Auction Hunter
Progress Update

Alexander Hull, Alexander Jacobson, Yufei Zeng

CS 462 - Winter 2019

March 19th, 2019

Stakeholders

- Client - Ryan Kalb
- Instructors - Kevin McGrath and Kirsten Winters
- Group - Capstone Group 4
- Organization - Oregon State University

Project Status

- Scrapes entries from IAAI
- Stores entries in a database
- Performs value calculations
- Displays information through a web interface.

Flow Diagram

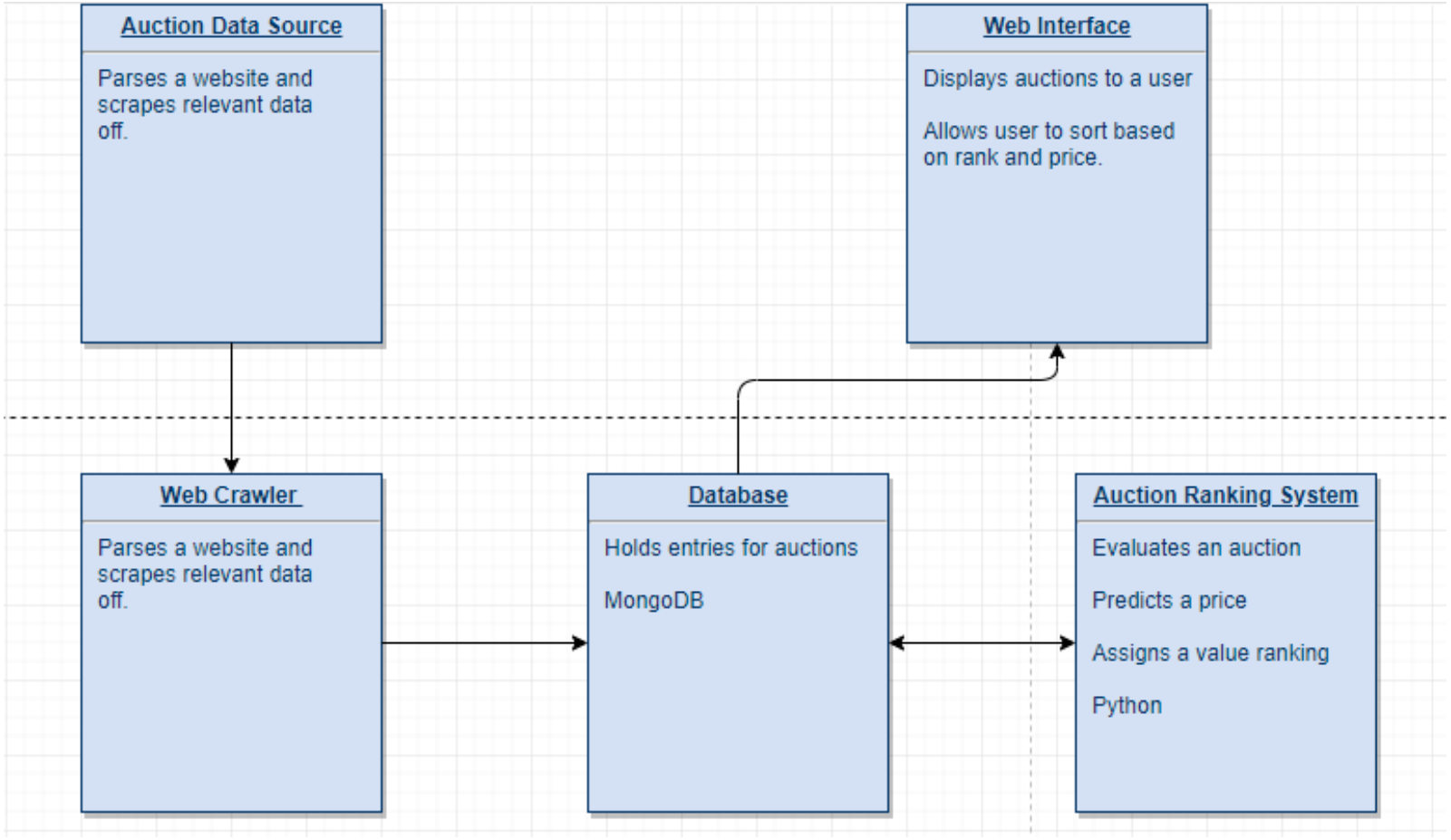

Figure 1: Flow Design

MongoDB Database

- Web scraper includes MongoDB pipeline
- Data is added directly to the database

```
def open_spider(self, spider):
    self.client = pymongo.MongoClient(self.mongo_uri)
    self.db = self.client[self.mongo_db]

def process_item(self, item, spider):
    self.db[self.collection_name].insert_one(dict(item))
    return item
```

Figure 2: Scrapy MongoDB pipeline

Database entries

Database entries in MongoDB:


```
_id: ObjectId("5c760ec27731f09709af5c35")
car name: "2008 FORD FOCUS S/SE"
miles: " 194k mi (Not Required/Exempt) "
vin: "VIN: 1FAHP34NX8W179989 "
primary damage: "Collision | FRONT END "
car image: "<img data-original="https://vis.iaai.com/singlethumbnail?imageKeys=231..."
value_est: 8.266666666666673
```

```
_id: ObjectId("5c760ec27731f09709af5c36")
car name: "2016 FORD EXPLORER LIMITED"
miles: " 33k mi  "
vin: "VIN: 1FM5K7FH3GGC66566 "
primary damage: "Collision | RIGHT FRONT "
car image: "<img data-original="https://vis.iaai.com/singlethumbnail?imageKeys=236..."
