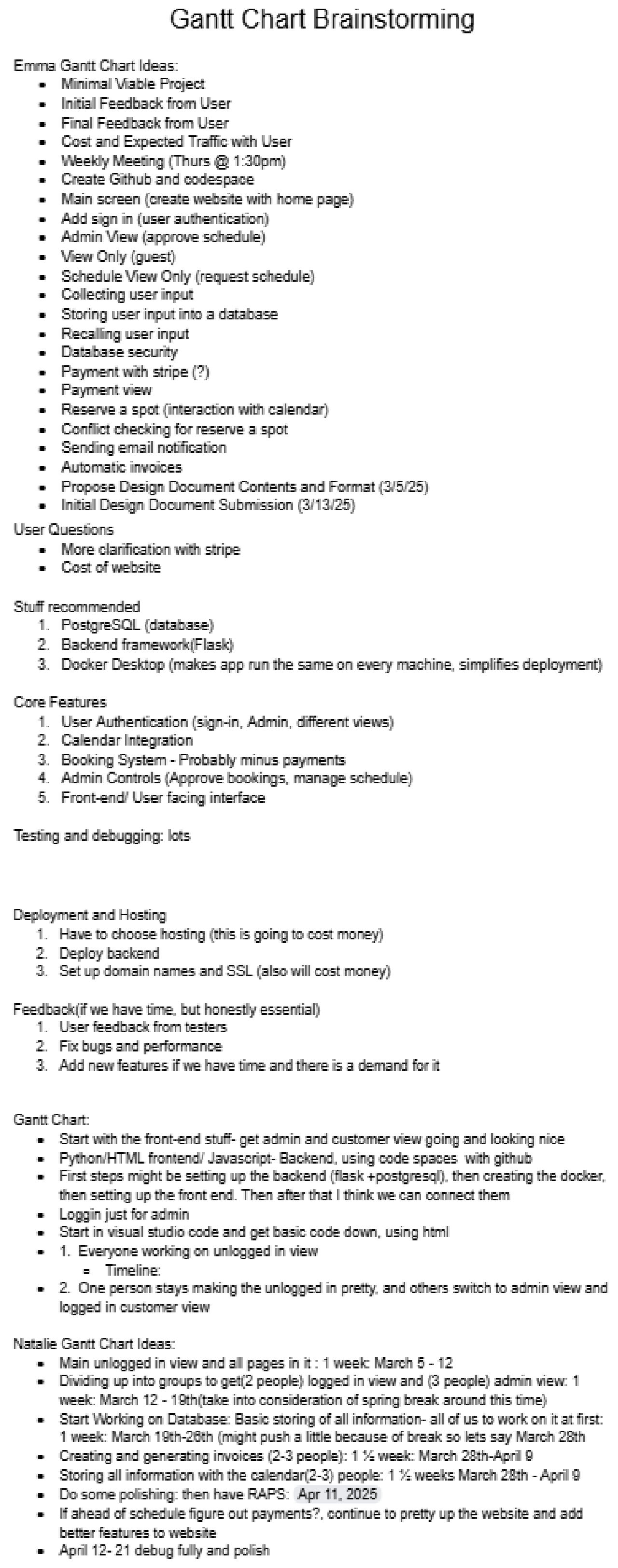
1) Take a picture/screenshot of your white board, post it notes, **Google doc**, or whatever you use for initial brainstorming.



Gantt Chart Brainstorming

Emma Gantt Chart Ideas:
- Minimal Viable Project
- Initial Feedback from User
- Final Feedback from User
- Cost and Expected Traffic with User
- Weekly Meeting (Thurs @ 1:30pm)
- Create Github and codespace
- Main screen (create website with home page)
- Add sign in (user authentication)
- Admin View (approve schedule)
- View Only (guest)
- Schedule View Only (request schedule)
- Collecting user input
- Storing user input into a database
- Recalling user input
- Database security
- Payment with stripe (?)
- Payment view
- Reserve a spot (interaction with calendar)
- Conflict checking for reserve a spot
- Sending email notification
- Automatic invoices
- Propose Design Document Contents and Format (3/5/25)
- Initial Design Document Submission (3/13/25)

User Questions
- More clarification with stripe
- Cost of website

Stuff recommended
1. PostgreSQL (database)
2. Backend framework(Flask)
3. Docker Desktop (makes app run the same on every machine, simplifies deployment)

Core Features
1. User Authentication (sign-in, Admin, different views)
2. Calendar Integration
3. Booking System - Probably minus payments
4. Admin Controls (Approve bookings, manage schedule)
5. Front-end/ User facing interface

Testing and debugging: lots

Deployment and Hosting
1. Have to choose hosting (this is going to cost money)
2. Deploy backend
3. Set up domain names and SSL (also will cost money)

Feedback(if we have time, but honestly essential)
1. User feedback from testers
2. Fix bugs and performance
3. Add new features if we have time and there is a demand for it

