BoilerBot

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

April 10th, 2017

Sprint 3 Planning

Contents

1 Sprint Overview

At this stage, the bot has gone through several revisions and provides a list of features critical to boilermakers. In this sprint, it is our goal to both refine the features(Dining courts etc.) and add more essential features that could make the bot far more versatile. Extending the dining courts features in this sprint would mean adding abilities such as finding, opening/closing times and providing entire menu listings for the previously implemented dining court cards. Additionally, this sprint has a much more critical update. Users will now be able to make time-sensitive decisions during the rather busy and stressful registration period. This includes getting information about the courses offered this semester, the rooms, times and credits associated with these and most importantly the remaining seats available course-wise.

1.1 Scrum Master

Eehita Parameswaran

1.2 Meeting Schedule

Tuesday, Thursday, Saturday at 6:00PM

1.3 Risks or Challenges

This sprint offers a lot of challenges and the risks associated with them, mainly due to the addition of the critical and heavy course-listing features that the bot must now support at the end of this sprint. The main challenge is to gain a sound understanding of the Purdue.io API that we must use in order to extract various pieces of information. The queries in Purdue.io are handled by OData protocol and hence an understanding of the REST requests through OData must be properly understood. Additionally, some features, particularly features such as seats available, require nested and deeper queries to OData structures. A thorough understanding of both OData protocols and the Purdue.io entities and catalog hierarchy.

1.4 Testing Framework

Team Boilerbot tried to automate most of the tests built in Node.js. We used a tool called 'Mocha' which is a feature-rich JavaScript test framework running on Node.js and in the browser. This tool helped us make asynchronous tests in a simple manner. These Mocha tests run serially thus, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. So, with Mocha we had an environment to make our tests but to actually test our HTTP calls, we required an add-on library.

Hence, to add the necessary logic, we utilized 'Chai' which is an assertion library. One of the main reasons we chose Chai over other assertion libraries was because it allowed us to choose the type of assertion style we'd like to use. This library comes with three different assertion types. It has the *should* style, the *expect* style and the *assert* style. Mocha essentially

provides *traction* for unit testing as there are a lot of libraries that are built on the *expect* package such as supertest. Supertest allows developers to test API endpoints very easily by querying the api directly and asserting the responses. The *expect* package can then be used to test the responses in more detail. We decided to use the *expect* style and by using chai-http, we were able to make the actual HTTP requests. And then, we tested the responses with the expected results.

2 Current Sprint Detail

2.1 User Story #19

As a user, I would like bot to provide me with information about the opening and closing times of dining courts at Purdue.

Task	Time(hrs)	Assigned To
Integrate Purdue dining court API with bot	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of timings	4	Yash
Retrieve relevant information from Purdue dining court API and feed into cards	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.1.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save dining courts information to the database .
- Given that the user requests opening and closing times of dining courts, we should be able to access the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get opening and closing times of dining courts at Purdue when the user requests for such information.

2.2 User Story #20

As a user, I would like the bot to provide me with information about food at dining courts at Purdue.

Task	Time(hrs)	Assigned To
Setup connection with Purdue dining court API	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of food	4	Yash
Retrieve relevant information from Purdue dining court API and feed into cards	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.2.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save dining courts menu information to the database .
- Given that the user requests food at dining courts dining courts, we should be able to access the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get food at dining courts Purdue when the user requests for such information.

2.3 User Story #22

As a user, I want the bot to have information about the courses being offered in the current semester.

Task	Time(hrs)	Assigned To
Integrate Purdue.io API with boilerbot	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of courses	4	Yash
Retrieve relevant information from Purdue.io API and feed into relevant template	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.3.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save course information to the database .
- Given that the user requests courses being offered in the current semester, we should be able to access the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get courses being offered in the current semester at Purdue when the user requests for such information.

2.4 User Story #23

As a student at Purdue, I would like to know the seats left in a course that I want to register for.

Task	Time(hrs)	Assigned To
Setup connection of Purdue.io API with bot	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of seats	4	Yash
Retrieve relevant information from Purdue.io API and feed into cards template	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.4.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save seats left in a course information to the database .
- Given that the user requests seats left in a course, we should be able to access the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get seats left in a course at Purdue when the user requests for such information.

2.5 User Story #24

As a student at Purdue, I would like the bot to provide me with information about the different professors teaching a course.

Task	Time(hrs)	Assigned To
Integrate Purdue.io API with boilerbot	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of professors	4	Yash
Retrieve relevant information from Purdue.io API and feed into relevant template	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.5.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save information about different professors teaching a course to the database .
- Given that the user requests information about different professors, we should be able to retrieve the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get information about different professors teaching a course at Purdue when the user requests for such information.

2.6 User Story #25

As a student at Purdue, I would like the bot to provide me with information about the different rooms that a specific course will be taught in.

