BotMan

An enterprise-grade framework for chatbot development

Project Workbook

By

Abhishek Madan

Aditi Shetty

Nachiket Joshi

Sushant Vairagade

November 15, 2017

Advisor: Prof. Rakesh Ranjan

Table of Contents

Chapter 1. Literature Search, State of the Art

Literature Search

With recent developments in the field of technology, people these days are adopting new means of communication. It has been observed that the use of messaging applications like Facebook, WhatsApp, HipChat are on the rise. In a study published by Flurry in their annual State of Mobile report 2016, it was found that the average time spent by an individual on social and messaging applications grew four times over the last year. The use of messaging applications is not only limited to the social platforms but also has penetrated the commercial space. Many companies are using chatbot on messaging channels as a means for customer service. Companies also use bots on their communication channels to take care of the day to day mundane tasks like answering the queries of employees, broadcasting a timely notification of important events, etc. Recent development in the field of Artificial Intelligence is helping us to make chatbots more intelligent and human-like. They can now even analyze and predict human sentiments and respond appropriately.

A blatantly evident reality faced by the app economy is that these days people are abandoning downloading application more than ever. According to a report by Digital Trends, only 16 percent of people try using an application more than twice. In the past 5-7 years, users have already grown tired of bouncing between too many applications or learning how to use a new interface after every new download. Studies show that the applications that users prefer to use on their mobile devices are the ones which act as a single access point to the services provided by multiple applications combined.

Moreover, there has been a shift in the way people prefer to communicate. They are now getting inclined towards chatting application over any other way of communication. People are now spending more time in messaging applications than on social media and that is a huge turning point. Messaging apps are the platform for the future and bots would be the way how users would access all sorts of services over the internet.

Natural language processing (NLP) is yet another area of research which is gaining increasing popularity. It aims at understanding how people organize their thinking, feelings, language and behavior to produce results competent to that of humans. The reason why so many bot platforms are popping up is the advent of many NLP services. Connecting to a channel and developing bots was not a problem, the missing link here was a platform which could understand the intention of the users and reply intelligently. NLP platforms like api.ai, wit.ai and

many others have now bridged this gap. This gives users an illusion that they are talking to a person rather than a machine.

While chatbots seem to have a promising future, many things need to be taken care of to successfully configure and maintain it. Selecting a cloud infrastructure to host the chatbot is one of the crucial things to think about. The cloud platform should be reliable enough to maintain high service availability. Another consideration is selecting an NLP platform which could quickly understand the user's intention and reply appropriately. The model should be able to learn quickly and scale its knowledge starting with a limited dataset. More importantly, the chosen platform should handle error conditions gracefully.

State-of-the-Art Summary

Cloud platforms are advancing at a rapid rate. They reduce capital expenditure on hardware and upgrades. They also help you cope with the changing demands. One can easily scale their cloud infrastructure based on the service demand. If your needs increase it's easy to scale up your cloud capacity, drawing on the service's remote servers. Likewise, if you need to scale down again, the flexibility is baked into the service. This level of agility can give businesses using cloud computing a real advantage over competitors and is making cloud hosting platforms an attractive option over hosting an in-house server rack. Cloud platforms these days have become more secure than ever before. They implement advanced security measures ensuring data integrity and preventing a data breach. With many cloud platforms providers like IBM's Bluemix, Amazon's AWS around, users can now have an option to choose a platform which they feel are more reliable and cost-efficient for their bots.

As per the article published on 451Research, *"Virtual assistants, bots and conversational software interfaces are sending the world's largest tech companies hunting for natural-language-processing (NLP) technologies and expertise"*. Attempts are being made to develop models which could understand natural language of human to gauge their needs. This helps machines to both interpret input and produce output in the form of human language. IBM, Facebook, Google, and Amazon are all making attempts to improve their NLP venture to gain a competitive edge over their competitors. Every day, attempts are being made to improve the accuracy of the NLP algorithm to closely mimic human thinking. With such available tool, bot developer could now engage users into a much insightful talk.

Tech companies are providing frameworks which facilitate the creation of chatbots for different channels. Gupshup, one of the world's largest Enterprise messaging platform is being

used by leading companies such as Flipkart, OLA, Facebook to develop chatbots for customer engagement and retention. Gupshup's platform handles on an average of 150 billion messages in total per day. It also provides NLP support and full lifecycle development for their bot service. The growing popularity of chatbot framework platform like Gupshup projects solidifies the potential that chatbots hold for the future.

References

1. Zdravkova K. (2000). Conceptual framework for an intelligent chatterbot. *Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No.00EX411)*, Pula, Croatia, 2000
2. Hung V., Elvir M., Gonzalez A. & DeMara R. (2009). *Towards a Context-Based Dialog Management Layer for Expert Systems*. International Conference on Information, Process, and Knowledge Management, Cancun.
3. Hung V., Elvir M., Gonzalez A. & DeMara R., (2009) *Towards a method for evaluating naturalness in conversational dialog systems*, IEEE International Conference on Systems, Man and Cybernetics, San Antonio.
4. Gartner Customer 360 Summit 2011. (2011). [ebook] Available at: https://www.gartner.com/imagesrv/summits/docs/na/customer-360/C360_2011_brochure_FINAL.pdf.
5. Pilato, G., Augello A. & Gaglio S., (2011). *A Modular Architecture for Adaptive ChatBots*. Retrieved from http://ieeexplore.ieee.org.libaccess.sjlibrary.org/
6. Augello A., Pilato G., Machi A. & Gaglio S. (2012). *An Approach to Enhance Chatbot Semantic Power and Maintainability: Experiences within the FRASI Project*, IEEE Sixth International Conference on Semantic Computing, Palermo.
7. Pramis, J. (2013, March 12). *Are you a rarity? only 16 percent of people will try out an app more than twice.* Retrieved from https://www.digitaltrends.com/mobile/16-percent-of-mobile-userstry-out-a-buggy-app-more-than-twice/
8. Satu M.S., Parvez M. H. & Shamim-Al-Mamun, (2015). *Review of integrated applications with AIML based chatbot*, International Conference on Computer and Information Engineering (ICCIE), Rajshahi.
9. Denne, S. (2016, September 20). *Google and other tech giants understand the value of natural language technology*. Retrieved from https://blogs.the451group.com/techdeals/ma/google-and-other-tech-giants-understand-the-value-of-natural-language-technology/

10. Reshmi S., Balakrishnan K. (2016, October 07). *Implementation of an inquisitive chatbot for database supported knowledge bases*. Retrieved from https://link.springer.com/article/10.1007/s12046-016-0544-1
11. Levy, P. H. (2016, October 18). *Gartner Predicts a Virtual World of Exponential Change*. Retrieved from http://www.gartner.com/smarterwithgartner/gartner-predicts-a-virtual-world-of-exponential-change/
12. Setiaji, B., & Wibowo, F. W., (2016). *Chatbot Using A Knowledge in Database*. 7th International Conference on Intelligent Systems Modelling and Simulation.
13. Smith S. (2016, October 18). *Mobile banking users to reach 2 billion by 2020, representing more than 1 in 3 of global adult population*. Retrieved from https://www.juniperresearch.com/press/press-releases/mobile-banking-users-to-reach-2-billion-by-2020

Chapter 2. Project Justification

Gartner predicted that by 2020, the average person will interact more with bots than with their spouses and that 85% of customer relationships with the companies will be managed by chatbots. A Chatbot is a software developed to complete a transaction by communicating with other entity using a predefined set of questions and answers. It is the support of technologies like artificial intelligence, machine learning and natural language processing because of which chatbots are now becoming even more humanlike or maybe be even better. Enterprises are investing a fair amount of their resources and time in trying to incorporate chatbots in their existing system to help them reduce the inflow of customer requests that can be handled by a chatbot. Chatbots help the enterprises to drastically cut their labor costs. A report released by Juniper states that by 2022 chatbots will be saving $8 billion per annum of labor costs. JPMorgan Chase, a bank that has a chatbot to help them with their back-office operations has saved them 360,000 human hours. Uber and Lyft have a chatbot that can book a cab using a conversation, this chatbot can replace their mobile application which means less application maintenance for them once most of their customers start using chatbot.

Enterprises develop their chatbot for a specific need that will serve their purpose. Chatbots are being used in different use cases; healthcare industry, customer support, e-commerce, HR systems. To develop a chatbot from scratch with a fair amount of customization could cost a company $40K-$100K and will take 2-3 months of time, that followed by the infrastructure maintenance cost which could be a few thousand dollars per month. It is highly unlikely for a small enterprise to develop a chatbot from scratch. Mid-sized or small enterprises usually either consult software companies or use online subscription-based services. Companies like Google, Facebook and IBM have invested a lot of resources in researching the fields of artificial intelligence and natural language processing. They offer conversation services (APIs) that can be easily used by chatbot to derive intents from the conversations, this saves the enterprises from investing a lot of time and cost for researching and developing their own in-house conversation service. A channel is a medium through which a user can interact with chatbot; Facebook messenger, Slack, Kik or even their own chat application. These channels also can easily be used by the chatbot with minimal configuration changes.

To address the issue of high cost of development and maintenance of chatbots we are proposing a chatbot framework that will drastically reduce the cost and the time required in the process. This framework will provide the user with a choice of using a channel, a conversation service and a cloud platform to deploy the chatbot from amongst the best in the industry. The

framework will provide templates to the user that needs a minimal amount of changes. The user will also be provided with a personalized dashboard which will show all the information about the user's currently and previously owned bots with versioning support. The dashboard will also provide analytics about the usage; conversation traffic, average time of conversation, uptime and downtime. The user will also have access to the detailed report on failed conversations that can be used to further train the bot and avoid any failures in the future.

Chapter 3. Project Requirements

Use Cases

1. As a framework user, I want to be able to maintain the history of my chatbots and their version history.

2. As a framework user, I want to be able to have a choice to select the suitable channel for my chatbot.

3. As a framework user, I want to be able to have a choice to select the suitable conversation service for my chatbot and integrate it with ease.

