1. Team info (10%)

- List each team member and their role in the project.
  - Anna Pettis: UI, Frontend
  - Danielle Maddox: Backend
  - Dylan Maring: UI, Frontend
  - Jacqui Benincosa: Backend
  - Kiann Skkandann: Backend
- Link to each project relevant artifact such as your git repo (this can be empty for now).
  - CrossWars Project
- Communication channels/tools:
  - Discord server

2. Product description (20%)

Abstract:

The NYT mini crossword has been locked away behind a paywall, leaving a void in the word game industry! We aim to create a free, interactive platform where users can challenge themselves individually, or race against their friends in real time. Our project combines the joy of a traditional crossword with the energy of social connection.

Goal:

Our system will enable users to enjoy crosswords for free, without paying for a subscription, and add the option to enhance puzzle solving with friendly competition.

Current practice: How is it done today, and what are the limits of current practice?

Currently, most crossword apps are either behind a paywall, or designed for solo play. For example Crosshare allows for different users to create, share, and play one another's crosswords but doesn't have a leaderboard to encourage community competition. And while the NYT Mini Crossword is arguably the most popular crossword platform and does encourage community with a leaderboard or the option to solve puzzles with partner mode it is unfortunately locked behind a paywall. This limits the user base, as new users are unlikely to easily find ways to play with their friends.

Novelty: What is new in your approach and why do you think it will be more successful than other approaches? Do not reinvent the wheel or reimplement something that already exists, unless your approach is different.

CrossWars is different because it combines a social aspect to puzzling. Allowing you to see in real time how your friends are faring at a puzzle will broaden the user base, and draw in people who are interested in competition as well as puzzles. Hopefully with enough time, CrossWars will reach the same popularity as the NYT mini - all while being free to use. We think the competitive addition, as well as having no financial barrier will make CrossWars more successful than competitors.

Effects: Who cares? If you are successful, what difference will it make?

Our audience is passionate puzzlers who feel the same level of outrage that at the increasing appearance of paywalls for simple games. While we will aim to bring in puzzle newbies, we hope that setting a standard for free games will encourage large gamemakers to make their products accessible to anyone who wants to play.

Technical approach: Briefly describe your proposed technical approach. This may include what system architecture, technologies, and tools you may use.

We plan to make a web app with a simple and appealing UI which we will map out using Figma or a similar design platform. We propose a python backend and java script frontend and hope to learn how to incorporate AI into the backend to aid in generating the daily puzzle. We will need a database (Supabase or Microsoft Azure) to track user stats which we will write SQL queries to to display the stats on the users profile. We will use FastAPI for our API endpoints.

Risks: What is the single most serious challenge or risk you foresee with developing your project on time? How will you minimize or mitigate the risk? Don't state generic risks that would be equally applicable to any project, like "we might run out of time".

We think that a risk with this project is creating an engaging UI that differentiates our product from others. In order to make CrossWars unique, we'll need to emphasize the competitive and social aspect, going beyond existing implementations. We will carefully design the UI to make sure it is not just a variation of a similar platform, but introduces a new way for users to experience crosswords.

- Major features:
  - Integrate AI to generate our daily crossword
  - Single player vs invite a friend to compete against options
  - Event syncing to see an opponent's live progress (green highlighting on 'opponent board' where they have found words correctly)
  - Timer to track play time taken to complete crossword

- Stretch goals:
    - User dashboard with stats such as number of wins and fastest crossword completion time.
    - Randomly pair with a user who happens to be online at the same time

3. Use Cases (Functional Requirements) (30%)

Use Case: Inviting a Friend to Play (Danielle)

1. Actors:
    a. Player (inviter)
    b. Friend (invitee)
    c. Our Crossword Platform
2. Triggers:
    a. Player selects to invite a friend when choosing to start a new game
3. Preconditions:
    a. Player is on the platform
    b. Player has selected a puzzle to play
    c. Friend has a way to receive the invite link (e.g., email)
4. Postconditions (success scenario):
    a. Friend is able to access the invite
    b. Invite navigates them to the game
    c. Both players are admitted to the game (same information is revealed to them simultaneously)
5. List of steps (success scenario):
    a. Player chooses a game to play
    b. Player selects to invite a friend over solo play
    c. Platform generates a unique invite link
    d. Player sends the invite to a friend
    e. Friend receives the invite (e.g., text or email notification)
    f. Friend clicks the link and accepts the invitation
    g. Platform allows both players into the game room
    h. Short count down and game begins
6. Extensions/variations of the success scenario:
    a. Invitee has an account and signs in before the game begins
    b. Inviter can cancel the invite
7. Exceptions: failure conditions and scenarios:
    a. Invitee declines invite
    b. Invitee never responds (link expires after set time)
    c. Invite is not delivered

    d. Error prevents players from being admitted to game

Use case: Solo game play (Jacqui)

