=

Lecture 15: Final Logistics

You're almost done!

- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



Schedule

4/4	Sign up for SEAS R&D Showcase
4/9 (lab)	Practice Presentation
Week of 4/14	Final Demo (code complete)
4/18	SEAS R&D Showcase Poster Submission
4/23 (lab)	Final Presentation
4/25	SEAS R&D Showcase
5/11	Final Package Due



- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



SEAS R&D Showcase

- Timeline

- 4/4: <u>Student application</u> due (one submission per team)
- 4/18: Poster submission due (one submission per team)
- 4/25: R&D Showcase (full agenda: https://showcase.engineering.gwu.edu/participate)

- Poster guidelines

- 36" x 48"
- Free printing: https://library.gwu.edu/3-d-and-large-format-printing
- Note: print the poster early may need up to 5 days notice

https://showcase.engineering.gwu.edu/participate



Example R&D Showcase Poster



BikeBuddy Ethan Cohen | Claes Boillot | Matt Gouvin | Adham Popal The George Washington University Department of Computer Science



The Problem

Despite having some of the best bicycle infrastructure in North America, Washington still sees cars hitting and killing cyclists at an alarming rate. BikeBuddy aims to route cyclists to their destination along safe, efficient, and reliable routes.

The Application



The Algorithm

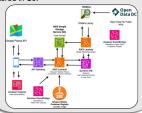
Our app pulls from OpenStreetMap as well as Open Data DC, an API provided by city government that contains data about cycling infrastructure, road safety, and crime across the District. The red dots in the map to the bottom left show the majority of data points collected.

We utilize the OSMnx in Python for graph creation, specialized for maps. Construction of graph data structures works by using our ingested data to add specific bike nodes and their attributes to the base graph for D.C. We utilize a cKDTree (plot on the left) for efficient lookup of bike nodes that will be added.

Routing works by modifying the weights of edges of our custom graphs. Thus, the shortest path from the graph's point of view will generate based on the weight modifications we have put in place. This can include user preferences, user reports, and bike attributes from our data incest.

The Architecture

BikeBuddy fundamentally lives in an AWS Lambda function that connects our front and back ends. From the mobile app, users can request routes as well as report unsafe conditions they encounter. An AWS EventBridge scheduler refreshes the data every morning so the map is always up to date. Our graphs and lookup structures are stored in S3.





Department of Computer Science School of Engineering & Applied Science



Engineering



Example R&D Showcase Poster



RTX CAPSTONE COMPETITION

MAE: Brendan Humphrey, Iftakhar Alam, Nitha Paulus, Ryan Rafati CS (UAV): Karl Simon, Leo Pham, Justin Park CS (UGV): Manue Alaimo, Dania Abdalla, Kayla Berne Advisors: Jarick Cammarato, Steven Shooter, Timothy Wood



Motivation

- The drone competition integrates advanced tech like AI, vision, autonomy, and 3D printing. This year's "Water Blast!" mission pushes the boundaries of unmanned vehicle technology
- Real-World Applications: By solving the "Water Blast!" challenge, teams contribute to the development of drone tech for tasks like package delivery and firefighting
- The competition is a high-energy event where months of hard work culminate in showcasing creations and competing against other top teams.

Dijective Engage students in the RTX drone competition.

- comprising seeker and evader challenges

 Develop algorithms for UAV navigation and object
- detection to meet competition requirements
- Design, prototype, and optimize water deployment and detection systems for UAVs and UGVs
- Execute precise coordination and strategic decision-making to overcome competition challenges
- Enhance students' understanding of engineering principles through hands-on application and experiential learning

Figure 1: UAV Circuit Accidedure

Figure 2: UGV Circuit Architecture

Implementation of Approach

Unmanned Aerial Vehicle (UAV)

- UAV is a hexacopter (6 propellers)
 Nvidia Jetson Orin Nano for image processing and route planning
- Flight controller equipped with three IMU sensors and a barometer
- 14.8 V 4000 mAh battery to power all UAV systems, capable of ~12 minutes flight time
- Camera Full HD, 60 fps, 128° FOV for
- ArUco marker detection
 GPS receiver for navigation
- Rangefinder for altitude measurement
- RF receiver for remote control

Water Deployment System

- Self-priming pump delivers a consistent flow rate of 8mL/second at 12v, controlled by a normally closed solenoid valve connected to the onboard computer.
- Adjustable nozzle optimizes spray pattern, while quick-connect tubing allows for fast assembly, disassembly, and future
- Custom-designed 3D-printed structures (ABS plastic) provide secure and reliable component placement on the drone frame