4. As a framework user, I want to be able to have a choice to select the cloud platform to deploy my chatbot.

5. As a framework user, I want to be able to have a detailed view of the statistics of my current as well as old chatbots.

6. As a framework user, I want to be able to see the failed conversations with the details about the chat, by using this information I should be able to train the chatbot to handle failed conversations by configuring relevant intents.

7. As a framework user, I should be able to change the channel, cloud platform or the conversation service with minimal configuration changes.

8. As a framework user, I should be able to deploy a selected version of my chatbot at a press of a button.

Functional Requirements

Req No.	Requirement Description	Type
Req 1	Framework should be able to create and maintain user profile	Essential
Req 2	The system should be able to maintain the history of the bots and the changes made to them by the user.	Essential
Req 3	The system should be able to display a dashboard with details of the chatbots owned by the user.	Essential
Req 4	The system should be able to provide a choice of conversation service (NLP/AI/ML) API user wants to use for a chatbot.	Essential
Req 5	The system should be able to provide a choice of channel user wants to use for a chatbot.	Desired
Req 6	The system should be able to provide a choice of cloud platform user wants to use for deploying a chatbot.	Optional
Req 7	The system should provide the user with the detailed report on the chatbot's failed conversations.	Essential
Req 8	The system should be able to provide detailed statistical analysis about the active chatbots of the user with many conversations, uptime, downtime, the average time of the conversations.	Desired
Req 9	The system should send the user notifications if there's a sudden increase of load or increased rate of failures.	Optional

Non-functional Requirements

Req No.	Requirement Description	Type
Req 1	The system should be able to scale easily as per the user's requirements.	Essential
Req 2	The system should be able to maintain high availability.	Essential
Req 3	The system should provide instructions to the new user when using the application.	Optional
Req 4	The system should use a data replication to have a backup of data during a system failure.	Desired
Req 5	The system should encrypt user's data before storing the data into the database ensuring the security.	Optional
Req 6	The system should generate detailed logs that will help to debug.	Essential
Req 7	The system should have at least 90% of test coverage.	Desired

Chapter 4. Dependencies and Deliverables

Dependencies

1. Integration support provided by the 3rd party service providers of messaging channels and conversation services.
2. Accessibility of the user's data.
3. The format of the user's data, it should be available to the framework in the JSON format.
4. Framework's availability and performance of the system will be directly impacted by that of the integrated external services.

Deliverables

1. Framework as a service accessible to the users to create and maintain chatbots.
2. A web interface to the user with a dashboard providing information of the user owned chatbots.
3. System generated statistical reports on chatbot's performance and failures.
4. Notification service for the user for an unusual change in chatbot's performance or incoming traffic.
5. Test cases.
6. Project documentation and user manual.
7. UI mockups and wireframes.

Chapter 5. Project Architecture

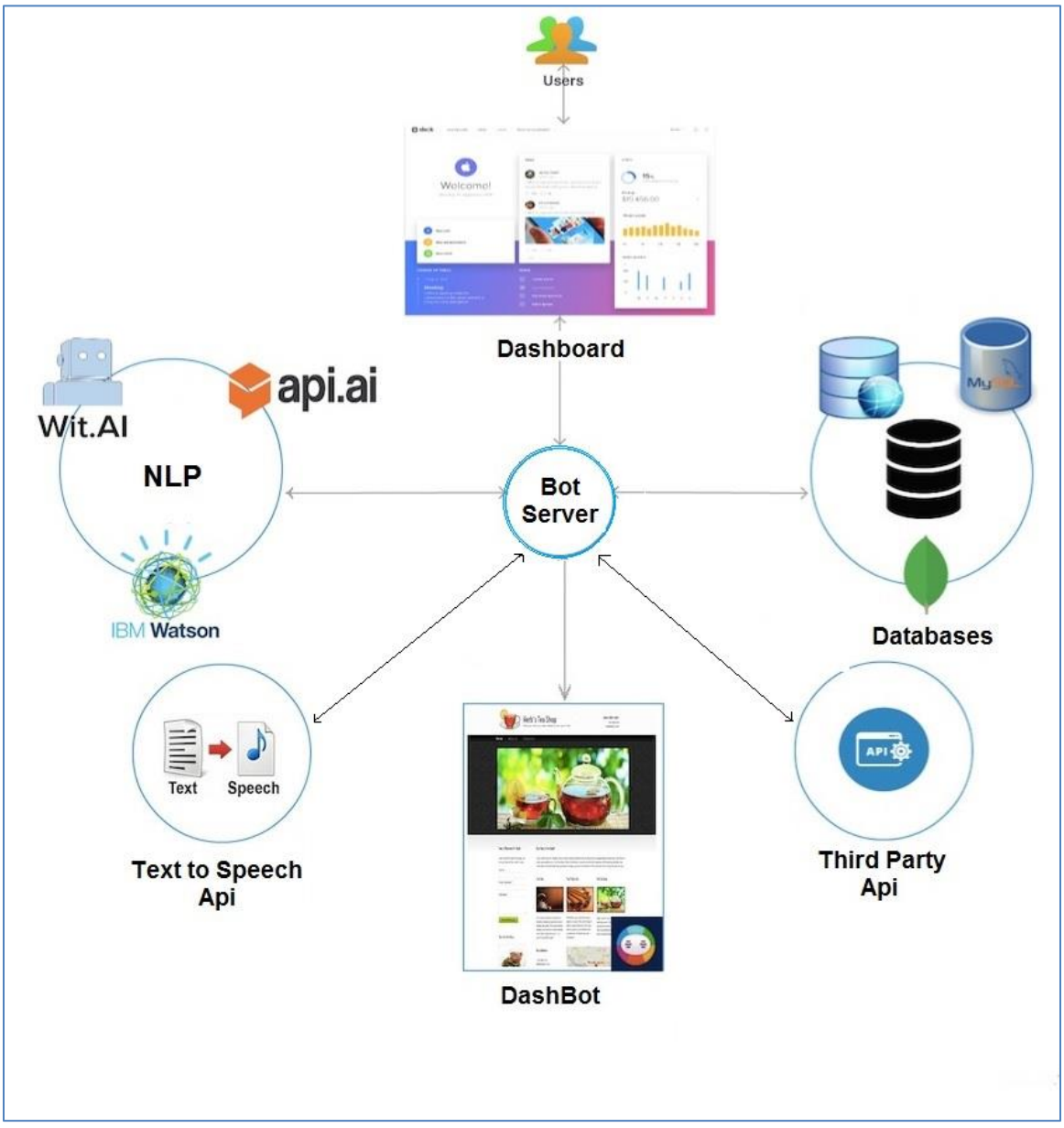


Fig. 5.1 System Architecture

1. **Database**: This is the customer's database which can be queried to complete business transactions. Depending on the output from the NLP after processing the user's input, the database will be queried to fetch the data for the response.

2. **NLP:** This is a set of conversation APIs that are available to the developer for processing user chat. Depending on the relative performance and features offered by an NLP platform, the developer can make a choice from a set of available options.

3. **Text to speech API:** API to convert text to speech and vice versa. For example, Google's Cloud Speech API.

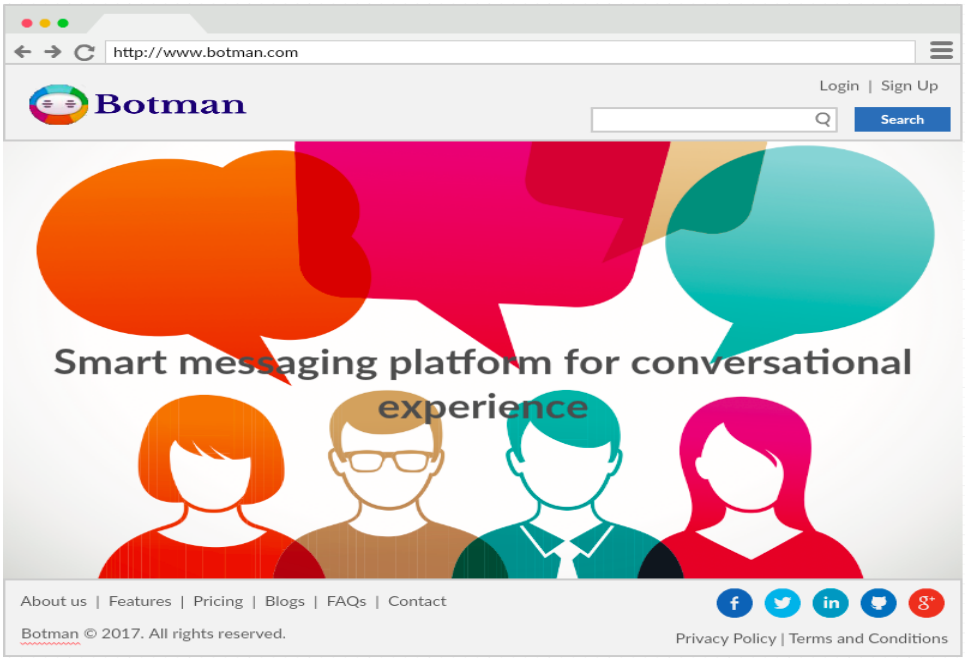
4. **Third party API:** The framework facilitates integration any third-party API which the developer can rely on for formulating the chatbot's response. For example, if a developer wants to include a real-time weather information as an intent to be handled by the chatbot, any third-party weather API could be easily integrated.

5. **Bot Server:** A cloud server platform for chatbot hosting; IBM, Amazon or Heroku Pass. This is where the operational logic for the chatbot would be hosted.

