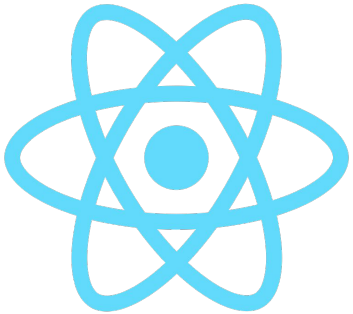
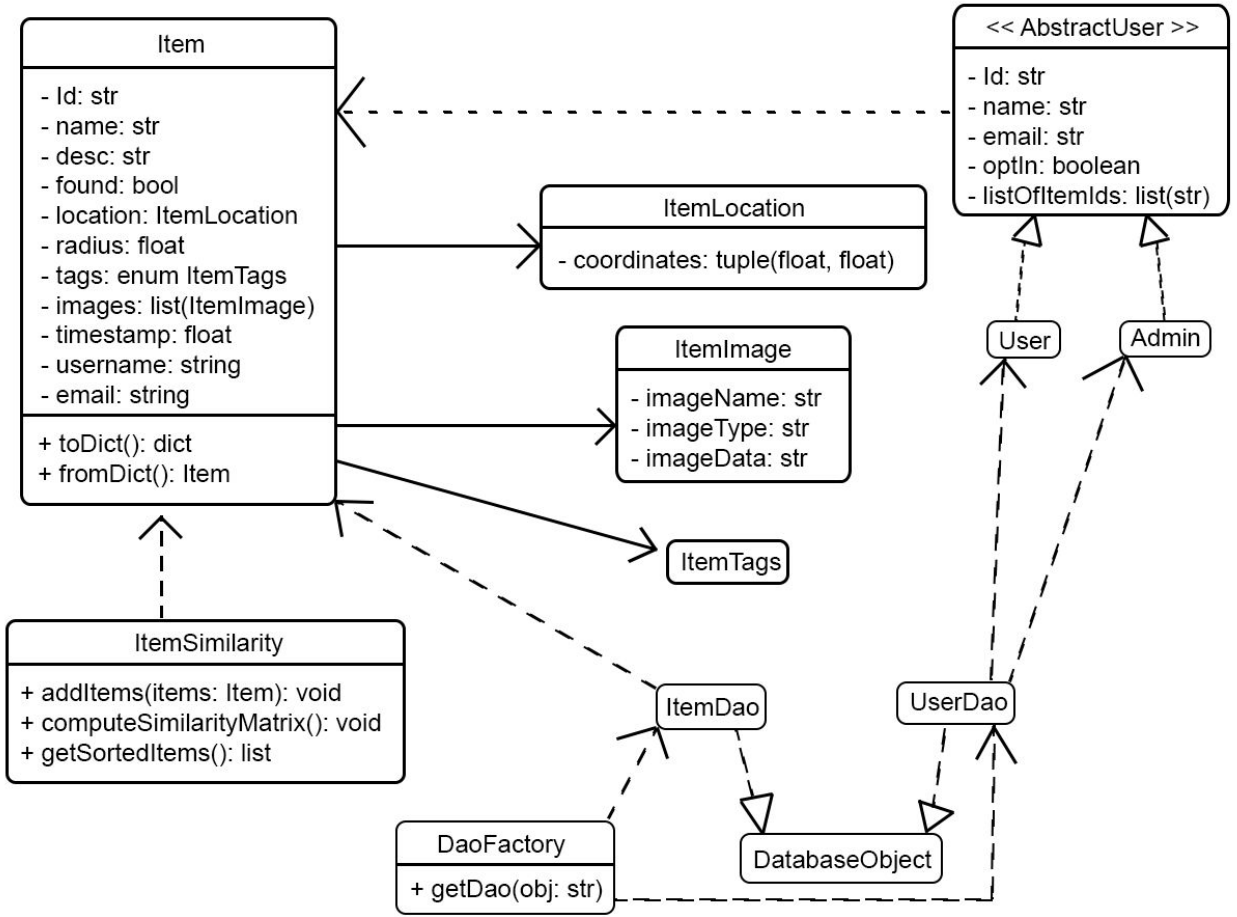
Seekr

