Chicago Wanderlust

The Ultimate Accommodation Finder
for Leisure Travelers

CS122 Final Presentation
Team: Chicago Explorer
March 8, 2017

Overview

1. Recap: Goals & Result
2. Structure & Algorithm
3. Filtering System Structure
4. Data Collection
5. Front-to-Back Linkage
6. Frontend Web Design

1. Recap: Goals & Result

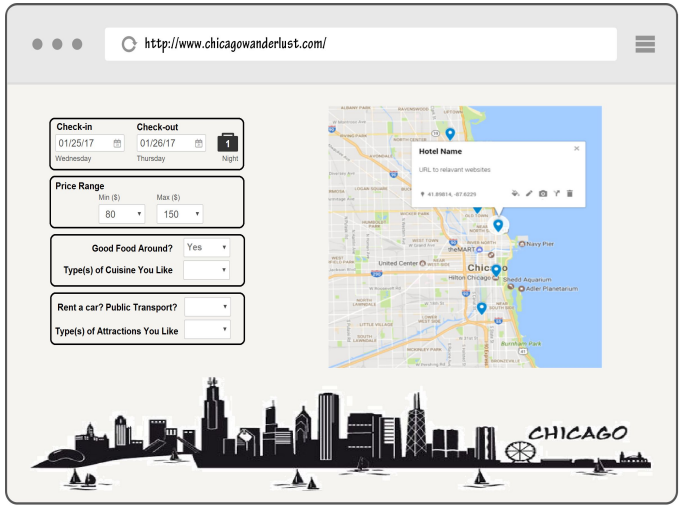
Target: Leisure travelers new to Chicago

AVOID

1. Wasting too much time on traveling to your favorite attractions
2. The hassle comparing hotels & Airbnbs
3. Dangerous neighborhoods
4. Unfavorably reviewed places
5. Places with no good food around

RECOMMEND

1. Top 5 places for your stay
2. Attractions to your taste
3. Daily itineraries



1. Recap: Goals & Result

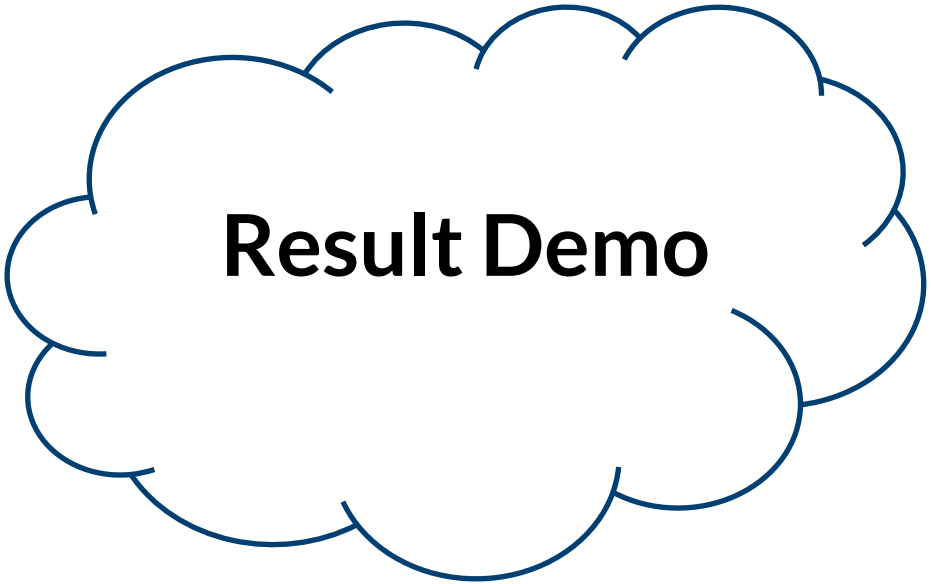
Goals

Develop a scoring algorithm that reasonably captures the accessibility of a particular location to multiple places of interest.	
Improve the efficiency of real-time data scraping.	
Effectively manage the relational databases of crimes and attractions.	
Design a user friendly and responsive interface.	