Gantt Chart:
- Start with the front-end stuff- get admin and customer view going and looking nice
- Python/HTML frontend/ Javascript- Backend, using code spaces with github
- First steps might be setting up the backend (flask +postgresql), then creating the docker, then setting up the front end. Then after that I think we can connect them
- Loggin just for admin
- Start in visual studio code and get basic code down, using html
- 1. Everyone working on unlogged in view
  - Timeline:
- 2. One person stays making the unlogged in pretty, and others switch to admin view and logged in customer view

Natalie Gantt Chart Ideas:
- Main unlogged in view and all pages in it : 1 week: March 5 - 12
- Dividing up into groups to get(2 people) logged in view and (3 people) admin view: 1 week: March 12 - 19th(take into consideration of spring break around this time)
- Start Working on Database: Basic storing of all information- all of us to work on it at first: 1 week: March 19th-26th (might push a little because of break so lets say March 28th
- Creating and generating invoices (2-3 people): 1 ½ week: March 28th-April 9
- Storing all information with the calendar(2-3) people: 1 ½ weeks March 28th - April 9
- Do some polishing: then have RAPS: Apr 11, 2025
- If ahead of schedule figure out payments?, continue to pretty up the website and add better features to website
- April 12- 21 debug fully and polish

2) Tell me what GANTT chart tracking software you've chosen to use (Excel, Google sheets, dotProject, etc.)

We've decided to move forward with Google sheets for our GANTT chart tracking software. It is relatively easy to use and is free. Everyone can also easily edit it and access it at any time.

3) Submit a PDF version of your initial schedule with half week intervals. It should show the critical path, dependencies among subtasks, who will do what, etc.

Submitted alongside this document.

4) Propose a plan for updating the GANTT chart each week. Will you meet to update it collectively? Will you identify one person who will be in charge of actually updating the GANTT chart and exporting a PDF version along with some high level notes about how it has changed (more subtasks added, some subtasks completed, critical path slipping, new estimates)?

We will all update the Gantt chart as we get done with our tasks and add any new stuff with features we might add if we do our plan A+. We are planning to have weekly meetings in which we will all discuss where we are at in our parts and look at the Gantt chart to see how we are doing. Here we will update the Gantt chart and take notes on the changes. Natalie will be the person in charge of the Gantt chart in basically writing updates/notes on changes made to the Gantt chart. Every group member will be involved with the Gantt chart, but she will be documenting the changes to it.

5) Describe your anticipated/target MVP as well as at least one "Plan B" MVP for if the schedule is slipping and a "Plan A+" if things are going really well.

The project is straightforward, The website works or it doesn't. The Plan B MVP would be a functional website which allows users to reserve slots with payments integration, and for admin to add/delete slots. Plan A+ product would have additional features like user accounts, adding dashboard to admin page, automatic invoice generation, google calendar integration, better ui/ux design with animations, flexibility for users to cancel slots and get refund, option for subscription service and discounts, a mobile app, notifications to remind users of their reservation.

6) What is on the critical path in your GANTT chart towards your MVP?
Plan B Product (MVP)

Create Github Codespace
Create basic booking calendar
Initial Design Document Submission
Frontend: Main unlogged in View
Frontend: Logged In View
Frontend: Admin View
Backend: Add sign in (user authentication)
Backend: Basic Storing Information
Backend: Creating and Generating Invoices
Backend: Storing Information with Calendar

Plan A Product (Ideal MVP)

In addition to the above critical path requirements, these would be completed to ensure an even more efficient minimal viable project to continue working on.

Cost and Expected Traffic with User
Testing: Collecting user input
Testing: Storing user input into a database
Testing: Recalling user input
Initial Feedback from User
Weekly Meeting
Frontend: Polish up the Views
Backend: Polish up the Database
Frontend: Debug and Look Better
Backend: Debug and Optimize
Final Feedback from User
Deploy the website

7) Identify a list of key risks to the success of your project. Make a plan for resolving the risks as early as possible.

Scope Creep & Time Constraints

Risk: Adding too many features may delay MVP completion.

Plan: Stick to the core MVP (scheduling, user roles, auto-invoicing). Additional features (payments, insurance tracking) can be a future agenda.

Technical Challenges

Risk: Issues with API integration (Google Calendar, Stripe) or database management.

Plan: Assign API research early, use well-documented services, and allocate extra time for integration testing.

User Adoption & Usability Issues

Risk: Users may find the system difficult or prefer existing solutions.

Plan: Get early user feedback, prioritize a simple UI, and ensure clear onboarding documentation.

Data Security & Privacy

Risk: Handling sensitive data (payments, customer info) may pose security risks.

Plan: Use secure authentication, encrypt sensitive data, and rely on trusted payment processors like Stripe.

Maintenance & Future Support

Risk: After the semester, who maintains the system?

Plan: Ensure clear documentation, automate key processes, and explore transition plans for long-term support.

Cost & Hosting Issues

Risk: Cloud costs may make the system unsustainable.

Plan: Optimize cloud usage, explore cost-effective hosting options, and limit unnecessary resource consumption.