CSE 5911: Capstone - Software Engineering

Individual Report Xiao Liang

The team responsible for the project iBot consists of four members, which are Xiao Liang (me), Yong Liu, Xuzhou Yin, and Yun Ma. The final product, iBot, was designed to be an assistant with natural language processing power built in in order to simplify the complex process of common daily tasks that involves with common business softwares like: Jira, Salesforce, and Jenkins.

As the project manager of the iBot team, my main job was to set up meetings with other team members every week, as well as set up the weekly meeting with our sponsor, Branden. Besides that, I was also in charge of setting up sprints and cards for the agile tools, as well as assigning them to corresponding team members and giving them an estimate.

At the same time, as the lead programmer, my responsibility for the team contains but not limited to the following: research and compare the pros and cons for different AI platform, programming languages as well as frameworks; set up the main structure of the code base; push the code to GitHub private repository as well as set up main collaboration platform; write testing code, as well as enable Travis-CI as the main continuous integration platform for the whole project; set up and publish the code to Heroku, which is one of the most famous PaaS service provider to enable interaction with Microsoft bot service; set up Microsoft bot framework account to link the bot's endpoint with Microsoft service in order to interact with Skype, Slack and Web chat; set up LUIS account and model, train it to make the natural language processing result more idea; purchase the extra resources required for the functioning of the project: both LUIS's service and Jira's monthly subscription.

As the frontman of the team, I set up the foundation for the design and basic implementation of the code. Having in mind the importance of reactive programming, we choose Node.js as the main platform of the project, which helps to archive high amount of processing power with little resources required. Nevertheless, we choose to use TypeScript as the language to program the bot, since it give the power of type system to JavaScript, it helps to find problems ahead of time, which is a huge plus. Finally, we use one of the main features added to ES6: the abilities to use async and await to setup coroutine, to simplify asynchronous programming, which reduced the possibility of having callback hell, as well as improving the readability of the code.

Other team members was mainly involved in manual testing, documentation composing, as well as doing various subtle task to ensure that the software work as intended.

Apart from that, we used LUIS as the main platform to incorporate Artificial Intelligence, since it has a monthly free amount of call quota, as well as a friendly web interface. By integrating LUIS into iBot, we was able to achieve the ability to let the user talk to the chat bot without having to learn any command prior to using it.

In terms of testing, I choose Mocha as the testing framework, since it provides a simple and effective syntax to do testing. At the same time, we integrate Travis-CI as a web hook for our GitHub repository, so that every time someone pushes code to the repository of set up a pull request, Travis-CI will be automatically run the tests as well as report the test coverage to the report. With all that in mind, we can be quite confident that the code will be able to work as intended without worrying about it.

We setup a private GitHub repository in order to collaborate online without having to go out and drive to school in order to meet face to face. When it comes to online communication, and documentation, we choose WeChat and Google Docs, since they are one of the most popular choices among students.

Looking back to the start of the semester, we have made a huge achievement, and learn a lot all the way. From the whole process of software engineering, to the way modern software companies work, as well as collaborating with other teammates to work out the software.

If there could be anything improved, I would suggest that we should make more efforts into making sure that the software is working before heading out to show them off. Though testing is harder, and requires more time and energy than we have ever expected.