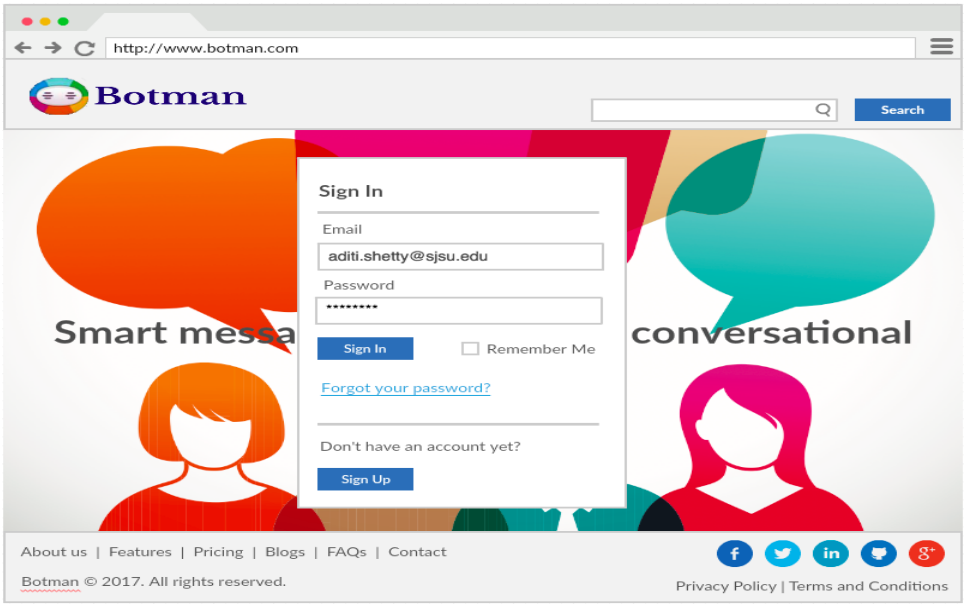
Chapter 6. Project Architecture

Please find our proposed UI mockup diagrams,

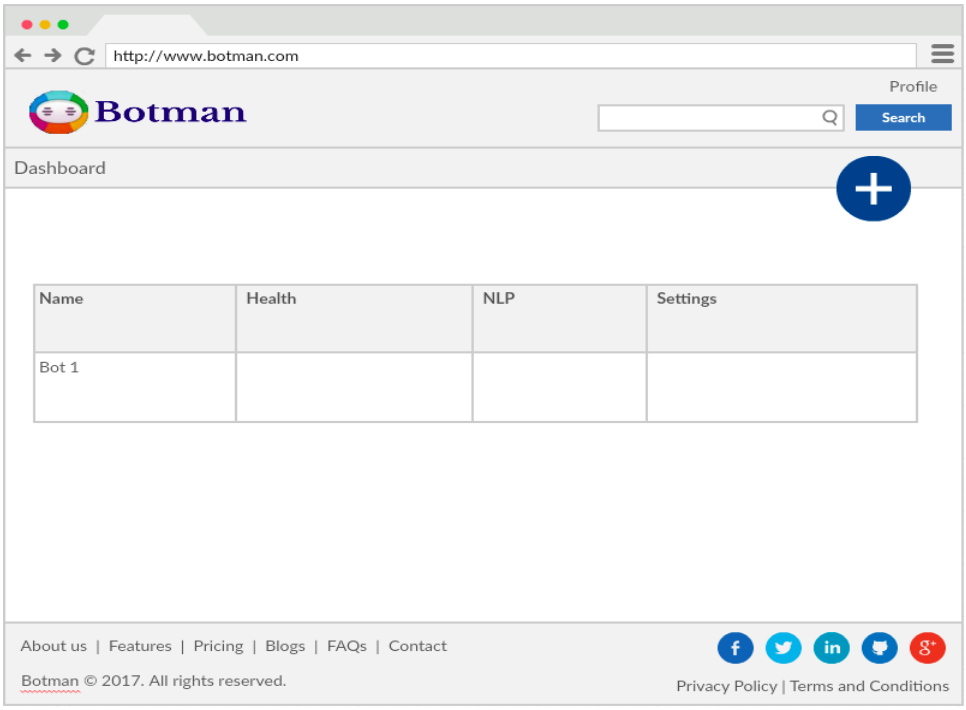
1. **Lading Page**



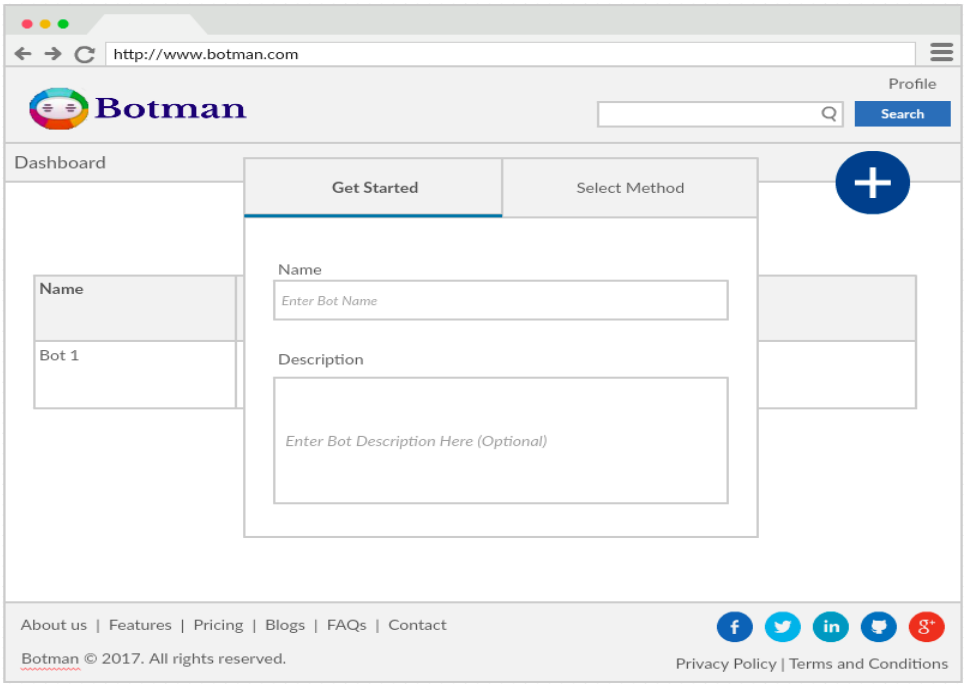
2. Login Page



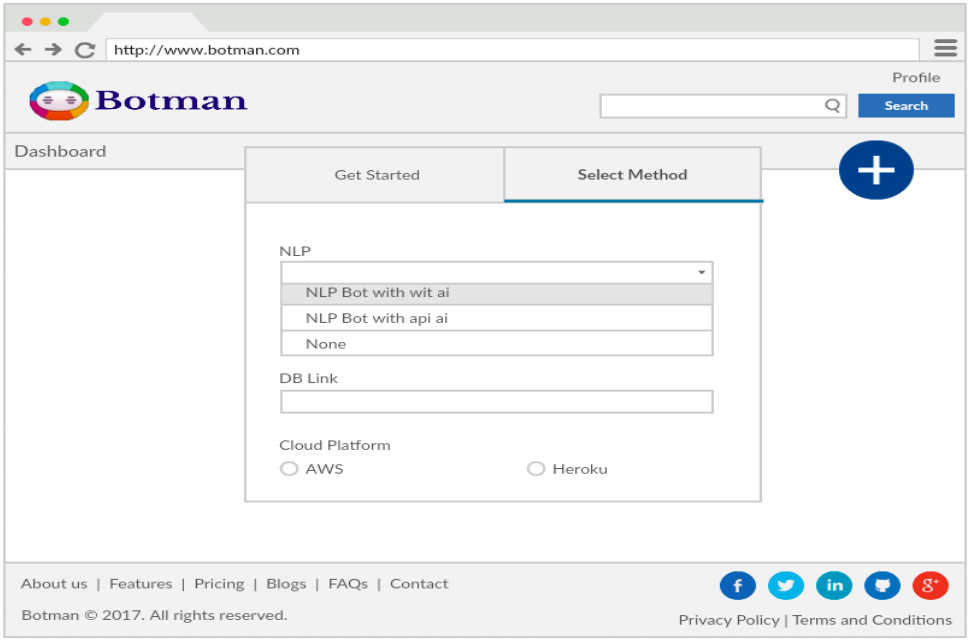
3. Dashboard



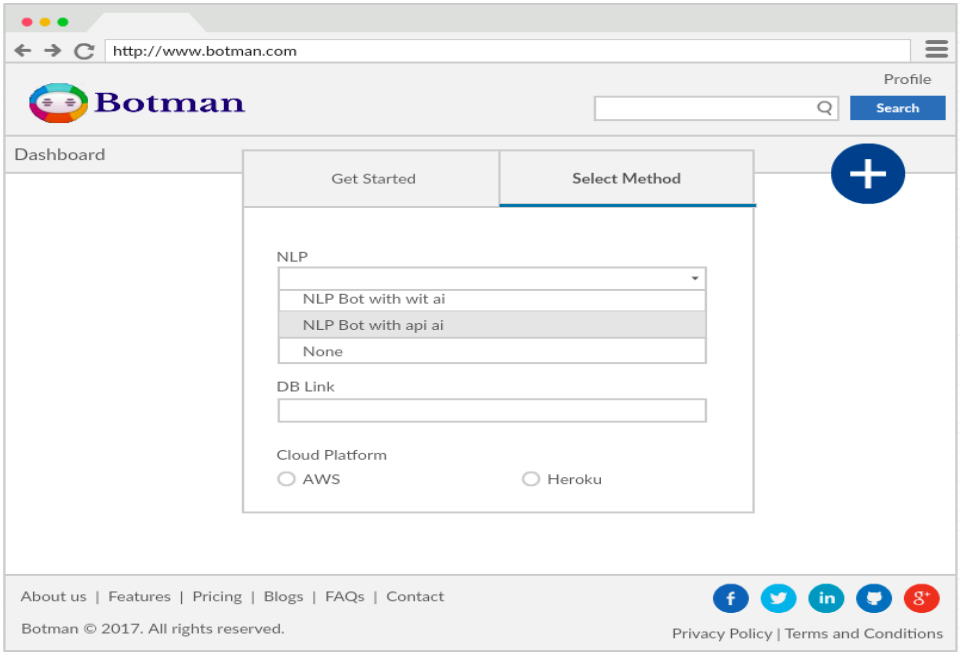
4. Get Started



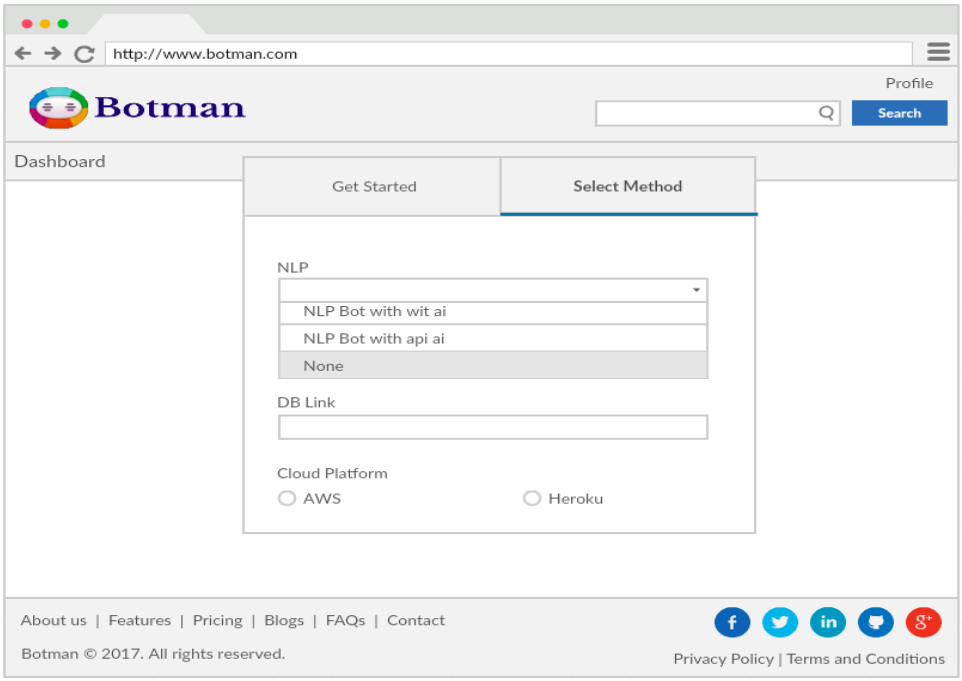
5. Get Method -1



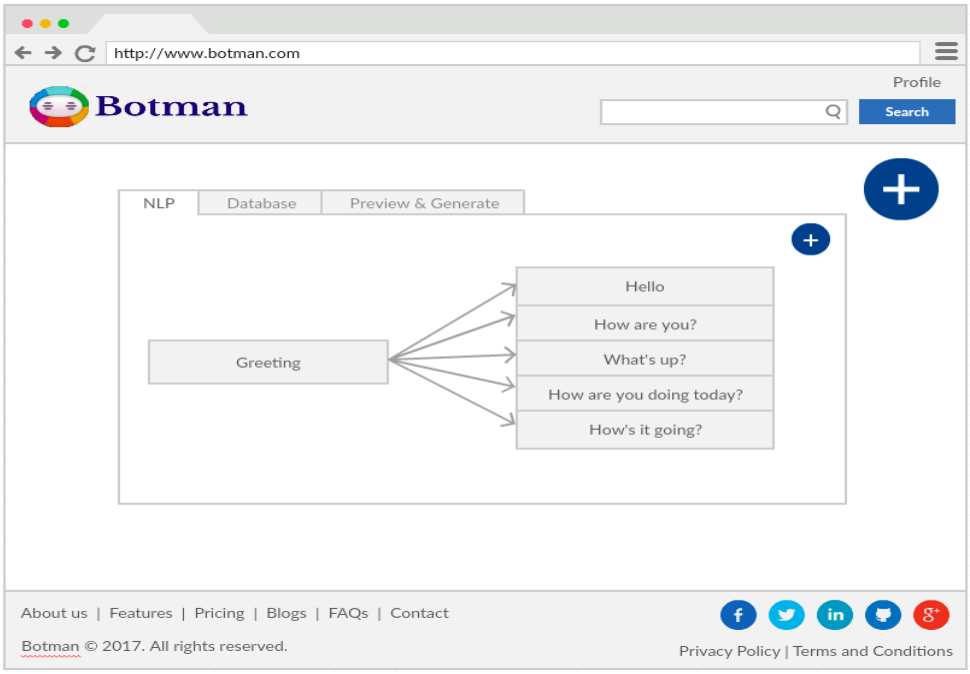
