Income Question Flow - Lovable

Using the persona above, recommend and outline the steps and prompts to create the income and expense items for a first time user. then recommend the steps and prompts to create a dashboard for that user, which becomes the standard start up screen for all users that have answered these questions. the start up screen will be use the information stored in the applicable tables for each user.

Create a form component that asks the user:

1. "How much do you receive monthly from your main job?" → Save as `income\_1\_amount`

2. "Do you have a side gig or second source of income?" (Yes/No)

- If Yes: Ask "How much do you earn monthly from this side gig?" → Save as `income\_2\_amount`

3. "Do you receive child support?" (Yes/No)

- If Yes: Ask "How much do you receive monthly in child support?" → Save as `child\_support\_amount`

- If No: Set `child\_support\_amount` = 0

Store all entries in the table `user\_income\_sources`, and link each row to the logged-in user via `user\_id`.

Ensure all numeric fields default to 0 if left blank.

Absolutely! Here’s how I would explain the **first step** and important **precautions** to take as an experienced developer guiding a beginner building in **Lovable**. I’ll also walk you through the setup for collecting income information, including child support.

**🧑‍🏫 👩🏽‍💻 Step-by-Step Guide (With Developer Guidance for Beginners)**

**✅ STEP 1: Start Simple – Build and Test One Piece at a Time**

**Before jumping into the full weekly calendar and financial flow,** your first focus should be:

✅ Building the **Income Source Setup Form** to ask users about their income streams, store their answers, and conditionally create a third source (e.g., child support) if needed.

**🛡️ Precautions to Avoid Bugs (as a beginner)**

1. **✅ Always test one feature at a time**  
   Don’t try to wire up the full calendar + income + Plaid integration all at once. Instead, test small parts (like a single form screen).
2. **✅ Use labels and variables consistently**  
   Stick with exact field names (like income\_1\_amount, not income1 and then income-one in another place). Inconsistent names cause silent bugs.
3. **✅ Default values are your friend**  
   Always set **initial values** (like 0 for income fields). It prevents undefined or null errors later when you run calculations.
4. **✅ Add simple validations**  
   Make sure users can’t skip required fields. This helps prevent storing incomplete data.
5. **✅ Preview the UI in Lovable often**  
   Use the **Preview** and **Simulate Responses** features in Lovable to test your logic visually as you go.

**🧾 INCOME SETUP FLOW – WHAT YOU'LL BUILD**

Here’s the user journey:

1. Ask the user about **Income Source 1**
2. Ask about a **Side Gig or 2nd Source**
3. Ask: “Do you receive child support?”
   * If **Yes**, ask for the amount
   * If **No**, skip
4. Store all values in the database

**💬 PROMPTS TO USE IN LOVABLE**

**💡 Prompt 1: Create the Income Question Flow**

Create a form component that asks the user:

1. "How much do you receive monthly from your main job?" → Save as `income\_1\_amount`

2. "Do you have a side gig or second source of income?" (Yes/No)

- If Yes: Ask "How much do you earn monthly from this side gig?" → Save as `income\_2\_amount`

3. "Do you receive child support?" (Yes/No)

- If Yes: Ask "How much do you receive monthly in child support?" → Save as `child\_support\_amount`

- If No: Set `child\_support\_amount` = 0

Store all entries in the table `user\_income\_sources`, and link each row to the logged-in user via `user\_id`.

Ensure all numeric fields default to 0 if left blank.

**🗂️ DATABASE SCHEMA YOU NEED**

Table: user\_income\_sources

Fields:

- user\_id (string) — linked to `users.id`

- income\_1\_amount (number)

- income\_2\_amount (number)

- child\_support\_amount (number)

- created\_at (date)

**🛠️ What You Can Do Next**

After this screen:

* You can **use these income values** in daily balance calculations.
* Pull them into your **daily\_cashflow** logic to compute “Total Income”.
* Later, allow the user to **edit or update** their income inputs.

**Final Beginner Tip 💬**

“Think of your app like building a sandwich — you add one layer at a time, test that it tastes right, then move to the next layer. If the bottom bread (data collection) is soggy (buggy), the whole thing falls apart.”

Let me know when you’re ready for the prompts that integrate this with the weekly calendar view or with Plaid API for auto-detected income!

Perfect! Let's walk through this like a patient developer mentoring a beginner in **Lovable**. We’ll first set up a simple onboarding flow to collect **income and expense items** from new users, then build a **personalized dashboard** that becomes the default screen once setup is complete.

**🧑‍💻🪜 PART 1: First-Time User Flow – Income & Expense Setup**

**🎯 GOAL:**

Ask the user to define their **income** and **recurring expenses** so we can calculate their daily/weekly/monthly budget view later.

**🪜 STEPS:**

**✅ Step 1: Create Income Setup**

Use the prompt below to build a form that collects:

* Main Job Income
* Side Gig Income
* (Optional) Child Support Income

plaintext

CopyEdit

Create a form titled "Tell us about your income" with the following questions:

1. "How much do you earn each month from your main job?" → Save as `income\_1\_amount`

2. "Do you have any side income or a second job?" (Yes/No)

- If Yes: Ask "How much do you earn monthly from this side job?" → Save as `income\_2\_amount`

3. "Do you receive child support?" (Yes/No)

- If Yes: Ask "How much do you receive monthly in child support?" → Save as `child\_support\_amount`

Store the results in the `user\_income\_sources` table. Link each entry to the current user with `user\_id`.

