Product Requirements Document

Overview

This document outlines a simple yet effective web-based tool designed to help new graduates track and manage their student loans. It clarifies team responsibilities, resolves conflicting requirements, and tightens the scope to ensure maintainability and strategic flexibility.

Goals and objectives

Goal

Enable new graduates to understand their student loan status, make informed payment decisions, and track progress toward complete repayment.

Objectives

- Define clear ownership for each feature (design, development, QA)
- Provide users with a clear interface to input loan details (total amount, interest rate, minimum monthly payment)
- Allow users to log individual payments and see an updated remaining balance in near real time
- Visualize loan balance decline with a chart
- **Optional**: Offer milestone notifications to celebrate key repayment progress

Responsibility and collaboration

To avoid confusion and ensure accountability, each major feature is assigned clear ownership:

- **Design team**: Create wireframes and user flows for loan setup, payment logging, and progress visualization.
- **Frontend development team**: Implement React components (Loan Setup, Payment Tracker, Progress Chart), integrate the chart library, and manage local storage or chosen backend solution.
- **QA team**: Test all user stories, focusing on both functional correctness and user experience consistency.
- **Product management**: Define success metrics, oversee the project timeline, and make final decisions on feature prioritization.

Scope

In scope

- Single-loan tracking (one loan per user)
- Basic monthly interest approximations to calculate remaining principal and payoff date
- Chart-based progress visualization
- Local data persistence as the primary method, **or** a single backend solution (Firebase **or** simple Node.js/Express), but not both
- Basic authentication if chosen; only one approach to be implemented if deemed necessary by product management

Out of scope

- Advanced multi-loan management
- Detailed daily compound interest or complex financial modeling
- Full-scale multi-user authentication systems
- Large-scale enterprise backend solutions with role-based access control

Key features

Loan setup and overview

- **Feature owner**: Design & frontend development
- **Description**: Users enter the principal, annual interest rate, and monthly payment. The tool calculates and displays:
    - Remaining balance
    - Estimated payoff date (approximate monthly calculation)
- **Strategic note**: Keep the calculation simple to meet the MVP timeline.

Payment tracking

- **Feature owner**: Frontend development
- **Description**: Users can add payment entries (amount, date) and immediately see:
    - Adjusted remaining principal
    - Updated payoff date

Progress visualization

- **Feature owner**: Frontend development (with QA ensuring chart accuracy)
- **Description**: A chart (using Chart.js or Recharts) that plots the remaining balance over time. Each new payment triggers a refresh, showing immediate progress.

Notifications and milestones (optional)

- **Feature owner**: Product management & frontend development

- **Description**: Users may receive alerts when significant repayment milestones are reached (25%, 50%, etc.). This feature is lower priority and should be implemented only after the core functionality is stable.

Architecture and tech stack

1. **Frontend**:
    - React for UI and state management (React hooks or Context API)
    - Chart.js or Recharts for visualizing the balance trend
2. **Data persistence**:
    - **Primary approach**: Local storage (single-user scenario)
    - **Optional approach**: Firebase **or** Node.js/Express backend (but not both). The team must decide on one if extended functionality or basic authentication is required.
3. **Deployment**:
    - Recommended: Netlify or Vercel for easy cloud hosting
    - Public GitHub repository (one central repo for all code)

Development approach

1. **Project setup**: Create a single-page React application. Decide (upfront) whether to use local storage **or** a simple backend.
2. **Loan setup**: Build a Loan Setup form for collecting principal, interest rate, and monthly payment.
3. **Payment logging**: Implement the Payment Tracker to log payments and recalculate the balance and payoff date.
4. **Chart integration**: Add a line or bar chart library and ensure it updates each time a new payment is logged.
5. **(Optional) Authentication**: If product management chooses authentication, integrate only one simple method (Firebase Auth **or** a Node-based session system).
6. **Milestones and notifications**: Implement optional alerts after ensuring core features are stable.
7. **Testing and validation**: QA verifies each user story.

User stories

US-001

- **User story**: As a new graduate, I want to input my total loan amount, interest rate, and monthly payment so that I can get an overview of my current debt and expected payoff date.
- **Acceptance criteria**:
    - The system accepts valid numeric inputs for loan amount and interest rate.
    - The system calculates a payoff date using the stated interest rate and monthly payment.

    - The loan overview (balance, estimated payoff date) displays immediately after saving.

US-002

- **User story**: As a user who has already entered loan details, I want to log each payment I make so that I can see how it affects my remaining balance in near real time.
- **Acceptance criteria**:
    - The system provides a form for logging payment amount and date.
    - The system deducts the payment from the remaining balance.
    - The updated balance and payoff date display immediately after logging the payment.

US-003

- **User story**: As a user, I want the system to update my estimated payoff date whenever I make additional or larger payments so that I understand how extra payments affect my loan duration.
- **Acceptance criteria**:
    - The system recalculates the payoff date based on any change in principal.
    - The new payoff date appears in the overview section without requiring a full page refresh.

US-004

- **User story**: As a visual learner, I want to see a chart showing my balance decreasing over time so that I can stay motivated by my progress.
- **Acceptance criteria**:
    - The system displays a line or bar chart plotting remaining balance against each recorded payment date.
    - Each new payment refreshes the chart with updated data.

US-005

- **User story**: As a user who occasionally makes additional payments, I want to log ad-hoc (non-monthly) payments so that I can accurately reflect my loan balance.
- **Acceptance criteria**:
    - The payment logging form allows any valid monetary amount.
    - Each payment triggers an update to the balance and payoff date.

US-006

- **User story**: As a user, I want to receive a milestone notification (e.g., 50% paid) so that I am encouraged to stay on track.
- **Acceptance criteria**:
    - The system identifies payoff milestones (25%, 50%, 75%, etc.).
    - A message or alert displays when a milestone is reached.

US-007

- **User story**: As someone who wants to keep my financial data secure, I want an optional login feature so that my loan information is visible only to me.
- **Acceptance criteria**:
    - If enabled, the system requires registration or sign-in before accessing loan data.
    - A simple authentication approach is consistently implemented (Firebase Auth **or** Node-based sessions).
    - Logged-in state persists until the user logs out or the session times out.

US-008

- **User story**: As a user prone to mistakes in data entry, I want to edit or delete payment records so that my loan balance remains accurate.
- **Acceptance criteria**:
    - The system allows selecting an existing payment from a list.
    - Users can modify or delete the payment, triggering an update to the balance and payoff date.

Changes from earlier versions

- **Clear accountability**: Defined which team owns each feature to prevent confusion.
- **Single data storage choice**: Pick either local storage or one backend (Firebase or Node.js), not both.
- **Optional authentication**: Retained but clarified that it applies only if the team decides to implement it (with one approach).
- **Real-time chart updates**: Still required, but clarified it should be “near real time” to allow basic refresh logic without overly complex performance constraints.

Final notes

- **Success metrics**: Product management will define measurable targets (e.g., daily active users or time to first meaningful payment log) to track adoption and user satisfaction.
- **Timelines and deliverables**: Design, development, and QA must collaborate on a shared project board to track tasks and responsibilities.
- **Future expansion**: If multi-loan support or advanced financial modeling becomes a priority, an architectural review will be required to determine the feasibility of extending the existing codebase.

This improved document addresses common pitfalls around collaboration, maintainability, and conflicting requirements. It provides enough clarity to guide the development team while leaving optional features flexible for future strategic considerations.