Task	Time(hrs)	Assigned To
Set up connection and integrate Purdue.io API with bot	4	Devaunsh
Train Natural Language Processing algorithm to understand intent of rooms for a course	4	Yash
Retrieve relevant information from Purdue.io API and feed into list or cards template	4	Kush
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Eehita

2.6.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save information about different rooms of a specific course to the database .
- Given that the user requests information about different rooms for a specific course, we should be able to retrieve the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get information about different rooms of a specific course at Purdue when the user requests for such information.

2.7 User Story #26

As a student at Purdue, I would like the bot to provide me with information about the different times at which a specific course is being offered.

Task	Time(hrs)	Assigned To
Setup connection of Purdue.io API with bot	4	Aditya
Train Natural Language Processing algorithm to understand intent of timings of a course	4	Aditya
Retrieve relevant information from Purdue.io API and feed into relevant template	4	Rahul
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Rahul

2.7.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save information about different times of a specific course to the database .
- Given that the user requests information about different times for a specific course, we should be able to retrieve the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get information about different times of a specific course at Purdue when the user requests for such information.

2.8 User Story #27

As a student at Purdue, I would like the bot to give me details like the credits offered for taking a course.

Task	Time(hrs)	Assigned To
Integrate Purdue.io API with Boilerbot	4	Aditya
Train Natural Language Processing algorithm to understand intent of details of courses	4	Aditya
Retrieve relevant information from Purdue.io API and feed into list or cards template	4	Rahul
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	4	Rahul

2.8.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, then it would be sent to Recast.ai for parsing.
- Given that we can access data from the Purdue.io page, we would be able to save credits offered for taking a course to the database .
- Given that the user requests information about credits offered for taking a course, we should be able to retrieve the correct details from the database.
- Test feature by using Mocha-Chai testing framework and check whether or not we get credits offered for taking a course at Purdue when the user requests for such information.

2.9 User Story #28

As a user, I want to be able to submit features that I would like to see on the bot.

Task	Time(hrs)	Assigned To
Train Natural Language Processing algorithm to understand intent of feedback	6	Devaunsh
Save feedback in the database using queries so that those features can be released in future versions	6	Yash
Test feature using automated testing i.e. mocha-chai or even manually in Messenger	6	Kush

2.9.1 Acceptance Criteria

- Given that our Natural Language Processing algorithm has been modified, when a user sends a message, then it would be sent to Recast.ai for parsing.
- Given that Recast.ai is set up, run queries such that the feedback gets saved to the database schema.
- Test feature by using Mocha-Chai testing framework and check whether or not our database is saving user suggested feedback.

2.10 Developer Story #4

As a developer, I would like to be able to handle the UI for mobile and web appropriately.

Task	Time(hrs)	Assigned To
Test all features on mobile and web manually in Messenger to ensure that the templates show up correctly	4	Rahul

2.10.1 Acceptance Criteria

- Test feature manually using Messenger and by adding multiple users as testers to ensure that our bot handles UI for mobile, web appropriately.

2.11 Developer Story #5

As a developer, I would like to implement further fixes and enhancements according to the usage of the user.

Task	Time(hrs)	Assigned To
Create a feedback model in the database.	4	Aditya
Store the feedback given by the user in the database schema.	4	Aditya
Test whether or not the feedback is getting saved in the database using stored procedure scripts	4	Rahul

2.11.1 Acceptance Criteria

- Given that the users are providing feedback to an image, when the user rates an image, then store the rating in the database.
- Given that the feedback is stored in the database, then analyze the feedback for further enhancements of filter recommendations to the user.
- Test feature using stored procedure scripts to ensure that feedback model in the database is correct and can be analyzed.

2.12 Developer Story #6

As a developer, I would like to be able to implement popular feature requests submitted by students.

Task	Time(hrs)	Assigned To
Create algorithm that can parse feedback saved in database	6	Aditya
Algorithm should identify the most popular feature requests from the database	6	Rahul
Test feature using stored procedure scripts to check if algorithm performs right action	4	Eehita

2.12.1 Acceptance Criteria

- Given that have feedback in our database model, then when user suggests feature, our algorithm should be able to parse it.
- Given that we have an algorithm that can parse feedback, then the code should be able to identify the most frequent user request.
- Test feature using stored procedure scripts to ensure that algorithm is identifying the most common feature request.

3 Remaining Backlog

The Boilerbot Team will complete the backlog in its entirety with Sprint 3. This includes all the functional (user and developer) and non-functional user stories.

3.0.1 Non-Functional Requirements

1. As a user of the chatbot, I expect my information to be secure from other users.
2. As a developer, I expect the chatbot to be scalable. The performance shouldn't be affected with increasing user base.
3. As a developer, I would like the Natural Language Processing model to be reusable and extensible.
4. As a developer, I would like to securely store user usage and telemetry for further understanding of user requirements without storing user credentials.
5. Build a database that is computationally cheaper to run and maintain.