Lost and Found

High level overview of stack



UML


Item
- Id: str
- name: str
- desc: str
- found: bool
- location: ItemLocation
- radius: float
- tags: enum ItemTags
- images: list(ItemImage)
- timestamp: float
- username: string
- email: string
+ toDict(): dict
+ fromDict(): Item
<< AbstractUser >>
- Id: str
- name: str
- email: str
- optIn: boolean
- listOfItemIds: list(str)
ItemLocation
- coordinates: tuple(float, float)
ItemImage
- imageName: str
- imageType: str
- imageData: str
ItemTags
User
Admin
ItemSimilarity
+ addItems(items: Item): void
+ computeSimilarityMatrix(): void
+ getSortedItems(): list
ItemDao
UserDao
DaoFactory
+ getDao(obj: str)
DatabaseObject

Design Principles & Patterns

- Single Responsibility Principle
  - Classes to store the more complex data of an Item
  - DAOs to handle interaction with the database
  - MVC architecture
- Open/Closed Principle
  - Adding more data/features to the Item class won't affect classes that already use it
- Interface Segregation Principle
  - Interfaces only contain basic methods, classes use all method of their implemented interface
- Factory Pattern
  - DAOs are retrieved in code through a factory
- Observer Pattern
  - Components in frontend manage their state and change if state is updated

Special features

- Gensim
- OpenCV
- Notifications
- Deployment

Using Gensim for Semantic Search Results

- Module in Python used for semantic modeling of text
  - Provides helpful data structures and tools
  - Needed to add some other tools to fit our need
- Results ranked using Soft Cosine Similarity
  - Scores from -1 to 1, results are shown in this order