6. Get Method -2



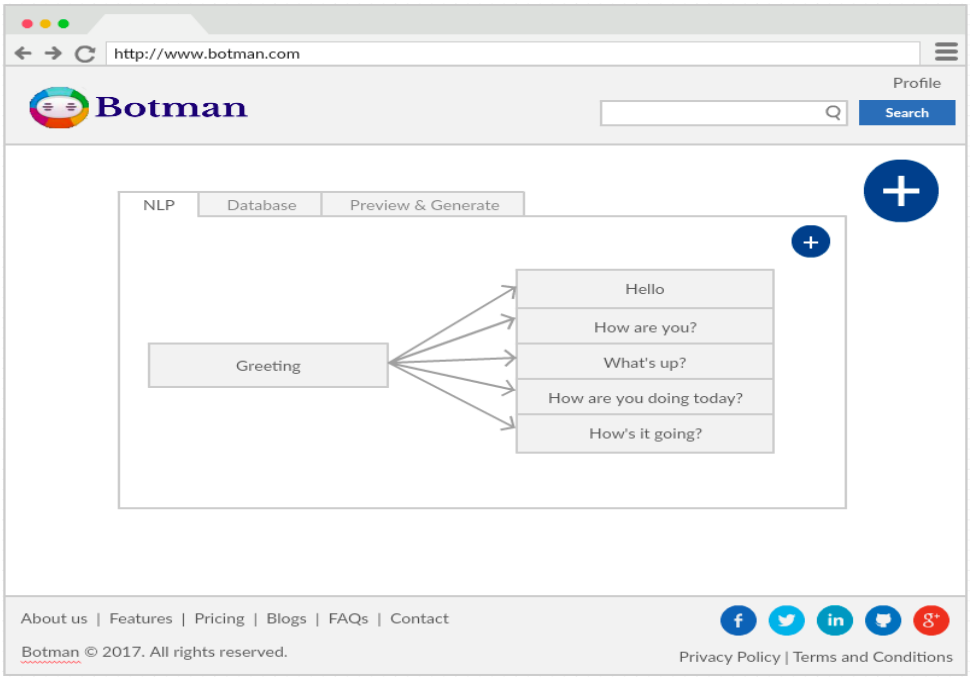
7. Get Method -3



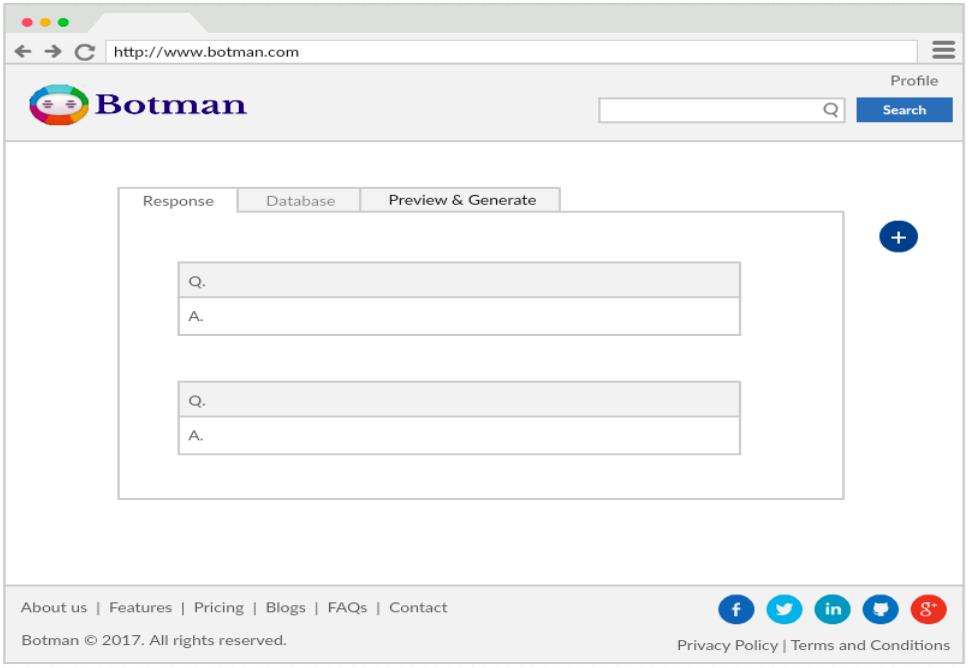
8. NLP -1


http://www.botman.com
Botman
Profile
Search
NLP
Database
Preview & Generate
Greeting
Hello
How are you?
What's up?
How are you doing today?
How's it going?
About us | Features | Pricing | Blogs | FAQs | Contact
Botman © 2017. All rights reserved.
Privacy Policy | Terms and Conditions

