AIDERA

Team 2: Arjun Bagla, Abhijit Edlabadkar, Rajalakshmy Iyer,
Eehita Parameswaran, Aakash Ranga, Akanksha Tripathy

February 10th, 2017

Sprint 1: Product Inspection

Contents

1 Design Inspection Defects

Product	Aidera Design Inspection		
Date	February 1st, 2017		
Authors	Arjun Bagla, Aakash Ranga		
Moderators	Rajalakshmy Iyer, Eehita Parameswaran		
Inspectors	Abhijit Edlabadkar, Akanksha Tripathy		
Recorders	Arjun Bagla, Aakash Ranga		
Defect #	**Description**	**How Corrected**	**Severity**
1	The files containing database models. For instance, the users.js should be in a separate directory/folder and not in the file where the setup of the application is.	The user model file users.js and other files were put in a new directory/folder called models.	1
2	The file containing system specific commands and functionality should be in a separate directory to avoid unnecessary exposure, create maintainable design structure and to avoid confusion.	The system functionality containing files were put in a new directory called system.	2
3	The file that holds the database connection settings (database.js) should be in a separate directory/folder and not in the file where the setup of the application is.	The user model file user.js and other files were put in a new directory/folder called models.	2
4	The files that holds the intents (greetings.js, help.js, goodbyes.js) should be in a separate directory/folder and not in the file where the setup of the application is.	The intents file greetings.js, help.js, goodbyes.js and other files were put in a new directory/folder called intents.	2
5	The files that are used to connect to Facebook should be kept as a separate module and not in the main server.js file. This module should not be exposed as it uses sensitive information like page access token etc.	The Facebook module with facebook connectivity files were put in a separate module of their own.	2

Defect #	Description	How Corrected	Severity
6	The main server.js file that was used to connect our application to ngrok uses express, a node module for node.js based servers. The environment should never specify a port because that may hinder connection to ngrok or facebook.	The environment never contains a variable with port value in it. In that manner, express doesn't get confused.	1
7	Large number of node modules are pushed that make the code base unnecessarily heavy. A solution is needed so that the node modules aren't pushed every time the code is committed.	A file named .gitignore had to be created to avoid node modules from being pushed to the code base.	3
8	Ngrok, the executable we are using for local testing should never be shut down since it generates a dummy url for query forwarding. The url is used to get and send information from and to our application.	Ngrok is never re-initialized on each of our laptops to keep testing url the same.	2
9	The access token for the Natural Language Processing API Recast.ai, Facebook page should be put in a separate file to maintain modularity, usability and maintainability.	A separate file config.js was created in the main directory.	1
10	Facebook Graph API should have its calling urls and tokens put in a separate file for safety and encapsulation so that sensitive information isn't exposed unnecessarily and all the information for calling API's are in one place.	A separate config.js file was made for the various access tokens and urls.js was made for the call urls.	1
11	In the Natural Language Processor API Recast.ai, user input expressions should be put under specific intents.	Various intents were made to recognise what the user is trying to say along with the suitable expression.	3

2 Code Inspection Defects

Product	Aidera Code Inspection		
Date	February 3rd, 2017		
Authors	Abhijit Edlabadkar, Akanksha Tripathy		
Moderators	Arjun Bagla, Aakash Ranga		
Inspectors	Rajalakshmy Iyer, Eehita Parameswaran		
Recorders	Abhijit Edlabadkar, Akanksha Tripathy		
Defect #	**Description**	**How Corrected**	**Severity**
1	When the user is trying to say something and the application is still not trained to handle those specific intents, the error received for the intent was null.	The null intent was gracefully handled by giving the user an appropriate message to try something else and continue the conversation.	2
2	API call would lead to asynchronous behaviour and subsequent code would be executed before reply from the API was received.	Avoid putting useful code after an async call method. Put all important code within async method.	2
3	Using the '+' operator to concatenate string objects was giving the error that the string is an undefined object.	The concat() function was used to achieve the functionality to concatenate strings objects.	2
4	Recast client did not connect to Recast.ai and was not able to analyze text.	Recast.ai node module was installed and object from that was created.	1
5	Application was not connecting to ngrok executable program and Facebook messages were not being received.	Server port was changed by environment, we handled that by setting port to 8080.	1
6	Self made modules were not importing due to issues with file location and syntax.	Put in a './' before file name while importing and managed files properly.	2
7	Parsing Facebook request body was not giving us user details and message data.	Printed out body of the request and parsed it correctly to get user details and message specific data.	1

