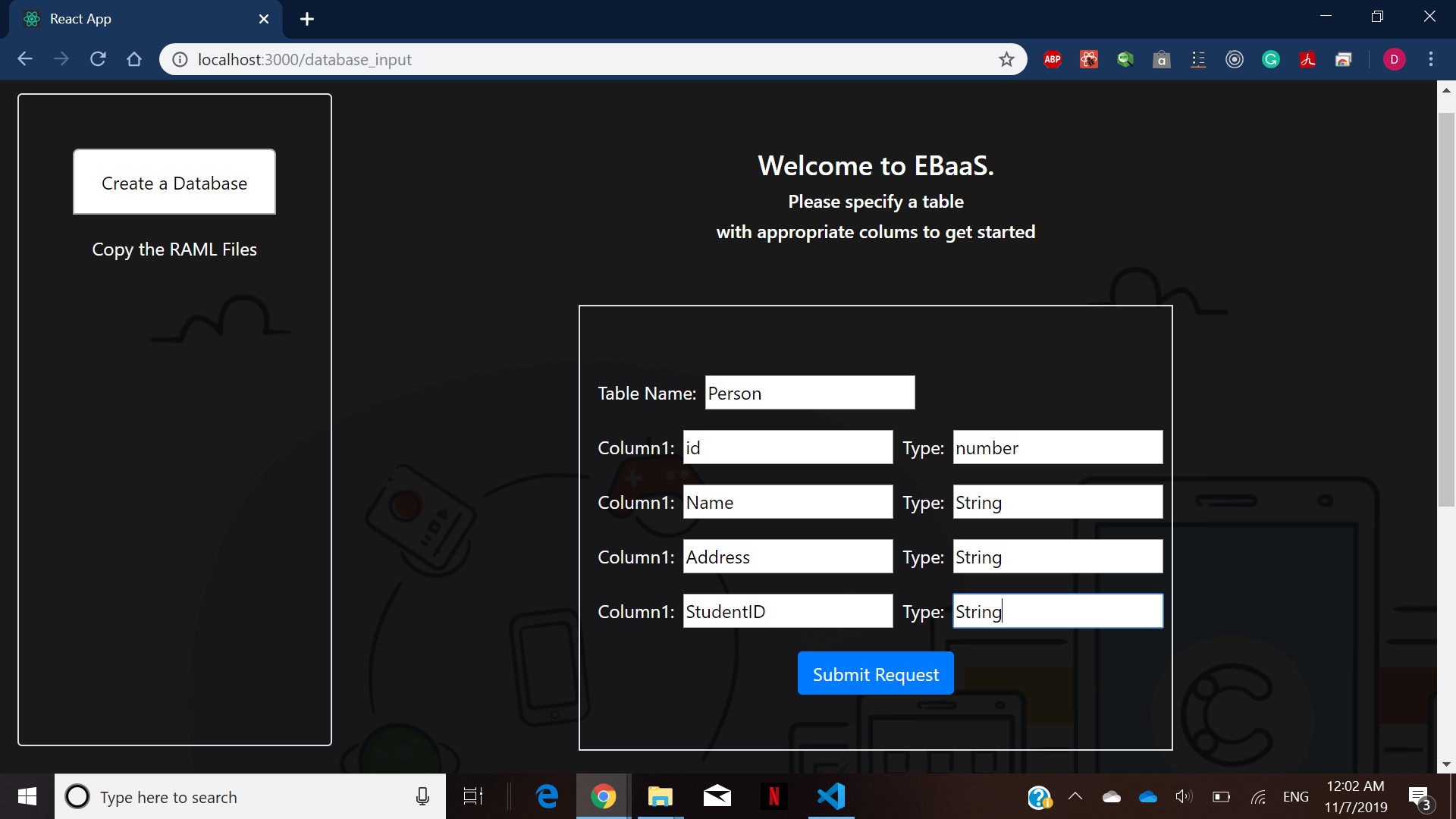


Fig6.1 Database Inputs

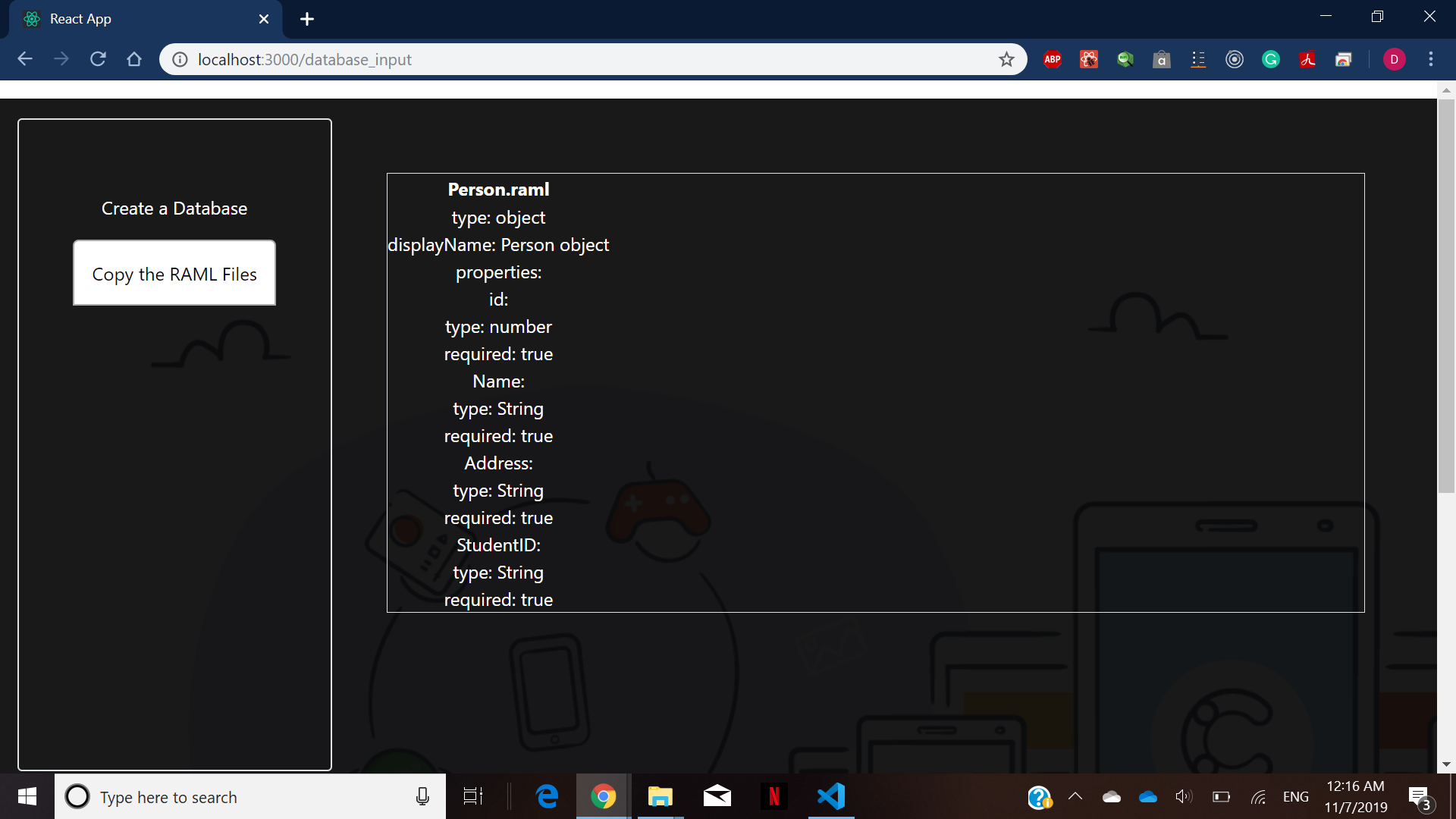


Fig 6.2 RAML File

After taking the necessary steps, it dumps the code which is generated to the output\_directory.

Performance:

We evaluated this methodology on the basis of amount of work that needs to be done by the user, the speed of the whole process and its efficiency.