Work Distribution

Minjia Z.	Weijie X.	Ran B.	Tommy Y.
Django Web Design	APIs Statistical Filters	Web Scraping Crime Data Front-Backend Linkage	Structure Design Selecting/Routing Algorithms

1. Recap: Goals & Result



2. Structure & Algorithm

32 Attractions

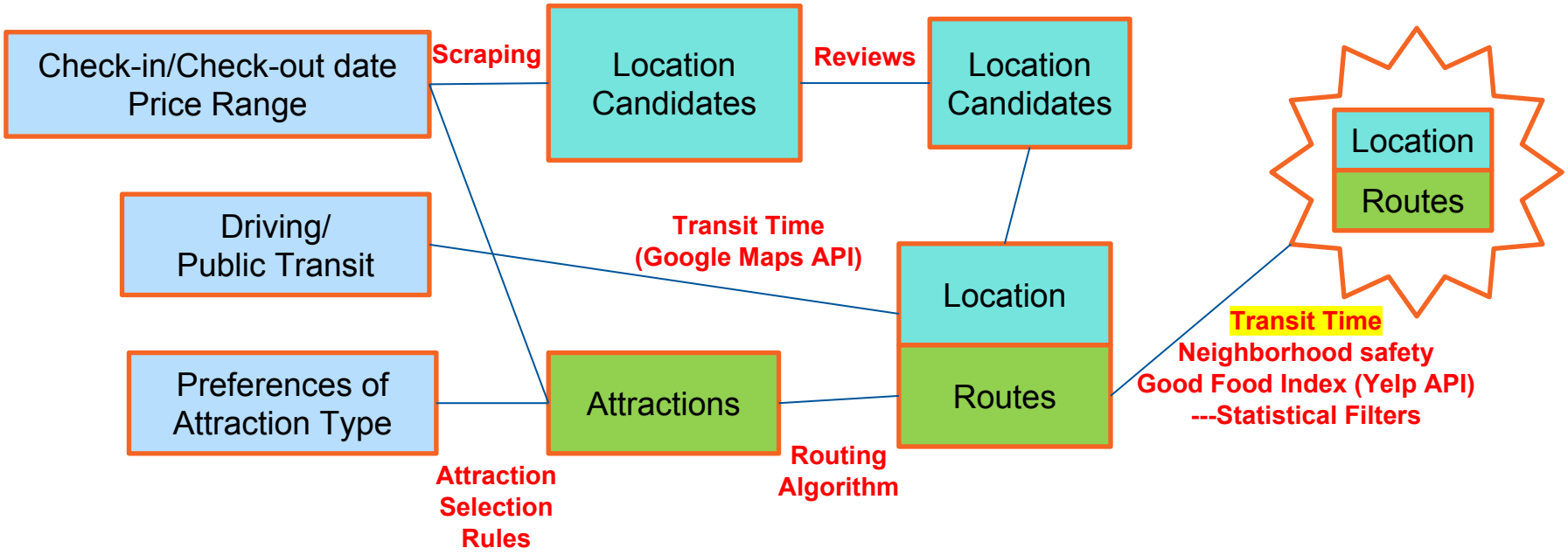
8 Types:

Architecture
Art & Culture
History
Kids/Family
Landmarks & Sightseeing
Museums
Parks & Nature
Sports & Entertainment

