Tama Tracky

FBLA Introduction to Programming (2025–2026

Shayaan Khan, Bryan Diaz, Curtis Newkirt

Team Members & Roles

- **Shayaan Khan** - (Lead Developer & Project Manager): Built core systems, debugging, integration, recording lead
- **Bryan Diaz** - (Quality Testing & Setup): Feature testing, validation, helped with scripting and setup
- **Curtis Newkirt/CJ** - (UI/Formatting & Rubric Alignment): Improved code formatting/comments, visual polish, matched rubric expectations

What is Tama Tracky?

Tama Tracky is a virtual pet education game that teaches financial responsibility through hands-on, realistic pet care. Players care for a virtual pet's health, happiness, and well-being while tracking expenses, earning coins, and learning about real-world cost-of-care concepts. Every action incurs a cost, every purchase is tracked, and every decision affects the pet's well-being, just like caring for a real pet.

The Problem It Solves

Younger people often overlook the financial responsibilities of pet ownership. Tama Tracky addresses this by making costs visible and using gameplay to teach budgeting, helping players manage both virtual pets and real-life financial decisions.
It not only makes costs easy to track and visualize and uses gameplay to teach budgeting, but it also helps them to manage pets and learn how to budget, manage, and spend in real-life financial decisions. They track expenses across categories (Food, Health, Toys, Supplies, Activities) and earn coins beforehand to spend, discovering budgeting's necessity for constant care. The game shows that having a pet involves more than love – it requires responsibility, planning, and money management.

What Makes It Above and Beyond for Us

- **Comprehensive Financial Tracking**: Every expense is automatically logged with categories, timestamps, and descriptions. Reports show income vs. expenses, category breakdowns, and CSV export for analysis and much more.
- **Realistic Cost-of-Care System**: Actions that cost coins in real life, Also require coins in game, (feeding, playing, cleaning, vet visits), items must be purchased before use, and players learn that good care requires budgeting.
- **Educational Reports System**: Financial reports with charts, filters, and insights help players understand spending patterns and make better budgeting decisions.

Key Features

Pet System

- **Pet Types**: Three selectable pet types (Cat, Dog, Rabbit), each with unique characteristics
- **Core Stats**: Five core statistics are tracked—Hunger, Happiness, Health, Energy, and Cleanliness—each ranging from 0 to 100
- **Mood System**: Pet mood is dynamically calculated from current stat levels, including states such as Happy, Sad, Sick, Energetic, Tired, Angry, and Neutral
- **Age Progression**: Four age stages (Baby, Young, Adult, Mature) determined by XP thresholds at 0, 20, 60, and 120 XP
- **Stat Decay**: Core stats naturally decrease over time, requiring consistent player interaction and care
- **XP System**: Experience points earned from care actions, tasks, and quests drive progression and age advancement

Cost-of-Care System

- **Expense Categories**: 5 predefined expense categories, Food, Health Toys, Supplies And Activities
- **Automatic Logging**: Every purchase and care action automatically records an expense with a timestamp and descriptive label
- **Realistic Costs**: Actions consume coins realistically, such as food being required for feeding and higher costs for veterinary care
- **Budget Awareness**: Players must earn coins before spending, reinforcing budgeting and financial planning concepts

Earning System

- **Tasks**: Four recurring tasks (Clean Room, Training, Pet Walking, Grooming) with cooldowns that reward coins and XP
- **Daily Quests**: Four daily quests that reset every day and provide coin and XP rewards upon completion
- **Daily Check-In**: A daily login reward encourages consistent engagement and routine pet monitoring
- **Weekly Allowance**: A weekly coin allowance system with cooldowns to prevent exploitation
- **Mini-Games**: Optional mini-games (Budget Blitz, Catch Food, Clean Up) reward coins and improve selected pet stats

Store + Activities

- **Store Items**: A variety of purchasable food items, toys, healthcare items, and supplies
- **Inventory System**: Purchased items are stored in an inventory and consumed upon use
- **Item Effects**: Items directly affect pet stats, such as food restoring hunger, toys increasing happiness, and healthcare items improving health
- **Special Activities**: Premium activities such as Spa Day, Training Session, and Park Trip cost coins and provide stat boosts

Reports

- **Financial Summary Cards**: At-a-glance cards display total spent, coins earned, net balance, and overall budget status
- **Category Breakdown**: Spending is broken down by category with percentage-based insights
- **Income Sources Breakdown**: Income is categorized by source, including Tasks, Quests, Allowance, Check-Ins, and Mini-Games
- **Date Range Filters**: Financial data can be filtered by predefined time periods (Today, Last 7 Days, Last 30 Days, All Time)
- **Category Filters**: Expense data can be filtered by specific category for targeted analysis
- Chart Visualizations: Line charts show trends over time, bar charts compare categories, and doughnut charts display proportional breakdowns
- **CSV Export**: Expenses, income, and statistics can be exported to CSV files for external analysis
- **Financial Insights**: Automated insights highlight major spending categories, primary income sources, and overall budget health

