BoilerBot

Team 1: Aditya Dhingra, Rahul Pai, Eehita Parameswaran,
Kush Rustagi, Devaunsh Sambhav, Yash Shiroya

March 6th, 2017

Sprint 1 Retrospective

Contents

1 What went well?

1.1 UserStory #1: As a first-time user, I would like the bot to introduce itself and provide me a set of instructions.

Status: Completed.

1.1.1 Check for a first time user:

1. Successfully able to differentiate between the returning (already existing) and a new user.
2. Our bot is able to reply to a new user with a descriptive / instructional welcome message and a returning user with a normal welcome message.
3. The reply message to a new user contains the description of Boilerbot and instructions on how to use it.

1.1.2 Recast training:

1. Recast tries to understand the intent of the message sent by the user.
2. Based on the understanding of recast, the bot replies with an appropriate message. More recast intent training is needed in order to get the intent accurately.

1.2 UserStory #2: As a user, I would like the bot to exchange pleasantries with me.

Status: Completed.

1.2.1 Create user's intents:

Added possible user intents to Recast.ai to recognize different possible intents of a user.

1.2.2 Understand user's intent:

Successfully parsed the user's message and got recast.ai to return a reply according to the user's intent.

1.2.3 Create greeting examples:

Created a set of messages to return to the user when the user sends a greeting or a goodbye to the bot.

1.3 UserStory #3: As a user, I would like to interact with the bot.

Status: Completed.

1.3.1 Node.js setup:

Created a project that includes the chatbot and Natural Language Processing module.

1.3.2 Heroku setup:

A Heroku container instance is required to run our chatbot server and keep it continuously responsive. The Heroku containers also allocates dynos which are set up so that every request from the server to the messenger bot causes the dynos to be restarted from heroku-sleep. Thus, the setup and required services are arranged correctly.

1.3.3 Database setup:

We setup a NoSQL mongoDB database which reflects our blueprint schema. The model is then built for our server. Finally a mongoDB service instance was attached to the Heroku container so that the data remains synchronized over devices and is always live.

1.3.4 Facebook Messenger Bot Setup:

We were successfully able to set up the webhook.

1.3.5 Recast Setup:

We successfully set up and trained intents using recast.ai in order to make our bot reply in context.

1.4 UserStory #4: As a first-time user, I would like the bot to repeat instructions when asking for help.

Status: Completed.

1.4.1 Acceptance Criteria

1. Given that the Recast.ai is setup, when the user asks for help, then the Recast.ai should be able to understand that the user is asking for help.

2. Given that the bot is able to communicate with the user, then a set of messages should be send to the user giving detailed information about the bot's functionality and usage.

3. Given that the help function is set up, when the user asks for help, then the chatbot should call the help function and render the set of messages that gives information about the usage and functionality of the chatbot at any point of time during the conversation.

1.4.2 Validation

1. We were able to understand users intent of asking for help by making use of the Natural Language Processing API - Recast.ai.

2. We were able to successfully identify user intent of asking for help and send a set of messages that will help the user to continue using the application when the user is confused about or stuck on some particular thing.

3. The help function is created in the codebase that is called when the user intent is 'help' and renders the set of information messages. The user can ask the bot for the set of instructions at any point of time.

1.5 UserStory #5: As a user, I would like to facilitate a forum-like environment on our Facebook page.

Status: Completed.

1.5.1 Acceptance Criteria

Given that the user is having any trouble in the usage of the bot, then the user should be able to post his questions on the Boilerbot's Facebook profile page.

1.5.2 Validation

We have successfully been able to facilitate a forum-like environment on Boilerbot's profile page. The users can now posts questions and concerns on the Boilerbot's facebook profile page.

1.6 UserStory #6: As a user, I want to be able to log into Facebook to use BoilerBot.

Status: Completed.

1.6.1 Acceptance Criteria

Given that the user wants to use the bot, then the user should be able login into the facebook account using his/her own credentials and be able to use the bot.

1.6.2 Validation

We were able to successfully allow all the facebook users to log in to their respective facebook account and like our page to access our application. This feature was largely handled by facebook platform itself.

1.7 UserStory #7: As a user, I want the bot to be platform-independent.

Status: Completed.

1.7.1 Acceptance Criteria

1. Given that the user wants to use the bot as a mobile application, then the user should be able to successfully access all the features of the bot on that particular mobile platform.

2. Given that the user wants to use the bot as a web application, then the user should be able to successfully access all the features of the bot on the web platform.

1.7.2 Validation

1. The bot was pushed on the messenger platform and as such, the user can switch to a mobile device without breaking the flow of the conversation or losing any functionality.

2. The bot was pushed on the messenger platform and as such, the user can switch to a desktop device without breaking the flow of the conversation or losing any functionality.

1.8 UserStory #8: As a user, I expect the bot to act in character even when asked about something that the bot doesn't yet understand.

Status: Completed.

1.8.1 Acceptance Criteria

1. Given that the user sends unexpected attachment or symbols to the chatbot, when detected by the recast.ai API, then the chatbot should respond with one of the predetermined messages.

2. Given that the user sends a text that is irrelevant in context with the chatbot, when detected by the recast.ai API, then the chatbot should handle it elegantly.

1.8.2 Validation

1. Since we are able to successfully detect whether or not the input is a text, we are correctly able to differentiate unexpected symbols from text.

2. Since we have programmed our NLP API to detect certain intents, if none is returned, we have a default string.

1.9 UserStory #21: As a user, I want the bot to be able to provide me with the location of dining courts at Purdue.

Status: Completed.

1.9.1 Acceptance Criteria

1. Given that Recast.ai is set up, when a user sends a message, then it would be sent to the API for parsing.

2. Given that we have established a connection with the Dining Court API, when a user requests location, then the interactive cards are displayed.

3. Given that our database is setup, when a user requests the location of a dining court, then requested location can be added to the card accordingly.

1.9.2 Validation

1. Since we are able to successfully understand a user's message, we were able to send it to our Natural Language Processing API for parsing.

2. The bot was integrated with Purdue's Dining Court API such that location requests would make the cards show up.

3. Since the bot was set up with our database, the location of the dining court is retrieved and displayed on the interactive card.

2 What did not go well?

- We were able to implement all of the planned and proposed user stories. However, setting up the project required us to work together, but instead we tried completing our tasks individually. This made it hard for us to understand how to connect all the components. So, we lost some time in the setup but we got back on track after meeting up regularly and understanding how to connect all the components.

- Setting up the project took us more time than expected and we found it hard to integrate all the individual components of our project.

- It was a challenge for the developers to understand the functionality of the tasks done by other team members as all the team members are not extremely comfortable with the full stack development. For example, some of the developers did not have the knowledge about NoSQL or functionality of the Natural Language Processing API, which created quite a bit of confusion amongst us. In conclusion, ramping up all team members to ensure that everyone knew about almost every component of the project took time.

3 What will we do better in the next sprint?

- Initially, we used to work individually and this resulted in several doubts, regarding directory structure etc. So, we fixed this by working on setting up the project together so that all of us followed a similar approach and we intend to continue to work together to ensure that we stay on the same page.
- Every developer will give some time on learning the complete stack (for instance, NoSQL, Node.js dependencies, working of the Natural Language Processing API) i.e. technologies which they aren't comfortable with. In that manner it would be easy to visualise the flow of our application. It will also allows other developers to do a Quality Assurance test for a task done by another team member.