2 Project Plan

2.1 Project Management/Tracking Procedures

Which of agile, waterfall, or agile+waterfall project management styles are you adopting? Justify it with respect to the project goals.

We are using agile as a software project management style. We are using agile because it is more dynamic and decentralized. With our project expected to take two semesters and our advisor changing in the second semester, the agile framework will allow us to procure pieces of the project in every step early on for review. If, at any point, the client decides to add, remove, or change a part of our implementation, this can be done easier than with a waterfall style. The agile project management style is more inclusive to our highly motivated team. Because each piece is being worked on simultaneously, the team is more involved and able to make each section their own, versus a centralized waterfall design that employs a single team lead to run the entire project.

The goals of our project are to; produce a functioning product that fits the description of the project proposal, learn to work as an engineering team would in the field, and ensure each team member contributes to the actual engineering of the final product. The agile methodology was selected to better accomplish these goals by providing greater flexibility to our final implementation, providing a more dynamic role for each team member, and splitting the engineering aspects of the project so that each member would practice their trade.

What will your group use to track progress throughout the course of this and the next semester? This could include Git, GitHub, Trello, Slack or any other project management.

Our team uses a combination of GitHub and Discord to track our progress in this project. GitHub allows us to create a repository of research and code we have been working on. Communication, announcements, and planning take place on Discord as our team has a few available open spots in our schedule to meet. As the semester progresses, if our current tracking method proves failable, we may move to ProofHub as it combines the features of many other tracking software such as time management, calendars, repositories, and communication.

2.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Breakdown of all tasks linearly

- Obtain hosting platform
- Obtain APIs
    - Obtain Google Map APIs
        - Create accounts to get the keys
        - Learn how to use the API through documentation
        - Test functionality of APIs
    - Obtain Zillow APIs
        - Email The responsible department
        - Learn how to use the API through documentation
        - Test functionality of the API
- Develop a front-end-only product for the first iteration
    - Design the visual appearance of the website
        - Begin with pencil and paper
        - Get feedback from the client
        - Proceed to the next step and revisit for alterations
    - Create a graphic interface with HTML and CSS
    - Add functionality with JS
        - Ensure button clicks correspond with desired actions
        - Take input locations and store them for future use
        - Delete input locations from the data structure
    - Process inputs to be used by APIs
        - JS functions rearrange user input to the proper format
        - JS functions use the input as a variable for the APIs
    - Modify API output for results
        - JS functions use the output to create a final result
- Develop a traditional front-end UI/UX version
    - Modify the graphic interface
        - Final implementation split into individual files
    - Modify the functionality to support forums
        - Inputs are sent to the backend for processing instead of being used on the front-end
    - Process inputs from forums with PHP to be used by APIs
        - Inputs are processed on the backend more efficiently using PHP
        - The same output will occur as on the front-end design
    - Modify API outputs for results
        - PHP used to alter data for final results
    - Send results back to the users
        - Send results to the front-end for viewing

- Develop a back-end
    - Create the logic to create an accounts table in the database.
        - User inputs from the front-end in the sign-up form are processed in the backend to create user accounts and save their data on the database.
    - Create the logic to manage users' accounts.
        - The backend takes user inputs and modifies the account based on user requests.
    - Create logic to manage heatmaps requests.
        - User inputs from front-end are used to make API calls, and results are saved in the database and sent to the front-end for user to see
    - Create logic to manage distance calculator requests.
        - User inputs from front-end are used to make API calls, and results are saved in the database and sent to the front-end for the user to see
    - Create logic to manage users' saved records.

- Create database
    - Identify the requirements.
        - Understand the requirements of the system or application that the database will support.
        - identify what data needs to be stored,
        - identify how data should be organized.
        - Identify what operations will be performed on data.
    - Design the schema
        - define the tables, columns, and relationships between tables in the database.
        - Maybe try to use tools like ER diagrams to create the schema.
    - Choose a database management system..
        - Select a database management system that meets our requirements.
    - Set up the database.
        - Set creating the files and configuring the settings..
    - Create tables and columns.
        - Create the tables and columns in the database based on the schema we designed.
    - Define relationships.
        - Define relationships between tables, including primary keys and foreign keys.

.

Overview of sprints using a parallel model (Backlog, In Progress, Completed, Future)

Sprint 1: 01/30 - 02/19 All Base Requierments	Sprint 2: 02/19 - 04/19 Team 1 Creative	Sprint 3: 03/19 - 04/19 Team 2 Functionality	Sprint 4: 04/19 - TBD All Integration	Sprint 5: TBD - TBD All Final Front-End	Sprint 6: TBD - EOS All Final Product
Get Hosting Platform	Visual Design	Buttons	Input to APIs	Process Results	Modify Front-End
Obtain APIs	Graphic Interface	Save Input	Output from APIs	Display Results	Create Back-End
Test APIs		Delete Input		Bug Review/Testing	Create database
		Process Input			Bug Review/Testing

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 2.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns per/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprints).