	Attraction Name	Identifi	Hours Needed	Architec	Art & Cu	History	Kids/Fa	Landma	Museur	Parks &	Sports &
2	Art Institute of Chicago	1	3	0	1	0	0	0	1	0	0
3	Millennium Park	2	1.75	1	0	0	1	1	0	1	0
4	Cloud Gate	3	0.5	1	1	0	1	1	0	0	0
5	Wrigley Field	4	0.75	0	0	0	0	0	0	0	1
6	Museum of Science and Industry	5	4	0	0	0	1	0	1	0	0
7	360 Chicago	6	2	1	0	0	1	1	0	0	0
8	The Field Museum	7	3.5	0	0	1	1	0	1	0	0
9	Lakefront Trail	8	2	1	0	0	1	1	0	1	0
10	Willis Tower Skydeck	9	2	1	0	0	1	1	0	0	0
11	Richard H. Driehaus Museum	10	1.75	1	1	1	0	0	1	0	0
12	Chicago Cultural Center	11	2	0	1	0	0	0	1	0	0
13	Maggie Daley Park	12	1.25	0	0	0	1	1	0	1	0
14	Lincoln Park Zoo	13	2.5	0	0	0	1	0	0	1	0
15	Chicago Riverwalk	14	1.5	1	0	0	1	1	0	0	0
16	The Chicago Theatre	15	0.5	1	1	0	0	1	0	0	1
17	Oriental Institute Museum	16	1.5	0	0	1	0	0	1	0	0
18	Shedd Aquarium	17	4	0	0	0	1	0	1	1	0
19	United Center	18	0.75	0	0	0	0	0	0	0	1
20	University of Chicago	19	1.5	1	0	1	0	0	0	0	0
21	Lincoln Park	20	1.25	0	0	0	1	0	0	1	0
22	Money Museum at the Federal Reserve Bank	21	1.75	0	0	1	0	0	1	0	0
23	Chicago History Museum	22	2	0	0	1	0	0	1	0	0
24	Navy Pier	23	1.5	1	0	0	0	1	0	0	0

Attraction

2. Structure & Algorithm



2. Structure & Algorithm

Attraction Selection Rules

1. Can take 0/1/2/3 ordered preference(s).
2. Preferences: ABC>AB>AC>BC>A>B>C.
3. Avoid too many similar attractions.
4. Range of accumulated hours: for 1 day-[6,8]; for 2 days-[12,15], as close to upper limit as possible.
5. Recommend higher-ranked attraction first.

2 DAYS
1st: Landmarks & Sightseeing
2nd: Kids/Family
3rd: Parks & Nature

Millennium Park
Lakefront Trail
Cloud Gate
360 Chicago
Lincoln Park Zoo
Shedd Aquarium
The Chicago Theatre
Navy Pier

14.75 Hours

1 DAY
1st: Museums
2nd: Art & Culture

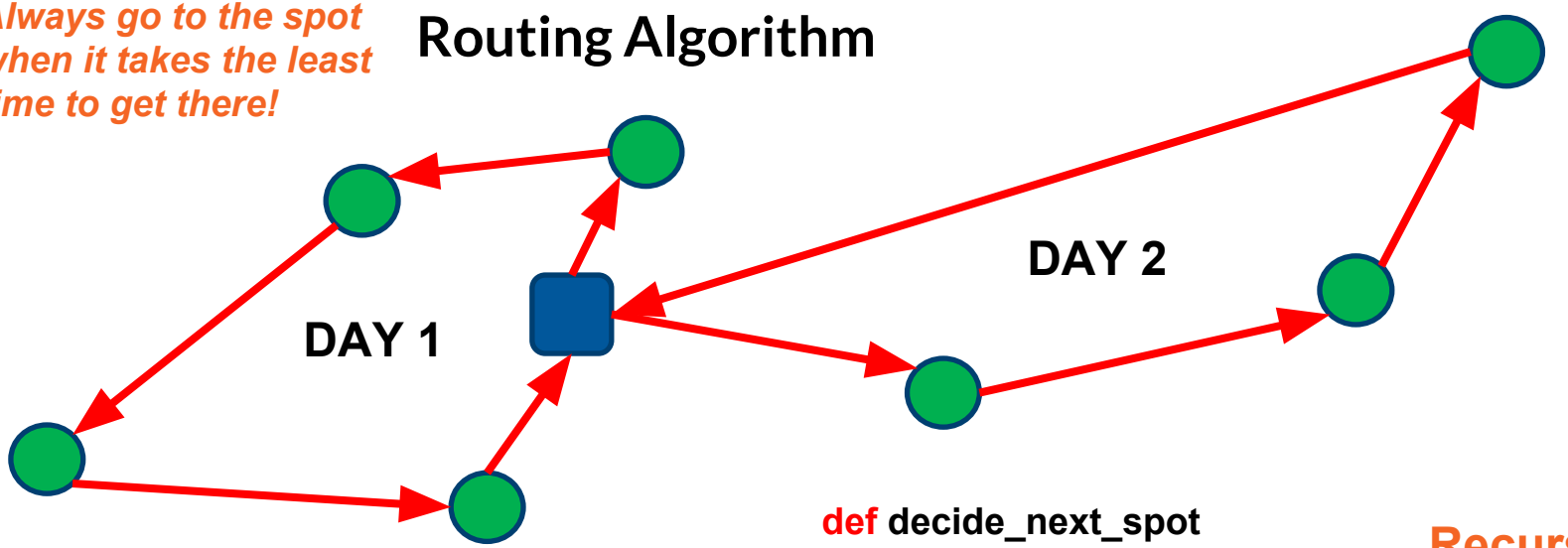
Art Institute of Chicago
Richard H. Driehaus Museum
Chicago Cultural Center

