# Blitz

Suchwinder Singh
David Yuen
Kent Zhang
Rohan Tohaan

12/9/2020

Hunter College - CS499 - Major Capstone Fall 2020 - Final Demo

https://github.com/Suchwinder/Blitz

# Product Definition

Problem: Splitting Expenses after Hangout

Target Audience: Any person who hangs out

Vision: Improve splitting expenses process

Goals: Make the bill splitting process collaborative, and easy to reference

Strategy:

1. Mobile friendly web app
2. Receipt Processing
3. Easy Group Joining

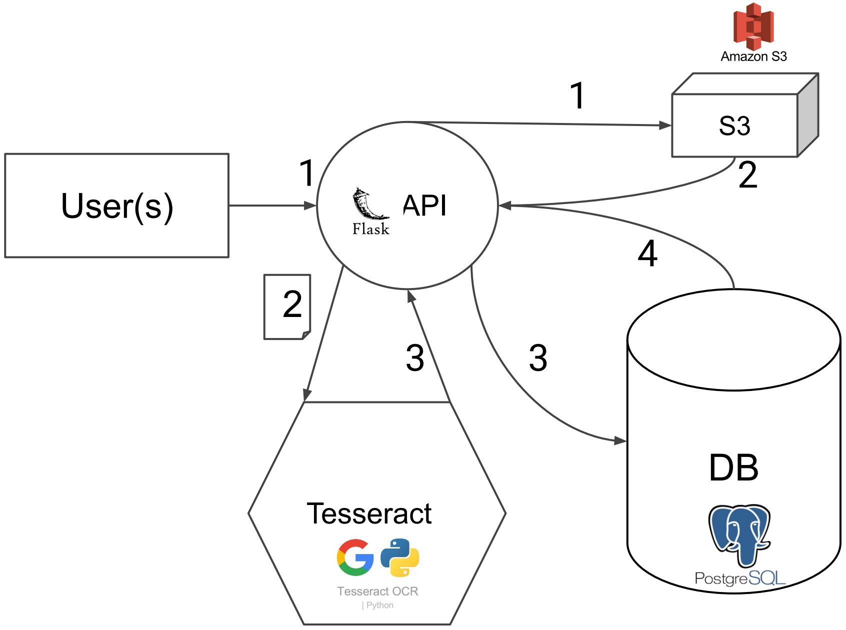
# Demo

# Tools/Technologies/Sources

| | | | |
|---|---|---|---|
| React.js | Flask/Python | Amazon S3 | SQLAlchemy ORM |
| Heroku | Pytesseract/OpenCV | PostgreSQL | Socket.io |

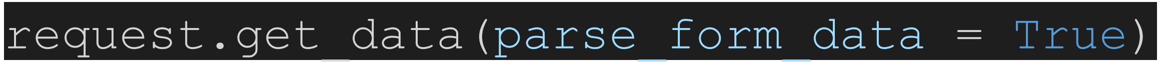
# General Architecture

1. User uploads image and sends information (user, location, etc.)
2. Image is sent to Pytesseract and the data is extracted and parsed
3. Parsed data and user information sent to database
4. Stored information returned to create split bill page



# Challenges: File Transmission

- Sending Form Data to Image Store Endpoint
- Couldn’t find the file key
- Realized the way requests handled need to use:



```
request.get_data(parse_form_data = True)
```

- Allows us to now access key to store image
- Tiny flask issue caused a few hour delay and long Sunday evening
- Lesson Learned:
  - Verify how the requests are handled by the framework
  - Print statements are old but gold

# Challenges: Deployment to Heroku

- Database wasn’t detected properly
  - Downgraded SQLAlchemy-Utils (to be discussed later)
- Getting Heroku to read from the correct file/directory
  - Gunicorn had an unfamiliar syntax in reading the app correctly
- Getting Heroku to have the correct build dependencies
  - Required Aptfile so that Tesseract and its dependencies are installed on the host machine
- Deployment with Socket.io
  - Having multiple workers in the Procfile created ‘rooms’ for concurrent users
  - Rooms split up users that were in the same group

- Lesson Learned: There are always new curveballs with deployment and Flask App deployment is time consuming

