Preliminary Project Proposal:
Unified lead generation service

Advanced Software Engineering - Fall 2021

Part 1

Meeting with IA Max Chen on 20th October 2021.

Part 2

Our service is meant to solve a very critical problem for small businesses. These businesses generate potential customers (often known as leads) from a variety of different sources. Managing leads individually across different sources is difficult. Furthermore, without analytics on lead sources, it can be difficult for businesses to identify their lead acquisition tactics and performance as a whole.

The business owners can access our service which will aggregate lead sources and allow them to have visibility into all their lead acquisition channels at once. In particular, our solution will let business owners see analytics on their lead funnel and send broadcast messages to their leads.

Our service can be viewed in 3 discrete parts: the integration layer where connections to various lead sources are made, the data layer where lead information from various sources is aggregated and cleaned in one database, and the manipulation layer where analytics are created and messages are sent.

First, the integration layer connects our API to various lead sources where customers make contact with businesses. In particular, our API will connect to Thumbtack, Facebook & WhatsApp.

Following is the list of APIs we plan to use for our integration layer:

1. Thumbtack

- Thumbtack will provide partners with username and password for them to call endpoints
- HTTP Endpoints
    - Send leads to partners (called by Thumbtack) :
      POST https://api.[partner].com/v1/lead
    - Send messages to customers:
      POST https://api.thumbtack.com/v1/business/:businessID/lead/:leadID/message
    - Send messages to partners (called by Thumbtack) :
      POST https://api.[partner].com/v1/message

2. Facebook Messenger

- In order to get started, you will need Messenger, a Facebook Page, and a url where the webhooks to be sent to
- Update: We now have an example business with a Facebook Page and developer account with access to the messenger and webhooks APIs.
- Through the webhook, we will retrieve the lead information along with the recent conversation.
- We'll use the "send" api to send messages to customers.

3. Whatsapp

- Whatsapp will also follow a similar flow. We will configure a webhook to receive new leads and messages.
- We'll use the messages endpoint of the whatsapp API to send messages to customers.

Second, in the data layer, we will record lead data and message data from all sources in one place. This is a non-trivial task as each lead source will have lead and message data in differently structured JSON responses.

For leads, the service will store information about customers and their corresponding requests. Customer information could include first name, last name, phone number, and email (if included). Request information includes details about the particular service/product requested. Note: Thumbtack will give more information than Whatsapp because Thumbtack already asks customers for details about the project; therefore, there will be some columns in the database that are only populated by leads from Thumbtack. As a result, it will be important to flatten out the various differently structured JSON files and normalize the data (same date formats, phone formats, etc).

Registered businesses (username, salted hashed password, credentials for lead sources), lead sources per business (customer information, request information, last entity to send message), and messages per lead source (message body, time sent, last sender) will persist in a relational database either locally or in the cloud.

For messages, the service will store the time of the message and the content of the message in a persistent way. Note that all of the APIs we suggest using in the first edition of the product do not allow users to collect historical message data. Our service enables persistent data storage of business critical information.

Third, in the manipulation layer, we will use the raw data from the data layer to provide analytics about a business's lead acquisition funnel and allow businesses to interact with all of their leads.

In terms of analytics, we will use the cleaned data in our database to display information about the number and frequency of leads, the number of conversations where the business is waiting for a response, and the number of conversations where the business needs to respond. The

businesses that use our service can query leads based on time to understand whether the number of new leads changes from month to month. Businesses can also use our service to query leads based on lead source to understand where their leads are coming from.

In terms of interaction between businesses and leads, we will allow businesses to broadcast messages to all their leads at once. This is in contrast to businesses messaging each lead individually on multiple platforms. We will again allow our users to add time filters as query parameters, so that businesses can choose to target their broadcast to specific types of leads; for instance, a business may want to send a message only to leads who haven't responded or even leads who haven't responded but contacted the business one month ago.

Here is the general flow of the service:

- New users/businesses will be asked to register an account with a username and password. Businesses will then be able to login to the service using this credential.
- New businesses will also have to provide their username and password used for their lead sources. This will allow our service to connect to the APIs of the lead sources, and gather lead data from these sources.
- Once businesses are logged on, they can query our service for lead source data with the ability to filter by time and/or lead source. They can also query for our service's analytics on this data. Lastly, they use our service to easily communicate with multiple leads from multiple sources.

Part 3

1. How will you test that your service does what it is supposed to do and provides the intended functionality?

In terms of the black box method: the user/business registration and login will be tested by creating an example business that registers and logs into our service. Thus, this also tests functionality of our table of registered users. Once logged in, we will test if our service adds user IDs to API calls so that each API call is specific to each user. Our service's ability to connect to lead sources (the integration layer) will be tested by creating an example business that has credentials to all of the lead sources described above. We will also create example leads that attempt to communicate with our example business for each lead source. If the communication between our example leads and business successfully gets posted to our API, then the integration layer is working as expected. If our service successfully cleans the data regardless of the source of the data, and then stores the cleaned data into their respective tables, then our data layer is working as expected. Lastly, to test the data manipulation layer, we will have to test our service's ability to query the database and run analyses on the queried data given API calls. Our analysis algorithms will have to be tested for correctness using the dummy data. The messaging broadcasting system will have to be tested by checking if all of our example leads get notified of incoming messages.

2. How will you check that your service does not behave badly if its clients use it in unintended ways or provide invalid inputs?

We will ensure our service aptly handles invalid use cases and inputs by implementing “unhappy” and edge test cases, as well as rigorous user input checking. For example, in the registration and login component of our service, we will require common security measures such as sophisticated password requirements, a limit on the number of password attempts, and an email requirement for registration. We plan to include tests for each of these measures, such as a business with invalid credentials, which our service should reject. For the integration layer, we plan to include edge cases where a user has no lead sources, or perhaps multiple of the same lead source. Likewise, we will include error handling and tests for the case where the user might unregister from one or more of the lead sources that they had previously been using. For the data layer, it is important to test the edge cases of the data cleansing (empty data, wrong format, etc). We also plan to check and build test cases for handling possibly malicious data, such as a SQL injection. Finally, for the data analysis layer, we need to handle the case where the data passed in is empty or incomplete, which might happen if the user requests non-existent data.

3. How will you test that your service handles its data the way it's supposed to?

For our testing, we plan to use static mock data, which will allow us to test that our service is handling the data consistently and correctly because we can check the output against a static expected response. Additionally, the use of both more granular unit tests as well as integration tests with this mock data should make it easy to identify where there might be a possible bug or vulnerability. We will use Postman for testing the different APIs which are part of our service.

Part 4

IDE - Pycharm & VSCode
Persistent data store - postgres or Google Big Table
Framework - Flask
Package Manager - pip
Style Checker - pylint
Static analysis bug finder - pylint or mypy
Test Runner - unittest or pytest
Coverage - coverage

APIs

Authentication - Auth0
Fetch lead info from Thumbtack & Facebook APIs
Send messages using Twilio API