Mood Meals

Eat With Feeling

Angie van Rooyen 241077

Open Window, School of Fundamentals

DV 200

Tsungai Katsuro

24 July 2025

Table of Contents

Proposal

B) Client Conceptualisation & Problem Statement

Overview:

Mood Meals exists in the health & wellness space, targeting people who feel emotionally overwhelmed when trying to choose what to eat.

The Problem:

People experience:

- Decision fatigue around food
- Poor awareness of their mood-food connection
- Grocery waste from unplanned meals

Why Mood Meals Matter:

Mood Meals helps users:

- Choose meals based on how they feel
- Get suggestions from ingredients they already have
- Reflect on how food impacts their mood over time

Constraints:

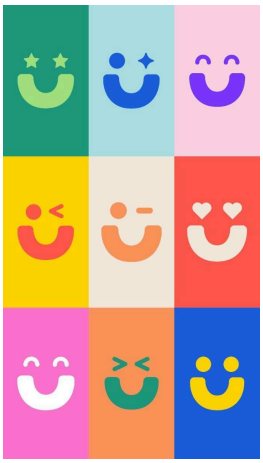
- Users need to self-report their mood accurately
- A recipe library must be curated or pulled from a reliable source
- Needs to remain minimal and fast to avoid adding friction

Branding:

- App Name: **Mood Meals**
- Concept Logo: Smiley bowl with pastel mood gradients or emoji-style facial icons.

Dynamic Logo System:

Mood Meals uses a flexible, mood-reactive brand mark. Based on the user's selected emotion, the app subtly changes its **logo icon** and **favicon** to reflect that state. For example, a smiling bowl may appear cheerful when "Happy" is selected, but shift to a sleepy or sad expression with softer colours when "Tired" or "Low" is logged. This dynamic branding deepens emotional connection and creates a feeling of being "seen" by the system.



C) System Architecture

High-Level System Flow:

User → React UI → Express API → MySQL DB

Tech Stack:

- **Frontend**: React (for dynamic UI and mood-state responsiveness)
- **Backend**: Node.js + Express (fast, scalable REST API)
- **Database**: MySQL (relational, structured for meal-mood tracking)

Justification for Tech Stack Selection

Mood Meals was designed to be a responsive, mood-sensitive, data-driven web application. The selected tech stack ensures scalability, speed, and ease of development, especially for a solo developer working within a tight academic timeframe.

Frontend: React

Why React?

- Component-based structure makes the UI modular and reusable, perfect for building mood-specific components like emojis, colour changes, and dynamic lists.
- Fast rendering through the Virtual DOM keeps user interaction smooth, which is essential when logging moods or browsing meal suggestions.
- Widely supported, with access to prebuilt UI libraries (e.g., Chakra UI or TailwindCSS) for fast and accessible design.

Backend: Node.js + Express

Why Node.js & Express?

- JavaScript is used on both the frontend and backend, reducing context-switching and increasing productivity.
- Node.js handles asynchronous operations well, great for dealing with mood logs, grocery input, and generating recipe suggestions in real-time.
- Express is minimalist and flexible, ideal for building a lightweight REST API that connects users to mood-meal logic.

Database: MySQL

Why MySQL?

- Structured data works best for this app (users, moods, meals, logs), and MySQL is perfect for that with its relational model.
- Strong support for JOIN operations allows easy querying between users, moods, meals, and logs.
- Well-documented and easily integrated with Node.js via Sequelize or MySQL2 npm packages.

Final Note:

This stack (React, Node, Express, MySQL) is known as a **full JavaScript stack with a relational database;** it's common in industry, easy to deploy, and scalable. It ensures **Mood Meals** will be:

- Lightweight
- Maintainable
- Fast to develop
- Familiar with future employers or collaborators

D) Feature Requirements & Scope

Scope:

Included:

- Mood logging
- Grocery input
- Recipe suggestions
- Meal-mood history

Excluded:

- AI-generated recipes
- External API integrations (for now)

SMART Objectives:

- Allow users to log moods & groceries in under 2 minutes
- Provide 3 relevant recipe suggestions per session
- Store & track mood-meal data over time

Feature Breakdown:

Must-Have:

- Auth system
- Mood selection & grocery entry
- Recipe suggestions
- Mood-meal history view

Nice-to-Have:

- Upload meal photos
- Mood-reflective UI animations
- Shareable “Meal of the Day” logs