value_est: 51.2
```


Figure 3: Database entries

Value Estimation

- Naive value estimation based on damage and mileage
- Certain attributes can be weighted
- Will be improved as more data is available

```
#Miles in thousands
value -= ((miles*2)/150.0 - 1)*self.milesWeight

#Damage from 0 to 5(most impactful damage)
value -= damage*(self.damageWeight/2.0)
```

Web Scraper

- Users want to see the prices at different platforms at a single place.

- Some scrape examples

```
#create a basic spider "timedauctions"
scrapy genspider timedauctions
https://www.iaai.com/TimedAuctions
```

Figure 4: Scrape Example

Extracting Info

Extracting the URLs include salvage car photo:

- Using Google extension SelectorGadget. We found image's CSS labels are ".lazy".

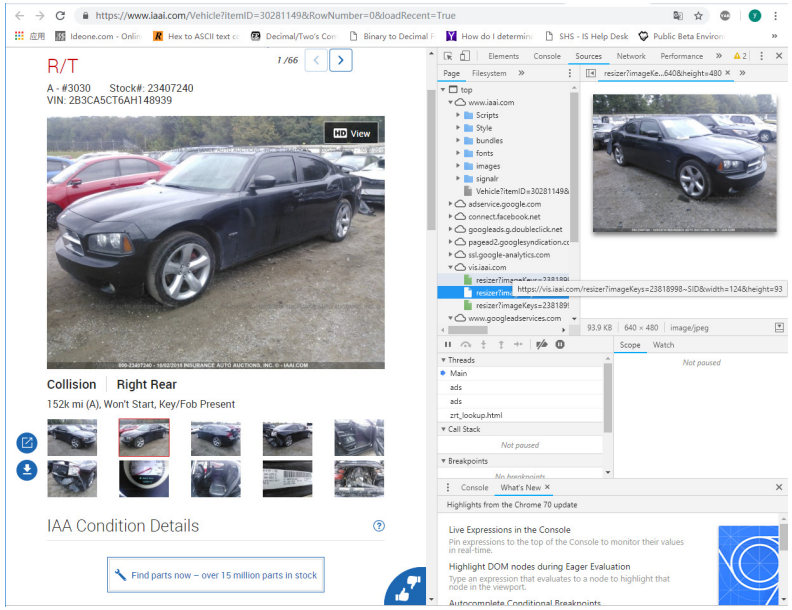

Figure 5: Developer tools to find keys

```
#the CSS labels ".lazy" can extract image URLs.
response.css(".lazy").extract()
```

Extracting Info

Extracting the URLs include salvage car vin: We can use a similar method to find VIN's location which is attributed of the <tbody>tag.
A screen shot is available on the next page.

```
response.css('a p:nth-child(3)::text').extract()
```

Figure 6: Extract Example

Extracting Info

Web Crawler Advancements

- Getting the anchor element like"next" or "next page" to scrape all pages on specific website.
- Downloading extracted photos to local.
- Consolidating the extracted data of salvage car into database.

