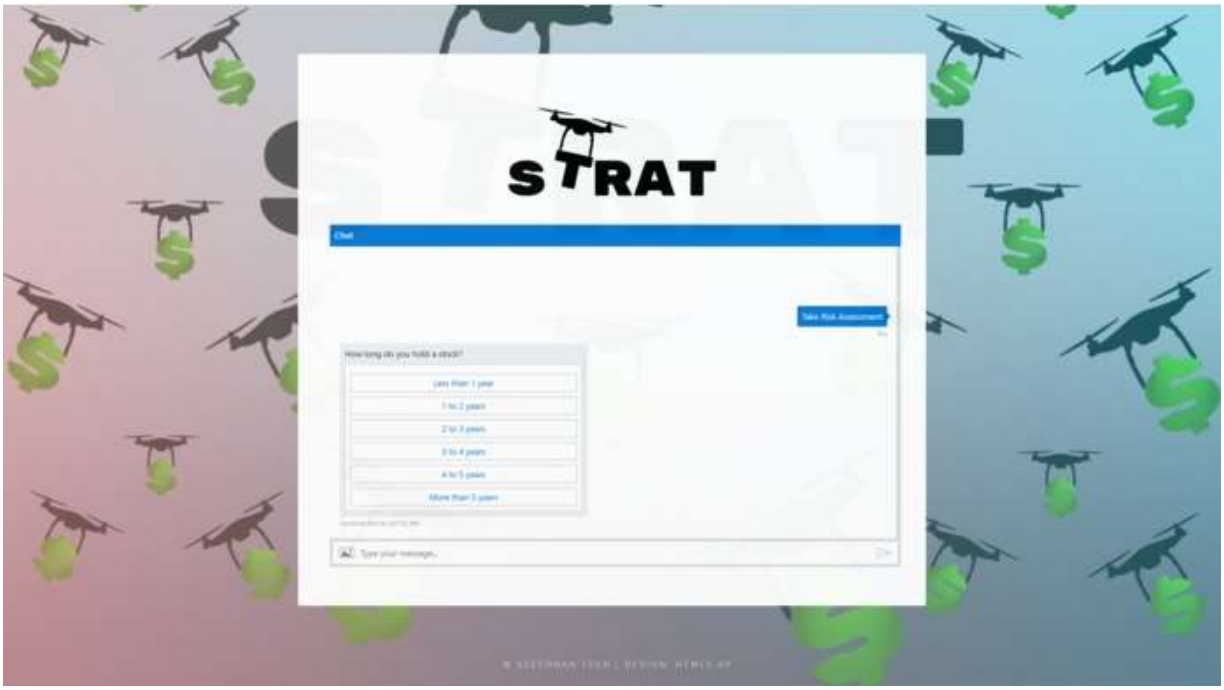
James Gan

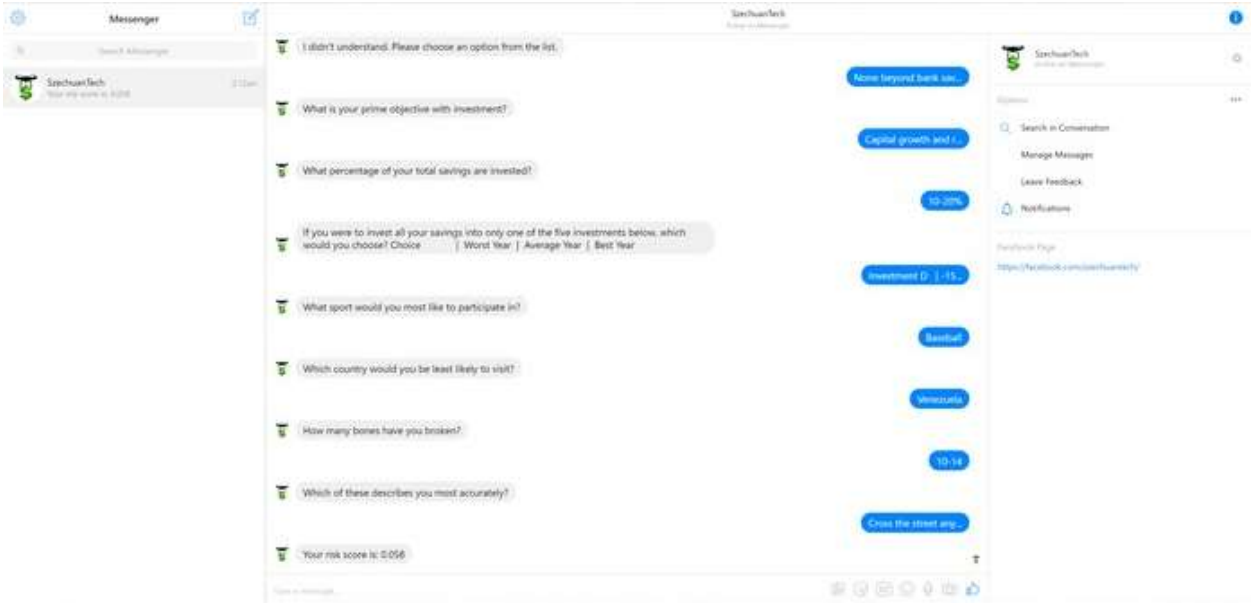
Digital Portfolio

More Projects @ http://bellevue.tech

Project 1: STRAT RoboAdvisor



Accessing STRAT through our website



STRAT is also available as a Facebook Messenger bot

STRAT brings professional risk management from BlackRock to individual investors looking to diversify investment portfolios and learn more about investments. Through an easy-to-use

chatbot, users can get suggestions to diversify and grow their portfolio, getting quick access to broken-down information about any stock.

Inspiration

Even in 2017, managing financial securities involves either long and tedious meetings with a financial advisor, cold and impersonal interactions with an algorithm, or a mix of both. We felt that there was an opening in the chat bot marketplace for providing personalized financial data and recommendations to every day users.

What it does

STRAT acts the user a series of personality-based questions to assess his or her risk profile, using the answers to calculate a number between zero (most risk-averse) and one (least risk-averse). These answers, influenced by university studies and traditional financial institutions' risk assessments, define the profile of the user for interacting with STRAT. Then, on prompt from the user, the bot can either (1) provide information about a specific financial security (e.g. "tell me about apple") or (2) make recommendations that fit the calculated risk profile of the user (e.g. "suggest a stock"). The bot makes recommendations for stocks that work to better match the user's investment portfolio's risk factor to their risk profile.

How we built it

The chat bot was built using the NodeJS implementation of the Microsoft Bot Framework. The bot runs on an Microsoft Azure instance and can be plugged into several different messaging services, such as Slack, Messenger, and Skype (or, more simply, on the Microsoft Bot Emulator). The bot invokes Python scripts on the back-end that fetch data from Blackrock's Aladdin REST API to be used to provide information about financial securities for the users and data for calculations about the risk of each security (so as to make the best recommendation to the user).