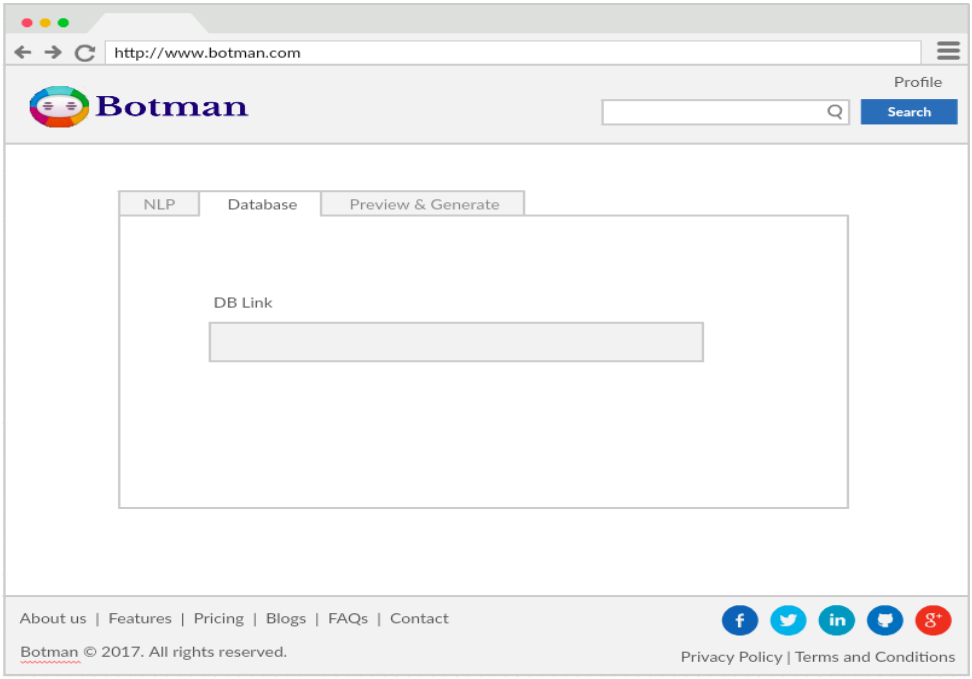
9. NLP -2


http://www.botman.com
Botman
Profile
Search
NLP
Database
Preview & Generate
Greeting
Hello
How are you?
What's up?
How are you doing today?
How's it going?
About us | Features | Pricing | Blogs | FAQs | Contact
Botman © 2017. All rights reserved.
Privacy Policy | Terms and Conditions

