BoilerBot

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

April 10th, 2017

Sprint 2 Retrospective

Contents

1 What went well?

1.1 UserStory #9: As a user, I would like to adjust the settings across the conversation.

Status: Completed.

1.1.1 Acceptance Criteria

1. Given that the user is able to converse with the bot, when the user wants to see the history of the conversation, then the user should be able to see the log of all the conversations held till date.

2. Using Mocha-Chai testing framework, we intend to test the stored procedures to ensure that each conversation is being mapped to the appropriate user by checking that it's a one-to-one mapping.

1.1.2 Validation

1. We successfully checked that when the user makes certain settings in the conversational state of the bot, then the settings do not eradicate until the user changes it.

2. We tested the stored procedures using automated frameworks and ensured that user settings are being mapped correctly by conversation.

1.2 UserStory #10: As a user, I expect the bot to keep a log of all the messages across the conversation.

Status: Completed.

1.2.1 Acceptance Criteria

1. Given that the user is able to converse with the bot, when the user makes certain settings to the default settings in the conversational state of the bot, then those settings should not eradicate until the user changes it.

2. Using Mocha-Chai testing framework, we intend to test the stored procedures to ensure that each user setting is being mapped to the appropriate user by checking that it's a one-to-one mapping.

1.2.2 Validation

1. We successfully checked that the user has permission to view his entire conversation history with the chatbot.

2. We tested the stored procedures using automated frameworks and ensured that user messages are being mapped correctly by conversation.

1.3 UserStory #11: As a first-time user, I would like to have a tutorial on how the bot works.

Status: Completed.

1.3.1 Acceptance Criteria

1. Given that the 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, we should create a set of messages that gives the user detailed information about the bot's functionality and usage.
3. Given that the tutorial function is set up, when the user asks for help, then the chatbot should call the tutorial function and we will test this by rendering the set of messages that give information about the usage and functionality of the chatbot.
4. Test feature manually by sending an input of 'help' or and checking the output against the predefined output. Another test case would be when a new user tries to use Boilerbot, a tutorial should show up for the user.

1.3.2 Validation

1. Since we are able to parse the help intent successfully, we can provide the user with the help they need.
2. Since we provide them with help successfully, we inherently also give enough detail to continue use.
3. In order to provide the help we were successfully able to create a subroutine to handle.

1.4 UserStory #12: As a user, I would like the bot to reply quickly.

Status: Completed.

1.4.1 Acceptance Criteria

1. Given that the user has been waiting for a reply for a while, we would send an intermediary message i.e. typing dots to ensure that the user knows that the bot will reply soon.
2. Test feature by sending multiple messages to BoilerBot and ensuring that our bot is 'typing' such that the user knows that a response will be received soon.

1.4.2 Validation

1. When a user has been waiting for a reply for quite a while, typing dots are sent to the user which shows that the bot is taking some time to respond.
2. Since our model is quite fast after taking care of asynchronous problems, we don't actually need to send a 'typing' message to our user.

1.5 UserStory #13: As a user, I expect the bot to inform me about server downtime.

Status: Completed.

1.5.1 Acceptance Criteria

1. Given that the chatbot is functional, we have to create a function that would render a message to a user regarding when the server downtime is expected.
2. Given the database is functional, when we are planning to stop the server, a function should get all the user id's present so that we can notify all users of our chatbot about the server downtime.
3. Given that the function to notify about server downtime is implemented, when we are planning to stop the server, we'll test the response in Boilerbot and ensure that the function sends the message about downtime to all the users listed in our database.
4. Test using automated testing framework such that when the server is shut down by the developers for sometime, the tester automatically gets a notification about the downtime message.

1.5.2 Validation

1. Whenever server-downtime is expected, a message is sent to all active users.
2. Since we store the users in our database, we can simply iterate over them in order to effectively send a message to all active users.
3. We have found that since we edit the server frequently during version testing, we need to stop the server often so testing using the mocha-chai framework was quite useful.

1.6 UserStory #14: As a user, I expect the bot to be versatile.

Status: Completed.

