# Generative AI Hackathon Project

**Project Title:**

**Financial chatbot Using Gemini Flash**

**Team Name:**

Team APP

**Team Members:**

* D.Srinivasa Pranavan
* V.V.S.Adithya
* M.Purandish

## Phase-1: Brainstorming & Ideation

**Objective:**

Develop a console-based personal finance assistant in Python that allows users to manage transactions, budgets, and generate financial reports—while identifying and addressing limitations around security, data persistence, and usability

**Key Points:**

1. **Problem Statement:**

○ Users struggle with tracking personal financial activity without relying on third-party apps that raise privacy or security concerns.

○ There is a gap in beginner-friendly finance tools that are both customizable and secure.

○ Existing basic tools lack scalability, data protection, and user-friendly functionality.

**2. Proposed Solution:**

* 1. A modular **Python application** (FinBot) using **pandas** for transaction and budget management**.**

○ Local storage via **CSV/JSON** for data persistence (prototype phase).

○ A future roadmap includes **database integration, secure authentication**, and **input validation.**

1. **Target Users:**

○ Individuals seeking local/offline **financial tools**.

○ **Students** or developers learning about **financial tracking systems**.

○ Early-stage fintech **prototypers**.

1. **Expected Outcome:**

○ A functional **CLI**-based financial assistant prototype.

○ Clear identification of **architectural** and security gaps.

○ A plan for future **iterations** with **database integration** and secure authentication.

## Phase-2: Requirement Analysis

**Objective:**

Define the functional and technical specifications, as well as acknowledge constraints in the current system.

**Key Points:**

1. **Technical Requirements:**

○ Programming Language: **Python,Streamlit**

○ Backend: **Google Gemini Flash API 2.5 pro**

○ Frontend: **Streamlit Web Framework**

○ Database: **Not required initially (API-based queries)**

1. **Functional Requirements:**

* 1. Add**/view/filter transactions**.

○ Set **check budgets**.

○ Generate **category-wise** and **monthly reports.**

○ Simple user **authentication** (non-persistent, prototype only).

1. **Constraints & Challenges:**

○ Lack of **persistent** and secure **user data storage**.

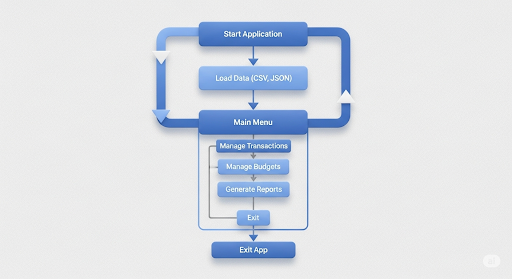
○ No support for **editing/deleting transactions**.

○ Minimal **input validation** and **no encryption**.

## Phase-3: Project Design

**Objective:**

**Define architecture and user flow for FinBot.**

****

**Key Points:**

1. **System Architecture:**

* 1. **Entry Point**: main.py handles CLI and state transitions.

○ **Modules**:

○ **auth.py: user login/registration**

**○ transaction\_manager.py: transaction tracking**

**○ budget\_manager.py: budget operations**

**○ report\_generator.py: summary/reporting**

**○ data\_manager.py: data persistence**

1. **User Flow:**

○ Launch **FinBot**.

○ Authenticate **user (non-secure).**

○ View **main menu** options (**Transactions, Budget, Reports**).

**○ Perform actions, save and exit**

3**. UI/UX Considerations:**

**○** **Text-based CLI for simplicity.**

**○** **Menu-driven structure for easy navigation**.

**○** **Future scope for TUI/GUI integration (e.g., using curses or Tkinter).**

## Phase-4: Project Planning (Agile Methodologies)

**Objective:**

Break down development tasks for efficient completion.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Task** | **Priority** | **Duration** | **Deadline** | **Assigned To** | **Dependencies** | **Expected**  **Outcome** |
| Sprint 1 | Environment Setup  & API Integration | 🔴 High | 6 hours  (Day 1) | End of Day  1 | Aditya | Google API Key,  Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day  1 | Aditya | API response format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search &  Comparison | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Pranavan | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling &  Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Pranavan | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI  Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Purandish | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation  & Deployment | 🟢 Low | 1 hour  (Day 2) | End of Day  2 | Purandish | Working prototype | Demo-ready project |

**Sprint Planning with Priorities**

**Sprint 1 – Setup & Integration (Day 1)**

**(**🔴 **High Priority)** Set up the **environment** & install dependencies.

**(**🔴 **High Priority)** Integrate **Google Gemini API**.

**(**🟡 **Medium Priority)** Build a **basic UI with input fields**.

**Sprint 2 – Core Features & Debugging (Day 2)**

**(**🔴 **High Priority)** Implement **search & comparison functionalities**. **(**🔴 **High Priority)** Debug API issues & handle **errors in queries**. **Sprint 3 – Testing, Enhancements & Submission (Day 2)**

**(**🟡 **Medium Priority)** Test API responses, refine UI, & fix UI bugs.

**(**🟢 **Low Priority)** Final **demo preparation & deployment**.

## Phase-5: Project Development

**Objective:**

**Key Points:**

**Build core functionality for FinBot in Python.**

1. **Technology Stack Used:**

* 1. **Frontend:** Streamlit

○ **Backend:** Google Gemini Flash API

○ **Programming Language:** Python

1. **Development Process:**

○ Created modular files for core functionality.

○ CLI-driven interaction.

○ Read/write operations via data\_manager.

○ Used pandas for aggregation and summaries.

1. **Challenges & Fixes:**

* 1. **Challenge:** Lack of persistent user registration.

**Fix:** Noted for future database migration

○ **Challenge:** **Input errors crashing app**.

**Fix:** Added **try-except** and basic **validation**

## Phase-6: Functional & Performance Testing

**Objective:**

Ensure reliability and usability of core features.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional  Testing | Add transaction and view. | Transaction saved & displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional  Testing | Set and  check budget. | |  | | --- | | Correct budget shown |  |  | | --- | |  | | ✅ Passed | |  | | --- | |  |  |  | | --- | | Tester2 | |
| TC-003 | Functional  Testing | Generate monthly summary. | Summary accurate by month. | ✅ Passed | Tester 2 |
| TC-004 | Performance  Testing | Load 100+ transactions | |  | | --- | |  |  |  | | --- | | Load time < 1 sec | | ⚠ Acceptable | Tester 3 |
| TC-005 | Bug Fixes & Improvements | Wrong input on date | App handles gracefully. | ✅ Fixed | Team APP |
| TC-006 | Security | Check if credentials persist | Login fails after restart | ❌ Fails (expected) | Tester 1 |

## Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**