Challenges we ran into

Definitions of the terms on the Aladdin API were few and far between, so we had to do a fair bit of research to determine what metrics were best suited for our purposes. The data from Aladdin, too, was not perfectly matched with our intentions, so we had to do some hacking to get the data and metrics we wanted without making a bajillion HTTP requests. Additionally, our group was much more comfortable to fetch data from Aladdin in Python instead of Node, so we had to use some creative workarounds to call the scripts we wrote from the app.js file. This inadvertently led to the Python integration (and thus the entire bot) failing much more than we would have liked. Finally, for the longest time we were running our bot on a purely free instance of Azure, which murdered the performance and responsiveness of our initial protoype. It was only until we set up our bot, with the help of a Microsoft employee, on an instance with a pass for credit that it became functional.

Accomplishments that we're proud of

This was the first real hackathon for most of the group, so to churn out a project we all found interesting and loved building is something we can tell our parents about. We also built something that we felt filled a need and helped solve a common issue (or at least make valuable steps to do so). Some of us also found that we were able to thrive mentally despite very little (see: none) sleep.

What we learned

Documentation of a framework or API is nearly as crucial to their effectiveness as how well the technology is actually implemented. A corollary to that, it turns out, is that the more the implementation of a technology deviates from your specific use-case, the more creative you have to get when incorporating said technology into the functionality of your project.