Future Considerations:

- AI recommendations
- Friends/community sharing
- Mood boosting suggestions (e.g. “Snack for Sad Days”)

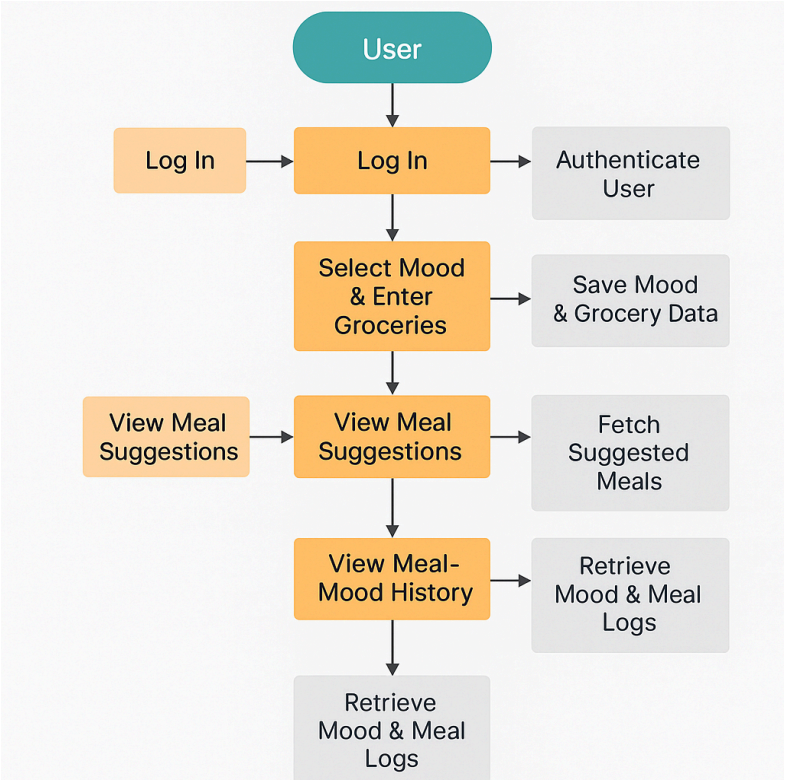
User Roles:

- Standard User
- Admin (for managing database content)

User Stories:

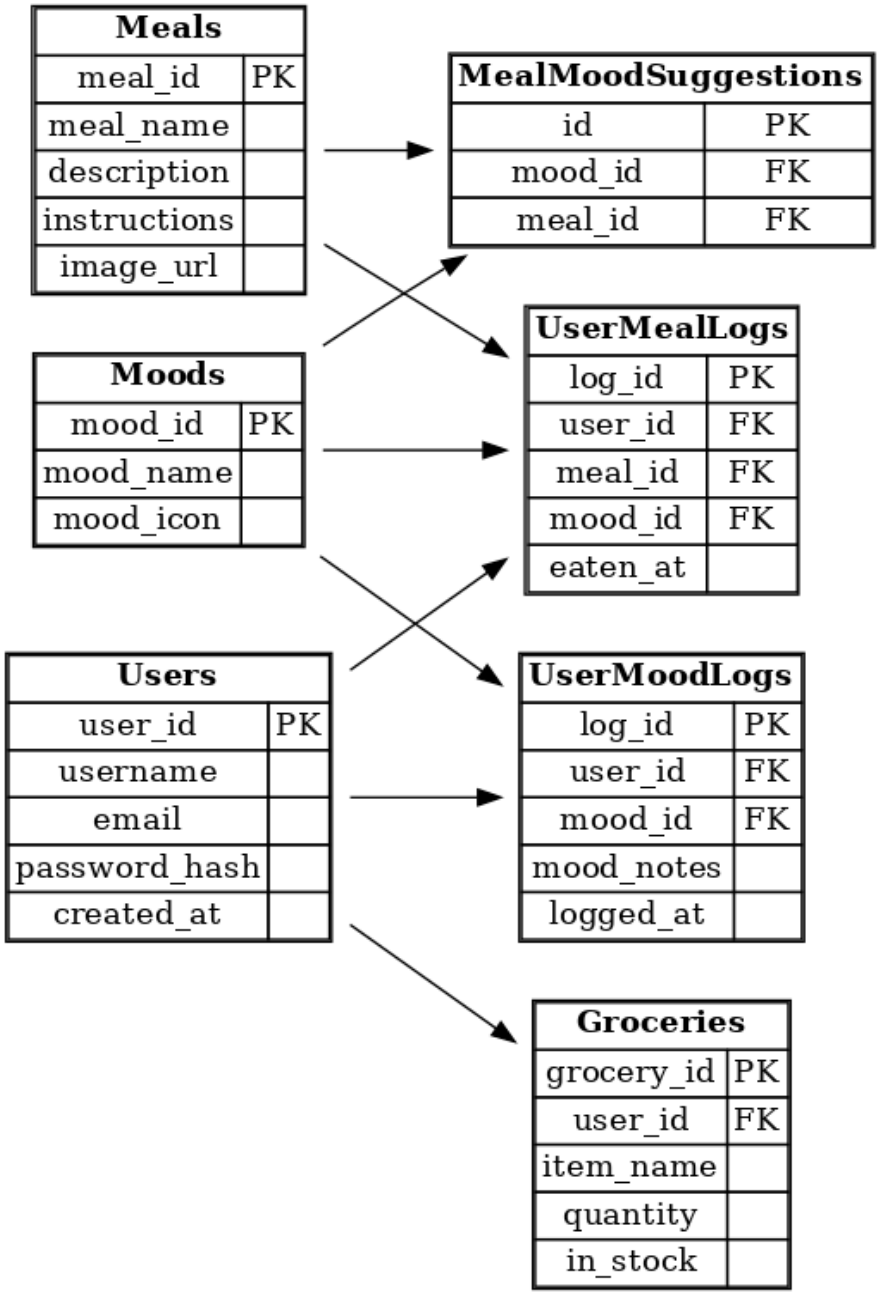
- “As a user, I want to log how I’m feeling and what I have at home so I get relevant meals.”
- “As a user, I want to track how my eating habits affect my mood.”