1.6.1 Acceptance Criteria

1. Given that the chatbot is functional, we will using Natural Language Processing to ensure that Boilerbot is trained well to answer almost any query and it responds appropriately to unusual queries.

2. Test feature using mocha-chai framework and send varied messages(help, feedback etc.) to the bot to ensure that Boilerbot responds with an appropriate message everytime (tested against valid output)

1.6.2 Validation

1. We successfully trained our Natural Language Processing algorithm to understand varied intents and print out correct messages to the console window.

2. Sending varied messages like 'hi', 'bye', 'help', etc to Boilerbot always gives the user a different response and this was tested using our automated testing framework too.

1.7 UserStory #15: As a user, I would like to get information regarding the last crime committed at Purdue.

Status: Completed.

1.7.1 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 scraped all of the data from the PUPD page, we will be able to keep all of the saved crime statistics in the database.
3. Given that the user requests the latest crime statistic, access the latest crime statistic from the database.
4. Test feature by using Mocha-Chai testing framework and check whether or not we get information about the last crime committed at Purdue when the user sends request like 'last crime?'.

1.7.2 Validation

1. On sending a message to our bot about daily crime statistics, our Natural Language algorithm parsed it correctly and printed the correct intent to the console.
2. When our code scraped data from the purdue university police department webpage, we were able to successfully view the information in our database.
3. When user requested for latest or daily crime statistics, our bot is able to respond appropriately with several cards that provide relevant information.

1.8 UserStory #16: As a user, I would like to get monthly crime statistics for the current year at Purdue.

Status: Completed.

1.8.1 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 scraped all of the data from the PUPD page, we will be able to keep all of the saved crime statistics in the database.
3. Given that the user requests monthly crime statistics, accesses the monthly crime statistics from the database.
4. Test feature by using Mocha-Chai testing framework and check whether or not we get the monthly crime statistics for the current year at Purdue when the user sends request like 'monthly crime stats'.

1.8.2 Validation

1. On sending a message to our bot about monthly crime statistics, our Natural Language algorithm parsed it correctly and printed the correct intent to the console.
2. When our code scraped data about monthly crime statistics from the purdue university police department webpage, we were able to successfully view the information in our database.
3. When user requests for monthly crime statistics, our bot is able to respond appropriately with several cards that provide relevant information.

1.9 UserStory #17: As a user, I would like to know about any active warrants 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 scraped all of the data from the PUPD page, we will be able to keep all of the saved crime statistics in the database.
3. Given that the user requests active warrants, accesses the active warrants saved in the database.
4. Test feature by using Mocha-Chai testing framework and check whether or not we get correct information about the active warrants committed at Purdue when the user sends request like 'active warrants?'.

1.9.2 Validation

1. On sending a message to our bot about active warrants, our Natural Language algorithm parsed it correctly and printed the correct intent to the console.
2. When our code scraped data about active warrants from the purdue university police department webpage, we were able to successfully view the information in our database.
3. When user requests for active warrants statistics, our bot is able to respond appropriately with several cards that provide relevant information.

1.10 UserStory #18: As a user, I would like to get yearly crime statistics for the current year at Purdue.

Status: Completed.

1.10.1 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 scraped all of the data from the PUPD page, we will be able to keep all of the saved crime statistics in the database.
3. Given that the user requests the yearly crime statistics, accesses the yearly crime statistics from the database.
4. Test feature by using Mocha-Chai testing framework and check whether or not we get yearly crime statistics at Purdue when the user sends request like 'yearly crime stats?'.

1.10.2 Validation

1. On sending a message to our bot about yearly crime statistics, our Natural Language algorithm parsed it correctly and printed the correct intent to the console.
2. When our code scraped data about yearly crime statistics from the purdue university police department webpage, we were able to successfully view the information in our database.
3. When user requests for yearly crime statistics statistics, our bot is able to respond appropriately with a list that provides relevant information.