Sprint 1; The use of the hosting platform and APIs will be at no cost to the development team.
The Hosting platform will be able to support front-end only, and UI/UX web pages

Sprint 2; The visual design will be pleasing to PC and Mobile users
The visual design will score 3.5 or higher out of 5 in a random campus review

Sprint 3; Input processing time will be negligible to the user
A maximum of 2 bugs in the styling implementation for this first iteration. (i.e. button does not highlight when hovered over)

Sprint 4; Total input processing time, including API usage, should be no greater than 800ms

Sprint 5; No more than 1000ms may elapse from the start of processing information to the final heatmap being generated.

Sprint 6; No bugs present in the program
No more than 1000ms may elapse from the start of processing information to the final heatmap being generated.

Performance: Our backend should be able to handle high traffic volumes and respond to quickly, with low latency and high throughput.
Scalability: Our backend should be able to scale up or down based on demand.
Security: Our backend and database should be secure and protected against common threats like SQL injection.
Flexibility: our backend has to be adaptable and handle a variety of frameworks, languages, and platforms.
Maintainability: Our backend should be maintainable, with clear and well-organized code and documentation.

2.4 Project Timeline/Schedual

• A realistic, well-planned schedule is an essential component of every well-planned project

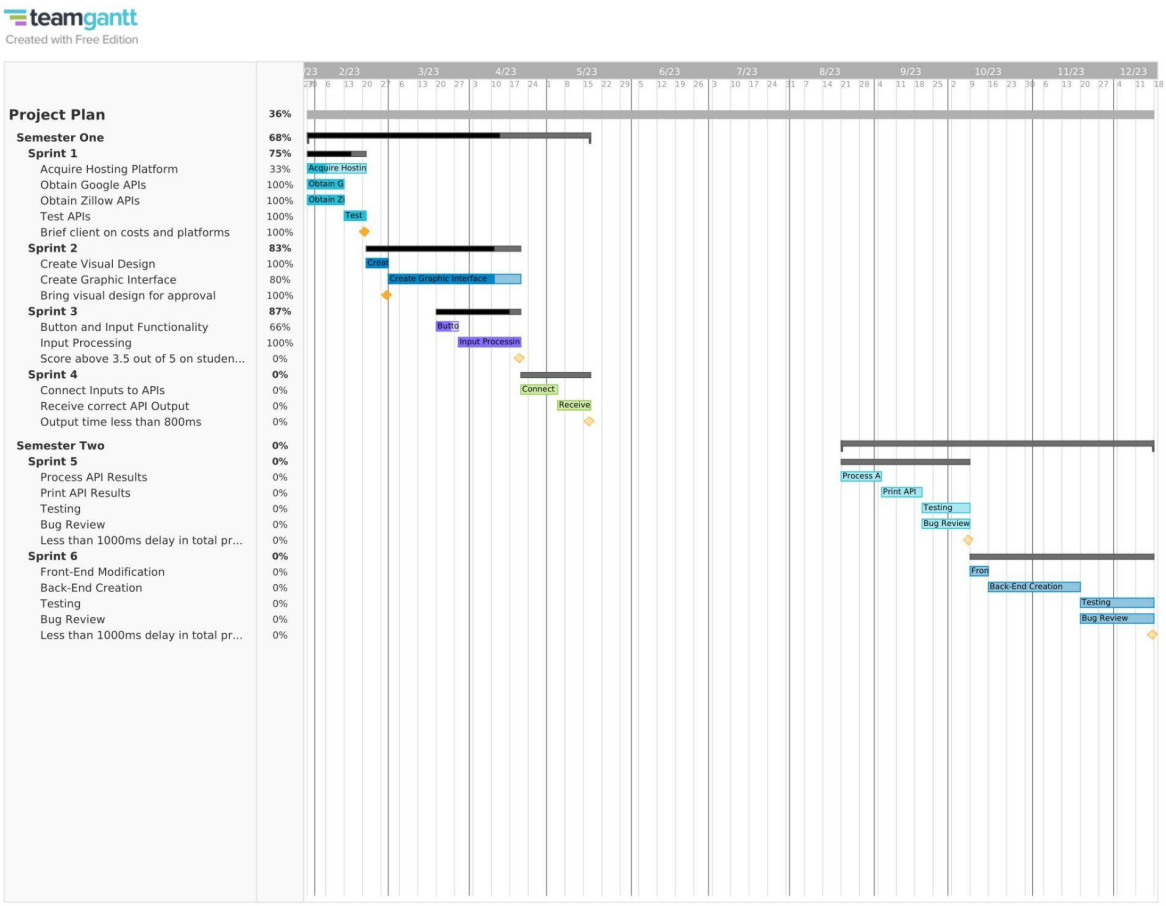
• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