What's next for STRAT

It would be cool to integrate the bot with the (unofficial) Robinhood API to allow the user to buy stocks on their Robinhood account after receiving a certain recommendation or quote. The bot could also be much more versatile if it took into account factors other than risk and volatility when making a recommendation (and also ask similar personality questions for those other factors). The next steps for this project would be to fetch investment data from services like Fidelity, Mint, or Robinhood and assess the risk of a user's portfolio as we make suggestions, in order to best diversity the user's portfolio.

Built With: python, node.js, javascrcipt, Azure, Microsoft Bot Framework, BlackRock's Aladdin API

See The Project In Action

- Public GitHub Repo
- Front-End Design (Bot no longer hosted)

HackGT 2017

WINNER Best Portfolio Data API Hack (BlackRock Winner)

Project 2: Easy Composites

Our Product: Fully Customizable, Generated By Software

```
Name of Organization?
tdx
College Name?
cornell
Border?
5
Separation?
2
Corners?
1 - Square
2 - Rounded
3 - Circle
2
Concatenate users yes -> y | no -> n
y
{'Background': ((160.625, 160.625, 160.625), '#a0a0a0'),
 'Border': 5,
 'College Name': 'cornell',
 'Name of Organization': 'tdx',
 'Separation': '2',
 'Text': ((200.78125, 158.6171875, 121.47265625), '#c89e79'),
 'concat': True,
 'files': {'ini': ['desktop.ini'], 'jpg': ['WIN_20160801_16_47_13_Pro.jpg']}}
```

Our Composite-Generating Software In Action

Easy Composites makes it cheap and easy to get composites for organizations. Traditionally, many Greek and Sport organizations spend nearly $1000 a year (or every few months) on a new composite. For a style of photography editing, it is simple and easy to automate: we managed to do just that. We worked to prototype code alongside some hired freelance developers and came up with a working product at a low financial cost. We are now able to use software to automate the creation of these photos, at a cost much lower than the current market price.

Inspiration

As a student at a school with a large Greek and Athlete communities, I learned that organizations were charged exorbitant prices for this service. As I studied Computer Science more for my minor, I realized that I could undercut the price of the two large photography service companies offering the service that hold the supermajority of market share.

What it does

We charge customers a set price, ranging from half off their previous price to $500 maximum for varying degrees of customization and support. At our highest price range, we offer similar services to the existing companies, including helping organizations print and frame the photos. We then run our software to create the photo for the customer, and send it along to the customer in various digital formats.

How we built it

My business partner and I built the software in Python with the assistance of some libraries, primarily svgwrite, which allowed us to manipulate photos in a perfectly scalable format (for high quality printing). At later stages of development, we hired freelance developers to help us add additional features, including a UI and customization functions.

Challenges we ran into

My primary issue with this project to date has been making sales. I was been able to find some level of interest, but unfortunately the market is saturated with long-term contracts (10+ years) which are already binding organizations to the two dominant companies. Even in cases where organizations have shown interest, I have usually been unable to sign them on due to their legal obligations to stick with other companies. In the future, I would gather more market research to ensure that we could make sales.

My other issue was making contact with the organizations which want my product: many of my relevant customers can be hard to reach, without reliable methods of contact across the board. I would hope to employ a larger sales budget and staff in future efforts to make sure that I can dedicate appropriate resources to connecting with a user base.

Accomplishments that I’m proud of

I was able to create a product that could theoretically generate hundreds of thousands of dollars a year. By automating a process which has traditionally been performed by graphic designers at a high cost, we would be able to take over a multimillion dollar industry with far fewer costs than existing companies.

What we learned

The potential for economic impact can be greatly reduced by implementation and roll-out. It will be important for me, in future projects, to make sure that my business plan has considered every aspect of sales, retention, operations, and growth.

