Project 1: Friday Cat Damage Design Doc

By: Brian Moses, David Lupea, Justin Chen, and Alex Thompson

Website Description:

- Gets user ip address using ipify and with that lat and lon using ip-api
- Navbar at the top
    - Traffic: shows traffic flow map in user area and traffic incidents nearby
    - Businesses: Shows businesses of a specific type in the area on a map and in a list each with a get directions button
        - Directions: gives user a map and directions to chosen place
    - Weather: Shows weather in user area over the next week

Design doc assignments:

- Component Map: Brian
- Component list and interaction: Alex
- Site Map: David
- Database: Justin

APIs:
-MetaWeather
-IP-api
-ipify
-Mapquest:
  -Directions API
  -Place Search API
  -Traffic API
  -Static Maps API

Project Assignments:

- Flask/approutes: Alex
- HTML/Bootstrap: David
- API interaction: Brian
- Database: Justin

Components:

1. Python file app.py:
    a. Site Paths:
        i. "/"
            1. Renders login.html
            2. Flashes error messages if received
            3. Sends user straight in if in session
        ii. "/register"
            1. Renders register.html
            2. User navigates here from register button on login.html page
        iii. "/auth"
            1. Takes user and pass and calls databaseUtils.validate(user, pass)
            2. Gets username and password for validation
            3. If validate returns 0, then the user is redirected to "/welcome" and entered into a session
            4. If validate returns 1, "login.html" is rendered with error message "wrong user"
            5. If validate returns 2, "login.html" is rendered with error message "wrong pass"
        iv. "/processRegistration"
            1. User navigates here when they create an account. This adds their user, pass, and address to the database with getAddress()
            2. Calls databaseUtils.register(user, pass, ip, lon, lat)
            3. Reroutes to "/welcome" or back to "/register" if there was a problem with the registration info.
            4. Flashes error message if username already in use
        v. "/welcome"

1. Renders welcome.html, wherein the user will see most of the content.
2. Displays content gathered about IP address
3. Requests API key for mapquest

- vi. “/log_key”
  1. Renders welcome.html
  2. Adds key to session for use in mapquest API requests
  3. Thanks user for adding the key
  4. Navbar to businesses, weather, and traffic
- vii. “/traffic”
  1. Uses mapquest’s traffic API to display a chart of traffic incidents in user’s area.
  2. Uses mapquest’s static map API to show traffic flow in your area
  3. Renders traffic.html
- viii. “/weather”
  1. Uses weather api to display weather forecast for the next week in your area
  2. Renders weather.html
- ix. “/business_form”
  1. Renders business_form.html
- x. “/business_list”
  1. Renders business_list.html
  2. Uses place search API with user-selected business type in their area
  3. Uses static maps api to display all of the businesses on a map
- xi. “/directions”
  1. Uses mapq route API to get a driving route to the destination.
  2. Uses static maps to form the route on a map
  3. Renders directions.html
- xii. “/inputAddress”
  1. Exactly the same as directions except uses address for starting point instead of ip latitude and longitude
  2. Renders directions.html

b. Functions:

- i. getIP()
  1. Gets IP address of the user and returns it

        ii. get_ip_data(ip)
            1. Gets all available location data using IP address
2. Python file: databaseUtils
    a. Functions:
        i. createUsers()
            1. Creates DB users with (user, pass, ip, lon, lat)
        ii. addToUser(user, pass, ip, lon, lat)
            1. Adds user entry to db
        iii. updateIP(user, ip, lon, lat)
            1. updates the location of a user when they login in another location
        iv. getUser(user)
            1. Returns an empty dict if user doesn’t exist, or a dict with all user info from the database if user does exist
        v. register(user, pass, ip, lon, lat)
            1. Uses getUser to make sure user is not in db
            2. Returns false if user in db
            3. Calls addToUser() on params and returns True if successfully registered.
        vi. validate(user, passw, ip)
            1. Uses getUser(user) to check if the fields match
            2. If user and pass are correct, returns 0
            3. If user is wrong, returns 1
            4. If pass is wrong, returns 2
            5. If IP has changed, returns 3
3. HTML Files:
    a. Login.html
        i. Displays simple login page with login + registration button
    b. Register.html
        i. Simple registration page with boxes for user and pass
    c. Welcome.html
        i. Bootstrap navbar at the top with links to “/traffic”, “/weather”, and “/business_form”
        ii. Chart in the middle displaying all location info received from ip geolocation api
    d. Directions.html
        i. Displays a list of directions from current location to the restaurant from approute.

