AIDERA

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

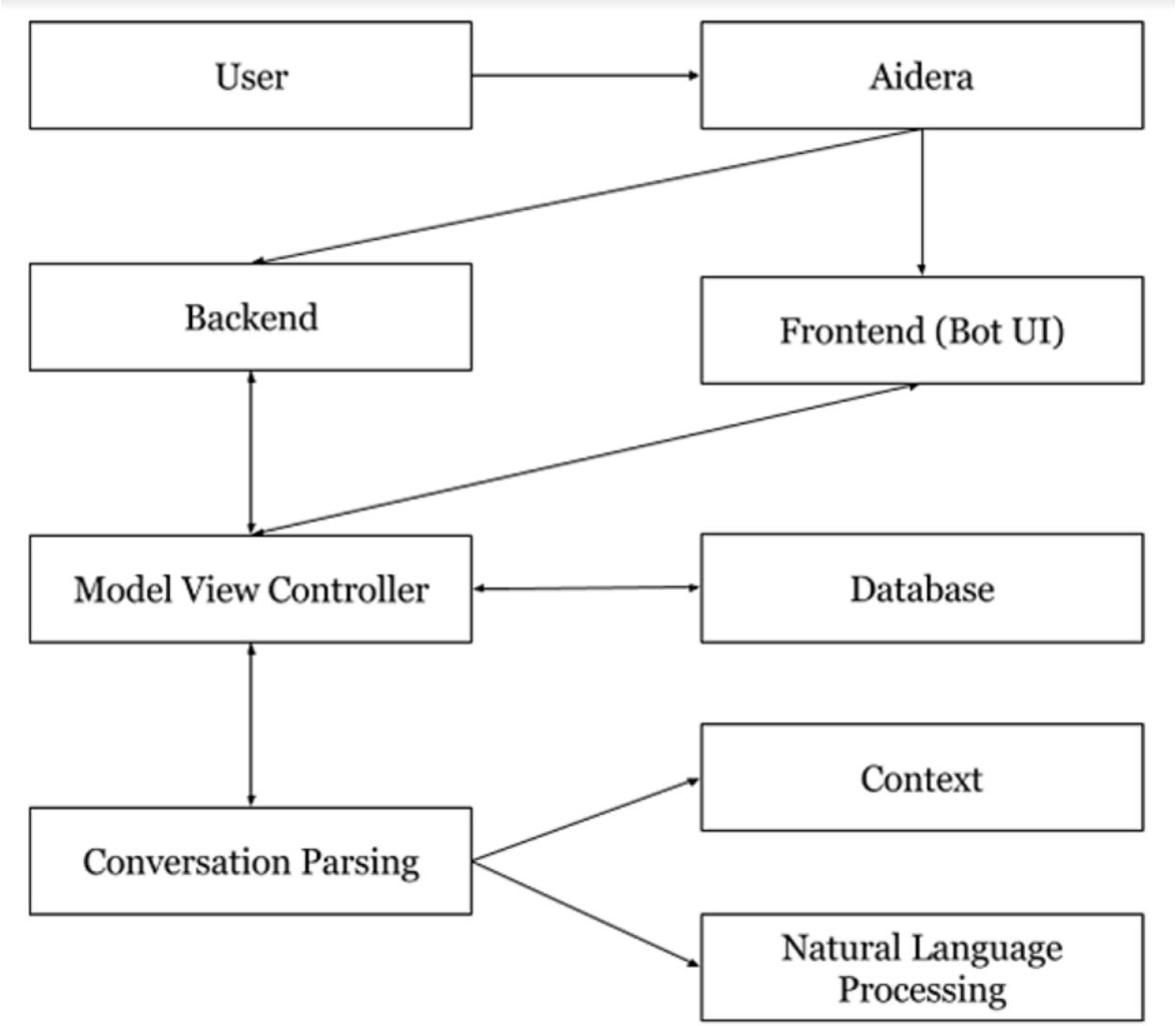
February 17th, 2017

Incremental & Regression Testing

Contents

1 Classification of Components

1.1 Define all Components



Component	Input	Output	Dependent Components
Component A: Aidera	User sends text or query to the bot in English sentences.	Aidera responds appropriately with answer or follow-up question	Aidera bot depends on mainly two components that are outlined below i.e. back-end and front-end.
Component B: Model View Controller	User doesn't directly interact with the MVC framework.	MVC middleware component receives and sends information to the dependent components (constant back-and-forth).	The Model depends on the database, View depends on the front end, Controller depends on the back-end and vice-versa.
Component C: Database	Yelp or Airbnb services require details like location, price range, etc. which are taken from user.	Database saves user details like preference of price range, location, dates, etc. in table and Aidera provides responses accordingly.	Database is largely independent of the other components. Interacts mainly with the MVC.
Component D: Front-end (UI)	User can provide feedback on resource page or choose to interact with the bot directly.	Developers provide responses to user inquiries and 'help', 'feedback' like keywords redirect user to correct resource.	User Interface is largely independent of the other components. Interacts mainly with the MVC.
Component E: Back-end	User queries are passed to the back-end and appropriate steps are taken.	GET, POST requests are made and the correct web-hooks are accessed to respond correctly.	Back-end is largely independent of the other components. Interacts mainly with the MVC.
Component F: Conversation Parsing	User doesn't interact directly with this component but it is essential for parsing to work perfectly.	Sends necessary details to context and Recast client to retrieve the intents, entities.	This component interacts with context, NLP parts, along with MVC for sending it the relevant information.

Component	Input	Output	Dependent Components
Component G: Context	Message sent to conversation parsing that is relayed to the context component.	Recast client enables developers to get intent from messages and thus provide an appropriate response.	Context component relies mainly on Conversation Parsing to get intent of message.
Component H: Natural Language Processing	Message sent to conversation parsing that is relayed to the natural language processing component.	Recast client enables developers to parse messages into entities and thus provide an appropriate response.	NLP component relies mainly on Conversation Parsing to get entities of message.

1.2 Form of incremental testing followed

Team Aidera decided to follow **Bottom-Up** incremental testing for this project.

Essentially, we decided to proceed from sub modules to main modules for development and testing using 'drivers'. Using the Bottom-up methodology, we believe we were able to find several flaws that occurred in the back-end/bottom of our program. Also, creating test modules for our functions was easier because each component at a lower hierarchy was tested individually. Then, the components that rely upon these components was tested. This ensured (or, will ensure) that our application is, as far as possible, bug-free for users to use. Hence, bottom-up testing will work as a specific type of integration testing that would test the lowest components of our code base first. So, it would test the middle phase of our software, before testing the entire system.

2 Incremental and Regression Testing

2.1 Automation