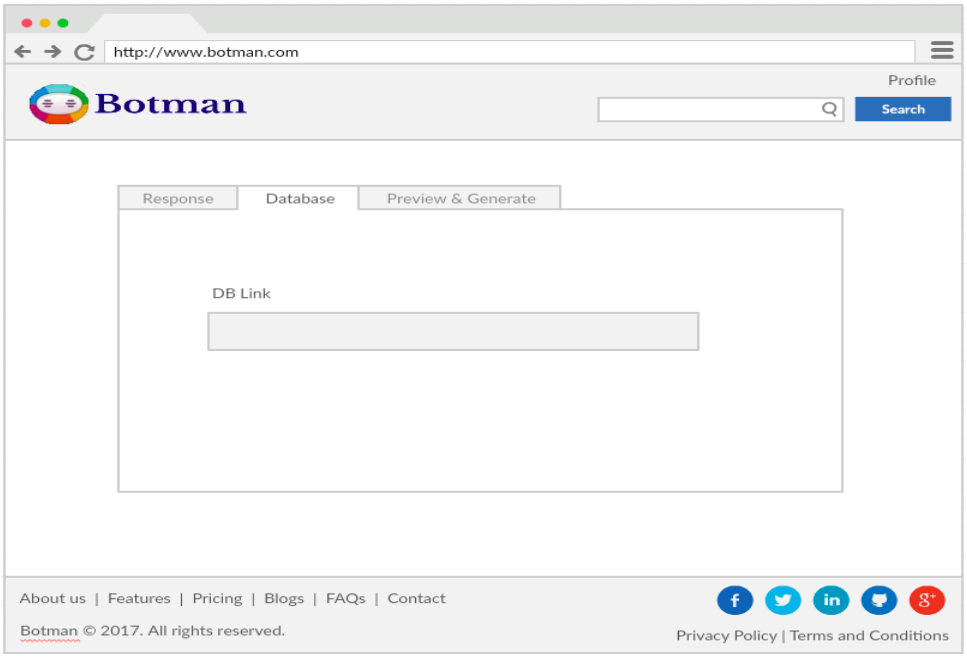
10. Resp



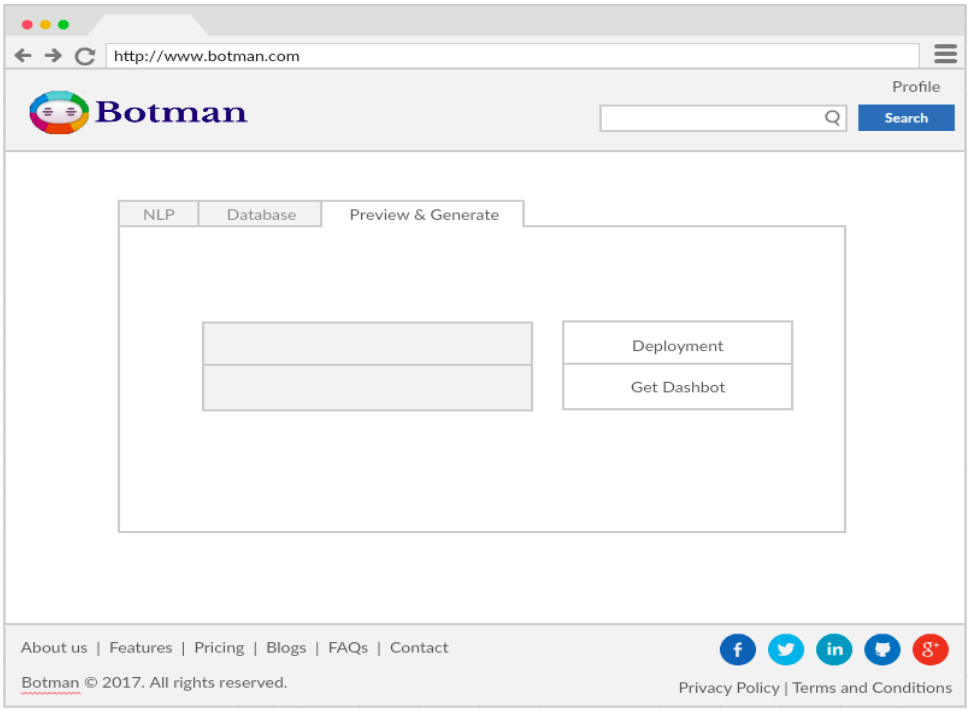
11. DB -1



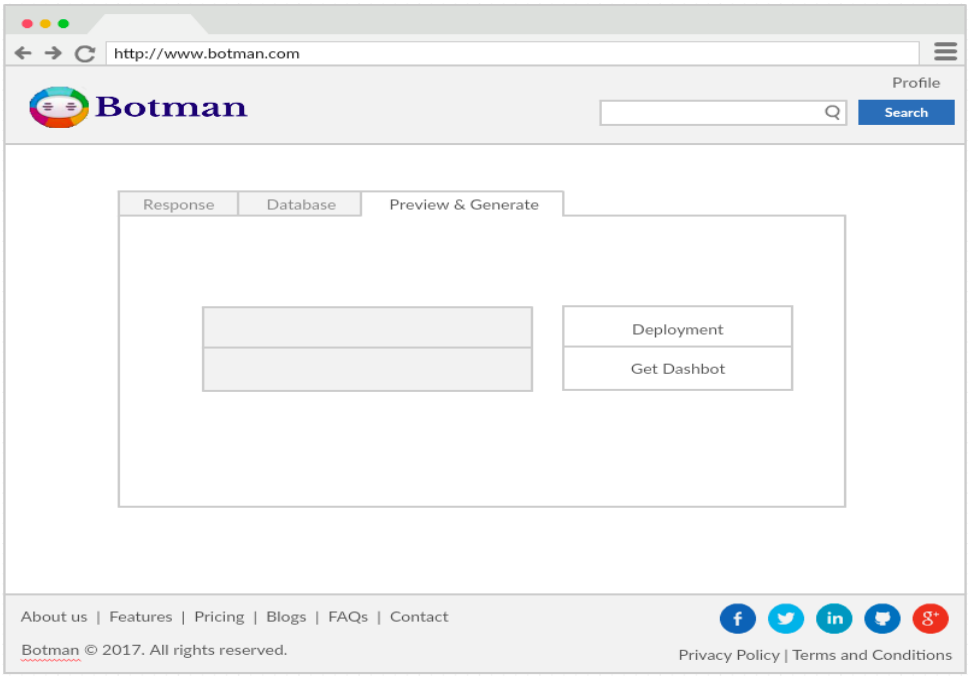
12. DB -2



13. Preview -1

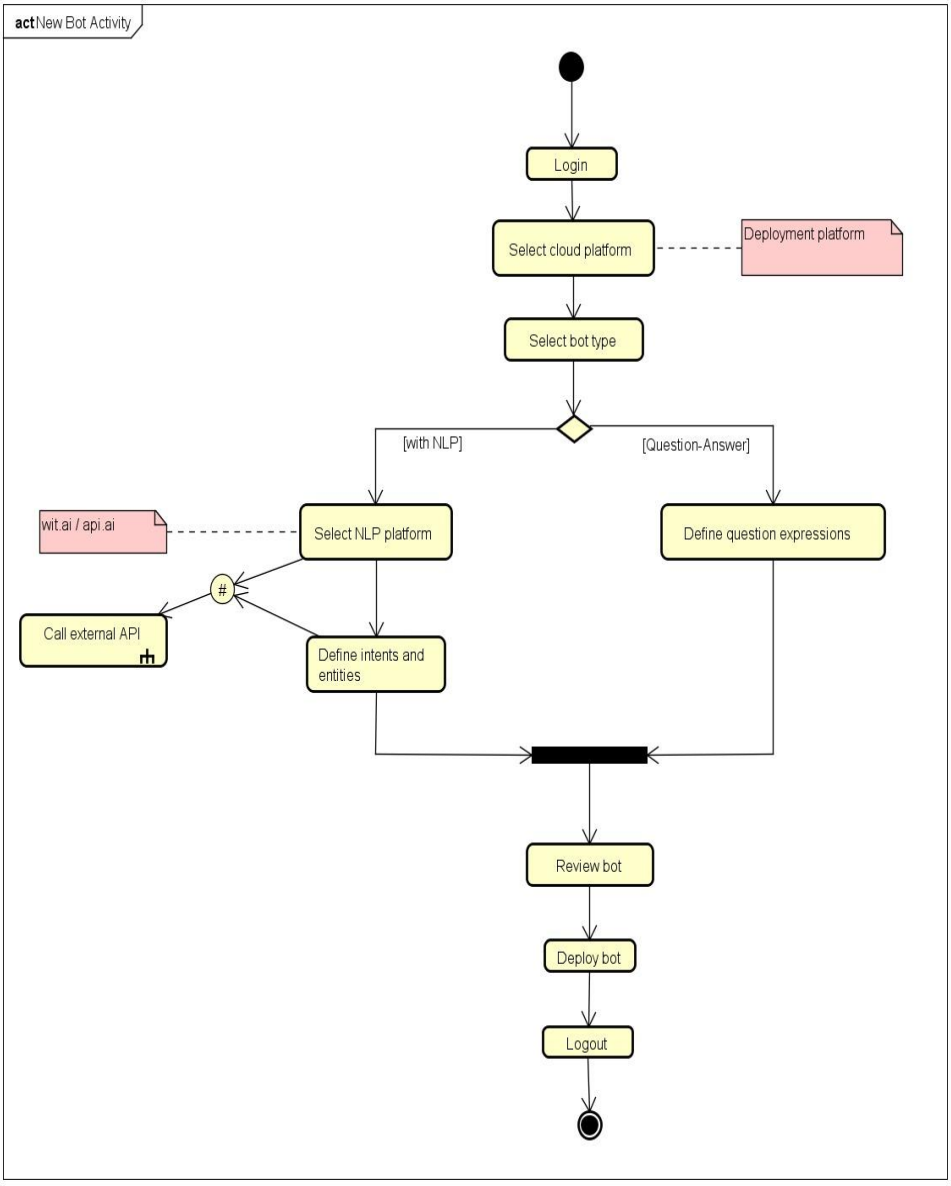


14. Preview -2

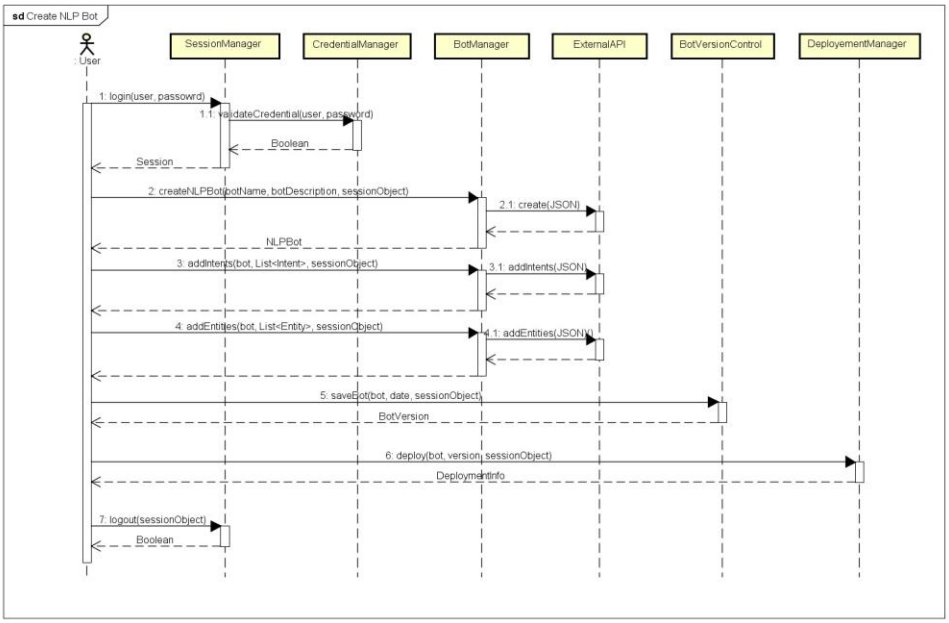

http://www.botman.com
Botman
Profile
Search
Response
Database
Preview & Generate
Deployment
Get Dashbot
About us | Features | Pricing | Blogs | FAQs | Contact
Botman © 2017. All rights reserved.
Privacy Policy | Terms and Conditions

UML Diagrams:

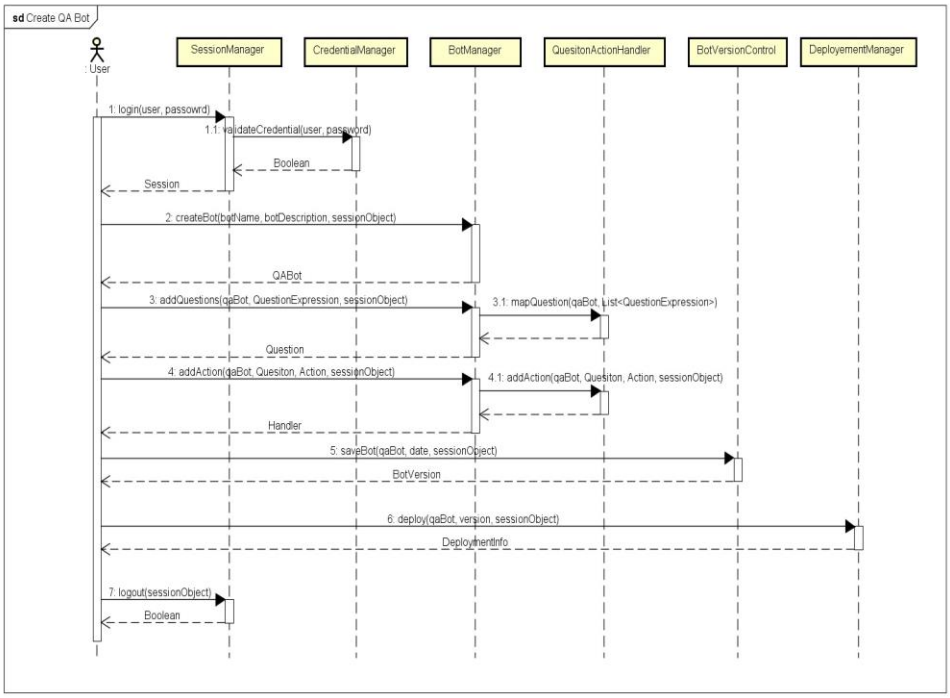
1. Sequence Diagram - 1



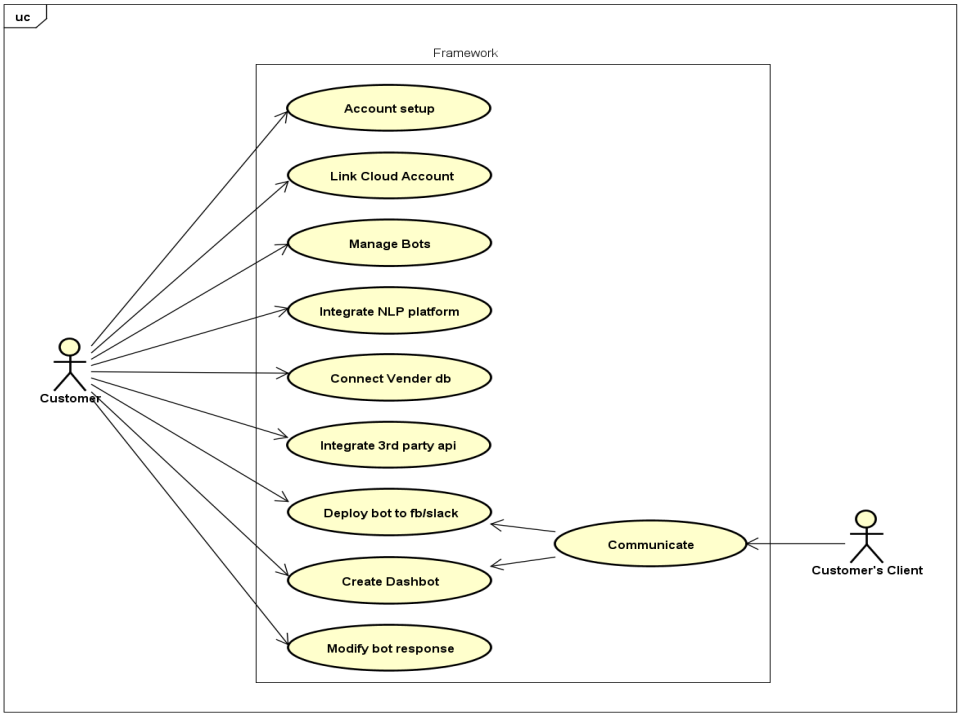
2. Sequence Diagram - 2



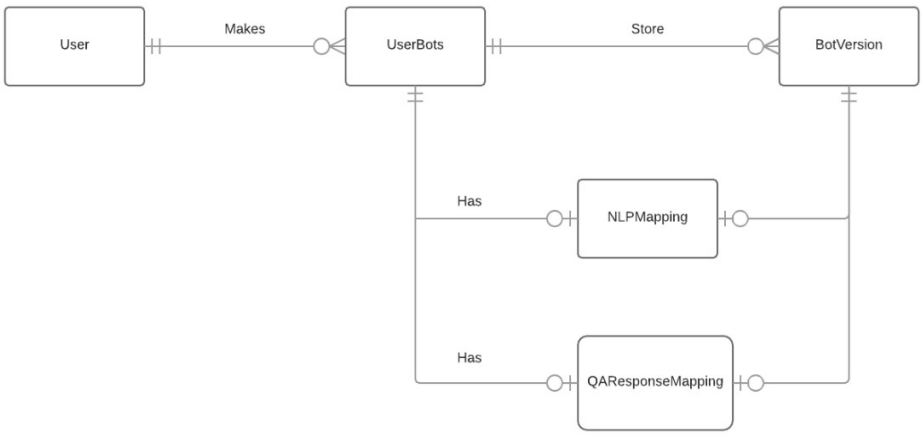
3. Sequence Diagram - 3



4. Use case diagram



5. ER Diagram



Chapter 7. QA, Performance, Deployment Plan

Testing Process Overview:

Software testing is one of the most important processes in any software development project. The testing includes validation and verifications both. Validation by definition is nothing but checking if the developed software complies with all of the standards and checking if all the user stories are met perfectly. The main purpose of testing is to uncover as many bugs possible in the initial stage itself.