ii. Also displays a static map generated by mapq API showing the direction route

e. Business_form.html

i. Simple form with a dropdown to select what kind of business you are looking for

f. Business_list

i. Displays a map with all of the businesses of that type near you marked on it

ii. List of the businesses and some brief info about them with a button that says “get directions” and routes to “/directions”

g. Traffic.html

i. Displays a list of traffic incidents in user area

ii. DIsplays a map of the traffic flow in user area

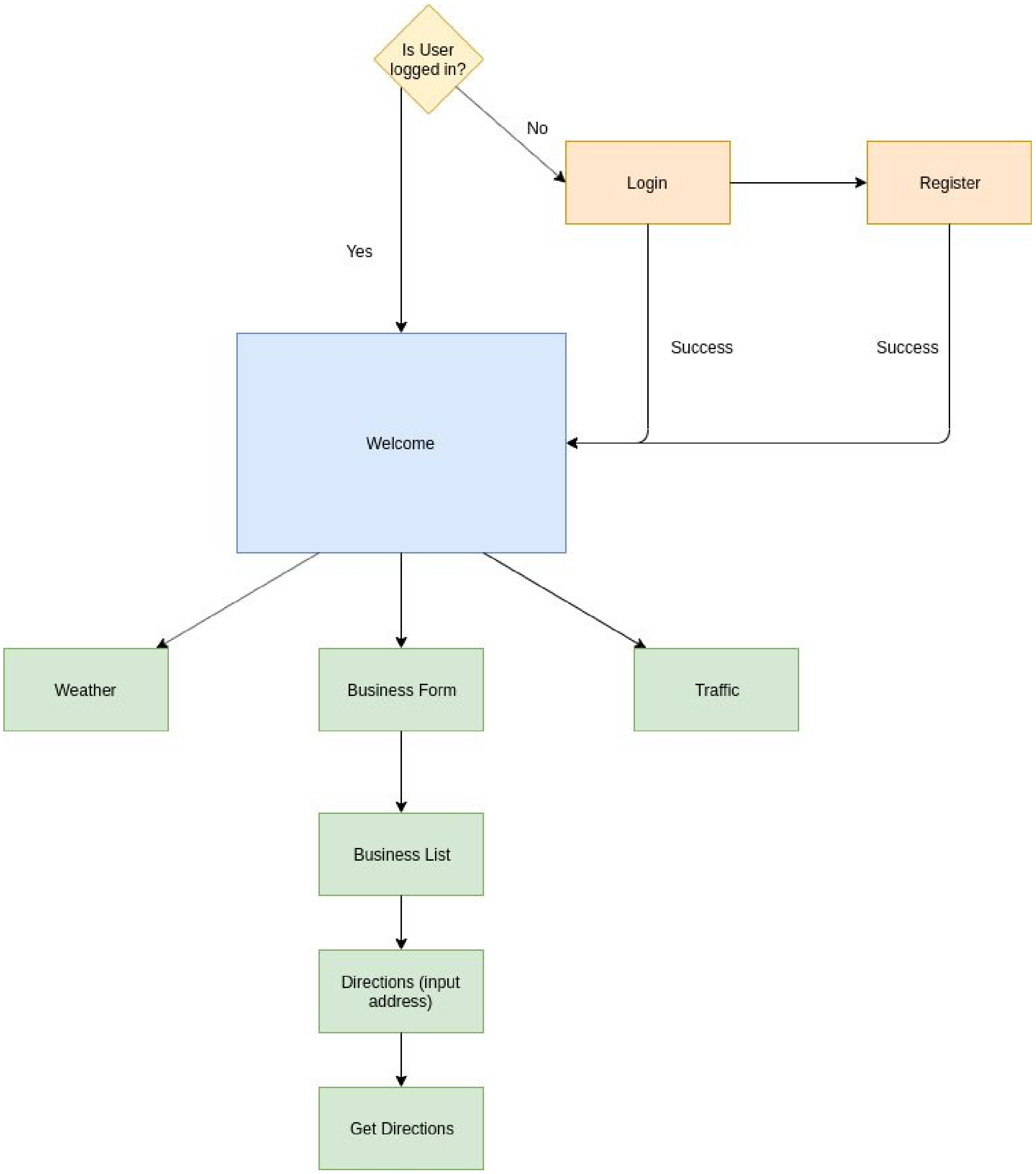
4. Database:

a. Tables

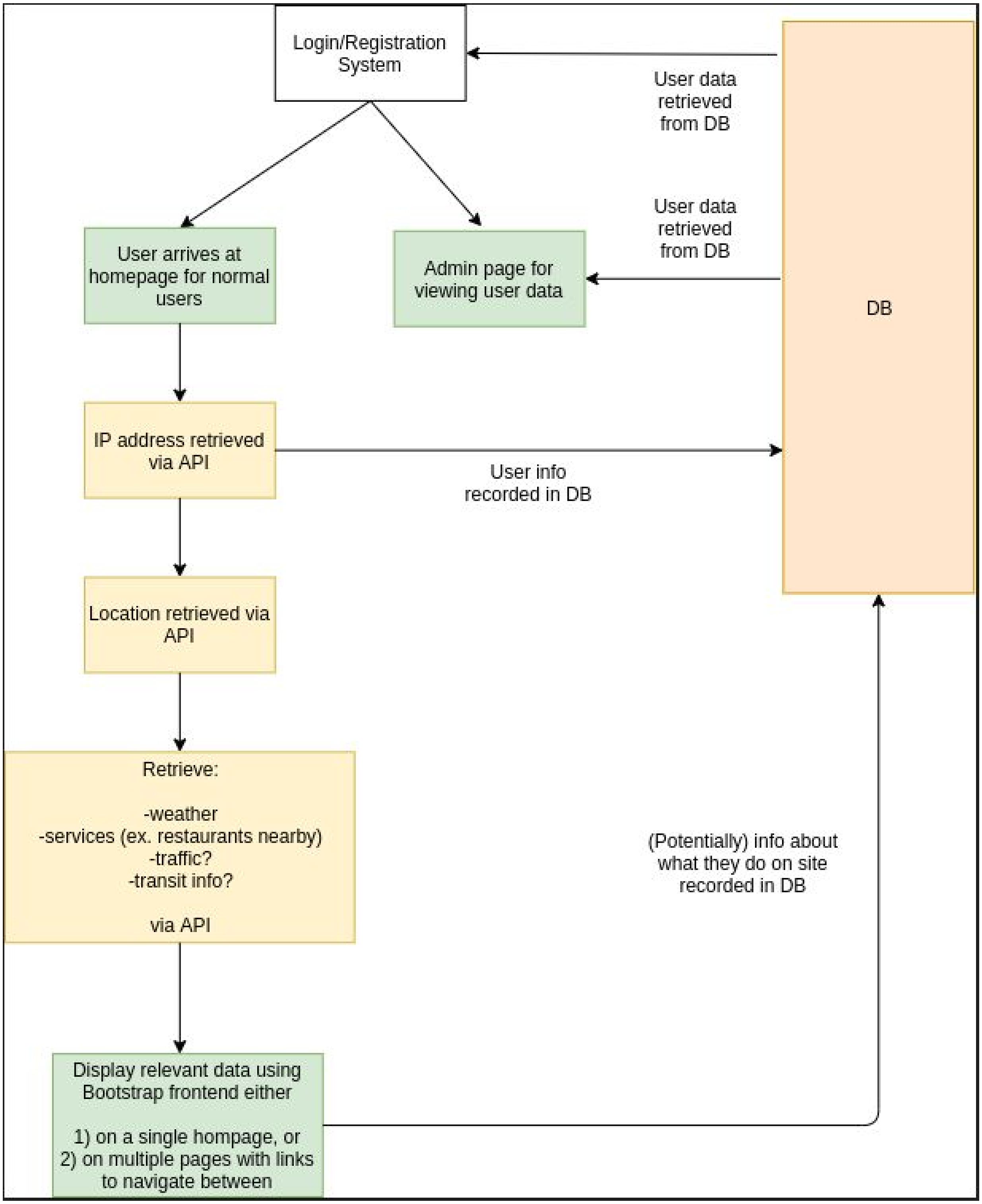
i. Users

1. Columns: username, password, ip, longitude, latitude

Site Map:


Is User logged in?
No
Login
Register
Yes
Success
Success
Welcome
Weather
Business Form
Traffic
Business List
Directions (input address)
Get Directions

Component Map:


Login/Registration System
User data retrieved from DB
User data retrieved from DB
User arrives at homepage for normal users
Admin page for viewing user data
DB
IP address retrieved via API
User info recorded in DB
Location retrieved via API
Retrieve:
-weather
-services (ex. restaurants nearby)
-traffic?
-transit info?
via API
(Potentially) info about what they do on site recorded in DB
Display relevant data using Bootstrap frontend either
1) on a single hompage, or
2) on multiple pages with links to navigate between

Database Layout:

-Table 1: userinfo

Username	password	Ip address	longitude	latitude
String	String	String	Double	Double