Purchase impact prompts:

using the persona of a Saas developer, create the prompts for use in lovable so that it can build a new screen for my app that asks the user a series of questions regarding 3 potential purchases that are about to be made in order to determine if cash will remain available when list of inputted expenses are due AND when the purchases on key dates are to occur. the screen should display the day(s) and potential expense(s) that will not get paid if the inputted purchase is made. recommend the preferred question sequence and screen layout that a UI/UX expert would employ for this type of questionnaire. also, use the existing color and formatting scheme that is commonly used on fintech personal finance platforms.

**ChatGPT said:**

Here's how I would approach this request as a **SaaS developer** building with **Lovable**, while collaborating closely with a **UI/UX expert** and staying consistent with industry-leading fintech design patterns (like Mint, YNAB, and Monarch).

**👨‍💻 Goal**

Create a **"Purchase Impact Forecast" screen** that:

1. Asks the user to input 3 upcoming **purchase decisions**
2. Cross-checks those against their known **recurring expenses**
3. Projects future **cash balance** daily
4. **Visually warns** the user of which expenses won’t be covered if purchases are made

**✅ Preferred UX Flow and Layout (Based on Fintech Standards)**

**Screen Layout**

* **Top**: Brief message like *“Let’s check if you can afford these upcoming purchases.”*
* **Section 1**: Purchase entries (3 item slots with name, cost, and purchase date)
* **Section 2**: Summary of fixed upcoming expenses (auto-filled or editable)
* **Section 3**: Result panel with:
  + Daily running cash balance chart
  + Highlighted dates showing a **negative balance**
  + Text warning: *“⚠️ You won’t be able to pay [expense name] on [date] if you make all 3 purchases.”*

**Colors**

* Use **green** for safe days, **yellow** for low balance warnings, **red** for days where expenses won’t be covered.
* Follow modern fintech UI style: clean white/neutral background, rounded cards, soft shadows, simple charts (line or calendar strip).

**🧠 Recommended Question Sequence**

“The logic is forward-looking, based on user-entered planned purchases vs. existing obligations.”

1. **What are you thinking about buying?**  
   (Ask for **3 items**: name, cost, and expected purchase date)
2. **Do you have any of these upcoming bills or obligations?**  
   (Load from user\_expense\_items if available, or allow quick edits)
3. **Let’s calculate your cash outlook with these purchases.**  
   (Pull current cash from daily\_cashflow.opening\_balance, subtract expenses and purchases on correct days)
4. **Show results with warnings**
   * “You will be $120 short on Sept 19 and won’t be able to pay ‘Car Loan’ unless you delay a purchase.”

**💬 PROMPTS FOR LOVABLE**

plaintext

CopyEdit

Create a new screen titled "Purchase Impact Forecast"

SECTION 1: Ask the user to input up to 3 planned purchases:

- For each purchase, ask:

- “What are you thinking of buying?” → purchase\_name

- “How much will it cost?” → purchase\_cost

- “When do you plan to buy it?” → purchase\_date

Store these entries in a temporary array or `user\_purchase\_forecast` table linked to the current user.

SECTION 2: Retrieve the user’s fixed monthly expenses from `user\_expense\_items`. Allow the user to confirm or edit them.

- For each expense, display the name, amount, and due date

- Assume due dates repeat monthly unless updated

SECTION 3: Simulate the cash balance across the next 30 days

- Use the latest `daily\_cashflow.opening\_balance` as the starting point

- Subtract expense amounts on their due dates

- Subtract each purchase cost on its `purchase\_date`

Highlight:

- Any day where projected balance < 0

- List which expenses will go unpaid if the purchases are made

Show a results panel with:

- A color-coded calendar strip (green = OK, red = shortfall)

- A table of days with failed payments (e.g., “Sept 19 – Unable to pay Car Loan”)

- An optional line chart of running balance

SECTION 4: Provide options:

- “Remove a purchase” → Deletes from forecast

- “Delay purchase” → Changes purchase\_date

- “Confirm I’m okay with risk” → Saves decision

Use modern fintech UI style: soft cards, green/red/yellow indicators, rounded inputs, light neutral backgrounds.

**🧑‍🎨 UI/UX Expert Guidance Recap**

| **UI Element** | **Reason** |
| --- | --- |
| **Top prompt message** | Sets user expectation and tone |
| **Progressive Disclosure** | Keep inputs minimal; reveal summary last |
| **Color-coded visual cues** | Matches mental model from Mint, YNAB, Monarch |
| **Line chart or dot strip** | Helps visualize “when things go wrong” |
| **Expense name callout** | Personalizes and contextualizes warnings |

Would you like me to convert this into a JSON config for Lovable or provide wireframe visuals next?