# Challenges: Packages & Dependencies

- SQLAlchemy-Utils was updated to 0.36.8 in July 2020
    - 0.36.8 has a bug with parsing database URLs
    - Logs pointed fingers at its dependencies Psycopg
    - Issues indicated bug appeared from 0.36.7 -> 0.36.8
- Lesson Learned: Error logs aren’t always accurate

- Socket.io had a major version update to 3.0 on Nov. 5, 2020
    - Flask Socketio was not up to date when we started working on sockets
    - Client-side Socketio was 3.x while Flask Socketio was 2.x
    - Socket Connection was unable to be established
- Lesson Learned: Check for versions when installing packages

- **Resolution:** Check package compatibility and downgrade

# Individual Contributions

- Suchwinder Singh - Backend (Database setup, API, Sockets), Frontend functionality, and deployment
- David Yuen - Backend (Text Recognition, Cost-Splitting API)
  - Resolved API related bugs
  - Frontend - Implement Image cropping
  - Assisted with Sockets and deployment
- Kent Zhang - Frontend (Developing the web application UI/UX)
  - Assisted with React components
- Rohan Tohaan -  Frontend(React components, UI/UX)
  - Assisted with S3 bucket

# Thank You!

Any questions?

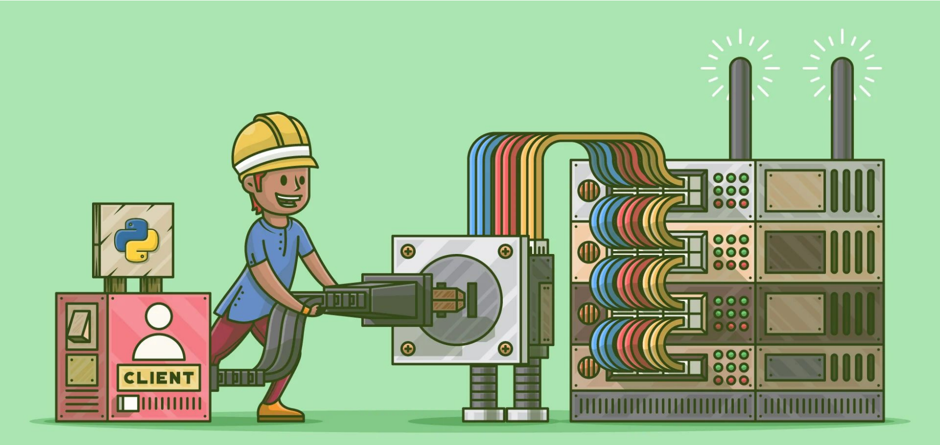
# WebSocket

Suchwinder Singh
David Yuen

12/9/2020

Hunter College - CS499 - Major Capstone Fall 2020 - Tech Talk

# What are WebSockets?



- An API that makes it possible to open a two-way interactive communication session between the user's browser (client) and a server
- Can send messages to a server and receive event-driven responses without having to poll the server for a reply
- Key here - not having to poll server: so no closing and recreating a connection

# WebSockets vs HTTP

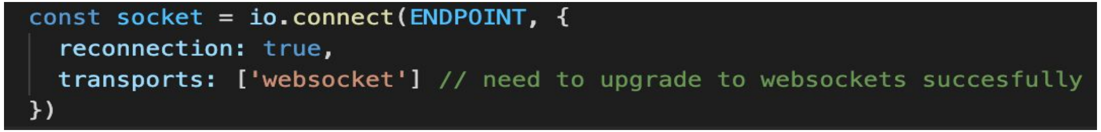
- HTTP - request/response protocol
- Rides on top of TCP (provides link between computers, and sends data described as packets)
- Endpoints have unique IP address, and do a three way handshake
    a. Client sends server a SYN packet with random number (ensures full transmission in correct order)
    b. Server receives segment and agrees to connection by returning SYN-ACK, and has clients sequence number + 1, and gives its own number
    c. Client acknowledges the receipt of SYN-ACK by giving its own ACK packet, also has number plus 1, and it begins transferring data

# WebSockets vs HTTP

- Bidirectional, full duplex communication
- Think of it as: it facilitates message passing between client and server (event driven)
- With HTTP client requests resource, server responds with requested data.
  - It is unidirectional, any data sent from server to client has to be requested first
- Long Polling was a means to work around this limit of HTTP where there is a long timeout period to push data to client, still takes lots of resources
- WebSockets allow for sending message-based data using TCP.
- It keeps TCP connection alive, using a different protocol that causes less strain on resources

