Preliminary Proposal:

CabX

Clyde Bazile (cb3150),
Benedikt Schifferer (bds2141),
Xiao Lim (xl2669) & Adiza Sumuna Awwal (asa2201)

Paragraph Description:

Our team would like to create a travel aggregator specifically targeting ride-hailing services. We envision being able to input our current and targeted location into the app and retrieve prices and travel times from popular ride-hailing services, such as Uber and Lyft.

We will create a database to store the user's sign-in info and saved addresses. We will leverage the Uber and Lyft APIs to get travel data and to request a ride.

For our MVP, we will focus on integrating data gathered from two popular ride-hailing services into the app. If successful (and if time permits), we will integrate other ride-hailing services available for use in the NYC area.

User Stories:

MVP:

CABX-001: User requests new ride (EPIC)

- As a user I would like to be able to view and compare the prices of various rideshare services in order to select the best price.
- Acceptance test:
  - If the user requests a location, the user sees the information (prices and travel times) from Uber, Lyft, etc. for his/her request

CABX-002: Receive information from Lyft

- As a developer, I want to receive price information and travel time from Lyft in order to present it to the user
- Acceptance test:
  - If the developer requests a ride from Lyft, the developer receives current price information and travel time from Lyft in a structured format

CABX-003: Receive information from Uber

- As a developer, I want to receive price information and travel time from Uber in order to present it to the user
- Acceptance test:
    - If the developer requests a ride from Uber, the developer receives current price information and travel time from Uber in a structured format

CABX-004: Get ride sharing information

- As a developer, I want to receive price information and travel time from all supported platforms in increasing price order in order to present it to the user
- Acceptance test:
    - If the developer requests a ride, the developer receives current price information and travel time from multiple platforms in a structured format

CABX-005: Get lat(itude) and long(itude) information from search keywords

- As a developer, I want to receive the lat & long coordinates from a searched place in order to use lat & long coordinates for the third-party platform APIs (probably use Google Maps to achieve this)
- Acceptance test:
    - If the developer searches for a keyword, the developer receives a list of possible places with their lat & long coordinates

Backlog:

- Users can sign up:
    - As a user I want a system in which I can create a profile within the application that will enable me to login using third party applications such as Facebook, Google or LinkedIn because this is easy to use.
    - Acceptance tests:
        - If the user fill out the form, a profile is generated with the entered information
        - After the profile is created, a user can login with his username & password
        - If the username or email did already exist in the database, the user gets an error message “Username/Email is already registered”
        - If the entered email address has not a valid email format, the user should get an error message “Invalid email address”

- Users can login (password authentication):

  - As a user I want some form of authentication to enable me to access information on previous rides purchased via the app.
  - Acceptance tests:
    - If a user has an account and the correct username & password combination is entered, the user gets a positive feedback and is logged in
    - If an invalid username & password combination is entered, the user gets a error message (“Username/password is wrong”) and is not logged in (or username does not exist)
    - Logged in user can see their own profile and profile information

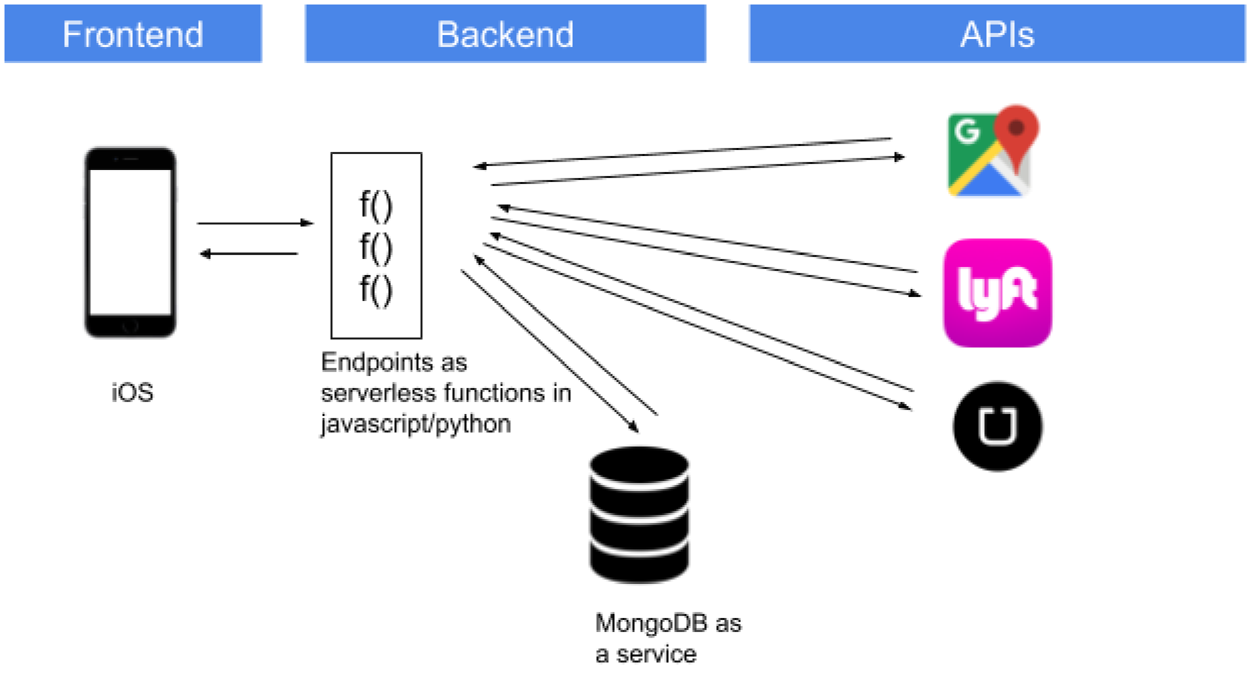
- Users change/save profile:
  - As a user I would like to customize specific stylistic and interactive functionalities within the application because this accommodates my personal preferences.
  - Acceptance tests:
    - After the (logged in) user changed/saved his profile, the information of this user is updated in the database
    - If the user log out and relogin, the user sees the new information in his profile