**✅ Step 2: Create Expense Setup**

Ask for common bills to get a basic picture of fixed outflows.

plaintext

CopyEdit

Create a form titled "Now let's look at your expenses" with the following inputs:

Ask the user to enter the estimated monthly amounts for:

- Rent or Mortgage → `housing\_expense`

- Utilities (Electric, Water, Gas) → `utilities\_expense`

- Phone/Internet → `communications\_expense`

- Groceries → `grocery\_expense`

- Transportation → `transport\_expense`

- Debt or Loan Payments → `debt\_expense`

- Childcare → `childcare\_expense`

- Subscriptions (e.g. Netflix, Gym) → `subscriptions\_expense`

Store these values in the `user\_expense\_items` table and link them to the user with `user\_id`.

**📂 DATABASE SCHEMAS**

**Table: user\_income\_sources**

| **Field** | **Type** |
| --- | --- |
| user\_id | string |
| income\_1\_amount | number |
| income\_2\_amount | number |
| child\_support\_amount | number |
| created\_at | datetime |

**Table: user\_expense\_items**

| **Field** | **Type** |
| --- | --- |
| user\_id | string |
| housing\_expense | number |
| utilities\_expense | number |
| communications\_expense | number |
| grocery\_expense | number |
| transport\_expense | number |
| debt\_expense | number |
| childcare\_expense | number |
| subscriptions\_expense | number |
| created\_at | datetime |

**🖥️🪜 PART 2: Personalized Dashboard for Returning Users**

**🎯 GOAL:**

Once a user finishes setup, they’re taken to a dashboard that shows:

* **Total Monthly Income**
* **Total Monthly Expenses**
* **Estimated Daily Budget**
* **Weekly Financial Status Overview (red/green dots)**

**🪜 STEPS:**

**✅ Step 1: Compute Dashboard Values**

plaintext

CopyEdit

Create a backend function that:

- Fetches all income values for the logged-in user from `user\_income\_sources`

- Fetches all expense values from `user\_expense\_items`

- Computes:

- `total\_income = sum of all income sources`

- `total\_expenses = sum of all expense items`

- `estimated\_daily\_budget = (total\_income - total\_expenses) / 30`

**✅ Step 2: Create a Visual Dashboard Component**

plaintext

CopyEdit

Create a startup screen titled "Your Financial Dashboard" that displays:

1. Total Monthly Income → Green text

2. Total Monthly Expenses → Red text

3. Estimated Daily Budget → Neutral color

4. A horizontal bar with 7 circular icons for the current week (S-M-T-W-T-F-S)

- Each icon is green or red based on `status\_color` from `daily\_cashflow`

- Tapping an icon opens that day’s transactions

Set this screen as the default home screen for users who have completed both income and expense setup.

**✅ Step 3: Gate Access Until Setup Is Complete**

plaintext

CopyEdit

If user does not have records in both `user\_income\_sources` and `user\_expense\_items`, direct them to the setup forms first.

Once setup is complete, redirect them to the dashboard screen by default on login.

**👨🏾‍🏫 Developer Advice:**

💡 *“Always test the forms separately before connecting them to the dashboard. Start by inputting test data into each table, and confirm it shows up correctly before wiring everything together.”*

Would you like me to turn this flow into a Lovable-ready JSON config or UI storyboard next?

JSON Config:

{

"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" }

]

}

]

},

"ui": {

"components": [

{

"name": "IncomeSetupForm",

"type": "form",

"fields": [

"income\_1\_amount",

{ "type": "conditional", "condition": "side\_gig == true", "fields": ["income\_2\_amount"] },

{ "type": "conditional", "condition": "receives\_child\_support == true", "fields": ["child\_support\_amount"] }

],

"data\_source": "user\_income\_sources"

},

{

"name": "ExpenseSetupForm",

"type": "form",

"fields": [

"housing\_expense",

"utilities\_expense",

"communications\_expense",

"grocery\_expense",

"transport\_expense",

"debt\_expense",

"childcare\_expense",

"subscriptions\_expense"

],

"data\_source": "user\_expense\_items"

},

{

"name": "FinancialDashboard",

"type": "dashboard",

"components": [

{

"type": "summary\_card",

"title": "Monthly Income",

"value": "SUM(user\_income\_sources.\*\_amount)",

"style": "green"

},

{

"type": "summary\_card",

"title": "Monthly Expenses",

"value": "SUM(user\_expense\_items.\*\_expense)",

"style": "red"

},

{

"type": "summary\_card",

"title": "Estimated Daily Budget",

"value": "(SUM(income) - SUM(expenses)) / 30"

},

{

"type": "calendar\_strip",

"data\_source": "daily\_cashflow",

"label": "first\_letter\_of\_day",

"color\_code": "status\_color",

"on\_tap": "navigate\_to\_daily\_transactions(date)"

}

]

}

]

},

"logic": {

"navigation": [

{

"condition": "NOT EXISTS(user\_income\_sources WHERE user\_id = current\_user)",

"navigate\_to": "IncomeSetupForm"

},

{

"condition": "NOT EXISTS(user\_expense\_items WHERE user\_id = current\_user)",

"navigate\_to": "ExpenseSetupForm"

},

{

"condition": "ELSE",

"navigate\_to": "FinancialDashboard"

}

]

}

}