* **Usability:** This methodology requires a lot of work to be done by the user. For example, installing util-raml-code generator which in itself is one tedious task to creating a particular project folder structure. It can be argued that the installation is a one-time thing but creating a particular project folder structure and copying files from our web component doesn’t provide smoothest of usability to the user. Hence, we are improving our component in to dynamically creating the RAML files, dynamically creating the project folder structure and directly giving users access to the generated RESTful API files in the form of zipped file.
* **Speediness:** Evaluating this methodology in terms of speediness is a bit tricky. If you consider the amount of work that goes into installing the generator and copy-pasting the raml files at particular places then the performance is not what users would expect from any software. But we are evaluating the speediness of this methodology after the setup is done i.e how much time the software itself is taking into giving users the access to code. Evaluating based on that, it performed as expected and the code was generated within a minute. So, by improving the usability we will be taking out the time required to setup the whole thing and thus would yield us the desired speed.
* **Efficiency:** We evaluated this methodology on how efficient is the code that is getting generated. We are checking the efficiency of the code on the basis of whether all functionalities are getting covered or not, and whether the errors exist or not. We found the code to be having all the basic CRUD operations without any errors. Hence, this methodology does provide an efficient RESTful Code.

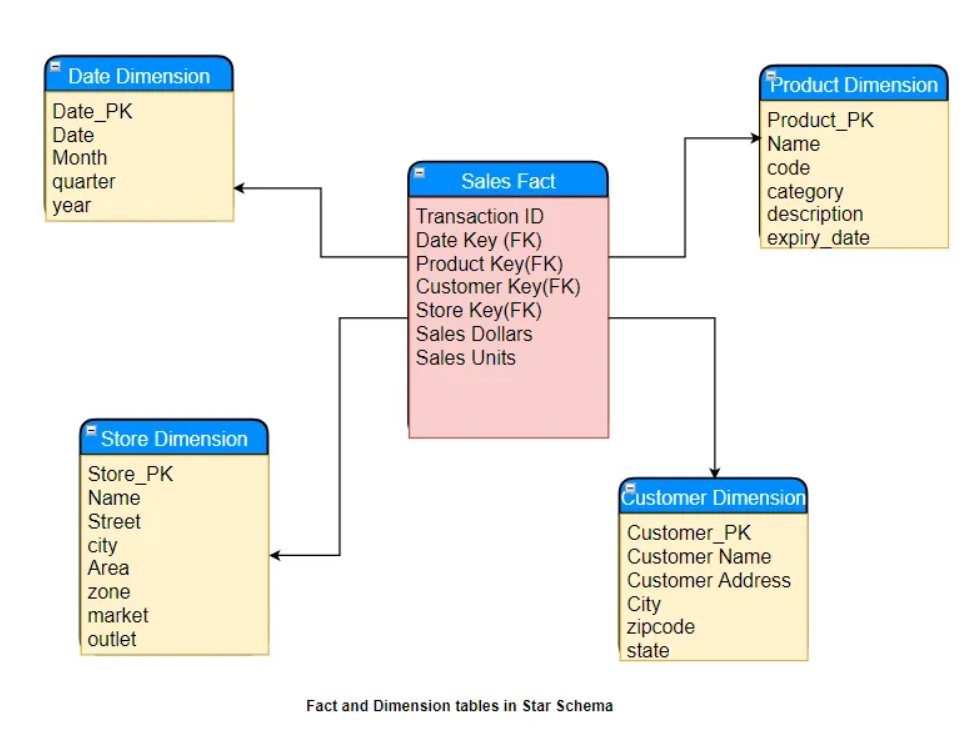
# System Design/Methodology

## Solution Overview

1. Communication with User for input
2. Preparation of Database Schema
3. Preparation of RAML
4. Code Generation and Docker Image generation
5. Deployment of Docker image
6. Communication with Users for Input :-  
     
   First and foremost step to start any thing is to look for the requirements.In our case the main requirement is to design the relational Database. We are implementing a User Interface where the user has to provide some predefined details like Database name , number of tables, names of tables, Columns required in to tables etc. This details we are planning to get through a form interface. As an extension we are planning to implement chatbot where our smart assistant will do the real time communication with the user and we will interpret all the details using Natural Language Processing.We are planning to use Natural Language Toolkit for NLP application.

