**Comprehensive System Specification: Family Budget Management Application (Version 1.0)**

1. **Introduction & System Goals**
   * 1.1. System Purpose
   * 1.2. Key Objectives
2. **Target Audience & User Needs**
3. **Core Principles**
   * 3.1. Design & UX Principles
   * 3.2. Technical Principles
4. **Functional Specifications by Module**
   * 4.1. User Authentication & Management
   * 4.2. Category & Subcategory Management (including HE/EN names, user-defined)
   * 4.3. Income Management (including future Recurring Income)
   * 4.4. Expense Management (Single, Recurring Instances, Installment Instances; isProcessed field)
   * 4.5. Budget Profile & Allocation Management
   * 4.6. Dashboard & Financial Overview (with period selection)
   * 4.7. Transaction History & Reporting (with filtering)
   * 4.8. User Onboarding Process
   * 4.9. User Settings
5. **Workflow Descriptions (Key User Flows)**
   * 5.1. New User Registration & Onboarding
   * 5.2. User Login
   * 5.3. Adding a New Transaction (Single Income/Expense)
   * 5.4. Adding a New Recurring Expense Definition (and instance generation)
   * 5.5. Adding a New Installment Transaction (and payment generation)
   * 5.6. Viewing Dashboard
   * 5.7. Managing Budgets (Profiles and Allocations)
   * 5.8. Viewing Transaction List & Filtering
6. **Data Model (Entities & Relationships - Detailed)**
   * (List each entity with its fields and relations as defined in our backend code)
7. **API Specification (Key Endpoints & Data Structures - Summary)**
   * (List main API groups and key endpoints as defined in our backend routes)
8. **Technology Stack & Architecture (Frontend & Backend)**
9. **Deployment & Environment**
10. **Future Considerations & Roadmap (Based on your selected items)**
    * Themes & Appearance
    * User Profile Customization
    * "What-If" Budget Scenarios
    * Debt Paydown Planner
    * Bill Payment Reminders (Alerts for unprocessed recurring items)
    * Advanced Customizable Reports
    * Trend Analysis Over Longer Periods
    * Budget vs. Actual Detailed Reports
    * Printable Reports
    * Responsive Web for Mobile (Optimized)
    * (Plus any others you confirm)  
      **Appendix A: Initial Category & Subcategory List (Hebrew & English)**

הבנתי מצוין. אם כך, אני אכין כל חלק מהמסמך המאוחד במלואו, כאילו הוא "מיני-פרומפט" בפני עצמו, ואשלח לך אותו. אתה תוכל פשוט להעתיק ולהדביק אותו. אחרי שתאשר, נעבור לחלק הבא.

נתחיל עם שלושת הסעיפים הראשונים של המסמך המאוחד.

**Prompt: Unified Specification Document - Sections 1-3  
(Copy and paste the entire content below into your document)**

**Comprehensive System Specification: Family Budget Management Application**

**Version:** 1.0  
**Date:** [Current Date]

**1. Introduction & System Goals**

**1.1. System Purpose:**  
The Family Budget Management Application is a comprehensive digital tool designed to empower users to effectively manage their personal and family finances. The system aims to provide clarity on financial inflows and outflows, facilitate adherence to budgetary goals, promote savings, and offer insights for informed financial decision-making. It is intended to be a robust, reliable, and long-term solution for users seeking financial control.

**1.2. Key Objectives:**

* To provide a secure, intuitive, and engaging platform for comprehensive financial tracking.
* To enable detailed and flexible recording, categorization, and management of all income and expense types, including one-time, recurring, and installment-based transactions.
* To offer powerful and adaptable budgeting tools, allowing users to define multiple budget profiles (e.g., for different life phases or financial goals) and allocate funds meticulously by category and subcategory.
* To deliver a clear, actionable, and customizable financial overview via a central dashboard, enabling users to quickly assess their status and progress.
* To facilitate a smooth and helpful user onboarding experience through a guided setup process for initial income, fixed expenses, and budget allocations.
* To support multilingual use, with initial full support for Hebrew (RTL) and English (LTR), including data and UI elements.
* To establish a scalable and maintainable technical foundation for future enhancements and feature expansions.

**2. Target Audience & User Needs**

**2.1. Target Audience:**  
The system is designed for a broad audience, including:

* Individuals seeking to gain control over their personal spending and savings.
* Couples managing joint household finances.
* Families tracking expenses for multiple members and diverse needs.
* Users at various stages of financial literacy, from beginners needing guidance to experienced budgeters looking for a powerful tool.

**2.2. Core User Needs Addressed:**

* **Clarity & Understanding:** "Where does my money go? What is my net cash flow?"
* **Control & Planning:** "How can I stick to a budget? How can I plan for future expenses and savings?"
* **Efficiency:** "I need a quick and easy way to log transactions, especially on the go."
* **Goal Achievement:** "How can I save for specific goals (e.g., vacation, down payment, debt reduction)?"
* **Insight & Foresight:** "What are my spending trends? What will my financial situation look like next month/quarter?"
* **Reduced Financial Stress:** Gaining a sense of order and predictability in their financial lives.

