

KTM Roadmap

Here's a summary of what you've been asking for with **Kitchen Table Mode**, along with a potential roadmap for development:

Project Summary: Kitchen Table Mode

Purpose:

A web application to organize and manage dynamic tournaments for the game Splinterlands. The site should:

1. Allow **10-200 players** to check in and register for a tournament.
 2. Manage **dynamic rounds** where players compete and get eliminated, typically reducing the player pool by 50% each round.
 3. Track **player stats** like wins, losses, and points, dynamically updating based on results.
 4. Provide a **visual bracket or dashboard** for players and administrators to see progress and matchups.
 5. Be accessible locally first, with the potential to expand to a hosted site with a domain in the future.
 6. Utilize **HTML, CSS, and JavaScript** (with Java for API calls if necessary).
 7. Integrate with **Splinterlands APIs** to fetch and process player data and battle results.
-

Core Functionalities

1. **Player Check-In:**
 - A page where players enter their names to register.
 - Display a list of all registered players.
2. **Tournament Dashboard:**
 - A visual representation of all players, their stats (wins/losses/points), and their matchups.
 - Automatically update after each round.
3. **Dynamic Matchmaking:**
 - Randomly pair players for each round.
 - Track who wins/loses and update brackets or player standings accordingly.

4. Points System:

- Accumulate points for players based on their wins and performance.
- Use this to determine the overall tournament winner.

5. API Integration:

- Call Splinterlands APIs to validate players or fetch battle results.
 - Handle data for automated tournament updates.
-

Roadmap for Development

Phase 1: Foundation

- **Set Up Project Repository:**
 - Create a GitHub repository (`kitchen-table-mode`).
 - Add the basic file structure (`index.html` , `styles.css` , `script.js` , `README.md` , etc.).
 - Ensure `.gitignore` is set appropriately.
 - **Build the UI:**
 - Design a **Player Registration Page** (HTML & CSS).
 - Design a **Tournament Dashboard** to display player stats and matchups.
 - **Basic JavaScript Logic:**
 - Implement functionality for player check-ins.
 - Randomize matchups for the first round.
-

Phase 2: Core Features

- **Dynamic Rounds:**
 - Automate player elimination and bracket updates after each round.
 - Add logic for points and stats (wins/losses).
 - **Local Data Management:**
 - Use JavaScript objects or `localStorage` to track tournament data.
 - **Interactive Dashboard:**
 - Show real-time updates for player progress and matchups.
-

Phase 3: Splinterlands Integration

- **API Calls:**
 - Fetch player battle data from Splinterlands.
 - Validate results or use battle outcomes for tournament progression.
 - **Player History:**
 - Optionally pull historical data for more insights or stats.
-

Phase 4: Hosting and Expansion

- **Local Deployment:**
 - Test the application locally to ensure full functionality.
 - Optimize the site for performance.
 - **Domain and Hosting:**
 - Purchase a domain if needed.
 - Deploy the application using platforms like Netlify, Heroku, or AWS.
-

Deliverables

1. A functional **dynamic web application** for managing tournaments.
 2. Integration with Splinterlands APIs for real-time data.
 3. An intuitive UI with real-time updates for players and administrators.
-

Great! Let's proceed step by step to **clone your repository** and get started locally.

Step 1: Copy the Repository URL

1. Go to your newly created GitHub repository on the website.
2. Click the green **Code** button at the top right.
3. Copy the HTTPS URL (it will look something like this):

```
https://github.com/your-username/kitchen-table-mode.git
```

Step 2: Clone the Repository Locally

1. Open **Git Bash** (or your preferred terminal).
2. Navigate to the folder where you want to clone the repository. For example:

```
cd ~/Documents/dev-projects/
```

3. Run the `git clone` command with the repository URL:

```
git clone https://github.com/your-username/kitchen-table-mode.git
```

4. Move into the newly created folder:

```
cd kitchen-table-mode
```

Step 3: Verify the Clone

- Run the following to see the repository contents:

```
ls
```

You should see the `.gitignore` file and any other files (like a `README.md`) you created in GitHub.

Step 4: Add Your Project Files

1. Place your project files (e.g., `index.html`, `styles.css`, `script.js`) in this folder.
2. Verify the files are in the directory:

```
ls
```

Step 5: Stage, Commit, and Push

1. Stage the files:

```
git add .
```

2. Commit the changes with a meaningful message:

```
git commit -m "Add initial project files"
```

3. Push the changes to GitHub:

```
git push origin main
```

Step 6: Verify on GitHub

1. Go back to your repository on GitHub.
 2. Refresh the page to confirm that all your files have been uploaded.
-

You're now all set to work on your project locally and sync it with GitHub! Let me know how it goes or if you need help with any part of this. 😊

Great question! To create and edit those files with the provided code, you'll need a text editor or an Integrated Development Environment (IDE). Here are some recommended options:

Best Options for Editing Code

1. Visual Studio Code (VS Code)

- **Why use it?**
 - It's lightweight, free, and designed for web development.
 - Supports syntax highlighting and extensions for HTML, CSS, JavaScript, and Markdown.
 - **How to Use:**
 1. Open VS Code.
 2. Navigate to your project folder (`Kitchen Table Mode`) by clicking **File** → **Open Folder** and selecting it.
 3. Inside VS Code, create new files (e.g., `index.html` , `app.js`) by right-clicking the folder in the left sidebar and selecting **New File**.
 4. Copy the code I provided into the appropriate files and save them.
-

2. Notepad++

- **Why use it?**
 - A simple, lightweight editor that supports syntax highlighting.
 - **How to Use:**
 1. Open Notepad++.
 2. Create a new file and paste the code into it.
 3. Save the file in the correct folder with the appropriate extension (e.g., `index.html` , `app.js`).
-

3. Any Basic Text Editor

If you don't have a coding-specific tool yet, you can use:

- **Notepad** (Windows)
 - Open Notepad.
 - Paste the code and save the file in the correct folder with the appropriate extension (e.g., `index.html`).
 - **TextEdit** (Mac)
 - Open TextEdit and set it to **Plain Text Mode** (Format → Make Plain Text).
 - Paste the code and save it with the correct extension.
-

Recommended Setup

Since you have plans for web development, I highly recommend installing **Visual Studio Code (VS Code)** if you haven't already. It'll make managing and editing your files much easier as your project grows.

- **Download VS Code:** [Visual Studio Code](#)
-

Let me know which tool you'd like to use, and I can guide you step-by-step for that specific app!



You said:

I happened to already download VS code but I'm a noobie at it

ChatGPT said:

ChatGPT

Update memory?

"Has downloaded Visual Studio Code but is a beginne..."

Yes

No

No problem! I'll walk you through the steps for creating and editing files in VS Code. It's beginner-friendly, and you'll get the hang of it quickly!

Getting Started with VS Code

1. Open Your Project Folder in VS Code

1. Launch **Visual Studio Code**.
2. Click on **File** → **Open Folder** (or use `Ctrl + K, Ctrl + O`).
3. Navigate to your project folder:
`C:\Users\david\Documents\dev-projects\kitchen-table-mode`.
4. Select the folder and click **Open**.

2. Create New Files

1. In the **Explorer** panel (left sidebar), right-click on a folder (e.g., `src`).
 2. Select **New File**.
 3. Type the file name (e.g., `index.html` , `app.js` , or `styles.css`) and press Enter.
 4. The new file will open automatically in the editor.
-

3. Add Code

1. Copy the code I provided earlier (e.g., for `index.html`).
 2. Paste the code into the file in VS Code.
 3. Press `Ctrl + S` to save the file.
-

4. Organize and Repeat

- Repeat the steps above to create all the files (`index.html` , `app.js` , `styles.css` , etc.) in their respective folders (`src` , `docs` , etc.).
-

Quick Tips for VS Code

- **Preview Files:** Right-click an HTML file (e.g., `index.html`) in the **Explorer** and select **"Open with Live Server"** (requires installing the **Live Server** extension).
- **Install Extensions:** Search for helpful extensions in the Extensions Marketplace (`Ctrl + Shift + X`):
 - **HTML, CSS, and JavaScript** support (built-in).
 - **Prettier** for formatting your code.
 - **GitLens** for Git integration.

It looks like everything is working now! 🎉

The message `Everything up-to-date` confirms that all your changes have been successfully pushed to the GitHub repository. There's nothing more to commit or push, and your repository is fully synced.

What You Can Do Next:

1. Verify on GitHub:

- Visit your GitHub repository page to confirm that all the files and folders (`assets/` , `docs/` , `src/` , etc.) appear there.

2. Open in VS Code:

- Open your project in VS Code to continue working:

```
code ~/Documents/dev-projects/kitchen-table-mode
```

3. Start Coding:

- Continue working on your project. If you make further changes, you can repeat the Git process:
 - `git add .`
 - `git commit -m "Your message"`
 - `git push origin main`

If you need assistance with the project or GitHub workflows, let me know. You're all set to keep developing now! 🚀

Here's an updated **Roadmap for the Kitchen Table Mode Project** to help keep you on track and organized:

Roadmap for Kitchen Table Mode

1. Repository Setup (Completed ✅)

- ✅ Create a GitHub repository: `kitchen-table-mode` .
 - ✅ Clone the repository to your local machine.
 - ✅ Add necessary project folders (`assets/` , `docs/` , `src/`) and track empty directories with `.gitkeep` .
 - ✅ Commit and push all changes to GitHub.
-

2. Initial Project Development

Frontend

- ☐ Design the layout using **HTML/CSS**.
 - ☐ Create a **landing page**.
 - ☐ Add placeholders for tournament brackets, player lists, and results.
 - ☐ Ensure responsive design for mobile and desktop views.