Rubric Mapping

Code Quality

- **Modular Architecture**: Code is organized into clear modules (`core/`, `components/`, `pages/`) with single-responsibility functions
- TypeScript Type Safety: All data structures use TypeScript interfaces for compile-time error checking
- **Comprehensive Comments**: Functions include clear documentation explaining purpose, parameters, and return values
- **Consistent Code Style**: Uniform formatting and naming conventions throughout the codebase

User Experience

- **Intuitive Navigation**: Clear navigation bar with labeled pages (Dashboard, Store, Tasks, Reports, Help)
- **Visual Feedback**: Pet mood indicators, stat bars, and animations provide immediate feedback on actions
- **Save System**: Three independent save slots allow multiple pets and easy switching
- Responsive Design: Works on desktop and mobile devices with adaptive layouts

Input Validation

- **Syntactic Validation**: Pet names are checked for length (1–30 characters), allowed character formats (letters, numbers, and spaces only), and correct whitespace handling
- **Semantic Validation**: Purchase actions check sufficient funds, numeric inputs are checked against valid ranges, and stat values are clamped between 0 and 100
- **Error Messages**: Clear, user-friendly error messages with recovery suggestions help guide users in correcting input issues
- **Edge Case Handling**: Null or undefined values, empty strings, and invalid data are handled neatly to prevent crashes and maintain stability

Functionality

- **Pet Management**: Create pets (Cat, Dog, Rabbit), track five core stats (Hunger, Happiness, Health, Energy, Cleanliness), and manage pet progression through defined age stages
- **Financial System**: Coin-based economy supporting earnings through tasks, quests, and daily rewards, and spending through store purchases and care actions
- **Expense Tracking**: Automatic logging of all purchases and care actions with clear categories and precise timestamps
- **Quest System**: Daily quests that reset each day, track progress, and reward coins and XP once completed
- **Task System**: Cooldown-based tasks that earn coins, grant XP, and positively affect pet vitals

Reports

- **Financial Summary**: Displays total spent, total earned, and net balance calculations with clear category-based breakdowns
- **Date Filtering**: Allows expenses to be filtered by predefined time periods (Today, Last 7 Days, Last 30 Days, All Time)
- **Category Filtering**: Enables filtering of expenses by category, including Food, Health, Toys, Supplies, Activities, and Other

- **Visual Charts**: Line, bar, and doughnut charts are used to visualize spending patterns and financial trends
- **CSV Export**: Supports exporting expenses, income, and statistics to CSV files for external review and analysis

Data Storage

- **localStorage Persistence**: All game data, including pet stats, expenses, income, quests, and badges, is saved to browser localStorage
- **Save Slot System**: Three independent save slots are supported, each with metadata such as creation date, last played timestamp, and slot number
- **Auto-Save**: The game state automatically saves after every user action to prevent any data loss while not messing with performance
- **Data Validation**: Data loading functions validate structure and gracefully handle missing or corrupted data

Documentation

- **Code Comments**: Functions include JSDoc-style comments that explain purpose, parameters, and return values
- **Architecture Documentation**: **docs/architecture.md** details system design, module responsibilities, and overall organization
- **User Guide**: A built-in Help page provides care instructions and an offline FAQ-style assistant
- **Attributions**: A complete list of third-party libraries and credits is maintained in **docs/attributions.md**

Final Notes

Use of External Resources and Tools

During the creation of Tama Tracky, our team used a combination of original programming and external learning resources to support the understanding and implementation of some concepts. These resources were used for reference, clarification, and learning purposes only.

The primary codebase, logic, and system design were written and implemented by the team. External resources were used in the following ways:

- **Artificial Intelligence tools** were used to assist with brainstorming, code explanations, heavy and extensive debugging, some of the comments, and improving documentation clarity. All AI-assisted content was reviewed, adapted, and integrated by the team.

- **Online documentation and tutorials** (such as official language documentation and educational videos) were referenced to better understand TypeScript, React, and general programming concepts.

- **Search engines and instructional videos** were used to research best practices, user interface ideas, and debugging techniques.

No external code was copied directly without full understanding or modification. All final implementations reflect the team's own work and design decisions.

This approach aligns with real-world software development practices, where developers responsibly use reference materials and tools to improve code quality, learning, and problem solving.

For technical details, see **docs/architecture.md**. For design decisions, see **docs/design_notes.md**. For third-party credits, see **docs/attributions.md**.

Thank you for judging our project!