Project Report: Urban Thread Loyalty Program (v1.0) - Prototype Build

Project Type: Business Analysis (Design) & Prototype Development

---

1. Executive Summary

This project addressed UrbanThread's strategic need to improve customer retention (currently 15% repeat-purchase rate) and reduce high Customer Acquisition Costs (CAC). The objective was to design and build a functional prototype for a new Customer Loyalty Program, aiming to support the business goal of increasing repeat-customer sales by 20% within 18 months.

Acting as the Lead Business Analyst and prototype developer, a full project lifecycle was undertaken. The Design Phase involved stakeholder analysis (Marketing, Finance, IT, Customer Service) to elicit requirements, resulting in a Business Requirements Document (BRD), Functional Requirements Document (FRD) using User Stories and Acceptance Criteria, and BPMN Process Flow diagrams ('As-Is' vs. 'To-Be'). The Build Phase involved creating an SQLite database (loyalty.db) via a Python script (create_database.py) and developing a multi-page Streamlit web application (app.py) demonstrating core loyalty functionalities (point earning simulation, balance view, redemption, manual adjustments). Finally, the Deployment Phase utilized Git for version control, GitHub for repository hosting, and Streamlit Community Cloud for deploying the live prototype.

The final deliverable is a functional, publicly accessible web application prototype, serving as a proof-of-concept that validates the design and provides a foundation for future full-scale development.

---

2. Project Objective & Scope

2.1. Business Problem Statement

UrbanThread faces low customer retention (15% repeat-purchase rate) and high Customer Acquisition Cost (CAC). The company struggles to generate long-term value, as many customers purchase only once.

2.2. Project Goal

To **design and build a functional prototype** of a new, in-house customer loyalty program. This prototype serves to validate the proposed solution design, gather early stakeholder feedback, reduce implementation risks, and ultimately support the strategic goal of **increasing repeat-customer sales by 20%** within 18 months of the *final* program launch.

2.3. Project Scope

In Scope (Design):

- Stakeholder identification and analysis (Marketing, Finance, IT, Customer Service).
- Elicitation and documentation of high-level business needs.
- Creation of a formal **Business Requirements Document (BRD)**.
- Modeling of current ('As-Is') and proposed ('To-Be') checkout processes using **BPMN**.
- Creation of a detailed **Functional Requirements Document (FRD)** using User Stories and Acceptance Criteria.

In Scope (Build - Prototype):

- Development of an **SQLite database schema** based on the FRD.
- Creation of a Python script (create_database.py) to initialize the database and insert sample data.
- Development of a **Streamlit web application** (app.py) demonstrating core functionalities:
  - Customer point balance and history lookup.
  - Simulation of point earning via a purchase form.
  - Reward browsing and point redemption.
  - Manual point adjustments (Customer Service view).
  - Basic tier checking logic (Standard/Gold).
- Version control using **Git** and **GitHub**.
- Deployment to **Streamlit Community Cloud**.

---

3. Methodology & Tools Used

This project integrated standard Business Analysis practices with rapid application development techniques.

Phase	**Methodology**	**Key Tools Used**
1. BA Design	Stakeholder Analysis, Requirements Elicitation, Process Modeling, Use Case Definition	Simulated Interviews, Text Editor/Word Processor (for BRD/FRD), diagrams.net (for BPMN)
2. Database Development	Relational Database Design, Scripting	Python 3, sqlite3 module, create_database.py script
3. App Development	Rapid Prototyping	Python 3, Streamlit, pandas, app.py script
4. Version Control	Git Workflow	Git CLI / GitHub Desktop, GitHub.com, .gitignore file
5. Deployment	Cloud Platform Deployment	Streamlit Community Cloud, requirements.txt file

---

4. BA Design Phase - Deliverables & Details

4.1. Stakeholder Analysis Summary

Interviews (simulated) identified key needs and potential conflicts:

- **Marketing (Sponsor):** Needs simplicity, excitement, a tiered system; concerned about complexity.
- **Finance:** Needs positive ROI, cost control, point expiration; concerned about financial liability.
- **IT:** Needs clear logic, speed/performance; concerned about checkout impact.
- **Customer Service:** Needs easy access to history, manual adjustment capability; concerned about resolving issues quickly.