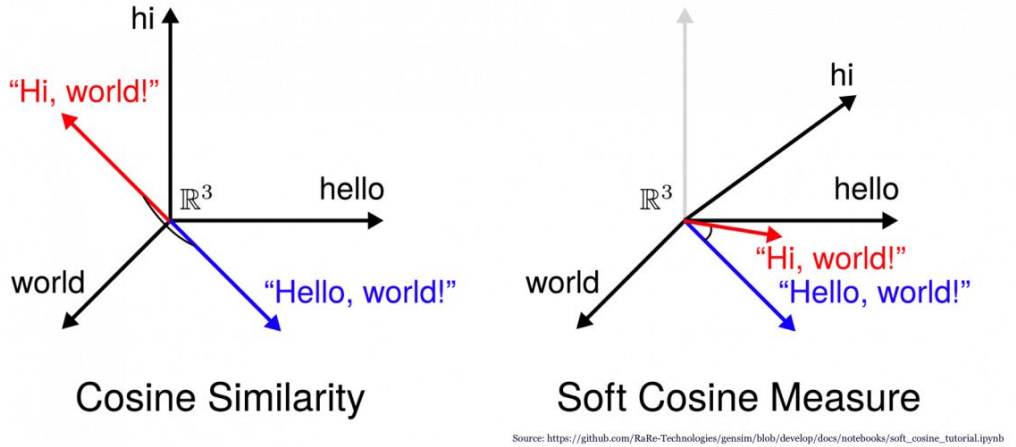
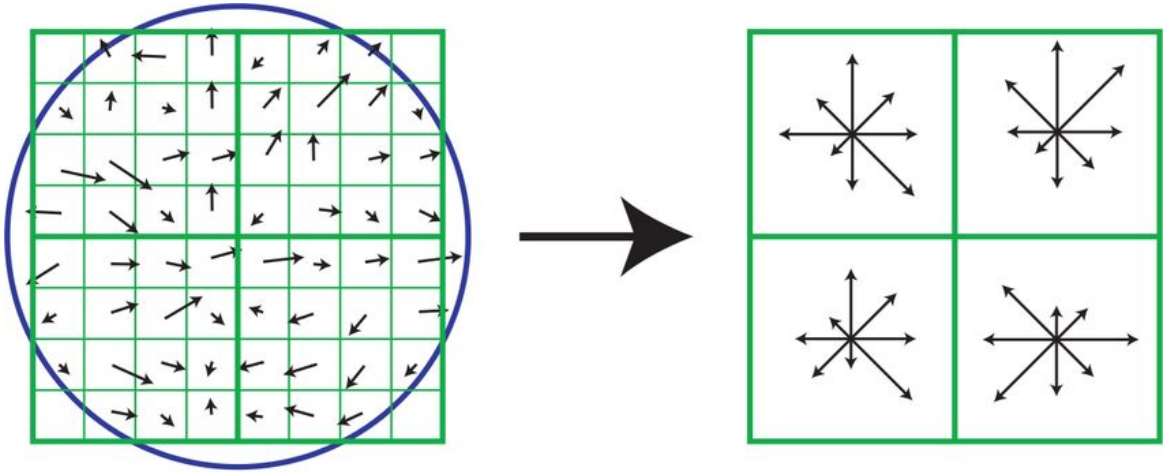
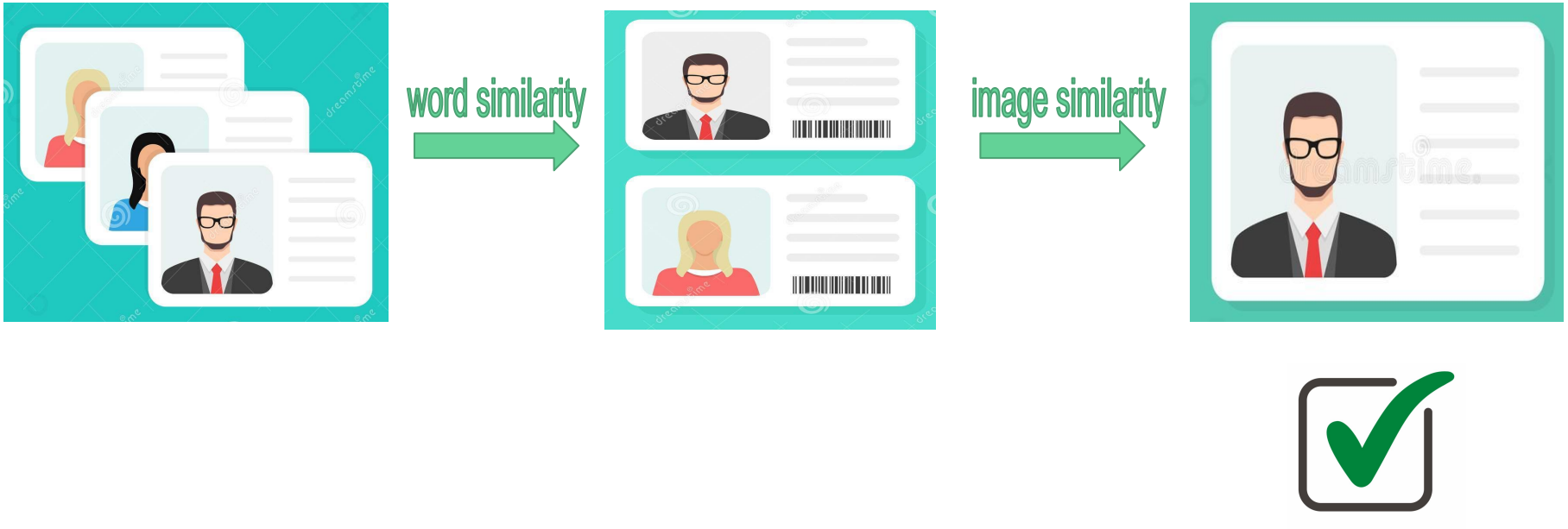


Image Recognition

- We use OpenCV and use SIFT as our feature detection algorithm
- Wanted something with a smaller memory footprint as a proof of concept
- We then use Lowe's ratio test to reduce noisy matches



Notifications



Deployment

- We use NGINX as our web server and reverse proxy
- NGINX proxies our API requests to our flask backend
- We deploy on AWS on a T2.medium, T2.micro was not enough compute power to house our models (why we wanted a smaller memory footprint)
- We also set up SSL Certs so our site has http (secure connection)
- We also use TMUX in order to keep a persistent terminal
- Travis for CI

Closing remarks

Limitations

- No built in communication system
- No verification system for claims
- Only FB and Google log in

Future

- Change model
- Add verification questions
- SIS Integration

Questions?