- Users can show profile/search history/add home/work address:
  - As a user I would like the capability to quickly access relevant information and relieve redundancy where my inputs are frequently the same.
  - Acceptance tests:
    - If the user is logged in and clicks on the search bar, the user will see their previous searches
    - If the user is logged in and clicks on the search bar, the user will see icons for their home/work address
    - If the user hasn’t set up their home/work address, the icon should be greyed out

Language & Platforms:

- iOS front-end (Swift)
- MongoDB Stitch
- Mac
- Database: MongoDB Atlas or AWS DynamoDB

Infrastructure:



iOS:

- We will develop the mobile UI or front-end in Swift using Xcode. User can sign up/login/request a ride and see the results in the app

MongoDB Stitch:

- We will use MongoDB Stitch, MongoDB's BaaS, to implement the server-side logic and handle integration with 3rd party APIs

MongoDB Atlas:

- We will use MongoDB Atlas, MongoDB's DBaaS, to store user profile data such as usernames and passwords, ride history, etc.

APIs:

- APIs are 3rd party APIs, which are not implemented by the team
- We access the APIs from the serverless backend

Next steps:

- Paragraph (Xiao) -- done!
- User stories (Adiza) - complete (edited in the first person)!
- Acceptance testing (Benny) -- done!
- Infrastructure draft (Benny) -- done!

BACKUP

Paragraph Description:

Our team would like to create a travel aggregator specifically targeting ride-hailing services. We envision being able to input our current and targeted location into the app and retrieve prices and travel times from popular ride-hailing services, such as Uber and Lyft.

We will create a database to store the user's sign-in info and saved addresses. We will leverage the Uber and Lyft APIs to get travel data and to request a ride. We will also use a publicly available Login API to create our user sign-up/login page.

For our MVP, we will focus on integrating data gathered from two popular ride-hailing services into the app. If successful (and if time permits), we will integrate other ride-hailing services available for use in the NYC area.

User Stories:

MVP:

Backlog:

- Users can sign up:
    - As a user I want a system in which I can create a profile within the application that will enable me to login using third party applications such as Facebook, Google or LinkedIn because this is easy to use.
    - Acceptance tests:
        - If the user fill out the form, a profile is generated with the entered information
        - After the profile is created, a user can login with his username & password
        - If the username or email did already exist in the database, the user gets an error message "Username/EMail is already registered"

        - If the entered email address has not a valid email format, the user should get an error message “Invalid email address”

- Users can login (password authentication):
    - As a user I want some form of authentication to enable me to access information on previous rides purchased via the app.
    - Acceptance tests:
        - If a user has an account and the correct username & password combination is entered, the user gets a positive feedback and is logged in
        - If an invalid username & password combination is entered, the user gets a error message (“Username/password is wrong”) and is not logged in (or username does not exist)
        - Logged in user can see their own profile and profile information