**You said:**

instead of asking if they have any upcoming expense, I'd like to automatically check what bills are coming due within the next two weeks as well as any key dates that are coming up within the next two weeks to use as a gauge to answer step 3 in the recommended question sequence. then I would like to show the results with warnings.

**ChatGPT said:**

Income Expense Dashboard Config

999

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

{

"database": {

"tables": [

{

"name": "user\_income\_sources",

"fields": [

{ "name": "user\_id", "type": "string" },

{ "name": "income\_1\_amount", "type": "number", "default": 0 },

{ "name": "income\_2\_amount", "type": "number", "default": 0 },

{ "name": "child\_support\_amount", "type": "number", "default": 0 },

{ "name": "created\_at", "type": "datetime" }

]

},

{

"name": "user\_expense\_items",

"fields": [

{ "name": "user\_id", "type": "string" },

{ "name": "housing\_expense", "type": "number", "default": 0 },

{ "name": "utilities\_expense", "type": "number", "default": 0 },

{ "name": "communications\_expense", "type": "number", "default": 0 },

{ "name": "grocery\_expense", "type": "number", "default": 0 },

{ "name": "transport\_expense", "type": "number", "default": 0 },

{ "name": "debt\_expense", "type": "number", "default": 0 },

{ "name": "childcare\_expense", "type": "number", "default": 0 },

{ "name": "subscriptions\_expense", "type": "number", "default": 0 },

{ "name": "created\_at", "type": "datetime" }

]

},

{

"name": "user\_purchase\_forecast",

"fields": [

{ "name": "user\_id", "type": "string" },

{ "name": "purchase\_name", "type": "string" },

{ "name": "purchase\_cost", "type": "number" },

{ "name": "purchase\_date", "type": "date" }

]

Edited Income Expense Dashboard Config

✅ The configuration has been updated to:

* **Automatically pull upcoming bills** and key dates within the next **14 days** from user\_expense\_items and user\_key\_dates
* Simulate cash availability after accounting for planned purchases
* Show a **calendar strip** and a **warning table** for days where users won’t be able to meet obligations

Would you like help wiring this into your Lovable app now or want a visual wireframe of the forecast screen?

**You said:**

give me the step by step prompts that I need to write to accomplish the above

**ChatGPT said:**

Absolutely. Below are **step-by-step prompts** to type or paste directly into **Lovable**, using their guided builder or command bar. These prompts are grouped by functionality so you can execute them sequentially and keep your build clean and modular.

**🪜 STEP-BY-STEP PROMPTS FOR LOVABLE**

**Feature:** Forecasting if purchases will cause missed bill payments

**✅ STEP 1: Create New Tables**

**📌 A. user\_purchase\_forecast table**

plaintext

CopyEdit

Create a new table called `user\_purchase\_forecast` with the following fields:

- user\_id (string, foreign key to users table)

- purchase\_name (string)

- purchase\_cost (number)

- purchase\_date (date)

**📌 B. user\_key\_dates table**

plaintext

CopyEdit

Create a new table called `user\_key\_dates` with the following fields:

- user\_id (string, foreign key to users table)

- event\_name (string)

- event\_date (date)

**✅ STEP 2: Build the Forecast Input Screen**

**📌 C. Create screen titled PurchaseImpactForecast**

plaintext

CopyEdit

Create a screen called `PurchaseImpactForecast`.

Add a top heading: “Will you still be able to cover your bills if you make these purchases?”

**📌 D. Add 3 purchase input blocks**

plaintext

CopyEdit

Add a repeatable group with 3 rows. For each row, ask:

- “What are you planning to buy?” → purchase\_name (text)

- “How much will it cost?” → purchase\_cost (number)

- “When do you plan to buy it?” → purchase\_date (date)

Save the input to the table `user\_purchase\_forecast`, filtered by current user.

**✅ STEP 3: Auto-load Expenses and Key Dates**

**📌 E. Create logic to retrieve upcoming expenses**

plaintext

CopyEdit

Query the `user\_expense\_items` and calculate estimated due dates within the next 14 days. Use recurring logic if needed (e.g., monthly due dates based on created\_at date).

**📌 F. Add query for user\_key\_dates**

plaintext

CopyEdit

Query `user\_key\_dates` for any event\_date within the next 14 days where user\_id = current\_user.