Backend (Optional for Now)

- ☐ Decide whether to include a **backend component**.
 - ☐ Use Node.js for handling dynamic API calls (later).
 - ☐ Set up a local database (if needed) for managing tournament data.
-

3. Core Functionality

Tournament Management

- ☐ Implement **round eliminations**:
 - ☐ Start with 10–200 players.
 - ☐ Eliminate 50% per round until 2 players remain for the final.
- ☐ Enable **dynamic bracket updates**:
 - ☐ Update player progress and standings automatically.
 - ☐ Show match history and results.

Points System

- ☐ Implement **point tracking** for players.
 - ☐ Accumulate points over multiple tournaments.
 - ☐ Determine an overall winner at the end of a series.
-

4. API Integration

- ☐ Research Splinterlands APIs for tournament data.
- ☐ Integrate APIs for:

- ☐ Fetching player details.
 - ☐ Recording match outcomes automatically.
-

5. Testing and Refinement

- ☐ Test the app locally:
 - ☐ Use mock tournaments for functionality testing.
 - ☐ Test UI/UX for responsiveness and ease of use.
 - ☐ Debug and refine any issues.
-

6. Deployment

- ☐ Choose a deployment method:
 - ☐ Deploy locally for testing.
 - ☐ Purchase a domain for live deployment (optional).
 - ☐ Use a platform like **Netlify**, **Vercel**, or a traditional web server.
-

7. Future Enhancements

- ☐ Add user authentication to save player profiles and progress.
 - ☐ Incorporate leaderboard functionality.
 - ☐ Include visuals like player avatars or team logos.
 - ☐ Expand tournament formats (e.g., single-elimination, round-robin).
-

Day-to-Day Workflow Checklist

1. **Work on Your Local Machine:**
 - Add/edit code in `src/`, `docs/`, and `assets/` as needed.
2. **Use Git Regularly:**
 - Stage changes: `git add .`
 - Commit changes: `git commit -m "Your commit message"`
 - Push changes: `git push origin main`

3. Sync with GitHub:

- Ensure your remote repository is always up to date.
-

Kitchen Table Mode Tracker: Updated Summary and Roadmap

Project Summary

The **Kitchen Table Mode Tracker** is a dynamic tournament management web application for Splinterlands. It aims to provide two types of tournaments:

1. **Live Tournaments:** Run in real-time with scheduled start times, active matchups, and fixed time limits for matches (e.g., 5 minutes).
2. **24-Hour Tournaments:** Players can play their matches anytime within a 24-hour period. This feature is expected to be more active and utilize Discord for notifications and coordination.

Key features include:

- **Landing Page:** Introduces the tracker and provides navigation to various features.
 - **Tournament Creation Page:** Allows tournament directors to set up tournaments with details such as name, description, start time, and type.
 - **Tournament List Page:** Displays active and planned tournaments with links to their respective pages.
 - **Individual Tournament Pages:** Each tournament has its own page with:
 - A running list of joined players.
 - Brackets, winners, and advancement details.
-

Updated Roadmap

Phase 1: Core Features

✓ ~~Landing Page:~~

- ~~Mention the two types of tournaments: live tournaments and 24-hour tournaments.~~
- ~~Provide navigation links to the Tournament Creation and Tournament List pages.~~

☒ ~~Tournament Creation Page:~~

- ~~Set up a form to create tournaments with:~~
 - ~~Name, description, start time, and type (Live or 24-Hour).~~

☒ ~~Tournament List Page:~~

- ~~Display a list of active and planned tournaments with links to individual pages.~~

☐ **Individual Tournament Pages:**

- Show players who joined.
 - Dynamically generate and display brackets, winners, and advancement.
-

Phase 2: Tournament Automation

☐ **Auto-Start Live Tournaments:**

- Automatically start at the scheduled time.
- Dynamically generate matchups and update pages with active matches.

☐ **Timer for Matches:**

- Implement a 5-minute timer for live matches.
 - Use a best 2/3 format for now, with customization options later.
-

Phase 3: Discord Integration

☐ **Public Discord Server:**

- Create a server where anyone can join to view and interact with tournaments.

☐ **Discord Bot Integration:**

- Notify players about tournament start times, active matchups, and round updates.
 - Utilize the bot for both live and 24-hour tournaments.
-

Phase 4: API Integration

☐ **Match Results via API:**

- Use APIs to fetch match results in real-time for live tournaments.
 - Dynamically update brackets based on winners and losers.
-

Phase 5: Future Enhancements

- ☐ **Customization Options:**
 - Allow tournament creators to customize match settings (e.g., best of 5, adjustable timers).
 - ☐ **Guild Wars:**
 - Add a guild-based feature where all members of one guild play against another guild.
 - ☐ **Leaderboards:**
 - Implement leaderboards for players and guilds based on tournament performance.
-

Immediate Next Tasks

1. **Finish Individual Tournament Pages:**
 - Add features to display players, brackets, and winners.
 2. **Set Up Discord Bot:**
 - Notify users about live and 24-hour tournaments.
 3. **Add Timer Logic:**
 - Implement countdown timers for live matches.
 4. **Prepare API Integration:**
 - Research APIs for real-time match results.
-

Let me know if you'd like help implementing any of these tasks or refining the roadmap further!

