Project Timeline

## Team: Santa’s Little Helpers

### **Project Overview:**

Our group was formed immediately prior to the release of the official project description documents. After deciding on a project idea on 11/18 and refining the project on 11/22 based on feedback from Professor Densmore, we set out on an ambitious 2.5 week target timeline to produce a functioning Secret Santa web application. As all group members had little to no experience with Python (our language of choice for the project) or Flask (our Python web application framework), the first 1.5 weeks were largely dedicated to gaining the technical skills (via online tutorials and other sources) we would need to complete the project. During this “study” phase, we divided the group in half to focus on learning specific skills relevant to our application: two group members (Josh and Panat) focused on developing an algorithm for optimized random assignment of Secret Santa “pairs” while the other members focused on web development (architecture, front end, databases) and email functionalities. Following the end of the Thanksgiving recess, our group met to compare progress and further refine assignments. Over the following 1.5 weeks, we focused on finishing individual assignments, integrating features, and testing functionality.

### **Objectives**

Our objective was to develop an “Optimal Secret Santa” web application with the following functionality:

* The web application would be hosted on Heroku, leverage an underlying Postgres database to store user information, and have a user-friendly, visually appealing graphical user interface.
* A user would go to our website ([**optimal-secret-santa.herokuapp.com**](http://optimal-secret-santa.herokuapp.com/)) and fill out a submission form with the names and emails of members in their Secret Santa group. This information would be stored in our database.
* Upon submission of this form, a random assignment (“graph”) algorithm would immediately assign “partners” (aka, the group member who would be receiving a gift) to each member of the group).
* In addition, following the group submission, each member of the group would receive an email from our ‘email bot’ ([**optimalsecretsanta@gmail.com**](mailto:optimalsecretsanta@gmail.com)**)** with a unique link to fill out a wishlist.
* Each group member would then separately use the provided link to fill out their wishlist and this information.
* Upon submission of the wishlist by each group member, we would query the database to find this member’s designated ‘gift-giver,’ and then our email bot would send the gift-giver the name of their assigned partner and this person’s wishlist.

To accomplish these objectives, we needed the following technological features:

* Web application hosted on Heroku with a framework designed using Flask web development tools and templates built in HTML (to provide a visually appealing interface).
* Postgres database to store user information. This database was accessed using SQL commands and the SQLAlchemy python package. To ensure optimal pairing and prevent issues with querying, the database does not allow an email to be entered more than once (ie each email must be unique).
* Graph algorithm developed to provide optimal pairing (ie, prevent simple A->B, B->, C->A assignments and ensuring that no user was ever assigned themselves as a partner -- a problem that had been noted with previous Secret Santa applications).
* An email bot designed to use both Flask mail and Thread package (to send emails asynchronously and prevent “hang-ups” caused by the server trying to send all emails at once (which might lead the site to crash or the user to try to re-submit). This bot also leverages the Flask ItsDangerous package to generate unique URLs for each group member

### **Detailed Timeline and Meeting Minutes**

An overview of group meetings and activities are included below:

* 11/18/19 - Group Meeting (Kickoff)
  + First group meeting to decide on project
  + Agreed to develop an “Optimal Secret Santa” web application
  + Developed list of featured and required technologies which were submitted as an initial project proposal
  + Initial GitHub repository was created
* 11/22/19 - Group Meeting (Proposal Refinement)
  + Updated project proposal based on feedback from Prof. Densmore
  + Re-submitted proposal with modified design and successfully received approval for project
* 11/26/19 - Group Meeting (Pre-Thanksgiving Check-in)
  + Refined functionality requirements for the application. Features include:
    - Webpage where a user could register a group by entering group member names and emails
    - Optimized Algorithm to ensure random, fair, and efficient assignment
    - Email ‘bot’ to send initial email message to group members. A group member would then reply to the bot with a wishlist. The bot would read this wishlist and send it to that person’s secret santa (note: this feature was redesigned in a subsequent meeting on 12/1 as improper user input could cause issues).
  + Assigned algorithm design to Panat/Josh and web development to Ben (hosting/database), Wiley(email bot design), and Ken(front-end development)
* 11/26/19-12/1/19 - Individual research and skills development
  + During Thanksgiving recess, each group member used internet resources to learn Python, Flask, and other required technical skill sets
  + Each member developed their own Flask page (running on local servers) to experiment with features and learn syntactic structures
* 12/1/19 - Group Meeting - Post Thanksgiving Check-in and Coding Session
  + Group convened to discuss progress and share learnings (including existing functional local flask pages)
  + Panat demonstrated his working random-assignment algorithm and began work on a group submission functionality
  + Wiley refined and demonstrated an initial email bot (based on SMTP mail). The group decided to forgo having the email bot read wishlists and instead have a wishlist form on the webpage
  + Ben secured web hosting on Heroku and set up a PostgreSQL database
  + Ken continued to refine UI and presented a prototype on Wix
  + Josh developed classes to encapsulate groups and group members (including associated attribute)
  + Group Meeting to discuss front end, algorithm efficiency, and debugging.
* 12/1/19 - 12/7/19 - Individual Work
  + Panat improved efficiency of algorithm, and researched SQL syntax to extract emails
  + Ben started building and testing the group submission page. Ben also focused on ensuring that data could be placed in the database each time a new group submitted information on the website.
  + Ken continued to refine app UI including designing backgrounds and graphics
  + Wiley refined email bot by developing and testing functionality of Flask mail, bulk-mail sending, and asynchronous sending (the ultimate selection) and extraction from different data sources (CSV, list, dictionaries, etc.)
  + Josh worked on database extraction to enable a final ‘assignment’ email to include key information stored in the database
* 12/7/19 - Group Meeting -- Feature Demonstration and Development
  + Team got together to work on front end and database
  + Ben demonstrated hosted Heroku application and group members discussed how to integrate individual pieces into framework.
  + Additional emphasis on ensuring that we could query the database to send emails following “wishlist” submission by each group member
* 12/8/19 - Group Meeting -- Initial Feature Integration
  + Email bot (for sending initial email) and algorithm (for pairing each member with a “partner”) were tested then integrated into master branch
  + Initial testing of code to extract query and extract information from database upon individual wishlist submission
* 12/9/19 -- Group Meeting -- App Security and Continuation of Feature Development
  + Group researched cryptography and implemented encryption methods to hide sensitive information needed by the application (i.e. API tokens and email passwords) by leveraging Heroku configuration settings and environmental variables. Previous passwords were reset.
  + Functionality was re-vamped for wishlist submission. While previous designs would not allow wishlist emails to be sent until all group members had filled out a wishlist, the group developed a new design that would send wishlist individually to partners whenever a group member submitted their wishlist. This eliminates the need for a “Group ID” (which we were having issues putting in our database) and allow for multiple groups to use the application.
  + Functions for querying database and sending wishlists were redesigned
* 12/9/19 - 12/10/19 -- Individual Work -- Refining Features and Documentation Development
  + Initial wishlist page was designed and implemented
  + Database querying function was refined
  + User Interface was finalized
  + Group members not working on these features began assembling and refining project documentation
* 12/10/19 -- Group Meeting -- User Interface Implementation and Documentation Work
  + User interface code integrated with master application and debugged
  + Additional work on Project Timeline, SOW, Project Architecture, and Project Documentation documents
  + Initial attempt to integrate function to extract information from the database (to send out a “pairing” email following a wishlist submission. Further de-bugging needed.
* 12/11/19 -- Group Meeting -- Final Integration/Testing, Documentation Completion, and Video Creation
  + Remaining features surrounding wishlist page implementation, database extraction, and email bot features (ie sending final assignments and wishlists to group members) integrated and tested.
  + Application was tested extensively by Team
  + Project documentation was finalized and submitted
  + Project video filmed
* 12/11/19-12/13/19-- Individual Project Wrap-Up
  + Final tweaks to project code (if needed)
  + Project Video edited and uploaded to EC327 YouTube page