- Users change/save profile:
    - As a user I would like to customize specific stylistic and interactive functionalities within the application because this accommodates my personal preferences.
    - Acceptance tests:
        - After the (logged in) user changed/saved his profile, the information of this user is updated in the database
        - If the user log out and relogin, the user sees the new information in his profile

- Users can show profile/search history/add home/work address:
    - As a user I would like the capability to quickly access relevant information and relieve redundancy where my inputs are frequently the same.
    - Acceptance tests:
        - If the user is logged in and click on the search bar, the user will see his/her previous searches
        - If the user is logged in and click on the search bar, the user see icons for his/her home/work address
        - If the user hasn’t setup the home/work address, the icon should be greyed out

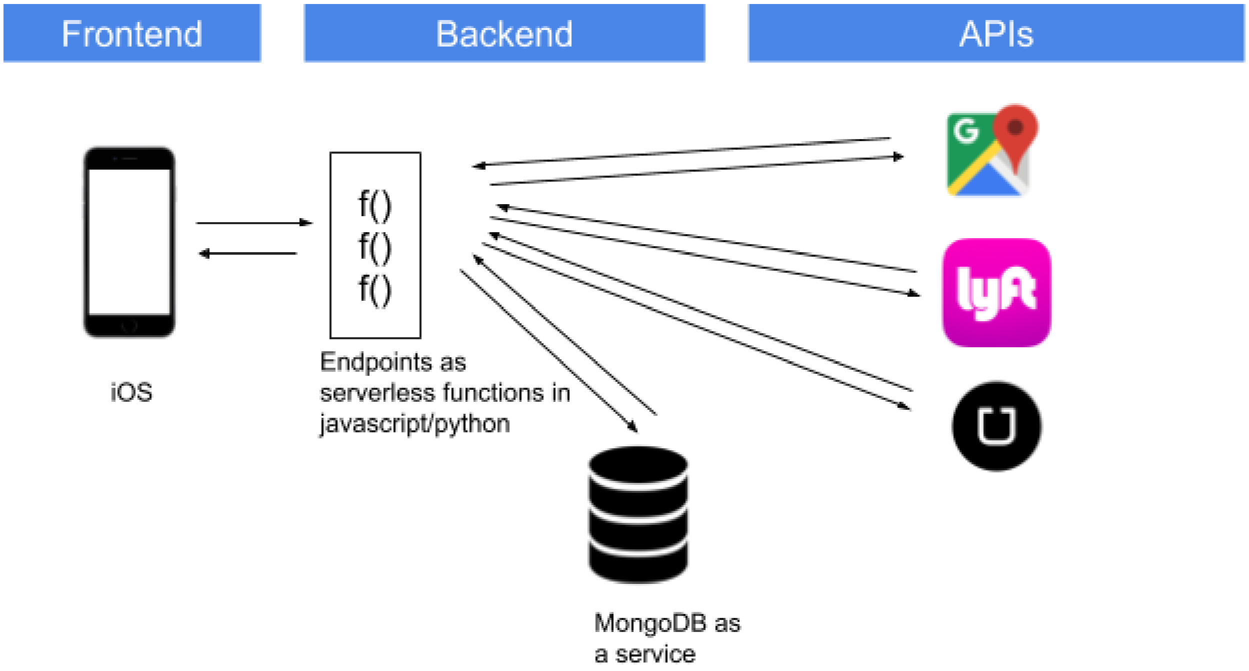
- User request new ride:
    - As a user I would like to be able to view and compare the prices of various rideshare services in order to select the best price.

- APIs:
    - Google Maps (show the map)
    - Uber
    - Lyft
    - (Juno) as alternative
- Acceptance tests:
    - If the user requests a search, the user sees the information (prices and travel times) from Uber, Lyft, Juno, etc. for his/her request

Language & Platforms:

- iOS front-end (Swift)
- Back-end (Python)
- Mac
- Database: MongoDB or AWS

Infrastructure:



iOS:

- iOS app is the frontend. User interact with the backend via app. User can sign up/login/request a ride and see the results in the app

Serverless endpoints:

- Serverless endpoints contains the main logic of the app
- The endpoints receive requests with parameters (e.g. location) and request from other endpoints necessary information (e.g. accesing database or APIs)

MongoDB:

- MongoDB as a service is the database, which important information. For example, the usernames and passwords, historical rides, etc

APIs:

- APIs are 3rd party APIs, which are not implemented by the team
- We access the APIs from the serverless backend

Next steps:

- Paragraph (Xiao) -- done!
- User stories (Adiza) - complete (edited in the first person)!
- Acceptance testing (Benny) -- done!
- Infrastructure draft (Benny) -- done!