What's next for Easy Composites

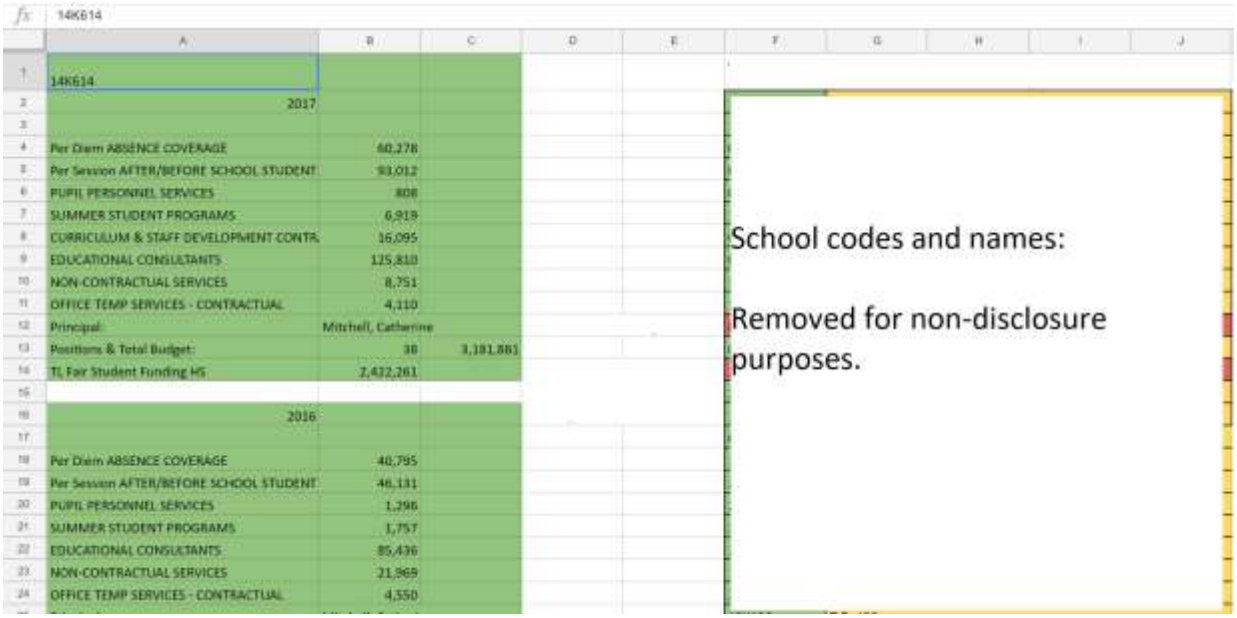
We are currently in the stages of finding on-campus representatives to connect us with organizations which will purchase our product. For every representative we hire at a school, we hope to be able to sign on 10 organizations with a steady revenue stream of $3000/school or more, and expand from there.

Built With: python, svgwrite, css, html

See The Project In Action

- Company Website With Pricing Tiers And Examples

Project 3: New York City Department of Education Budget Scraper



	A	B	C
1	14K614		
2	2017		
3			
4	Per Diem ABSENCE COVERAGE	60,278	
5	Per Session AFTER/BEFORE SCHOOL STUDENT	93,012	
6	PUPIL PERSONNEL SERVICES	808	
7	SUMMER STUDENT PROGRAMS	6,919	
8	CURRICULUM & STAFF DEVELOPMENT CONTR	16,095	
9	EDUCATIONAL CONSULTANTS	125,810	
10	NON-CONTRACTUAL SERVICES	8,751	
11	OFFICE TEMP SERVICES - CONTRACTUAL	4,110	
12	Principal:	Mitchell, Catherine	
13	Positions & Total Budget:	38	3,181,881
14	TL Fair Student Funding HS	2,432,261	
15			
16	2016		
17			
18	Per Diem ABSENCE COVERAGE	40,795	
19	Per Session AFTER/BEFORE SCHOOL STUDENT	46,131	
20	PUPIL PERSONNEL SERVICES	1,296	
21	SUMMER STUDENT PROGRAMS	1,757	
22	EDUCATIONAL CONSULTANTS	85,436	
23	NON-CONTRACTUAL SERVICES	21,969	
24	OFFICE TEMP SERVICES - CONTRACTUAL	4,550	

School codes and names:

Removed for non-disclosure purposes.