1. Actors
    a. Player
    b. Our Crossword Platform
2. Triggers
    a. The Player chooses solo play from the main menu
3. Preconditions
    a. Player is on the platform
    b. Player has selected Solo Play
4. Postconditions (success scenario)
    a. Player is navigated to game
    b. Puzzle is revealed
5. List of Steps (success scenario)
    a. Player chooses Solo Play
    b. Player is navigated to short countdown
    c. Puzzle is revealed and Player begins game
6. Extensions/variations of success scenario
    a. If Player is logged in, their stats for the new game they are in will be recorded for their stat dashboard
    b. Player's time will be recorded and presented upon game completion
7. Exceptions/Failure conditions:
    a. Time out– if players game time exceeds 30 minutes, time out, log this stat as DNF for their dashboard

Use case: Viewing personal stats (Anna)

1. Actors
    a. Player
    b. Our Crossword Platform
2. Triggers
    a. The player chooses the option to view their personal stats from the main menu
3. Preconditions
    a. The player has a valid account and is logged in
    b. The player has used the platform to record some gameplay data
4. Postconditions (success scenario)
    a. The player successfully views a dashboard of their personal stats (races completed, wins/losses, times)
5. List of steps (success scenario)

    a. The player logs into their account
    b. The player navigates to the main menu
    c. The player selects “Stats”
    d. The system retrieves the player’s statistics from the database
    e. The system loads and displays the player’s personal stats in a table
6. Extensions/variations of the success scenario
    a. The player can filter stats by time period (this week, this year)
    b. The player can view their stats in comparison to their friends/a global leaderboard
7. Exceptions: failure conditions and scenarios
    a. No data is available (the player has not played any games)
    b. Database connection fails

Use Case: Backend puzzle generation (Kiann)

1. Actors
    a. Backend service (CrossWars server)
    b. LLM / crossword-generator pipeline (3rd-party libs / APIs)
    c. Database (Supabase)
    d. Admin or scheduled job
    e. Player (solves puzzle)
2. Trigger
    a. Player clicks “Play” (solo or multiplayer) — the backend is asked to return a puzzle instance immediately. Players may request puzzles multiple times in the same session.
3. Preconditions
    a. Backend service is running and reachable.
    b. Generation pipeline (crossword-generator ± LLM) credentials/config are available.
    c. Database is available for storing user accounts and previous puzzles
    d. Potential cost constraints for LLM calls are configured
4. Postconditions (success)
    a. A puzzle instance (grid + clues + solution) is produced or retrieved and returned to the player.
    b. The puzzle instance is stored in the Database (so repeats are not possible).
    c. If possible, the instance is linked to a multiplayer room or a player session.
5. List of steps (success)
    a. Player selects “Play” (solo or invite friend). Frontend calls backend endpoint.
    b. Backend checks generation policy and cache and creates a fresh puzzle instance on-demand.
    c. Backend runs generation (LLM optional) → crossword-generator → validation (fillable, clues present, safety).

   d. Backend returns puzzle JSON to client and keeps a short-lived server-side record for the active session only.
   e. Player begins play; if solo, player can request a new puzzle at any time (regenerate) — except when in a locked competitive room
   f. Player begins play; completion times and results are recorded for that instance.
6. Extensions
   a. Regenerate on repeat (solo only): If the generator detects the returned puzzle is a repeat for that player (or the player thinks it's a repeat), the player may request to regenerate instantly. Regeneration is disabled in competitive rooms
   b. Multiplayer (competitive): Generate exactly one shared instance per room; lock it so neither player can regenerate
7. Exceptions
   a. Generator/LLM fails → retry up to N times; if still failing, return user-friendly error (try again later)
   b. Too many generation requests (abuse/cost) → return HTTP 429 and show cooldown message

Use Case: Racing against a friend (Dylan)