1. DataBase Preparation :-

Database design we are planning to do based on the details provided by the users. Currently we are following use case of preparing backend for the sales department of any company. Below is the example of start schema distribution of sales data based on the tables and columns details provided by the user.



Database design can follow different structures like snow flex schema based on the Data we are getting from the user or based on the structure best suits the data. Here we have provided a simple example of sales data schema distribution for the products of the company.

#### Preparation of RAML

On the EBaaS server the code will gather all the inputs required for the API from user. Reading through the input submitted by the user through forms or to our Chatbot we would get the relevant and important information from the complete input. Once we have the relevant data out code would properly format the data to generate a RAML file from it.

The basic structure of a **RAML** file for an object/category is like:

|  |
| --- |
| *type: object*  *displayName: A display name for the object*  *properties:*  *ID:*  *type: string*  *required: true*  *description: Unique id for the object*  *Name:*  *type: string*  *required: true*  *description: The name of the category* |

The basic structure of a **RAML** file for an api is like:

|  |
| --- |
| *title: object API*  *version: v2.5*  *baseUri: https://getouroutput.com/object/v2.5*  *mediaType: application/json*  *protocols: [ HTTP ]*  */object:*  *get:*  *description: Get all the objects list*  *is: object*  *post:*  *description: Add an object*  *body:*  *application/json:*  *type: object*  *responses:*  *200:*  *body:*  *application/json:*  *type: object* |

The objective of the EBaaS server code would be to edit this template for RAML file according the user requirements. Once this file is generated we are ready to go ahead for our next step where we will use the RAML code generator to generate our code.

#### Code and Docker Image generation

One the RAML files are ready we can use the RAML code generator to generate the code for our APIs. The RAML file which we have generated after parsing the user input will be the input to the command line tool and this tool will generate the folder which will have the complete code for APIs. The execution of the command would be like:

|  |
| --- |
| *bin/console js-generator:package {path\_to\_raml} {output\_dir} {client\_name} --library\_name={library\_name}*   * *path\_to\_raml location of raml files.* * *output\_dir The directory to store output files.* * *client\_name name of main JS client.* * *library\_name name to be given to the generate library* |

Once this command is executed the output directory location will have all the files for the JavaScript client. All the class files in entity folder and the service files in service folder. Now we want to give these files to user. The docker file which will be used for generation of the docker image can do this task for us. The sample simple structure for dockerfile would be like:

|  |
| --- |
| # Base Image: Node.js 8.X  FROM node:carbon  # Create app directory  WORKDIR /usr/src/app  # Install app dependencies  # A wildcard is used to ensure both package.json AND package-lock.json are copied  # where available (npm@5+)  COPY package\*.json ./  RUN npm install  # If you are building your code for production  # RUN npm install --only=production  # Bundle app source  COPY {output\_dir} ./  # Expose Port  EXPOSE 8080  # Startup  CMD [ "npm", "start" ] |

Using this Dockerfile we can generate the docker image which would have the code of APIs generated by out raml util and we can then we can publish this image to docker hub and user can use the link to deploy as many containers as he wants to balance the load of his API requests.

#### Deployment of Docker image

Once the docker image is published we just need to pull the docker image and spawn a container of the image. We need to take care to have the docker engine is installed on the API server provided by the user. This will be a prerequisite which user has to take care of. Once we have the API server address and it has docker engine installed and running on it, the code on the EBaaS server can shoot a command on the users API server to have the docker container up and running which will have all the APIs running. User can actually ssh into the container can change or make required changes into the code and republish the docker image.

The future prospect for this part can be that we can have multiple docker containers started using Docker swarm to have proper load balancing. User can also use AWS services like EKS, ECS which are automated container management services, Also use can use GCP services like Kubernetes cluster. All these services has UI configuration where the user just needs to provide some basic details of the container which include the image location, port and number of containers for load balancing etc.