The Resulting Document From Budget Scraping: Itemized And Formatted Line Items By Query

5. DOWNLOAD THE "budget_webscrape.py" FILE AND MOVE TO DESKTOP
   - THE GREEN "CLONE OR DOWNLOAD" BUTTON ABOVE AND TO THE RIGHT --> DOWNLOAD ZIP --> UNZIP FILE --> MOVE TO DESKTOP
6. MOVE TO STEP 7

Returning Users

7. IN THE TERMINAL, USE CD COMMANDS ("cd .." GOES UP A FOLDER, "cd [FOLDERNAME]" GOES DOWN) TO NAVIGATE TO FOLDER CONTAINING BUDGET_WEBSCRAPE.PY

8. ADD SCHOOL ATS CODES INTO A FILE CALLED "schools.txt" SEPARATED BY ONLY NEW LINES. THE FILE MUST BE IN THE SAME FOLDER AS BUDGET_WEBSCRAPE.PY. EXAMPLE PROVIDED.
9. ADD KEYWORDS INTO A FILE CALLED "keywords.txt" SEPARATED BY ONLY NEW LINES. THE FILE MUST BE IN THE SAME FOLDER AS BUDGET_WEBSCRAPE.PY. EXAMPLE PROVIDED.
10. TYPE "python" AND HIT ENTER
11. TYPE "import budget_webscrape" AND HIT ENTER
12. TYPE "budget_webscrape.main()" AND HIT ENTER

FAQ

Clear Documentation And Non-Technical Usage Instructions

The Budget Scrape Program was created for my grant-funded internship with Practice Makes Perfect. It allowed users to input a list of school codes and keywords, and get concise budgets relating to those keywords for each school. For example, if the school district wanted to reduce spending on substitute teachers, then they would be able to quickly identify which schools overspend in that area by searching for all school codes, and the keyword “substitute”. If a company wanted to expand into water polo equipment, they could search all the schools for keyword “water polo”. This allows a quick querying system for the education environment in New York City.

Inspiration

At my internship I was working with the Founder and CEO of the company. He assigned me to a project to identify growth potential with current partners, and it required me to analyze over 100 budget pages. I realized that the time it took me to write software to automate the process would be nearly identical to doing that work manually, and save the company a lot of time in the future, so I requested time to work on figuring out how to automate the process.

What it does

The software helps users search through multiple web pages of the New York City Department of Education’s system for relevant information related to their work. It has many use cases for the Department of Education, education companies, and anyone invested in the education system.

How I built it

The software uses Selenium Webdriver to go through web pages, and parses them using Beautiful Soup 4. By going through the resulting data structures, I was able to parse and output relevant information to the user. The program is written in Python.

Challenges I ran into

Creating this software was my first time using web automation or large amounts of data. I was able to overcome these barriers and learn how to use these technologies fairly quickly by looking at existing open source projects for web automation and applying them to the problem I faced. I also had to watch a few hours of video lectures to learn how to use the two libraries to automate data collection.

Accomplishments that I’m proud of

My software was used to determine the growth strategy of a multimillion dollar education that has been recognized by the Clinton Foundation and TED. Practice Makes Perfect was founded as a nonprofit and has since become a B-Corp to aid in growth and impact; I was able to help a company with low budget and huge impact save hundreds of hours a year, which will allow them to more effectively grow and reach more students with their mission of social good.

What I learned

Many people and organizations in our society underutilize technology. Technology has the potential to positively impact millions of jobs in the United States; with the right knowledge, a simple solution can have an incredible impact. I hope to be able to make similarly large impacts in the future by learning more and more technical skills.

What's next for Budget Scrape

I wrote extensive usage instructions for non-technical users, and trained 25% of the company staff on how to use the software. They have used the data I gathered to help determine their annual growth and sales strategy, and plan to use it in the future to adjust their corporate strategy as they grow.