1. Actors
   a. Player
   b. Friend
   c. Crossword platform
2. Triggers
   a. Player selects option from main menu to race a friend
3. Preconditions
   a. Player and friend have accounts
   b. Player and friend are online
   c. Friend will accept to race
4. Postconditions (success scenario)
   a. Player and friend match into the same puzzle instance
   b. Countdown synchronizes revealing and starting the game
   c. Completion ends race and shows and records scores
5. List of steps (success scenario)
   a. Player accesses main menu
   b. Player selects race mode
   c. Platform lists friends sorted by online status
   d. Player may invite an online friend to race
   e. Friend accepts
   f. Countdown
   g. Player and friend compete to solve puzzle

    h. Player or friend completes puzzle
    i. Winner and scores shown
    j. Player or friend may choose to play again or return to menu
6. Extensions/variations of the success scenario
    a. Multiple friends are invited
    b. Sending expirable invites to offline friends
    c. Disconnected player may or may not be able to rejoin
    d. Player and friend finish at the same time?
    e. Anti-cheat? (e.g. Player or friend inputting impossibly fast)
7. Exceptions: failure conditions and scenarios
    a. Invited friend declines to race
       (Player is notified, removed from matchmaking process)
    b. Friend tries to use revoked or expired invite
       (Friend receives error message, or prompt to invite player again, or a new invite is sent silently)
    c. Puzzle fails to generate (*see § Backend puzzle generation*)
    d. Connection becomes unstable or drops
       (Disconnected user receives error message hiding puzzle and blocking game input; timeout counter starts)
    e. Player or friend times out due to inactivity
       (Connected user receives error message, wins by default)
    f. Scores cannot be recorded to server

4. Non-functional Requirements (10%)

- Secure database for users passwords and accounts.
    - All users must have a valid password and username
    - Players should not be able to alter their own stats through the client side
- Reliable platform even during peak hours
    - The platform should be available 99% of the time during peak hours
    - The system should be able to automatically restart in the event of a crash
    - If one player drops, it should not crash the race session for the other
- In the case that a user leaves a game open and running for days, the site should not crash. We will have a set time out (eg: 30 min) on each game session.
- Scalability
    - The system should handle scaling from a small project environment (less than 40 users), to potentially having hundreds of concurrent users with minimal reconfiguration

5. External Requirements (10%)

6. Team process description (20%)

- Specify and **justify** the software toolset you will use.
  - Bolt.new for rapid prototyping. None of our team has much experience with design and building a platform from scratch so being able to rapidly iterate on design and platform flow will help us to iterate on design and features quickly.
  - Socket.io (frontend and backend) For handling and displaying racing against friends. Will be helpful for managing multi-player events.
  - React.js (frontend) — Our team is most familiar with React, and we're confident we'll be able to find resources and support to aid us while we learn how to integrate it.
  - Figma (UI design) — We chose figma as it is an industry standard and simple to use tool that the front end team has the most experience in.
  - Python (backend) — We chose to use a Python-powered backend because the backend team has the most experience with it, thus minimizing the potential risk of a learning curve.
  - Supabase, a secure free to low cost cloud-hosted and managed PostgreSQL database. It will serve as a helpful database for storing user information from login to stats.
  - crossword-generator for crossword generation. Using an existing source for creating valid grids. This is helpful as it helps us leverage existing tools rather than re-inventing the wheel.
  - LLM to generate crossword input word list, clue generation, and puzzle theming. Typically the generator would use a static wordlist such as a dictionary. The LLM can help us generate a set of words based on specified themes which will allow us to create interesting theme-focused vocab along with generating clues for the words.
- Define and **justify** each team member's role: why does your team need this role filled, and why is a specific team member suited for this role?
  - Anna will be part of UI and frontend development. She will be responsible for working on the user-facing elements of CrossWars, and work with JavaScript to ensure the smooth interaction with the Python backend. With prior web design experience, she'll work to make the implementation unique and engaging for users, an important part of mitigating the aforementioned risk.
  - Dylan will be part of UI and frontend development. They have previous experience working with web technologies: HTML and CSS in particular, as well as TypeScript (a superset of JavaScript) and React. The team needs multiple people working on UI design and the frontend, to be able to work in parallel and focus on multiple areas at once.