1.11 DeveloperStory #1: As a developer, I would like to come up with regular updates to the chatbot. Updates that include giving more relevant information based on user feedback.

Status: Completed.

1.11.1 Acceptance Criteria

1. Given that the user sent their recommendations, then use the stored user recommendations to analyze user likes and dislikes as a whole.
2. Given that Recast.ai is trained to understand the suggestion intent, the bot can send the appropriate message to the user accordingly.
3. Given that the bot is updated to a particular version, then all users receive updates immediately.
4. Test analytics manually to ensure that our algorithm is saving the feedback correctly to the database and we are able to come up with correct assumptions about user suggestions.

1.11.2 Validation

1. We have been able to store the user's recommendations and thus used them to analyze a user's likes, dislikes.
2. When user suggests anything to our bot, our Natural Language algorithm understands the intent and prints the correct message to the console window.
3. As soon as the bot gets updated with any feature, all users in the database get a message and this was tested using mocha-chai testing framework.

1.12 DeveloperStory #2: As a developer, I would like my bot to handle at least 100 users at once.

Status: Completed.

1.12.1 Acceptance Criteria

1. Given that multiple users are using the bot at the same time, the bot should be able to serve their individual requests concurrently and in a timely manner.
2. Given that multiple users send multiple queries/texts i.e. more than can just handling it concurrently, the bot should be able to efficiently queue tasks so as not to cause clashes between multiple users and their respective data.
3. Given that the server is running a specific version, when the new version of the server comes out, then check the version and any issues pertaining to it and upgrade to it accordingly.
4. Test feature by using automated testing framework of mocha-chai such that there are at least 100 'pseudo-users' who access/send Boilerbot a message/query at the same time.

1.12.2 Validation

1. We were successfully able to handle responding to multiple users fast and ensured there weren't any issues or time lag when multiple users used our chat bot.
2. We were able to successfully check that there were no issues when multiple users sent in multiple messages.
3. When a new version of the bot rolled out i.e. new features were included, all users of bot were sent a message at the same time.

1.13 DeveloperStory #3: As a developer, I would reduce the downtime for maintenance to 2 hours per month.

Status: Completed.

1.13.1 Acceptance Criteria

1. Given that the user base increases, then we would like to have a fast maintenance pattern so as not to hinder user experience for very long.
2. Given that the user is providing the feedback, then analyze the feedback and implement the functionality in a 2 hour time span once a month.
3. Given that the some new functionality is implemented and is in the testing phase, when the update is due, then inform the all the users about the time during which the application will not be available.
4. Test feature by setting the downtime for maintenance to around 10 mins and check whether or not it takes place by receiving a console log message. Developers can then take necessary action accordingly.

1.13.2 Validation

1. Developers of the team have special access to inform all users in real-time about server downtime for maintenance (before it happens).
2. To ensure that server downtime for maintenance is as less as possible, team Boilerbot has created a copy of the application that can be run on a local machine.
3. Before pushing the functionality live, our team will test it thoroughly on the local machine.
4. During server downtime for maintenance, team Boilerbot will quickly push the tested functionality to production and thus get the bot running again.

2 What did not go well?

- Team Boilerbot was able to meet all its targets and user stories but one problem was that we hadn't thought far enough into the future when we were initially writing the documentation. Our documentation conveys what we need but details like handling a large user base isn't something that we had thought of completely.
- We spent a lot of time in refactoring our code to get it up to scratch. If we had started off with a better mindset of writing elegant code, we would not have needed to allocate time to refactoring and cleaning up our codebase.
- We chose to continue to work in our own individual tasks after we integrated all our features. This led to some communication gaps while we were piecing the code segments together.

3 What will we do better in the next sprint?

- We have also decided to make our planning documents even more articulate and well thought out so that during the development phase, the developers can add/create the feature without too much confusion i.e. with enough clarity.
- Our team has written different parts of code for this project. Everyone on our team has different coding practices and etiquette so from now on, we have decided to place heavy emphasis on writing code that is not only reusable but also is easy to maintain, read and verify i.e. even by members of other sub-teams.