System flow diagram:



E) Data Planning

- **ERD**: Designed with Users, Moods, UserMoodLogs, Groceries, Meals, MealMoodSuggestions, and UserMealLogs.
- **Relationships**: One-to-many between Users and their logs, Moods and suggestions, etc.
- **Constraints**: Emails must be unique, user_id is foreign key across all relational data.



Explanation of Key Tables & Relationships

1. Users

This table stores all registered users. Each user has a unique ID and can log moods, groceries, and meals.

- **Relationships:**
    - One user → many mood logs
    - One user → many grocery entries
    - One user → many meal logs

---

2. Moods

Stores a fixed list of moods (e.g. Happy, Sad, Anxious) which users select when logging.

- **Relationships:**
    - One mood → many user mood logs
    - One mood → many meal suggestions (mood → meals)
    - One mood → many meal logs (for tracking what meals were eaten when in a specific mood)

---

3. UserMoodLogs

Logs each user's selected mood (and optional notes) daily.

- **Relationships:**
    - Belongs to one user

- Linked to one mood

---

4. Groceries

Tracks all groceries input by the user, including quantity and availability.

- **Relationships:**
  - Belongs to one user
  - Used to filter meal suggestions (no direct foreign key to meals, but connects logically)

---

5. Meals

Stores meal suggestions (e.g. recipes) with instructions and optional image.

- **Relationships:**
  - One meal → many mood suggestions (via MealMoodSuggestions)
  - One meal → many meal logs

---

6. MealMoodSuggestions

A link table that maps moods to suggested meals. This controls what meals are suggested based on the user's mood.

- **Relationships:**
  - mood_id → references Moods
  - meal_id → references Meals

---

7. UserMealLogs

Logs what meal a user actually selected/eaten and what mood they were in when it happened.

- **Relationships:**
  - user_id → Users
  - meal_id → Meals
  - mood_id → Moods

Data Types & Constraints for Key Fields

Field Name	Table	Type	Constraints
user_id	Users	INT	PK, AUTO_INCREMENT
username	Users	VARCHAR(50)	UNIQUE, NOT NULL
email	Users	VARCHAR(100)	UNIQUE, NOT NULL
password_hash	Users	VARCHAR(255)	NOT NULL
created_at	Users	DATETIME	DEFAULT CURRENT_TIMESTAMP
mood_id	Moods	INT	PK, AUTO_INCREMENT
mood_name	Moods	VARCHAR(50)	UNIQUE, NOT NULL
log_id	UserMoodLogs	INT	PK, AUTO_INCREMENT
mood_notes	UserMoodLogs	TEXT	NULLABLE
logged_at	UserMoodLogs	DATETIME	DEFAULT CURRENT_TIMESTAMP
grocery_id	Groceries	INT	PK, AUTO_INCREMENT
item_name	Groceries	VARCHAR(100)	NOT NULL

