AIDERA

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

January 27th, 2017

Product Backlog

Contents

1 Problem Statement

Individuals use innumerable applications everyday. Integration of essential applications in a simple chat service would be a useful tool for many. We intend to build **Aidera**, a Facebook Messenger bot that helps people access different services in one window. Using Natural Language Processing, Aidera pushes the bounds of creativity and the hope is to provide a seamless conversation for the user with the bot. Our bot uniquely provides an aggregation of different applications by giving users access to their favourite services through a portal. Thus, the bot will serve as an everyday companion to enhance the user experience.

2 Background Information

- There are several service aggregator chat bots that exist. For instance, Google Home has doubled up to be a personal assistant keeping the user informed about news, weather and other Google related features. Even Alexa has similar functionality with Amazon. More specifically, these bots integrate only features available through a single platform.
- Accessing multiple applications for different functionality on a daily basis is tedious for users. Using various services like Yelp, Airbnb on different portals increases memory usage on the individual's device and incurs a toll on battery life.
- Our chatbot will allow Facebook Messenger users to access these different resources through one window. Thus, avoiding the need to create several accounts with different services or even download their standalone applications.
- Aidera is different than it's competition since it provides a unique compilation of amenities for users. Our chatbot further incorporates a simple interface to communicate with people, making the experience more personal and realistic to the user. Also, since Aidera uses Messenger, it is platform independent.

3 Environment

The Messenger platform by Facebook works as a tool to facilitate complex conversational experiences. This enables our team to create conversational bots that can automatically chat with users. We intend to use a Natual Language Processing API like Recast.ai that effectively turns natural language into structured data as a simple way to manage context and drive conversations based on our users intent. A large number of modules are required for the problem we are trying to address and Javascript is very well suited for this purpose. We intend to leverage Node Package Manager and hence, use a Node.js service that provides a functional backend to merge multiple platforms in a single window. A NoSQL database like MongoDB works perfectly to suit our needs as it stores the information the bot requires from the user to process requests to the connected services. Additionally, the database helps with context management and facilitates smooth conversation between the bot with the user. All the environments we plan to utilize have comprehensive documentation allowing for further development.

4 Functional Requirements:

4.1 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	Planned for sprint 1
2	As a user, I would like the bot to exchange pleasantries with me.	5	Planned for sprint 1
3	As a user, I would like to interact with the bot.	5	Planned for 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	Planned for sprint 1
6	As a user, I want to login to Facebook to use Aidera.	1	Planned for sprint 1
7	As a user, I want the bot to be platform-independent.	1	Planned for sprint 1
8	As a user, I expect the bot to act in character even when asked about something that the bot doesn't 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	Planned for 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	Planned for 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	Planned for sprint 1
16	As a user, I would like the bot to provide public ratings and reviews of the restaurants.	4	Planned for sprint 1
17	As a user, I would like the bot to provide restaurants based on a price range I can afford.	5	Planned for sprint 1
18	As a user, I would like the bot to provide a PDF of the menu card of a specific restaurant.	4	Planned for 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	Planned for 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**	

4.2 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 chatbot. 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	Planned for 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	Planned for 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**	

5 Non-Functional Requirements:

Backlog ID	Functional Requirement	Time(Hrs)	Status
1	As a user of the chatbot, I expect my information to be secure from other users.	3	Planned for sprint 2
2	As a developer, I expect the chatbot 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	Planned for sprint 1 & 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	Planned for sprint 1
	Total	**58**	

6 Use Cases:

6.1 User:

6.1.1 Case #1: First-time user interaction with bot

Action	System Response
1. Choose Aidera from list of bots on Facebook.	2. Aidera gets a callback response, gives user a first-time greeting and displays a "Get Started" button.
3. Click the "Get Started" button.	4. Aidera responds with a set of instructions.

6.1.2 Case #2: Bot exchanges pleasantries with user

Action	System Response
1. Open Aidera on Messenger.	2. Aidera responds with an appropriate greeting.
3. Type in a greeting.	

6.1.3 Case #3: Bot interacts with user

Action	System Response
1. Request for information.	2. Aidera gives the most suitable response to the user's inquiry.
3. Type in a message.	

6.1.4 Case #4: Bot repeats instructions when user asks for help

Action	System Response
1. Type in "help" on the Aidera chat window.	2. Aidera gets the request and sends back standardized helpful information (functionalities of APIs).

6.1.5 Case #5: Facilitate a forum-like environment on Aidera Facebook page

Action	System Response
1. Discuss bot feedback with other users.	2. Aidera Facebook page information is available to all users. Developers provide responses to problematic situations.

6.1.6 Case #6: Login to Facebook to use Aidera

Action	System Response
1. Facebook users search for "Aidera" in Messenger text window.	2. The user is able to access the Aidera's chat window.

6.1.7 Case #7: Bot is cross-platform

Action	System Response
1. Open Aidera bot on any mobile or desktop Operating System.	2. Aidera works and responds to user questions.

6.1.8 Case #8: Bot always acts in characte

Action	System Response
1. Type in general questions like, "what do you think of Siri?", etc.	2. Aidera responds appropriately by having a standard error reply to irrelevant texts.