    - Danielle will be part of the backend development team. She will focus on managing server logic and database interactions. She has prior full-stack experience, including integrating APIs, handling GET and POST requests, and managing regularly updated data in a database. She has also worked with Supabase in the past. Her front-end experience will allow her to effectively collaborate with members of the team focused on the front-end to ensure smooth communication between the client and server.
    - Jacqui will be part of the backend development team and assist with frontend as needed. Jacqui has worked as a full-stack engineer intern for a company which used a similar stack. She is excited about the AI integration into the backend for crossword puzzles and her interest in this aspect of the project combined with past experience makes her suited for the backend team.
    - Kiann will be part of the backend development team. Kiann worked as a backend engineer intern for one year at a local startup, where they built multiple websites and maintained production servers. They have also completed several React + Python projects (such as interactive portfolio websites, ecommerce) that required backend design and implementation, giving them practical skills in server-side development, deployment, and API design
- Provide a schedule for each member (or sub-group) with a *measurable*, concrete milestone for each week. “Feature 90% done” is not measurable, but “use case 1 works, without any error handling” is.

Schedule			
Week	Milestone	Team	Status
Week 3	Structural UI complete. Work on UI mockup via Figma or other design platform (for main menu, solo play, sending friend invite screen, competitive play, and stat dashboard). Should have at least the main menu and solo play designs by the end of week.	Frontend	Not started
	The backend is set up and ready to begin dev. We have a domain name and the site is hosted on an external server, as well as having a database set up to be hosted externally. Start thinking	Backend	Not started

Schedule			
Week	Milestone	Team	Status
	about database table schema (user table, puzzle table). We can spin up our localhost.		
Week 4	All mockups are done. Start on developing the menu page. Should be functional (buttons take user to new API endpoint) by end of week. Implement React routing.	Frontend	Not started
	Able to generate a crossword using the crossword generator where we just pick words.	Backend	Not started
Week 5	Implement crossword board grid UI and implement basic handling for filling in answers.	Frontend	Not started
	Work on AI generating the words	Backend	Not started
Week 6	Connect the puzzle grid to backend generated puzzles. Add a "generating puzzle…" loading state	Frontend	Not started
	AI can generate the words, we have working for solo play	Backend	Not started
Week 7	Create system for checking answers and showing users their crossword progress	Frontend	Not started
	Set up storage of old crosswords in db so that we don't get repeat crosswords/repeat hints	Backend	Not started
Week 8	Begin to integrate Socket.io, build race mode UI including a progress tracker, timer, and player usernames	Frontend	Not started
	Start on invite link logic and username storage	Backend	Not started
Week 9	Build the leaderboard and profile	Frontend	Not started

Schedule			
Week	Milestone	Team	Status
	paces, display user stats (win rate, average time, best time, number of puzzles completed)		
	Finish invite link logic and username storage, start on live syncing for competitive play	Backend	Not started
Week 10	Usability testing, add finishing touches. Fix remaining UI/UX issues.	Frontend	Not started
	Finish live syncing	Backend	Not started
Week 11		Frontend	Not started
	Any final touchups	Backend	Not started

- Specify and explain at least three major risks that could prevent you from completing your project.
    - Learning curve
        - Since the project involves integrating multiple technologies — including backend services, frontend frameworks, and possibly real-time multiplayer systems — We may face a steep learning curve. Some of us might need extra time to get familiar with new tools or frameworks (e.g., React components, WebSocket handling, or database schema design). This could slow down early development and testing phases if not managed with clear task distribution and documentation
    - AI integration
        - The project depends on integrating an AI or LLM-based system for crossword generation. This brings technical uncertainty: the API might produce inconsistent outputs, require high latency, or have cost and rate-limit constraints. Additionally, validating AI-generated puzzles for fairness, solvability, and appropriateness will require careful filtering and testing. If this pipeline fails or becomes too unreliable, gameplay could be disrupted
    - Time and Coordination Constraints

        - When several parts of the project (like the frontend, backend, AI, and multiplayer features) are being built at the same time, good teamwork is very important. If one part gets delayed — for example, if the backend isn't ready when the frontend needs it — the whole project can slow down. Since everyone has different schedules and workloads, clear communication and good time management are needed to keep things on track.
- Describe at what point in your process external feedback (i.e., feedback from outside your project group, including the project manager) will be most useful and how you will get that feedback.
    - User interface feedback will be useful throughout the UI design process to ensure that our implementation is functional and intuitive.
    - Would also be nice to get test users for our crosswords to see if people think our LLM is successfully generating reasonable puzzles/hints for the words. We can ask our various friends/peers to test it out and ask them at the end of their experience if they thought the hints were the right amount of challenging.