Unmanned Ground Vehicle (UGV)

- Basic frame with patterned holes for variable mounting
- 2 motor drive with 0.69 Nm, 294 RPM motors powered by a 7.4 V 5200 mAh battery
- Powered by a 7.4 V 5200 mAh battery
 Raspberry Pi 4 Model B powered by a separate 5 V battery
- ROS2 workspace with nodes and topics to facilitate communication between all sensors

Water Detection System

- Hydrophobic-coated ArUco marker placed on a angled (3.5 degrees) funnel allows water to flow seamlessly into the funnel while remaining easily detectable by the sensor
- The water detection chip, angled to prevent water build-up, triggers a dual response upon water contact. First, a signal alerts the team of water detection, and second, LED lights and an alarm on the UGV visually confirm the detection

Current Design Alton OPS Receiver Flight Controller Water Tank Weter Tank Weter Tank Veter Nozzle



Eleuro 3: Eull CAD Assembly





Figure 4: Photographs of current assembly

Algorithms and Testing



Figure 5: Camera footage from camera on UAV

- We tested our UAV at GW VSTC and UMD campus, outside DC flight restricted zone
- Users can view real-time camera footage and sensor readings from the UAV
- The UAV can be controlled remotely via remote controllers and computers
- The UAV flies autonomously 3 meters in the air to scan for ArUco Markers
- Once detected a non-ally marker, the UAV flies toward it and releases water
- When all non-ally markers have been targeted, the UAV flies back to its base

UGV

- We tested our UGV following each of the challenges
- Motor controls UGV at 3 constant speeds
- When UGV has been hit, alerts with lights and buzzer and logs the hit with time
- When hit UGV will immediately disable the motors and the UGV will stop
- UGV uses on board IMU sensor to make precise 90 degree turns and stay on a straight linear path

Acknowledgments

Special thanks to Jarick Cammarato, Professor Steven Shooter, Professor Timothy Wood, the GW 3D Printing Lab, the GW Machine Shop, & the Drone Lab

Example R&D Showcase Poster

CHAT GPTRAVEL

SIDRA HUSSAIN, EVAN FRIES, COLIN RANCK, SARAH JAGERDEO

PROBLEM STATEMENT

Our project aims to bridge the gap between knowing where you want to visit and having personal travel preferences, but not knowing what activities align with those preferences or how to optimize your itinerary at your travel destination.

Currently, many travel services create itineraries for users based on their preferences. For example, Wonderplan.ai creates itineraries based on factors such as budget, number of travelers, and activities of interest. However, these products often do not optimize their itineraries with objective measures like cost, ratings, or travel time.

Our project addresses this gap by generating travel itineraries based on user preferences, using activity suggestions from sources like Yelp and Foursquare. We then filter these suggestions with ChatGPT to align them with personal travel preferences, while also optimizing for objective measures such as cost. ratings, and travel time.

SYSTEM ARCHITECTURE

Our application is hosted on an AWS EC2 server running an ubuntu operating system. Therefore, the architecture design is all contained within AWS. The purpose of chatGPTravel is to take a destination, user preferences and an optimization metric from the user and return an itinerary for said destination using generative AI and the corresponding optimization algorithm. First, the user must log in to the application using the system UI. We maintain and update a user database. A logged in user can view their saved itineraries and generate new itineraries through the user interface.

The core of our application is the itinerary generation functionality. When the user enters a destination, dates, preferences and optimization metric, we acquire activities in the specified destination city from the Yelp and Foursquare APIs to yield activities with specific information, such as location, phone number, operational hours, rating and cost. We next filter the activities from the APIs based on the user's specified preferences using the openAl API and a set of natural language queries designed to yield the most accurate results. Finally, the system needs to decide which optimization algorithm to apply: fastest, cheapest, or best rated, which was provided during the user input period. The appropriate algorithm is applied to the activities to generate an itinerary for the specified number of days and then the itinerary is returned to the user.

Activity 1 Activity 3 Activity 4

YELP API

Generates restaurants located near the user's hotel address

FOURSQUARE

Generates activites near the user's hotel address

API

OPENAI API

Filters the Calculates the amount of time it restaurants and takes to travel activities based on users food and activity preferences

GOOGLE MAPS API

between

activities.

hotels

restaurants and google maps api.