Team Aidera automated most of the Incremental and Regression tests for our bot built in Node.js. We used a tool called 'Mocha' which is a feature-rich Java-Script 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 super-test. Super-test 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.

2.2 Defects log

2.2.1 Component A Module

Component A Module	Aidera

2.2.1.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Aidera was not connecting to the backend of our server.	1	Fixed callback URL to the correct ngrok URL for each user.
2	Aidera was not giving appropriate callbacks for different message types.	2	Fixed permissions for Aidera on Facebook developer portal.
3	Application was not connecting to ngrok executable program and Facebook messages were not being received.	1	Server port was changed by environment, we handled that by setting port to 8080.
4	When trying to access user information by calling Graph API, the call would fail and user details could not be found.	2	Handled error by using request module for node and creating the request URL from scratch.

2.2.1.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Fixing connection issues with backend caused no messages to appear on the terminal.	1	Emptied pending message queue by subscribing and unsubscribing Aidera to the server.
2	Fixing callback issue led to postbacks for buttons to stop working.	2	Fixed code issues regarding JSON compatibility with the data sent by facebook and parsed it appropriately.
3	Fixing ngrok connection led to breaking connection with facebook due to use of unsecure url.	1	Fixed ngrok compatibility issue with Facebook by switching to a secure https url instead.
4	Fixing the problem with accessing user information in Aidera did not ensure that sensitive information was stored securely.	2	A separate config.js file had to be made for the various access tokens and urls.js was made for the call URLs.

2.2.2 Component B Module

Component B Module	Model View Controller

The **Model View Controller Module** depends on the following three components. The Model refers to the database schema, View refers to the frontend, Controller refers to the backend system. Essentially, we will first build and test these three units and once they are working correctly, the MVC framework will work seamlessly too.

2.2.3 Component C Module

Component C Module	Database

2.2.3.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Bot was unable to connect to MongoDB database.	2	Create a MongoDB object for modeling our code-base that is made in Node.js.
2	MongoDB database would crash if any of the values in the schema were null.	1	Create an invalid entry in the database that adds a null value.
3	MongoDB would crash if mongod instance was not running.	1	Mongod server was instantiated such that it always ran in the background.
4	Bot was providing feedback responses but asynchronously saving user responses.	3	Updated schema for database to ensure that feedback is getting saved.

2.2.3.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Fixing connection to MongoDB database by creating a schema was not ensuring the functionality of the MVC framework.	2	Used mongoose connection and a token to connect to database.
2	Fixing null value error in component C (Database) causes component B (Model View Controller) not to return correct values.	1	Knowing that a value can be null at any point of time, value is initialized as *default = null*.
3	MongoDB would run but mongod would stop functioning so the View portion of the MVC was running into errors. .	1	Created error handling such that the if the instance stopped functioning as desired, the database operations would be stopped and developers would be notified by message.
4	Fixing the saving of user responses to component C(Database) did not take care of the problem in the dependent component i.e. Model View Controller.	3	Had to update schema along with algorithm to ensure that user IDs are getting mapped to feedback.