Web View

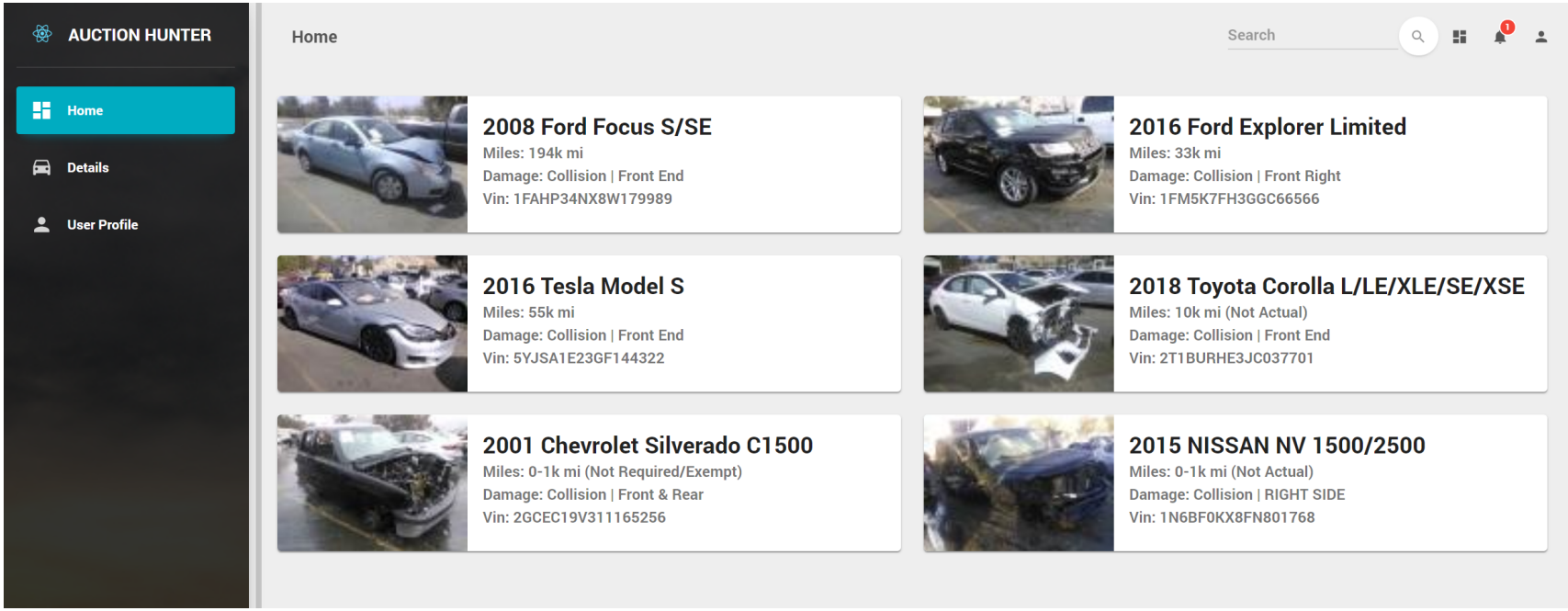

Figure 7: HomePage of Auction Hunter

Website Interface

- Backend written in Python using Django library
- Backend interacts with database to display data to user
- Frontend written in JavaScript using React framework
- Frontend uses Rest API to interact with the backend

Web Code snippet

```
return (
  <div><Grid container spacing={24}>
    {this.cars.map((car) => {
      return ( <Grid item xs={6}>
                  <AuctionCard
                    carImage={car.carImage}
                    carName={car.carName}
                    miles={car.miles}
                    vin={car.vin}
                    damage={car.damage}
                  />
                </Grid>
              )
          })} </Grid></div>); })
```

Figure 8: web

Demo