COST OPTIMIZATION

The cost optimization algorithm filter through a list of activities and then sorts them from cheapest to most expensive. The algorithm adopts a greedy strategy, iteratively choosing neapest restaurant for each meal: breakfast, lunch, dinner, and non-me ntinues until the top five cheapest artivities are chosen for the day within their appropriate category. The itinerary for each day. Cost data is obtained using an approximation of costs, based on the dollar (\$) rating system provided by Yelp and

RATINGS OPTIMIZATION

Initially, the ratings optimization algorithm filters through a list of activities and selects clusters of toprated restaurants with ratings above stars for breakfast, lunch, and dinner. The algorithm adopts a greedy strategy, iteratively choosing the highest-rated restaurant within the specified cluster for each meal: breakfast lunch and dinner and nonmeal category. The selection process continues until an itinerary is created for each day. We obtain rating information from the Yelp or Foursquare API when the activity is initially generated.

TRANSIT TIME OPTIMIZATION

The transit time optimization algorithm adopts a greedy approach, it iteratively finds the activity that is closest to the hote then finds the activity that is closest to the first activity, second activity, and so on until the itinerary is completed to create an itinerary that is optimized based on transit time for each day. The transit time between two locations is found by finding the directions between two locations using the

COSTS AND RISK

Our project has a few costs associated with it. Particularly, we pay for an API key to use chatGPT, Google Maps directions API, AWS database services and AWS Web hosting services. We also use the free Yelp and Foursquare APIs. Both of these APIs have usage limits, therefore we have created multiple free keys to cycle through in instances when the usage limit is hit.

FUTURE WORK

n the future, we hope to improve our project in a few ways. First, we would like to improve the runtime of our application because it can sometimes take 5+ minutes to generate an itinerary. Additionally, we would like to add more features to improve our users' experience, such as the ability to customize generated itineraries by adding and removing to it, links to direct booking services, and better filtering of activities. Most importantly, we would like to improve our error handling when the application crashes or an API key limit is hit.



- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



Practice Presentation (4/7)

Logistics

- 7-8 minutes long
- Very similar to presentation 3 use slides as a starting point
- Everyone should be in-person
- Everyone should stay for the full duration (will likely run past 8pm, reschedule mentor meetings if needed)
- Upload slides <u>here</u>

- Goals

- Describe the problem & motivate the project
- Show an overview of the technical components
- Explain the full scope of the project
- Include a comprehensive demo to illustrate your work (can be pre-recorded)



Presentation Template (similar to presentation 3)

INTRO

Problem to be solved

Key Concepts from Writing 1

Users

- User Personas: 1-2 key users max

Impact

- How is this different from other apps?

Implementation

- Technical Designs: Diagrams, Algorithms, APIs, etc.
- If using Diagram, use a shortened version with the fewest components max (*No more than 5 is recommended*)

Algorithmic Components

- Explain the algorithmic complexity

Final Deliverable

- Describe all implemented functionality
- Recorded demo or screenshots (brief demo ~ 1-2 min)
- Can be a small portion of the presentation or interspersed throughout



TECH

DEMO

Final Presentation (4/23)

- Same as practice presentation, refined based on instructor feedback
- Will be held in Lehman Auditorium



- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



Final Demo (Week of 4/14)

Logistics

- Signup for a demo slot <u>here</u>; reach out ASAP if none of those times work
- Demo will be over Zoom

Pre-read Submissions

- 48 hours before your demo, you must share a team pre-read with instructors (send it over slack)
- Each team member must prepare at least one slide highlighting their top contributions throughout the year, including github links to key code
- These contributions should describe what was done technically, as well as how the work ties back to the project as a whole

Expectations

- Show end to end working system (can be recorded)
- Present the complete project, not just the work done after Demo 4
- Each individual should showcase their main contributions, which do not necessarily need to be the most recent work
- Come prepared to make efficient use of the time 30 minutes is not a lot!



- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



Final Package Submission (5/11)

- Team Website (update to include the following)
 - Project description
 - Bio of team members
 - Writing 4 (from the fall semester)
 - Final presentation slides (PDF)
 - SEAS R&D Showcase poster (PDF)
 - Link to GitHub repo(s) with all code
- Google Drive (upload the following to your team folder <u>here</u>)
 - README.md including project description + link to team website
 - Writing 4
 - Final presentation slides (PDF)
 - SEAS R&D Showcase poster (PDF)
 - Zipfile of GitHub repo(s) with all code



- Rest of Year Schedule
- SEAS R&D Showcase
- Practice Presentation + Final Presentation
- Final Demo
- Final Package Submission
- Mentor Meetings



Mentor Meetings

- Please practice your demos & presentations with your mentors and ask them to review your slides
- Your last required meeting with your mentor is next week (4/7), but feel free to coordinate with them if you want to meet the week of 4/14
- We've invited your mentors to your final presentations (in-person or over zoom) so they can see all the great progress you all have made!