6.75 Hours

2. Structure & Algorithm

Always go to the spot when it takes the least time to get there!

Routing Algorithm

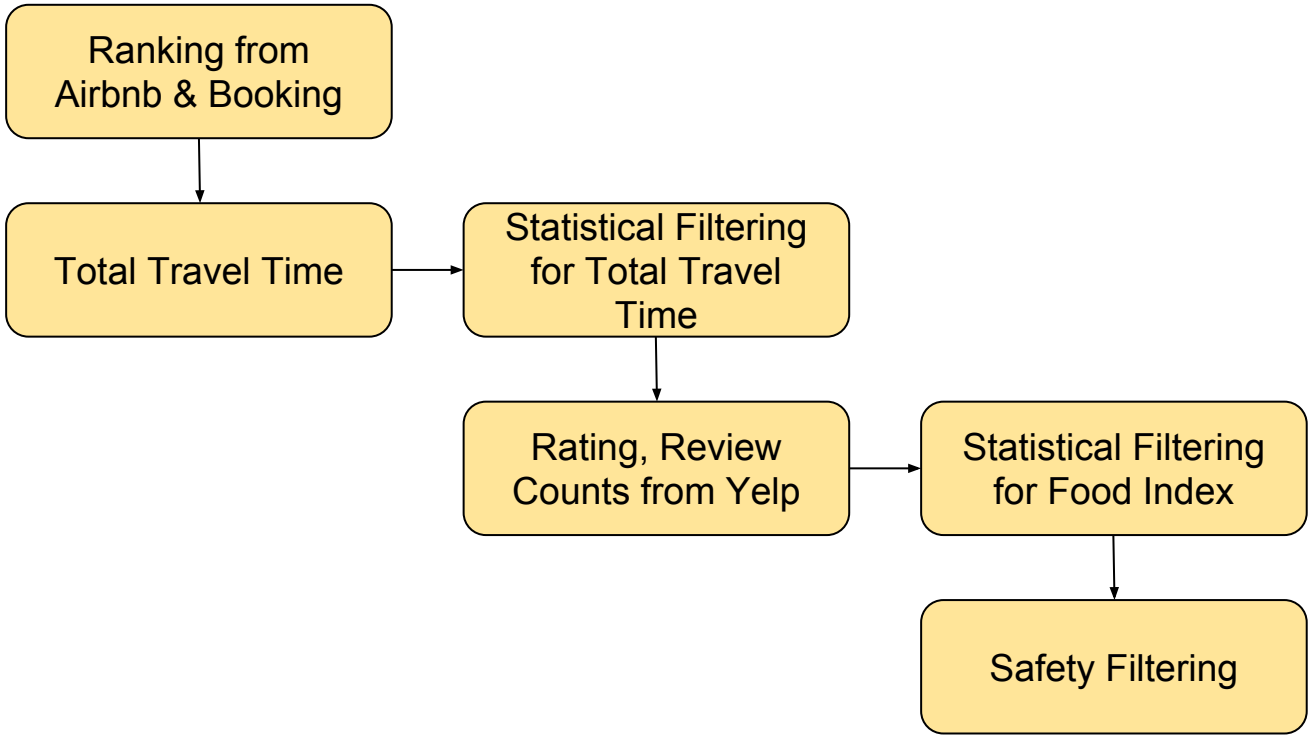


For 2-day itinerary, 1st day tends to be fuller than the 2nd day.

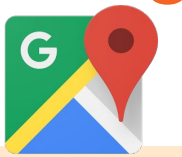
```
def decide_next_spot
def single_day_route
def first_day_route
def start_place_and_routes
def route_from_hotels
```

Recursion.
Efficiency.
Limitation.

3.1 Filtering System Structure



3.2 APIs for Filtering



API for Transit Time Calculation

- Public API
  - API key needed
- Distance Matrix API
  - Standard Usage Limits
  - 2500 free elements per day
- Transit Time Function
- Details
  - reasonable departure time
- Improvement
  - Make maximum use of API key

API for Food Index

- Public API
  - API key needed
- Search API
  - Radius: 40,000 meters(25 miles) max
  - Return results: 40 max
- Yelp filter Function
  - Rating, Review Count
- Corner Case