2.2.4 Component D Module

Component D Module	FrontEnd (UI)

2.2.4.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Forum-like resource page was inaccessible by users so questions couldn't be answered or feedback couldn't be received.	1	Created and publish Aidera page such that it is viewable by individuals.
2	Users unable to select options for Yelp services like PDF menu card for a restaurant.	3	Modified algorithm such that user is redirected to menu card details on selecting option.

2.2.4.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Fixing viewing component of the resource page (Frontend) do not give users the permission to post questions.	1	Updated codebase such that the page could take questions as well as feedback from users.
2	Fixing viewing component of the resource page (Frontend) do not give users the permission to post questions.	3	Created new buttons that allowed a user to select '$', '$$' or even '$$$' for a particular restaurant.

2.2.5 Component E Module

Component E Module	Backend

2.2.5.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Using the '+' operator to concatenate string objects was giving the error that the string is an undefined object.	1	Created a check of undefined objects and assign them to 'string' type.
2	Self made modules were not importing due to issues with file location and syntax.	2	Moved modules to the input location i.e. where server code was located.
3	Bot was unable to connect to Node.js codebase.	2	Used page information to obtain appID and secret to establish connection.
4	Parsing Facebook request body was not giving us user details and message data.	3	Printed out body of the request and parsed it correctly to get user details and message specific data.

2.2.5.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Fixing the error in component E (Backend) by defensive coding didn't help in concatenating string objects in component B (Model View Controller).	1	The concat() function was used to achieve the functionality to concatenate strings objects.
2	By fixing the import error in component E (Backend), it did not restore the correct management of files in component B (Model View Controller).	2	Putting in a './' before file name fixed problem with importing and helped manage files properly.
3	Fixing connection to Node.js codebase in backend component caused in turn, the HTTP callback to malfunction in component B.	2	Used webhook to connect bot to a callback URL which we obtained from our ngrok server.
4	Fixing the parser for the Facebook request body in component E (Backend) didn't solve the problems with sending message data in the form of replies in component B (MVC framework).	3	Handled issue by creating JSON object from scratch and building the JSON based on specifications detailed on Facebook developer page.

2.2.6 Component F Module

Component F Module	Conversation Parsing

The **Conversation Parsing** module depends on Context and Natural Language Processing components so once these two units are working, the parsing component will work appropriately too.

2.2.7 Component G Module

Component G Module	Context

2.2.7.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Aidera was unable to provide appropriate response to feedback texts.	1	Added feedback intents to Recast.ai collection and updated algorithm to respond appropriately.
2	Aidera was unable to provide appropriate response to goodbye texts.	3	Added goodbye intents to Recast.ai collection and updated algorithm to respond appropriately.
3	Aidera was unable to provide restaurants nearby without location.	1	Added search_nearby_restaurants intents to Recast.ai collection and updated algorithm to respond appropriately.
4	Aidera was unable to differentiate between different Yelp related services like restaurant search and restaurant recommendation.	2	Trained recast intents more thoroughly so that they could parse intents with a higher accuracy level.

2.2.7.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Fixing feedback intents in component G (Context) did not solve the problem of Aidera responding slowly to texts in component F (Conversation Parsing).	1	Modified algorithm such that it understands context faster and provides a speedy reply.
2	Fixing goodbye intents in component G (Context) caused component F (Conversation Parsing) to provide 'goodbye' messages to help texts too.	3	Added help intents to Recast.ai collection and updated algorithm to respond appropriately.
3	Fixing search_nearby_restaurants intents in component G (context) did not ensure that the correct location was passed in component F (Conversation Parsing).	1	Updated algorithm to ensure that it could get location from user and save it to our database.
4	Fixing the different intents by training them in the context component led to bot putting incorrect information in different fields while parsing the conversation.	2	Fixed this issue by training Aidera even further and providing multiple examples of different entities to help with categorization.

2.2.8 Component H Module

Component H Module	Natural Language Processing

2.2.8.1 Incremental Testing

Defect No.	Description	Severity	How to Correct
1	Sending a text like, 'Hi Aidera!' to the bot would give no response whatsoever.	1	Added code to recognize an input text and always respond.
2	Bot would responds with generic, impersonal responses to user queries.	2	Implemented body-parser module to extract user information.
3	Bot would only responds appropriately to greetings and was unable to provide responses to texts with unspecified intents.	2	Modified algorithm such that Aidera would ask provide standardized error messages.