#### Pseudo code

1. Get the server details of database
2. If input has create database
   1. Get the database requirements
   2. Get the type of data which user wants to store (Tables)
   3. Get details about each type the user wants (Columns) / Suggest columns (Applying the ML on previously created tables with similar type of data)
   4. Connect the database server from the EBaas server
   5. Create database and schema according to above requirements
3. If input has connect to existing database
   1. Get the database name from user
4. Ask user input for if they want any particular query API
   1. If Yes
      1. Ask for Category details
      2. Ask for query parameters details
      3. Create a RAML file for this filter
5. Read the inputs provided for database or read the database to get the category information
6. For each attribute (Category)
   1. Generate a RAML file for this category
   2. Generate a Category filter for this category
7. End For
8. Use the command line tool to generate the code for API in JS.
9. The output folder with the code will be generated for these categories
10. Edit the docker file with the location of code
11. Generate the docker image with base image of node JS and add the code to the docker image and update the docker command to run the server with this code
12. Publish the docker image
13. Provide user the URL of the published docker image.
14. Deploy the docker container on user provided API server
15. Make the container connect with Database and have it up and running

# Implementation Plan and Progress

“We will be starting with creating a website which provides the user to provide us inputs in a form of story. At the end of the interaction, we will have all the requirements for creating a database and we will host a database with required tables and fields on the server provided by the user. After that, we will be creating RAML files to develop the specification files which in turn translate the database operation and manipulation to APIs. We will be using React.js for the front-end and Python for the back-end. The database will strictly be created as a SQL Database. For the insights we are thinking of using cloud services like Google Cloud and AWS.“

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PHASES** | **TASK NAME** | **ACTIVITIES** | **ASSIGNED TO** | **% Completed** |
| Planning | Business analysis | Selection of Topic | Team | 100 |
| Determining Project Scope | Team | 100 |
| Requirement Elicitation | Identification Risk involved | Team | 100 |
| Defining the use cases | Team | 100 |
| Determining Functional requirements | Aditya | 100 |
| Identifying Infrastructure Requirements | Darshil, Aditya | 100 |
| Abstract submission | Devashish | 100 |
| Design | High Level Design | Project architecture | Aditya | 100 |
| Outlining use cases to be implemented | Maulin | 100 |
| Detailed Design | UML diagrams | Maulin | 100 |
| Backend Design | Darshil,Devasish | 100 |
| UI Mockup | Darshil,Aditya | 100 |
| Work Book 1 submission | Devashish | 100 |
| Prototype Development and Testing | Prototype Development | UI development/ChatBot | Devashish, Darshil | 30 |
| Backend Development | Aditya, Darshil | 30 |
| Smart Predictor & Recommender Development | Devashish,Maulin | 0 |
| Integration | Aditya | 0 |
| Deployment Setup | AWS setup | Maulin, Darshil | 30 |
| AWS Deployment | Team | 0 |
| Work Book 2 submission | Devashish | 50 |
| Testing | Functional testing and Bug fixing | Team | 0 |
| Integration testing and bug fixing | Team | 0 |
| Documentation | Documentation | Project Report | Team | 0 |