Defect #	Description	How Corrected	Severity
8	When the user sends anything other than a message, (for instance, image) the bot crashed.	Handled the error gracefully by sending the received image back to the user with a message that the bot handles only text.	2
9	When trying to access user information by calling Graph API, the call failed and user details weren't found.	Handled error by using request module for node and creating the request URL from scratch.	2
10	Had issues sending message data to Facebook in the form of replies.	Handles issue by creating JSON object from scratch and building the JSON based on specifications detailed on Facebook developer page.	1
11	Due to the asynchronous nature of Node.js, problems occur when a variable is used before a query is finished that will assign a value to that variable.	The solution was to assign the variable within the async function and return it from within there as well. This way, we force Node to wait for the asynctask to finish before moving forward to the next line of code.	1
12	MongoDB would crash if mongod instance not running.	Mongod server should always be running in the background. If the instance stops functioning as desired, stop any database operations and notify developers via message.	2
13	MongoDB database will crash if any of the values in the schema are null at any point of time.	If it is known that the value will be null at any point of time, then initialize it as *default=null* while declaring it.	3
14	Different message types led to confusion and application crashing.	Saw the different message types on Facebook developer website and handled for all different types of messages like text, attachment, quick reply, echo etc.	1

3 Unit Testing Defects

Team Aidera automated the tests for our bot 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. 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.

Product	Aidera Unit Testing		
Date	February 5th, 2017		
Authors	Eehita Parameswaran, Rajalakshmy Iyer		
Defect #	**Description**	**How Corrected**	**Severity**
1	Tedious process to host Aidera service on Heroku cloud-platform for local testing	Decided to not use Heroku for initial testing because multiplatform tunnelling is possible using other servers	1
2	Unable to connect to a public endpoint(i.e. internet) using a locally running network service	Used ngrok to host a server on which we run Aidera	1
3	Bot wouldn't show up on Facebook Messenger service	Created a forum-like environment for Aidera on Facebook that generated Page Access Token	1
4	Bot was unable to connect to Node.js codebase	Used page information to obtain appID and secret such to establish connection	1
5	Aidera didn't POST a message to a URL i.e. HTTP callback was malfunctioning	Used webhook to connect Aidera to a callback URL which we obtained from our ngrok server	1
6	receiveMessage function was unable to differentiate between text messages and other messages such as attachments	Added code such that the function would print to the console if the input text was an attachment or a query	2

Defect #	Description	How Corrected	Severity
7	Aidera wouldn't provide usage instructions. It will instead start interaction with the user.	Added code to identify new users and thus provide a tutorial/instructions about how to use the bot	2
8	Input: Hi Aidera! Output: No output	Added code to recognize an input text and always respond Correction: 'Hi!'	1
9	sendToRecast function would try to run attachments through our Natural Language Processing codebase	Added code to differentiate between messages that can be scanned for intent and those that can't	1
10	callSendAPI function is unable to parse the response data appropriately as text or image before sending it	Created a specific method sendTextMessage to parse the text based response.	1
11	Aidera always responded with the same text to any user message	Established connection with Natural Language Processing API i.e. Recast.ai using token	1
12	Aidera responds with the same string every time. Input: Hi Aidera! Output: 'Hi'	Added Natural Language Processing and provided varied greetings Correction: 'Hello!'	1
13	Aidera responds with generic, impersonal responses for user queries Input: 'Hi Aidera!' Output: 'Hello!'	Implemented body-parser module to extract user information and return personalized responses Correction: 'Hi Chi!'	1
14	Aidera only responds appropriately to greetings and is unable to provide responses to texts with unspecified intents Input: 'jgehfbsd' Output: No output	Modified algorithm such that Aidera would ask user to clarify instructions i.e. standardized error message Correction: 'Didn't really get that. Could you try something else?	3

Defect #	Description	How Corrected	Severity
15	Bot was unable to provide appropriate response to goodbye texts Input: 'bye' Output: 'Didn't really get that. Could you try something else?'	Added goodbye intents to Recast.ai collection and updated algorithm to respond appropriately Correction: 'Bye Chi, Hope you have a nice day!'	1
16	Bot was unable to provide appropriate response to help texts Input: 'help' Output: 'Didn't really get that. Could you try something else?'	Added help intents to Recast.ai collection and updated algorithm to respond appropriately Correction: 'Aidera helps use two services : Yelp, Airbnb. Please access discussion forum for detailed instructions.'	2
17	Complex method to integrate and use a NoSQL database because of the JSON tree structure	Decided to use a SQL database i.e. MongoDB because of ease of creating tables and accessing user details	2
18	Aidera was unable to connect to MongoDB database	Used mongoose connection and a token to connect to databaset	2
19	Aidera was unable to provide appropriate response to feedback texts Input: 'need to add new feature' Output: 'Didn't really get that. Could you try something else?'	Added feedback intents to Recast.ai collection and updated algorithm to respond appropriately Correction: 'Thank you for your feedback! Team Aidera will look into it.'	2
20	Aidera was providing feedback responses but asynchronously saving user responses	Updated schema for database along with algorithm to ensure that user IDs are getting mapped to feedback	2
21	Bot was unable to provide appropriate response to cuisine recommendation queries Input: 'recommend me a cuisine' Output: 'Didn't really get that. Could you try something else?'	Added cuisine recommendation intents to Recast.ai collection and updated algorithm to respond appropriately Correction: 'Try Chinese.'	2
22	Bot responds to text after a considerable wait Input: 'Hi Aidera!' Output: ... (after wait) Hello!	Modified algorithm such that it processes data faster and provides a speedy reply	3