# How to use WebSockets (High Level)

- In order to reach WebSockets you need to do a HTTP first
- The standard three way handshake performs once
- To change from HTTP to WebSockets need to add something to the request

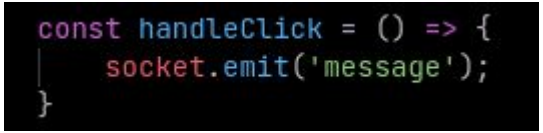


```
const socket = io.connect(ENDPOINT, {
  reconnection: true,
  transports: ['websocket'] // need to upgrade to websockets succesfully
})
```

- With this addition we can now let our server know we want to upgrade it to a WS connection.
- With this established we can now create socket communication

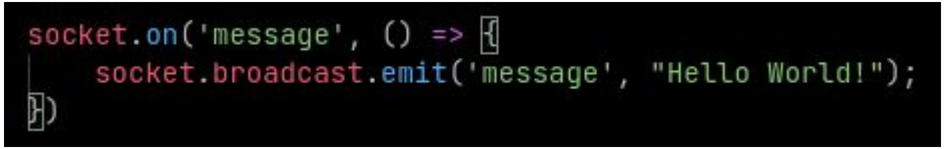
# How to use WebSockets (Example)

- A simple usage of sockets is to have a button on the client side that sends a message



```
const handleClick = () => {
    socket.emit('message');
}
```

- On the server side, it will broadcast the message “Hello World!” to all other connected clients



```
socket.on('message', () => {
    socket.broadcast.emit('message', "Hello World!");
})
```

# Thank You!

Any questions?

# Tesseract

Kent Zhang
Rohan Tohaan

12/9/2020

Hunter College - CS499 - Major Capstone Fall 2020 - Tech Talk

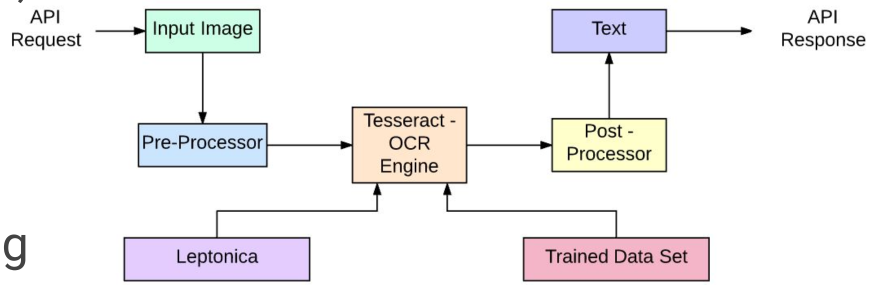
# What is Tesseract?

- Tesseract is an open source text recognition (OCR) engine
- OCR uses artificial intelligence for text search and its recognition on images
- Has the ability to recognize more than 100 languages out the box
- It is compatible with most programming languages with the use of wrappers
    - Java, Python, Swift, R, Ruby, C, etc.
    - Latest version of Tesseract (4.x)

# How does it work and how do we use it?

- Tesseract is finding templates in pixels, letters, words and sentences.
- User uploads image and apply any pre-processing to make it clearer
- Image sent to tesseract returns a string which is then parsed using regular expressions
- Print out the parsed regular expressions as text

OCR Process Flow



# Why should you use it?

- Tesseract is recognized as one of the most accurate open source OCR engines available
- Tesseract can read binary, grey, or colour images and output text.
- Can be configured to detect only digits, whitelist and blacklist certain characters
- Simultaneously detect multiple languages

# Cons of Tesseract

- Tesseract is not nearly as accurate as commercial solutions
- Needs the data to be pre-processed before the OCR can detect characters accurately
- Has a lot of errors due to distortion, noise, and poor background quality
- Is not capable of detecting handwriting
- Is known to find gibberish and include it in the output

Improving OCR Accuracy
Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut
enim ad minim veniam, quis
nostrud exercitation ullamco
laboris nisi ut aliquip ex ea
commodo consequat. Duis aute
irure dolor in reprehenderit in
voluptate velit esse cillum dolore
eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat
non proident, sunt in culpa qui
officia deserunt mollit anim id est
laborum."
Clean Up and Enhance Scanned Images

# Thank You!

Any questions?