  - No business found - pass

4. Data Collection

airbnb

API: Requests+BeautifulSoup

- No Public API
  - Unofficial API
- Missing search keys
  - URL Encoding ASCII
  - HTTP POST request
- Making sense of search ranking
  - An algorithm encouraging exploration

Booking.com

Web Scraping: Selenium+BeautifulSoup

- AJAX request within webpage
  - Set header in HTTP request
  - Selenium Webdriver
- Using Selenium
  - Scrape dynamic contents
  - Simulate a real browser
- Website updates
  - Test scraping code regularly
- Thoughts on ethical issues

5. Front-to-back Linkage

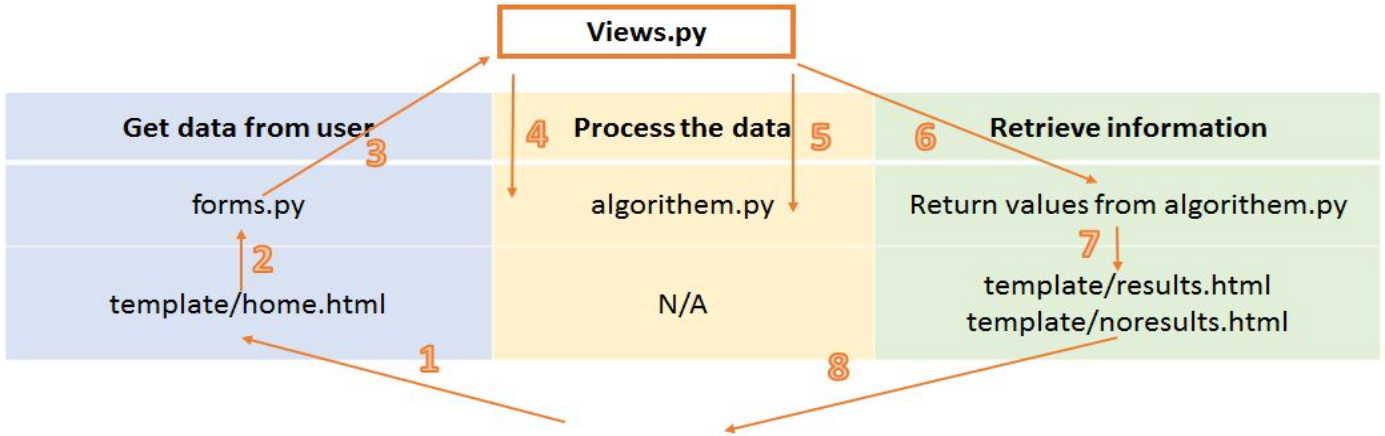
Crime Rate: Pandas+Shapely

- Source
  - 2016 Crimes - City of Chicago data portal
  - Population by communities - 2010 census
- Steps
  - Aggregate at community level
  - calculate crime rate per 1,000
  - Get boundary data of dangerous communities
- Geojson Data
  - Shapely - multipolygon
  - From first screening to sanity check