quantity	Groceries	VARCHAR(50)	NULLABLE
in_stock	Groceries	BOOLEAN	DEFAULT TRUE
meal_id	Meals	INT	PK, AUTO_INCREMENT
meal_name	Meals	VARCHAR(100)	NOT NULL
description	Meals	TEXT	NULLABLE
instructions	Meals	TEXT	NULLABLE
image_url	Meals	VARCHAR(255)	NULLABLE
id	MealMoodSuggestions	INT	PK, AUTO_INCREMENT
meal_id	MealMoodSuggestions	INT	FK → Meals(meal_id)
mood_id	MealMoodSuggestions	INT	FK → Moods(mood_id)
log_id	MealMoodSuggestions	INT	PK, AUTO_INCREMENT
eaten_at	MealMoodSuggestions	DATETIME	DEFAULT CURRENT_TIMESTAMP

F) Wireframes & UI/UX Considerations

Moodboard:

- Soft, pastel palette for a calming experience
- Mood icons (emoji-style), clean typography, white space

Mood-Reactive UI

Mood Meals' UI adapts not just via data, but visually. Key elements like the logo, favicon, background gradients, and icon sets reflect the user's mood. This contributes to a more personalised, emotionally supportive digital experience.

Key Screens:

Figma link -

https://www.figma.com/design/rX3bXzQlqHJXstfr5Clhda/Mood-Meals?node-id=0-1&t=v67xBpVEU6d0v3uu-1

- Homepage
- Login/Register
- Mood & Grocery Input Dashboard
- Suggested Meals Page
- Meal-Mood History Log

Justification:

- React allows for component reusability and fast UI state changes
- UX focused on low cognitive load for overwhelmed users
- Accessible colour contrast and minimal animations to avoid sensory overload

Accessibility Concerns & Considerations

1. Colour Contrast

- Pastel colours (used in the UI) may cause low contrast on certain backgrounds.
- **Solution:** Use tools like WebAIM's Contrast Checker to ensure all text and icons meet **WCAG AA** or **AAA** contrast ratios.
- Ensure minimum contrast of **4.5:1** for normal text and **3:1** for large text.

---

2. Icon Clarity

- Mood emojis and icons must be easily distinguishable, especially for users with visual impairments or colour blindness.
- **Solution:** Pair icons with **clear text labels** (e.g. "Happy 😊") and avoid using colour as the only cue.

---

3. Keyboard Navigation

- Users should be able to navigate the app **entirely via keyboard** (Tab, Shift+Tab, Enter).
- **Solution:** Add proper focus states to all buttons and inputs using semantic HTML and ARIA roles.

---

4. Screen Reader Compatibility

- All dynamic content (meal suggestions, mood logs, etc.) should be announced correctly to screen readers.
- **Solution:** Use aria-live, aria-labels, and descriptive alt text for all images and buttons.

---

5. Form Input Feedback

- Forms like the mood logger or grocery list must include **clear error messaging** and success indicators.
- **Solution:** Add accessible form validation using aria-describedby for inline feedback.

---

6. Motion Sensitivity

- Avoid excessive animations, especially mood transitions or loading spinners, which can trigger motion sensitivity.
- **Solution:** Respect the prefers-reduced-motion media query to offer a simplified, low-motion version.

G) Project Timeline & Workflow

Gantt Chart (Week-by-Week)

- Week 1–2: Backend setup + DB schema
- Week 3–4: Frontend dev with static data
- Week 5: API integration & dynamic suggestions
- Week 6: Testing & bug fixes

Milestones & Deliverables – Mood Meals

Week	Milestone	Deliverables
Week 3(Proposal & Setup)	Project Planning & Approval	- Final proposal PDF - ERD + system diagrams - Initial project repo created on GitHub - Database schema drafted
Week 3(Frontend Foundation)	UI Wireframes + Static Screens	- Figma wireframes for key screens - React components for Login, Dashboard, Mood Log, and Grocery List - Placeholder data for meals
Week 4(Backend Foundation)	Backend Setup & API Planning	- Node.js + Express server setup - MySQL database connected