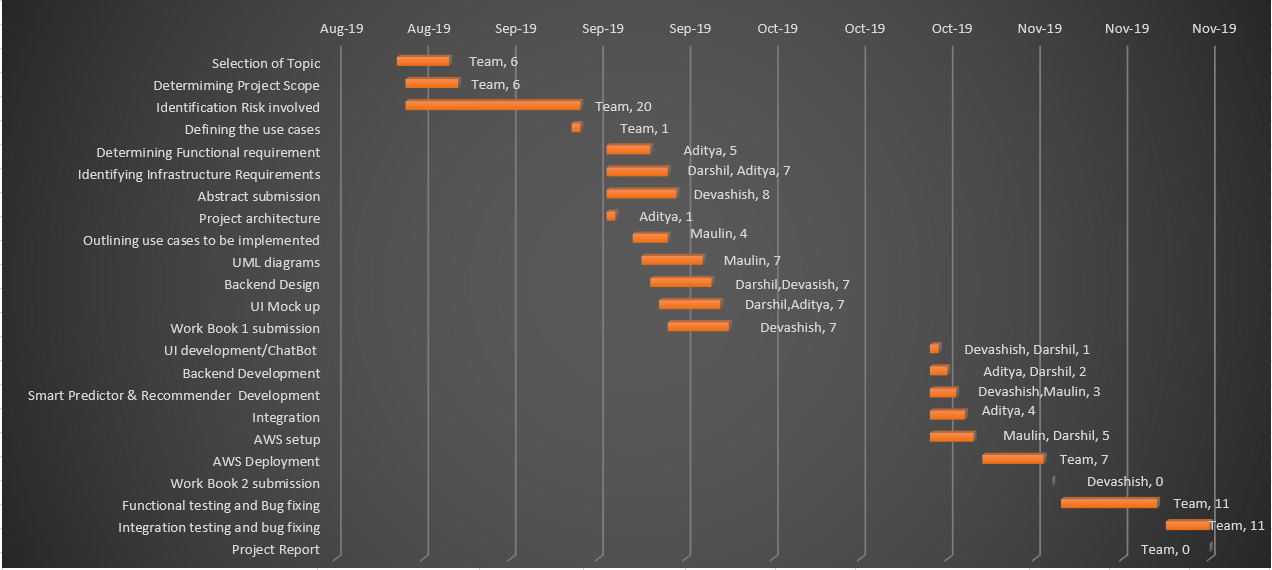
# 

# Project Schedule

“We have divided our project into different modules like business analysis, requirement gathering, prototype development, deployment, testing & documentation and have scheduled our whole project based on those modules and their respective deadlines with the subtasks for each module.

The table includes the activities, the people who that is assigned to (very important to delegate the responsibilities to deliver results within a deadline), start date and an end date. The duration field in our table is expressed in days. We have followed that with a gantt chart to show the division of work as well. “

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ACTIVITIES** | **ASSIGNED TO** | **START DATE** | **END DATE** | **DURATION** |
| Selection of Topic | Team | 8/20/2019 | 8/26/2019 | **6** |
| Determining Project Scope | Team | 8/21/2019 | 8/27/2019 | **6** |
| Identification Risk involved | Team | 8/21/2019 | 9/10/2019 | **20** |
| Defining the use cases | Team | 9/9/2019 | 9/10/2019 | **1** |
| Determining Functional requirements | Aditya | 9/13/2019 | 9/18/2019 | **5** |
| Identifying Infrastructure Requirements | Darshil, Aditya | 9/13/2019 | 9/20/2019 | **7** |
| Abstract submission | Devashish | 9/13/2019 | 9/21/2019 | **8** |
| Project architecture | Aditya | 9/13/2019 | 9/14/2019 | **1** |
| Outlining use cases to be implemented | Maulin | 9/16/2019 | 9/20/2019 | **4** |
| UML diagrams | Maulin | 9/17/2019 | 9/24/2019 | **7** |
| Backend Design | Darshil,Devasish | 9/18/2019 | 9/25/2019 | **7** |
| UI Mockup | Darshil,Aditya | 9/19/2019 | 9/26/2019 | **7** |
| Work Book 1 submission | Devashish | 9/20/2019 | 9/27/2019 | **7** |
| UI development/ChatBot | Devashish, Darshil | 10/20/2019 | 10/21/2019 | **1** |
| Backend Development | Aditya, Darshil | 10/20/2019 | 10/22/2019 | **2** |
| Smart Predictor & Recommender Development | Devashish,Maulin | 10/20/2019 | 10/23/2019 | **3** |
| Integration | Aditya | 10/20/2019 | 10/24/2019 | **4** |
| AWS setup | Maulin, Darshil | 10/20/2019 | 10/25/2019 | **5** |
| AWS Deployment | Team | 10/26/2019 | 11/2/2019 | **7** |
| Work Book 2 submission | Devashish | 11/3/2019 | 11/3/2019 | **0** |
| Functional testing and Bug fixing | Team | 11/4/2019 | 11/15/2019 | **11** |
| Integration testing and bug fixing | Team | 11/16/2019 | 11/27/2019 | **11** |
| Project Report | Team | 12/1/2019 | 12/1/2019 | **0** |



(Fig 2. Gantt Chart)