Thoughts on Front-to-back Linkage

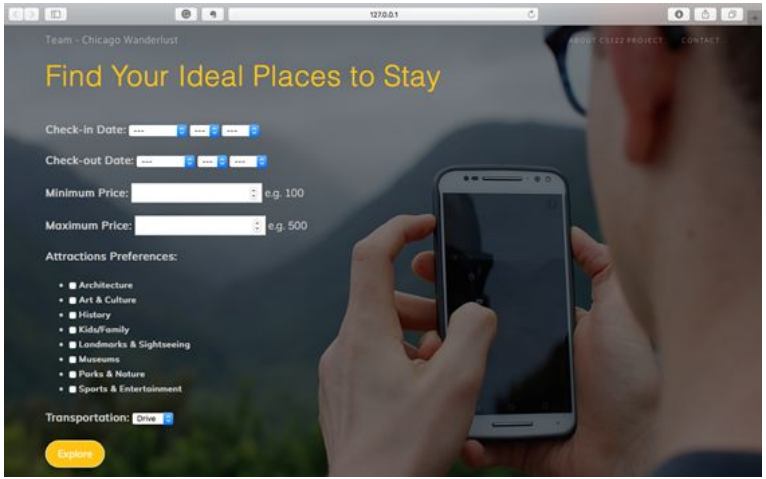
- Plan, build, test, and iterate as quickly as possible.
- Start out early. Expect the unexpected.
- Holistic understanding of the both.
- Communication is the key.

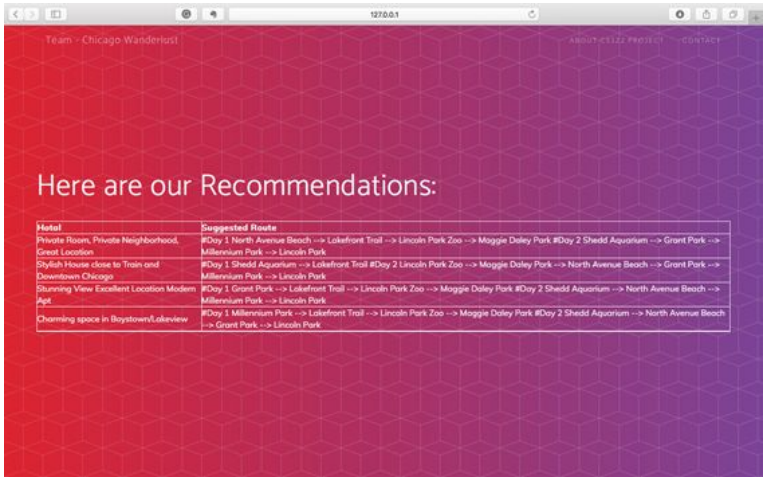
6.1 Django Framework



1. User access home page
2. User input formatted by forms.
3. Input sent to views after being aggregated as dictionary
4. In views, call the Algorithms function to process data
5. Return values sent back to views
6. Retrieve “hotel name”, “reservation link”, “suggested route” from the returned dictionary
7. Send to display in html
8. Back to user

6.2 CSS





Boostsrap template from: [Start Bootstrap]

- **New versions have changed a lot**--Be careful because it has not been trivial to update
- **Unfamiliarity**--it seems faster and easier to use an template. However, it takes time to get familiar with it and customize it.

Thank you

CS122 Final Presentation
Team: Chicago Explorer
March 8, 2017