The scope of testing is not only to uncover the bugs but also to make legit documents and reports from time to time. This helps in maintaining the data for future reuse. The goal is to make sure that every bug and error that is discovered during the development process should be documented to reduce the efforts.

Testing is a process that starts when we start writing the first line of code or rather before even that. We are going to follow a Test Driven Approach for development. Our development will be driven by the Unit Tests. A JavaScript framework for writing unit tests will be used called the Chai and Mocha JavaScript.

We will be testing our application keeping following aspects in mind,

1. Unit Testing:

The aspect will test individual units of the code. Intuitively one can assume a unit test as the smallest testable part of any application. We will be using unit testing for API testing and functional testing. We will try to incorporate these aspects in standalone test scripts.

2. Performance Testing:

Performance is a very important criterion of our application to test its ability to show extensive results. This also helps us to test whether our app yields appropriate results in stress or extreme input. High-performance results in higher sales and profit thus performance testing becomes very important part of our application. We will test using JMeter. The mail aim will be to test the application for multiple users and for multiple concurrent connections. The performance will be tabulated and studied in different graphs.

A simple format of our test case will be as follows,

S. No	Test Case	Expected Result	Test Output
1	Validating JavaScript included at the bottom of HTML	JavaScript should be included at the bottom of the page	Pending
2	Validating CSS files included at the top of the file	CSS should be included at the top of the HTML page	Pending
3	Validating the DNS connections.	Limited DNS connections for the app.	Pending
4	Checking if the app crashes for certain number of users	App should not crash for at least 5k concurrent users.	Pending
5	Checking if the app crashes for certain concurrent connections.	App should not crash for at least 5k concurrent connections	Pending

3. Decision Coverage Testing:

This testing will make sure to test that for any code, the decision will yield in tangible outputs. In any branch of code the true/false result should result in a separate logic. This result helps in validating all the branches in the code to make sure that no branch will ever lead any abnormal behavior. We will divide the code in flow charts and eventually create decision tables and tabulate each branch result.

4. Automation Testing:

Basically, software test automation means automating all the processes and tasks using tools instead of us doing them manually. By definition, automation testing means testing existing software by using another software, which is developed for the sole purpose of testing. Our primary objective in using the Automation Testing will be to reduce the time that we will invest while testing all the APIs when a new addition is done. We will obviously want to test everything when a new API is developed or a new git commit is done. We will develop some automation Java Scripts that will check the sanity of the application after a new change has been pushed or a major commit has been done.

Automation Testing Objective:

The main purpose of using Automation Performance Testing is to conclude on an overall test coverage to detect any possible bugs or errors that might have been introduced.

Our focus will be,

1. Testing API sanity

2. Testing Database Sanity

3. Testing functional sanity

Following is a simple overview of the testing process that we will follow,

1. We will plan our requirements around the test cases:

We will first try to understand the basic scenarios and know how test cases can be devised; this approach is useful while developing the Unit Tests. The things that will come under scrutiny here are, test exit Criteria, Contingency plans, expected output and test results, pre-condition and post-conditions.

2. Design the test cases which fit the use requirement:

The primary focus will be on developing Unit tests around the requirements using simple JavaScript modules like Chai and Mocha.

3. Implement Black box and White Box testing approaches:

We will implement both the black box and white box testing for our application.

4. Analyze and document test results

5. Reuse the rest results.

Test Component Items

1. UI functionalities on web platform
2. Performance Testing
3. Statistics of each bot owner
4. Check the login credentials testing

5. Check Bot AI testing

Software Requirements

1. Development Tool : Web Storm/ Atom/ Eclipse
2. Bug Tracking Tool : BugZilla
3. Project Management Tool: Kanban
4. API Testing: Chai/ Mocha/ Postman
5. Automation Testing: Chai/ Mocha
6. Performance Testing: JMETER

Test Cases Sample Table Example

SR No	Test Case Description	Expected Output	Status
1	User Registration	Successful with github login and metadata grab should also be done	Planned
2	Log In	No unauthorized entry should be there	Planned
3	Use of multi AI platforms	Template should be according to AI platform chosen	Planned
4	Deployment	Bot should be deployable on different platforms	Planned
5	Database Sanity	Database should be sane and bot code should be tangible	Planned
6	Load Testing	Concurrent users and concurrent connections	Planned

Chapter 8. Implementation Plan and Progress

Environment setup

Action	Tools/IDE	Assigned to	Status
Setting up MEAN stack	MongoDB, ExpressJS, AngularJS/ReactJS, Nodejs	Aditi, Nachiket	Done
Setting up IDE	Webstorm	Aditi, Nachiket	Done
Setting up Cloud Logins	Putty	Abhishek, Sushant	Done
Setting up NLP platform accounts	Api.ai, Wit.ai, Watson	Abhishek, Nachiket, Sushant	Done
Cloud Environment Setup	Amazon AWS/ IBM Bluemix	Abhishek, Sushant	Planned

Development and Testing tools

Tools/IDEs	Status
Webstorm	Done
Postman	Done
Amazon AWS/ IBM Bluemix	Planned
Apache JMeter	Planned

Code Samples

Implementation	Tool/IDE/Platform	Assigned to	Status
Implementing sample MEAN stack project	MongoDB, Express, AngularJS/ReactJS, Nodejs	Aditi, Nachiket	Done
Connecting bots to messaging channels like Facebook, Slack.	Facebook messaging API, Slack bot API	Abhishek, Sushant	Done
Deploying bots to cloud platforms	Amazon AWS/ IBM Bluemix	Abhishek, Sushant	Done
Experimenting with NLP services to link Chabot.	Api.ai, Wit.ai and Watson	Abhishek, Aditi, Sushant, Nachiket	Done
Creating a working model of a Dashbot.	MongoDB, ExpressJS, AngularJS/ReactJS, Nodejs	Aditi, Nachiket	Planned

Project feature Implementation

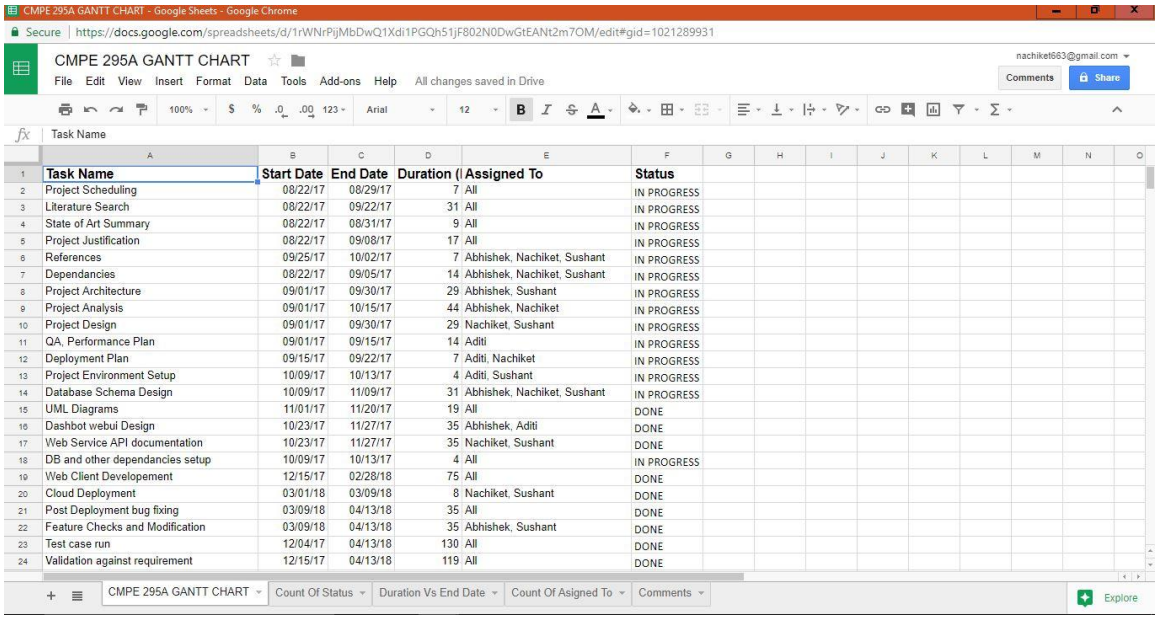
Features	Status
Creating a centralized management portal for managing Chatbots for all users.	Planned
Implementing login functionality to maintain user accounts.	Done
Maintaining a history to the various Chatbot versions developed by the user.	Planned
Developing a generic bot template in NodeJS which would include the basic housekeeping code required for the health status checkup and linking to third-party APIs.	Planned
Establishing the connection between the central management portal and the cloud service providers to enable one-click deployment of the developed bots.	Planned
Implementing a generic UI chat template with data traffic control logic for the dashbots. The user can easily deploy this interface onto their website and control its behavior from the framework.	Planned
Providing an easy to plug interface where a developer can experiment with different NPL services supported by the framework.	Planned
Providing an easy to plug interface where a developer can link database and the bot.	Planned
Implementing an interface that would enable the developers to train the chosen NLP model by creating a mapping between the likely questions that a user would ask and intents that the bot should understand.	Planned

Performance testing and quality assurance

Performance Factor	Status
Performing load test on the framework using JMeter to determine the number of simultaneous users that the framework can support and scale up the infrastructure according to the demand.	Planned
Writing unit test case.	Planned
Writing database cleanup and optimization scripts to improve framework performance in user account management.	Planned
Implementing an admin dashboard which would provide the summary of the availability status of all the services that are being utilized by the users for their Chatbots.	Planned

Chapter 9. Project Schedule

Gantt Chart



CMPE 295A GANTT CHART - Google Sheets - Google Chrome

Secure | https://docs.google.com/spreadsheets/d/1rWNrPijMbDwQ1Xdi1PGQh51jF802N0DwGtEANt2m7OM/edit#gid=1021289931