4.2. Business Requirements Document (BRD)

A high-level document outlining the project vision:

- **Objective:** Increase repeat sales by 20% via a new loyalty program.
- **Scope:** Defined core features (points, tiers, redemption, expiration, manual adjust) and explicitly excluded others (physical cards, complex rewards).
- **Key Requirements:** Listed high-level business rules (e.g., BR-001: Earn points per $1; BR-004: Points expire after 12 months).

4.3. Process Flow Diagrams (BPMN)

Visual blueprints created using diagrams.net:

- **'As-Is' Diagram:** Depicted the current simple checkout process, highlighting the lack of loyalty engagement.
- **'To-Be' Diagram:** Detailed the proposed new workflow, crucially placing loyalty logic *after* the payment success gateway to ensure checkout speed, and outlining the sequence of system actions (calculate points, add points, check tier, send email).

4.4. Functional Requirements Document (FRD)

The detailed specification for developers, using User Stories and Acceptance Criteria:

- **Format:** "As a [User Type], I want to [Action], so that [Benefit]."
- **Acceptance Criteria (AC):** Provided specific, testable conditions for each story (e.g., *GIVEN* X, *WHEN* Y, *THEN* Z).
- **Key Features Covered:** Point Earning (incl. ratio, timing), Tier System (incl. threshold, automation), Reward Redemption (incl. store view, deduction), Point Expiration (incl. timing, FIFO), Manual Override (incl. lookup, adjustment, notes).

---

5. Technical Implementation Phase - Deliverables & Details

5.1. Database Design (SQLite)

A lightweight, file-based database (loyalty.db) was implemented using Python's sqlite3 module.

- **Schema:**
  - Customers: customer_id (PK), first_name, last_name, email (Unique), tier (Default 'Standard').
  - Rewards: reward_id (PK), name, points_cost.
  - PointsLedger: transaction_id (PK), customer_id (FK to Customers), points_change (positive for earn/adjust, negative for redeem/expire), transaction_type ('earn', 'redeem', 'manual_adjust', 'expire'), timestamp, note.
- **Creation:** Managed entirely within the create_database.py script.

5.2. Web Application (app.py)

A functional prototype was built using Streamlit.

- **Structure:** Multi-page application navigated via st.sidebar.radio.
- **Core Logic:** Python functions were created to handle database interactions (CRUD operations - Create, Read, Update, Delete - simplified here) triggered by user input via Streamlit widgets.
- **Key Features Implemented:**
    - Customer View: Fetches customer details, calculates balance (SUM(points_change)), displays history.
    - Add Points: Takes email/amount/order ID, calculates points (amount * POINTS_PER_DOLLAR), inserts 'earn' record into PointsLedger, calls update_customer_tier.
    - Redeem Reward: Fetches rewards, checks balance against cost, inserts 'redeem' record (negative points_change) into PointsLedger.
    - Customer Service: Takes email/points/reason, inserts 'manual_adjust' record into PointsLedger.
- **Database Initialization:** Includes an initialize_database() function that runs on startup, creating the tables if the database file is empty (essential for Streamlit Cloud deployment).

---

6. Deployment Phase

6.1. Version Control (Git/GitHub)

- **Git:** Used locally to track code changes (git init, add, commit).
- **.gitignore:** Configured to exclude the loyalty.db file and other non-essential files from being uploaded.
- **GitHub:** Served as the remote, public repository for code hosting (git remote add, push). Repository: https://github.com/RBhardwaj2080/loyalty-app

6.2. Deployment (Streamlit Community Cloud)

- **Configuration:** App deployed via Streamlit Cloud by linking the GitHub repository and specifying the branch (main/master) and main file (app.py).
- **Dependencies:** Required libraries (streamlit, pandas) specified in requirements.txt for automated installation in the cloud environment.
- **Outcome:** Generated a live, publicly accessible URL for the prototype.
- **Limitation:** The deployed app uses ephemeral storage; the loyalty.db file resets to its initial (empty or minimally seeded) state on app restarts or inactivity.

---

7. Final Outcome

This project successfully delivered a functional web application prototype demonstrating the core features and logic of the proposed UrbanThread Loyalty Program. It includes the necessary BA documentation (BRD, FRD, Process Flows) that defined the prototype's requirements. The live prototype serves as a tangible proof-of-concept.