6.1.9 Case #9: Settings and log of all messages across the conversation

Action	System Response
1. Click on button to view previous messages.	2. Aidera shows the previous texts between user and the bot.
3. Change settings of chat bot like color, emoticons etc.	4. Aidera changes color, emoticons etc. of user messages with the bot.

6.1.10 Case #10: First-time user tutorial on how the bot works

Action	System Response
1. First-time user clicks on tutorial button.	2. Aidera provides tutorial details to help user get better acquainted with our chat bot.

6.1.11 Case #11: Bot provides quick reply

Action	System Response
1. Request for information from bot.	2. Aidera immediately provides an appropriate response to user's text.

6.1.12 Case #12: Server downtime alerts

Action	System Response
1. Text the bot when the server is down.	2. Before server is down, Aidera sends automated text informing all users about the server.
	3. Once server is running again, Aidera sends automated text informing all users about the server.

6.1.13 Case #13: Bot provides cuisine of each month

Action	System Response
1. Request recommended cuisine of the month.	2. Aidera responds with the recommended cuisine of the month.

6.1.14 Case #14: User sends request for cuisine based restaurant search

Action	System Response
1. Request for restaurants based on cusine.	2. Aidera uses Yelp API to retrieve restaurants that serve that specific cuisine and responds appropriately.

6.1.15 Case #15: User sends request for a restaurant near the user

Action	System Response
1. Request for a restaurant based on vicinity.	2. Aidera uses Yelp API to retrieve restaurants near the user's current location and responds appropriately.

6.1.16 Case #16: User sends request for public ratings and reviews of a restaurant

Action	System Response
1. Request for public rating of a restaurant.	2. Aidera uses Yelp API to retrieve public ratings and reviews of the restaurant and responds appropriately.
3. Request reviews of a restaurant.	

6.1.17 Case #17: User sends requests for price based restaurant search

Action	System Response
1. Request for restaurants based on price.	2. Aidera leverages Yelp API to retrieve restaurants within the price range and responds appropriately.

6.1.18 Case #18: User sends request for PDF of menu card of specific restaurant

Action	System Response
1. Choose postback button to request PDF of menu card of a particular restaurant.	2. Aidera leverages Yelp API to retrieve PDF of menu card of specific restaurant and responds appropriately.

6.1.19 Case #19: User sends request for business information of a restaurant

Action	System Response
1. Choose postback button to request business information of a restaurant.	2. Aidera uses Yelp API to retrieve business information (contact number, email etc.) and responds appropriately.

6.1.20 Case #20: User sends request for date based listing search

Action	System Response
1. Request for accommodations based on dates.	2. Aidera leverages Airbnb API to retrieve accommodations based on dates and responds appropriately.

6.1.21 Case #21: User sends request for public ratings and reviews for a particular listing

Action	System Response
1. Request for listing based on public ratings and reviews.	2. Aidera uses Airbnb API to retrieve listing based on public ratings,reviews and responds appropriately.

6.1.22 Case #22: User sends request for location based listing search

Action	System Response
1. Request for accommodations based on locations.	2. Aidera uses Airbnb API to retrieve listing based on locations and responds appropriately.

6.1.23 Case #23: User sends request for host based listing search

Action	System Response
1. Request for accommodations based on hosts.	2. Aidera leverages Airbnb API to retrieve listing for a specific host and responds appropriately.

6.1.24 Case #24: User sends request for specific listing

Action	System Response
1. Click the postback button to get more information on a particular listing.	2. Aidera uses Airbnb API to retrieve listing based on a specific name and responds appropriately.

6.2 Developer:

6.2.1 Case #1: Bot allows bot-to-bot interactions

Action	System Response
1. User types in a request for more information of a particular service.	2. Aidera connects to a chatbot of the service and gets more information if possible.

6.2.2 Case #2: Regular updates to the chatbot

Action	System Response
1. User types in feedback of the bot for the developers to improve the functionality and add new features.	2. Aidera takes in and stores all ratings and reviews which developers can then review so as to get a better idea of what to change/add in subsequent updates.
	3. Team Aidera will implement the features, add new functionalities and inform the user about it.

6.2.3 Case #3: Bot should be scalable

Action	System Response
1. User types in a request for any service provided and wait for a reply.	2. Aidera should quickly dispatch requests and get replies even when the number of users is high.

6.2.4 Case #4: Reduce downtime for maintenance

Action	System Response
1. User can type in a request at any time whatsoever.	2. Aidera should be robust and useful enough to always be able to handle requests at any time.
	3. The server downtime for maintenance should be minimal.

6.2.5 Case #5: Appropriate User Interface across all platforms

Action	System Response
1. User types in a request on a mobile device and follows up with the conversation on desktop.	2. Aidera will handle the user interface across all platforms and create a seamless user experience.

6.2.6 Case #6: Follow the usage of the user

Action	System Response
1. Keep track of the user activities and usage.	2. Store the user telemetry for further understanding about user requirements.
3. Do this without storing the user credentials to generalize the requirement across all users.	4. Team Aidera will comprehend the user requirements and add new features and functionality.