– Start with a Gantt chart showing the tasks (that you developed in 2.2) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Project schedule/Gantt chart can be adapted to Agile or Waterfall development model. For agile, a sprint schedule with specific technical milestones/requirements/targets will work



2.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance target may not be met; certain tool may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Agile project can associate risks and risk mitigation with each sprint

- Obtain Host Platform
    - Many free hosting platforms consist of maximum requests
        - Buy a platform with no restraints
- Obtain APIs
    - We need to find the APIs to be able to have correct and current dated information

        - If a company denies us access we must find a different way to display the data
            - Find alternative real estate applications to use data from
- Develop the front-end-only product for the first iteration
    - Additional requirements are added by our client
        - Could cause a restart or run into issues such as needed further APIs
- Develop a traditional front-end UI/UX version
    - Risk level - low
        - At this point in the project we will have everything we need to develop a secure, cache oriented application that saves user data.
- Develop a back-end
    - Risk level - low
        - Security risks: include unauthorized access, data breaches, and hacking.
        - Scalability Risks: The backend may be challenging to scale up or down.
        - Integration Risks: our Backend must interact with a variety of systems and services, which, if done incorrectly, can result in compatibility issues, data loss, and other concerns.
        - Performance Risks: To provide a high-quality user experience, our backend must be quick, responsive, and effective.
- Create database
    - Risk level - medium
        - Data Integrity Risks: Our databases must ensure that data is consistent, accurate, and reliable. We can test our database to ensure that and if we have any problems we can redefine our tables and relations.

2.6 Personal Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in total number of person-hours required to perform the task.

Task	**Sprint**	**Time** (hours)
1. Acquire Hosting Platform	1	.5
2. Obtain Google APIs	1	.5
3. Obtain Zillow APIs	1	.5
4. Create Visual Design	2	5
5. Create Graphical	2	10

Interface		
6. Button and Input functionality	3	5
7. Input processing	3	10
8. Connect Inputs to APIs	4	5
9. Test APIs for correct output	4	3
10. Process API results	5	2
11. Print API Results	5	2
12. Testing & Bug review	5	7
13. Front End Modification	6	10
14. Backend creation	6	10
15. Backend input processing	6	10
16. Backend output generation	6	15
17. Database	6	10
18. Testing & Bug review	6	10

- Obtain Host Platform & APIs (Tasks: 1-3)
    - This portion of the project has been done for the requirements and functionality to the current date.
    - Each of these tasks is a basic acquisition step such as:
        - Contacting Zillow to request access to their API.
    - These tasks require minimal time.
- Develop the front-end-only product for the first iteration (Tasks: 4-8, 10-11)
    - These are the first steps in the design process
        - Visual appearance - 5 hours
        - Graphic interface with functionality - 10 - 15 hours
            - Create Graphical interface
            - Button & Input functionality/processing
        - Add API interfacing - 3 hours
        - Modify outputs for results - 2 hours
    - Overall these tasks will initially require a high input of hours, but as features are

designed and implemented other micro-tasks of this type will be easier and quicker to complete.

- Develop a traditional front-end UI/UX version (Tasks: 13-14)
    - Modify the graphic interface - 10 hours
    - Create Backend - 10 hours
        - Process inputs from forums with PHP to be used by APIs - 10 - 15 hours
        - Modify outputs and send them back to the user - 15 - 20 hours
    - After testing initial front end work, a good amount of time will be spent fixing and improving features and the overall design. Along with this work, we will start the creation of the backend. Backend work will continue after the improvements are implemented.
- Testing - 20 hours (Tasks: 9 & 12 & 15)
    - Tasks that require a large amount of time invested to upscale the quality and function of the site.

2.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) required to complete the project.

Because this is a software project, there are no other resources needed to complete it.

- Our project is all web-based, so we will not need material except purchasing of domains if needed, or further APIs if required.

- Programming Languages: our group can write the code for our web application using programming languages like Python, Ruby, JavaScript, PHP, and others.

- Frameworks: Our team could use frameworks like React, Angular, Django, Laravel, and Express to quicken development.

- Databases: Our group can store data for our web application in databases like MySQL, MongoDB, or SQLite.

- Cloud services: We can use cloud services for hosting, storage, and other infrastructure for online applications, provided by cloud services like Amazon Web Services (AWS).

- APIs: Our team can add features and functions to our web application by using APIs offered by third-party services like payment gateways, maps, and others.