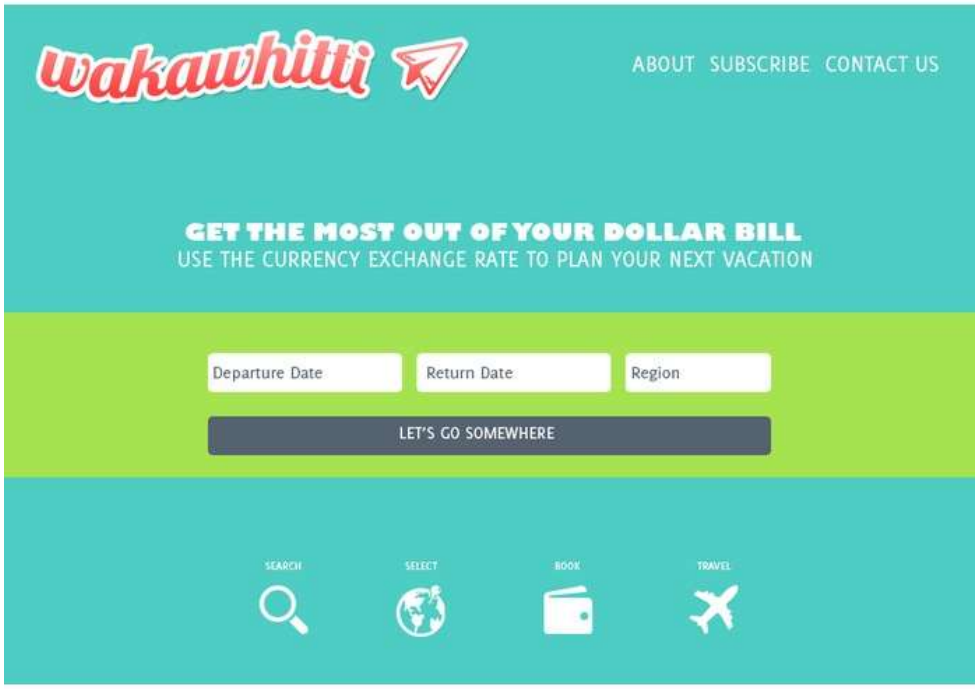
Built With: python, Beautiful Soup 4, Selenium Web Driver

See The Project In Action

- Public GitHub Repo

Practice Makes Perfect

Project 4: Wakawhitti



Wakawhitti Landing Page

Wakawhitti Destination Suggestion

Louisa is a poor graduate student but she really wants to go somewhere cool for spring break! The problem is she doesn't have a lot of money for her travel budget. Luckily, there is a travel site that can help her plan a vacation that can maximize the money she does have!

Enter Wakawhitti.

Wakawhitti is a recommendation engine for vacation destinations. Pulling real-time data from the global markets, Wakawhitti ranks countries to visit based on the relative change in currency exchange rate. When users are interested in a particular country, Wakawhitti will link super-sweet hotel deals for each major city from Priceline.com. Future updates will allow users to book their lodging directly from the site and take advantage of the changes in exchange rate and cost of living in foreign countries, as well as sign up for email alerts to find the best deals.

Inspiration

As a kid, I visited London with my family. Everything there was exorbitantly expensive: their currency traded practically 2:1 with the USD, making a bottle of water cost nearly $6. Since then, the GBP has dropped to 1.4 USD, and the purchasing power of USD in the United Kingdom has skyrocketed. This metric is not a perfect indicator of real-time purchasing power, but it can expose impressive chances to travel on a budget.

What it does

Wakawhitti uses Python, Django, and Bloomberg API to access the most accurate financial data and intelligently analyzes it to optimize your vacation by your budget. In particular, it compares currency value data in real-time to determine relative increases in purchasing power.

How we built it

The chat bot was built by connecting with Bloomberg's servers to access currency data, running analysis to compare it to historical data, and suggesting countries with the best purchasing power currencies as travel destinations through Priceline's deals,

Challenges we ran into

We had two back-end developers, one designer, and me. At the time, I had little technical experience. Regardless, I was able to step in as a product manager, QA developer due to some history coding, and generalist. I worked on design, front-end, back-end, and testing.

Accomplishments that I'm proud of

This was my first hackathon, and I was able to win an award from a top financial technology company in the world. I attribute this fact to our attention to detail throughout our project, including having a concrete business plan and a working minimum viable product.

What I learned

Communication is key when working with a team: my team was writing code for about a day before I realized that two of our developers working on back-end were actually writing overlapping functions. I helped them determine which of them would work on which aspects of our project, and we managed to finish the project on time.

What's next for Wakawhitti

Wakawhitti lived and died as a hackathon project. It could be easily extended with better financial data to better analyze historical trends to come up with better results: as is, our product primarily suggested countries with low currency value (e.g. Zimbabwe) instead of countries where the currency value dropped (e.g. United Kingdom after Brexit). By improving our system and releasing online, Wakawhitti would be a viable competitor to websites like Expedia and Kayak.

Built With: python, Django, html, css, Priceline's API, Bloomberg's API

See The Project In Action

- Public GitHub Repo

BigRed//Hacks
WINNER Best Use of Bloomberg API