CMPE 295A GANTT CHART

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

nachiket663@gmail.com

Comments Share

Task Name

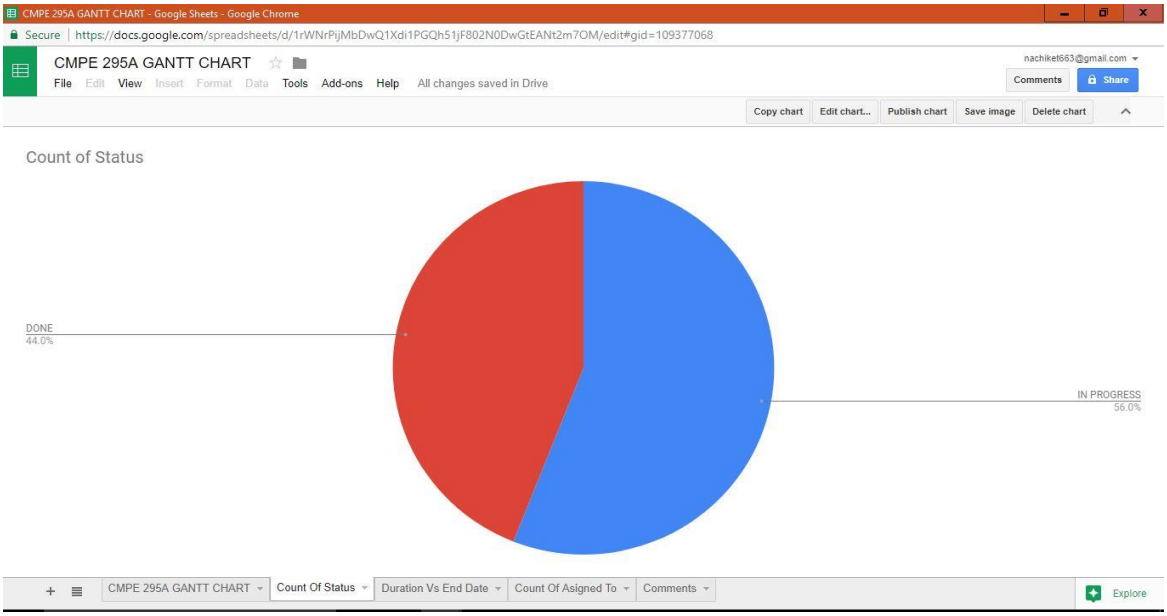
	Task Name	Start Date	End Date	Duration (d	Assigned To	Status
2	Project Scheduling	08/22/17	08/29/17	7	All	IN PROGRESS
3	Literature Search	08/22/17	09/22/17	31	All	IN PROGRESS
4	State of Art Summary	08/22/17	08/31/17	9	All	IN PROGRESS
5	Project Justification	08/22/17	09/08/17	17	All	IN PROGRESS
6	References	09/25/17	10/02/17	7	Abhishek, Nachiket, Sushant	IN PROGRESS
7	Dependancies	08/22/17	09/05/17	14	Abhishek, Nachiket, Sushant	IN PROGRESS
8	Project Architecture	09/01/17	09/30/17	29	Abhishek, Sushant	IN PROGRESS
9	Project Analysis	09/01/17	10/15/17	44	Abhishek, Nachiket	IN PROGRESS
10	Project Design	09/01/17	09/30/17	29	Nachiket, Sushant	IN PROGRESS
11	QA, Performance Plan	09/01/17	09/15/17	14	Aditi	IN PROGRESS
12	Deployment Plan	09/15/17	09/22/17	7	Aditi, Nachiket	IN PROGRESS
13	Project Environment Setup	10/09/17	10/13/17	4	Aditi, Sushant	IN PROGRESS
14	Database Schema Design	10/09/17	11/09/17	31	Abhishek, Nachiket, Sushant	IN PROGRESS
15	UML Diagrams	11/01/17	11/20/17	19	All	DONE
16	Dashbot webui Design	10/23/17	11/27/17	35	Abhishek, Aditi	DONE
17	Web Service API documentation	10/23/17	11/27/17	35	Nachiket, Sushant	DONE
18	DB and other dependancies setup	10/09/17	10/13/17	4	All	IN PROGRESS
19	Web Client Developement	12/15/17	02/28/18	75	All	DONE
20	Cloud Deployment	03/01/18	03/09/18	8	Nachiket, Sushant	DONE
21	Post Deployment bug fixing	03/09/18	04/13/18	35	All	DONE
22	Feature Checks and Modification	03/09/18	04/13/18	35	Abhishek, Sushant	DONE
23	Test case run	12/04/17	04/13/18	130	All	DONE
24	Validation against requirement	12/15/17	04/13/18	119	All	DONE

CMPE 295A GANTT CHART | Count Of Status | Duration Vs End Date | Count Of Asigned To | Comments

Explore

Fig: Gantt chart

Count of Status



CMPE 295A GANTT CHART - Google Sheets - Google Chrome

Secure | https://docs.google.com/spreadsheets/d/1rWNrPijMbDwQ1Xdi1PGQh51jF802N0DwGtEANt2m7OM/edit#gid=109377068

CMPE 295A GANTT CHART

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

nachiket663@gmail.com

Comments Share

Copy chart | Edit chart... | Publish chart | Save image | Delete chart

Count of Status

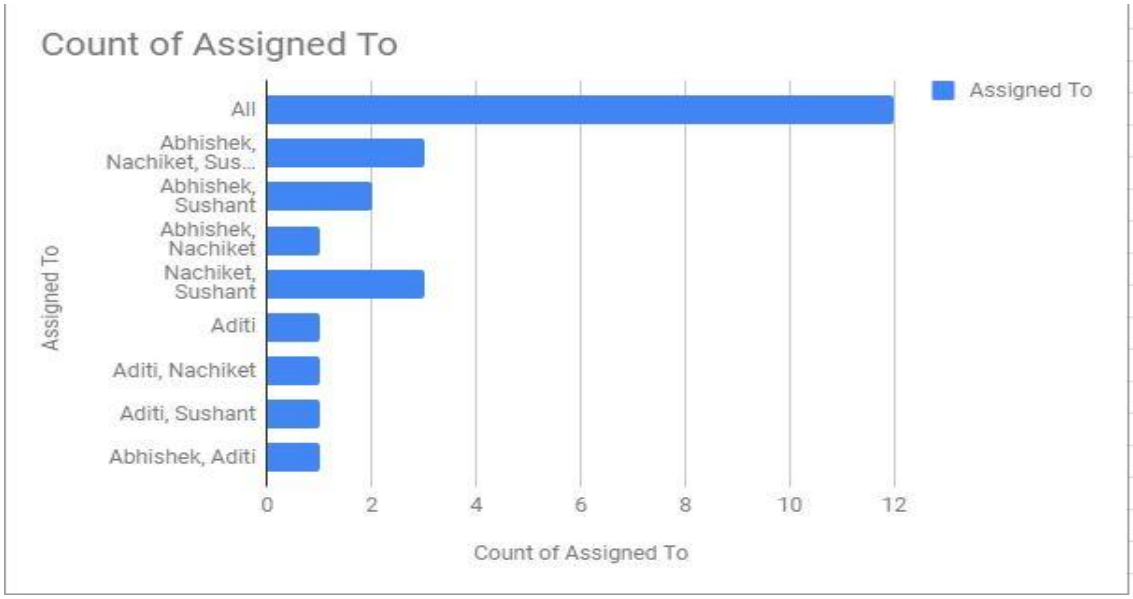
DONE
44.0%

IN PROGRESS
56.0%

CMPE 295A GANTT CHART | Count Of Status | Duration Vs End Date | Count Of Asigned To | Comments

Explore

Count of assigned to



Pert Chart

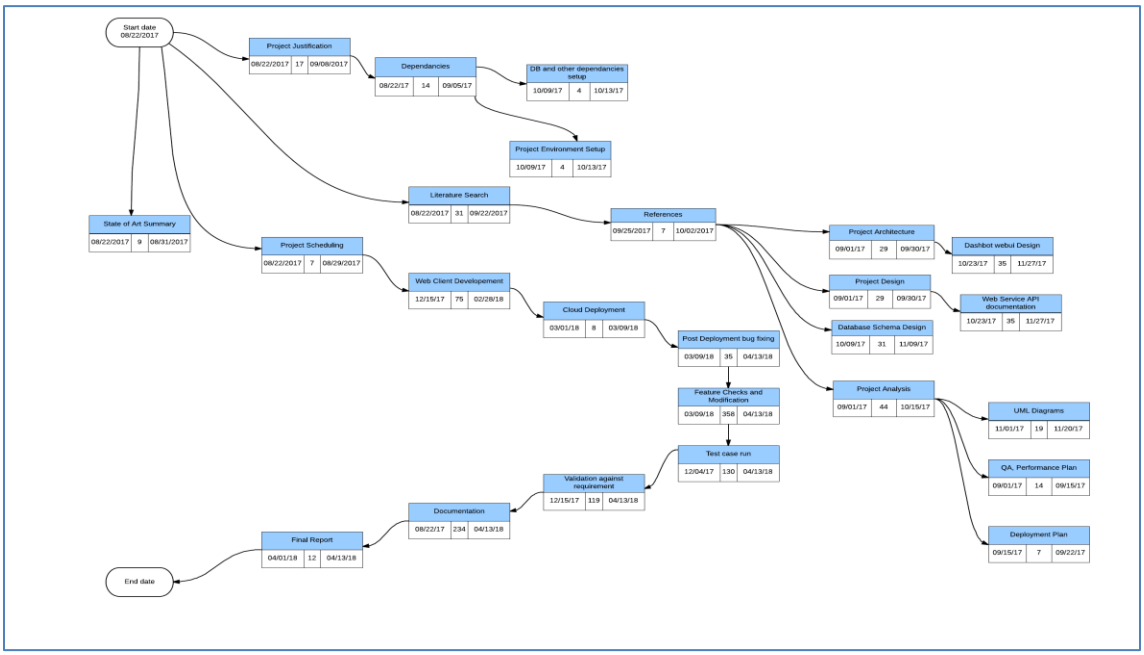


Fig 9.2 Pert Chart