		- User auth routes (register, login) - API endpoints outlined (CRUD for mood, meals, groceries)
Week 9(Core Features)	Mood + Grocery Functionality	- Frontend CRUD for mood logs and grocery list - The backend API is fully functional for these modules - Mood-based meal suggestion logic implemented
Week 12(Integration & Testing)	Linking Systems & Testing	- Full frontend-backend integration - User can log mood + groceries and receive real meal suggestions - Unit testing & bug fixes - MVP completed
Week 16(Polish & Handover)	UI Polish + Deployment Prep	- UI/UX refinement (colour, contrast, usability) - Accessibility audit - Deployment to Vercel/Render - Final testing, documentation & walkthrough

---

Deliverables Summary

- Project Proposal PDF
- ERD + System Diagrams
- Working REST API (Node + Express)
- React Frontend (Fully Integrated)
- CRUD Features: Mood, Groceries, Meals
- Final Documentation & Live Demo

Workflow:

- **Method**: Agile sprints
- **Tools**: Trello (task tracking), GitHub (version control), Figma (UI/UX)
- **Role**: Solo development — all areas handled by Angie

H) Risks, Challenges & Conclusion

Risks:

- Data mismatch between mood & meals → **Solution:** Standardise mood categories
- Scope creep → **Solution:** MVP-first development
- User testing delays → **Solution:** Create dummy users for simulation

Potential Risks & Mitigation Strategies

Technical Risks

Risk	Impact	Mitigation Strategy
API Bugs or Route Failures	Core features (e.g., meal suggestions, mood logging) could break	- Use Postman or Thunder Client to test all endpoints before frontend integration - Write clear route comments & modularise controllers
Frontend-Backend Sync Issues	Data may not render or update properly	- Define API contracts (request/response shape) early - Use consistent naming between backend and frontend
Database Query Errors or Data Loss	Broken app logic or user frustration	- Validate all SQL queries and test with dummy data - Implement error handling + fallback responses
Poor Mobile Responsiveness	Weakens UX, especially for mobile-first users	- Use responsive design with CSS Grid or Tailwind

		- Test early on mobile breakpoints
Deployment/Environment Config Problems	The app may not work online after the dev phase	- Use .env files correctly, test on Render/Vercel early - Document deployment steps for reproducibility

---

Non-Technical Risks

Risk	**Impact**	**Mitigation Strategy**
Scope Creep	Trying to add too many features = burnout	- Stick to MVP first, log all "Future Features" for later- Prioritise tasks weekly using Kanban
Time Management	Missed deadlines or rushed work	- Set mini-deadlines per feature using a Gantt chart- Reserve time for testing + refinement in Week 6
Lack of User Testing	Features may not meet real needs	- Create 2–3 dummy user personas to simulate feedback - Gather peer/class feedback in Week 5
Burnout or Overwhelm	The project stalls due to pressure	- Use mood tracking tools] - Take breaks, reflect, and manage tasks realistically
Data Entry Fatigue (for users)	Users may not want to log daily	- Keep UI minimal and fast (e.g. tap-to-log moods) - Use autofill and toggle states where possible

Final Thoughts:

Mood Meals addresses a common, relatable problem with an empathetic, tech-savvy solution. It seamlessly merges emotional wellness and food choices into a unified experience, driven by user data and minimalist design.

References

Macht, M. (2008). How emotions affect eating: A five-way model. *Appetite, 50*(1), 1–11. https://doi.org/10.1016/j.appet.2007.07.002

Jacka, F. N. (2017). Nutritional psychiatry: Where to next? *Epidemiology and Psychiatric Sciences, 26*(5), 468–473. https://doi.org/10.1017/S2045796017000062

Norman, D. A. (2004). *Emotional design: Why we love (or hate) everyday things*. Basic Books.

Garrett, J. J. (2010). *The elements of user experience: User-centered design for the web and beyond* (2nd ed.). New Riders.

Crockford, D. (2008). *JavaScript: The Good Parts*. O'Reilly Media.

Holovaty, A., & Kaplan-Moss, J. (2009). *The definitive guide to Django: Web development done right*. Apress. *(Helpful for full-stack thinking, even if not using Django)*

MySQL. (2023). *MySQL 8.0 reference manual*. Oracle Corporation. https://dev.mysql.com/doc/refman/8.0/en/

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Prentice Hall.

Trello. (n.d.). *Trello for agile teams*. Atlassian. https://trello.com/