**✅ STEP 4: Calculate Cash Flow Forecast**

**📌 G. Simulate forecast logic**

plaintext

CopyEdit

Using the latest `daily\_cashflow.opening\_balance` as the start, simulate cash changes over the next 14 days:

- Subtract purchase\_costs on each purchase\_date

- Subtract expenses on estimated due dates

- Flag any day where the resulting balance < 0

Store these flagged days in memory or create a derived output for UI binding.

**✅ STEP 5: Display the Results**

**📌 H. Add a calendar strip (UI component)**

plaintext

CopyEdit

Add a 14-day horizontal calendar strip to `PurchaseImpactForecast`:

- Each day displays first letter of day

- Use colors:

- Green for safe days

- Yellow if low cash (< $100)

- Red if cash < $0 (shortfall)

Bind color logic to simulated forecast result.

**📌 I. Add a warning table**

plaintext

CopyEdit

Create a table view that lists:

- Date

- Expense Name or Purchase Name

- “At Risk” reason (e.g., “Insufficient funds to pay Rent”)

Only show rows where balance forecast is negative.

**✅ STEP 6: Add Confirmation Buttons**

plaintext

CopyEdit

Add two buttons below results:

- “Remove a purchase” → Opens the entry to edit or delete

- “Confirm I’m okay with the risk” → Saves the record to `user\_purchase\_forecast` and returns to dashboard

**🧠 PRO TIPS:**

* Test this with mock purchases and fixed expenses (e.g. rent due on the 15th, subscription on the 10th)
* Use default colors and rounded card UI styles common in Lovable and fintech apps (neutral backgrounds, shadows, hover states)

Would you like this turned into a complete visual storyboard or JSON export next?

**You said:**

complete a storyboard

**ChatGPT said:**

Purchase Impact Storyboard

**Storyboard: Purchase Impact Forecast Screen**

**Screen Name:** PurchaseImpactForecast

**User Goal:** Determine whether 3 upcoming purchases will cause them to miss any scheduled expenses or key events within the next 14 days.

**Scene 1: Introduction Header**

* **Title:** "Will You Still Cover Your Bills?"
* **Subtitle:** "Check your finances before making upcoming purchases."
* **Purpose:** Orient user to why this screen matters — decision support before discretionary spending.

**Scene 2: Planned Purchases Input**

* **Layout:** Stack of 3 purchase cards
* **Fields per card:**
  + *Item Name*: purchase\_name
  + *Cost*: purchase\_cost
  + *Planned Date*: purchase\_date
* **Action:** Save each purchase entry to user\_purchase\_forecast
* **UX Pattern:** Repeatable form component with card-style layout

**Scene 3: Auto-loaded Forecast Data**

* **Logic:**
  + Pull user’s fixed expenses from user\_expense\_items
  + Derive upcoming due dates within 14 days
  + Pull user\_key\_dates for events within 14 days
  + Fetch current balance from daily\_cashflow.opening\_balance
* **Processing Logic:**
  + For each of the next 14 days:
    - Subtract scheduled expenses
    - Subtract planned purchases
    - Log any day with balance < 0

**Scene 4: Visual Feedback**

**A. Calendar Strip**

* **Display:** 14 dots labeled S, M, T, etc.
* **Color Logic:**
  + Green: Sufficient funds
  + Yellow: Low balance (< $100)
  + Red: Negative balance
* **Interactivity:** Tap a dot to highlight forecast value and flagged expenses

**B. Expense Risk Table**

* **Columns:**
  + Date
  + Expense or Event Name
  + Description of Shortfall
* **UX Style:** Dynamic warning list with red or yellow alert icons

**Scene 5: Recommendations / Actions**

* **Callouts (if shortfalls exist):**
  + "⚠️ You won’t be able to pay your car loan on May 18."
  + "⚠️ Grocery budget shortfall expected May 19."
* **User Actions:**
  + [🛠️ Edit Purchase] – return to input card
  + [✅ Confirm Risk] – save entries and return to dashboard

**Success Criteria:**

* User understands if discretionary spending will cause problems
* System provides forward-looking insight in a clean, helpful interface
* User can easily modify plan or acknowledge risk

**Visual Style:**

* Clean white background, soft shadows, sans-serif font
* Card components for forms and alerts
* Color-coded feedback (green/yellow/red)
* Mobile-first layout with responsive strip and stacked panels

Answer in chat instead

✅ The **Purchase Impact Forecast storyboard** has been created and saved under the canvas title **“Purchase Impact Storyboard.”** It includes:

* Input scenes for up to 3 planned purchases
* Automatic analysis of upcoming expenses and key dates
* Visual calendar and warning system for potential shortfalls
* User options to adjust purchases or proceed despite risks