2.2.8.2 Regression Testing

Defect No.	Description	Severity	How to Correct
1	Bot would respond with same greeting or text every time i.e fixing the error of sending texts in component H (NLP) didn't solve the problem of component F (Conversation Parsing).	1	Modified Natural Language Processing code such that it provided varied greetings.
2	Fixing the error of impersonal responses in component H (NLP) didn't solve the problem of parsing information in component F.	2	To provide personalized responses to user questions, algorithm had to be updated with user details.
3	Fixing the error of irrelevant texts in component H (NLP) didn't solve the problem of real-life conversations in component F (Conversation Parsing).	1	Updated the entities, intents section of the Natural Language Processing code-base such that the user is redirected to helpful information too.

3 Updated Product Backlog

3.1 Functional Requirements

3.1.1 As a User:

Backlog ID	Functional Requirement	Time(Hrs)	Status
1	As a first-time user, I would like the bot to introduce itself and provide me a set of instructions.	15	Completed in sprint 1
2	As a user, I would like the bot to exchange pleasantries with me.	5	Completed in sprint 1
3	As a user, I would like to interact with the bot.	5	Completed in sprint 1
4	As a first-time user, I would like the bot to repeat instructions when asking for help.	3	Planned for sprint 2
5	As a user, I would like to facilitate a forum-like environment on our Facebook page.	1	Completed in sprint 1
6	As a user, I want to log-in to Facebook to use Aidera.	1	Completed in sprint 1
7	As a user, I want the bot to be platform-independent.	1	Completed in sprint 1
8	As a user, I expect the bot to act in character even when asked about something that the bot does not yet understand.	6	Planned for sprint 2
9	As a user, I would like to keep the settings and a log of all messages across the conversation.	6	Planned for sprint 2
10	As a first-time user, I would like to have a tutorial on how the bot works.	6	Completed in sprint 1
11	As a user, I would like the bot to reply quickly.	8	Planned for sprint 2
12	As a user, I expect the bot to inform me about server downtime.	4	Planned for sprint 2
13	As a user, I would like to get a recommended cuisine each month.	6	Planned for sprint 2
14	As a user, I would like the bot to provide restaurants based on type of cuisine I want to eat.	4	Completed in sprint 1

Backlog ID	Functional Requirement	Time(Hrs)	Status
15	As a user, I would like the bot to provide restaurants based on my current vicinity.	4	Completed in sprint 1
16	As a user, I would like the bot to provide public ratings and reviews of the restaurants.	4	Completed in sprint 1
17	As a user, I would like the bot to provide restaurants based on a price range I can afford.	5	Completed in sprint 1
18	As a user, I would like the bot to provide a PDF of the menu card of a specific restaurant.	4	Completed in sprint 1
19	As a user, I would like the bot to provide information about restaurants such as reservations, delivery or a take out of food, contact number and picture of a restaurant.	5	Completed in sprint 1
20	As a user, I would like the bot to provide a list of available accommodations based on preferable dates.	5	Planned for sprint 2
21	As a user, I would like the bot to provide public ratings and reviews of a particular listing.	5	Planned for sprint 2
22	As a user, I would like the bot to provide a list of available accommodations based on preferable locations.	5	Planned for sprint 2
23	As a user, I would like the bot to provide a list of available accommodations from a specific host.	5	Planned for sprint 2
24	As a user, I would like the bot to provide information for a specific listing.	5	Planned for sprint 2
	Total	**118**	

3.1.2 As a Developer:

Backlog ID	Functional Requirement	Time(Hrs)	Status
1	As a developer, I expect my bot design to allow bot-to-bot interactions.	8	Planned for sprint 2
2	As a developer, I would like to come up with regular updates to the chat-bot. Updates that include giving more relevant information based on user feedback.	8	Planned for sprint 2
3	As a developer, I would like my bot to handle at least 100 users at once.	10	Completed in sprint 1
4	As a developer, I would reduce the downtime for maintenance to 2 hours per month.	5	Planned for sprint 2
5	As a developer, I would like to be able to handle the UI for mobile and web appropriately.	2	Completed in sprint 1
6	As a developer, I would like to implement further fixes and enhancements according to the usage of the user.	8	Planned for sprint 2
	Total	**41**	

3.2 Non-Functional Requirements:

Backlog ID	Functional Requirement	Time(Hrs)	Status
1	As a user of the chat-bot, I expect my information to be secure from other users.	3	Planned for sprint 2
2	As a developer, I expect the chat-bot to be scalable. The performance shouldn't be affected with increasing user base.	10	Planned for sprint 2
3	As a developer, I would like the model wrapper build around the Yelp and Airbnb API's to be reusable and extensible.	15	In Progress: moved to sprint 2
4	As a developer, I would like to securely store user usage and telemetry for further understanding of user requirements without storing user credentials.	15	Planned for sprint 2
5	Build a database that is computationally cheaper to run and maintain.	15	Completed in sprint 1
	Total	**58**	