**3. Core Principles**

**3.1. Design & User Experience (UX) Principles:**

* **User-Centricity:** All design and functionality decisions are driven by the needs and goals of the end-user.
* **Simplicity & Elegance:** A clean, uncluttered, and aesthetically pleasing interface that avoids overwhelming the user. Focus on essential information and actions.
* **Intuitive Navigation & Workflow:** Logical, predictable, and easy-to-learn pathways for all tasks, minimizing cognitive load.
* **Efficiency & Speed:** Streamlined processes for frequent actions (e.g., transaction entry, budget review) to save user time and effort.
* **Informative Feedback:** Clear, concise, and timely feedback for all user interactions (loading states, success messages, validation errors, confirmations).
* **Consistency:** A cohesive and uniform design language (color palette, typography, iconography, component behavior, terminology) throughout all parts of the application.
* **Responsiveness & Cross-Device Experience:** A fully responsive design ensuring an optimal and consistent experience across desktops, tablets, and mobile devices (web-based, mobile-optimized).
* **Accessibility (WCAG Adherence):** Commitment to making the application accessible to users with disabilities, considering color contrast, keyboard navigability, screen reader compatibility, and semantic HTML (to be progressively enhanced and audited).
* **Language & Directionality:** Full support for Hebrew (RTL) and English (LTR) from the outset, including UI text, data display (e.g., category names), and layout mirroring. A user-selectable language preference mechanism will be provided.
* **Encouragement & Positive Reinforcement:** Design elements and feedback that motivate users and reinforce positive financial behaviors.

**3.2. Technical Principles:**

* **Robustness & Reliability:** Ensuring data integrity, accurate calculations, and stable system performance.
* **Scalability:** Architecting the backend and database to efficiently handle a growing number of users, transactions, and data complexity.
* **Maintainability & Extensibility:** Writing clean, modular, well-documented, and testable code to facilitate ongoing maintenance and future feature development.
* **Security:** Implementing industry best practices for data security at rest and in transit, secure authentication and authorization mechanisms, and protection against common web vulnerabilities (OWASP Top 10 considerations).
* **Performance:** Optimizing database queries, API response times, and frontend rendering speed for a fluid and responsive user experience.
* **Testability:** Designing components and services with testability in mind, enabling comprehensive unit, integration, and end-to-end testing.
* **Modularity (Backend & Frontend):** Separating concerns into distinct modules/layers (e.g., API routes, services, repositories in backend; container/presentational components, hooks, contexts in frontend).
* **Adherence to Technology Best Practices:** Utilizing the chosen technology stack (React, Node.js, TypeORM, PostgreSQL, Tailwind CSS) according to established best practices and conventions.

בסדר גמור, הנה כל **סעיף 4: Functional Specifications by Module** מהמסמך המאוחד, המפרט את הפונקציונליות של כל מודול מרכזי באפליקציה. זהו סעיף ארוך ומפורט.

**( המשך המסמך המאוחד )**

**4. Functional Specifications by Module**

This section details the specific functionalities and user interactions for each core module of the Family Budget Management Application.

**4.1. User Authentication & Management**

* **4.1.1. User Registration:**
  + **Functionality:** Allows new users to create an account.
  + **Input Fields:**
    - Full Name (שם מלא) (Optional, User.name)
    - Email Address (כתובת אימייל) (Mandatory, Unique, Valid Format, User.email)
    - Password (סיסמה) (Mandatory, Minimum length [e.g., 8 characters], complexity requirements - future, User.password - to be hashed)
    - Confirm Password (אימות סיסמה)
  + **Process:**
    - User fills the registration form.
    - Client-side validation for format and matching passwords.
    - On submit, data is sent to POST /api/auth/signup.
    - Backend validates data (e.g., email uniqueness), hashes the password, and creates a new User record.
    - Backend returns a success message.
  + **User Feedback:** Clear error messages for validation failures (e.g., "Email already exists," "Passwords do not match," "Password too short"). Success message upon completion, guiding the user to log in.
  + **(Future):** Email verification process (send confirmation link).
* **4.1.2. User Login:**
  + **Functionality:** Allows registered users to access their accounts.
  + **Input Fields:**
    - Email Address (כתובת אימייל)
    - Password (סיסמה)
  + **Process:**
    - User fills the login form.
    - On submit, credentials sent to POST /api/auth/login.
    - Backend validates credentials (compares hashed password).
    - On success, backend generates a JWT and returns it along with essential user information (e.g., { token, user: { id, name, email } }).
    - Frontend stores the JWT securely (e.g., localStorage or HttpOnly cookie if web-only and server-side rendering is considered) and updates the auth state (e.g., via AuthContext).
    - User is redirected to the main dashboard.
  + **User Feedback:** Error messages for invalid credentials. Loading indicator during login attempt.
  + **(Future):** "Remember Me" functionality (extending JWT validity or using refresh tokens).
* **4.1.3. User Logout:**
  + **Functionality:** Allows authenticated users to securely end their session.
  + **Process:**
    - User clicks a "Logout" button.
    - Frontend removes the JWT from storage.
    - Frontend clears the authenticated user state.
    - User is redirected to the login page.
    - (Optional Backend) API endpoint to invalidate the token on the server-side if using a token blacklist/whitelist.
* **4.1.4. Session Management & Route Protection:**
  + **Functionality:** Ensures that only authenticated users can access protected parts of the application.
  + **Mechanism:**
    - Frontend uses a ProtectedRoute component that checks authentication status (via AuthContext).
    - If not authenticated, redirects to the login page.
    - API requests to protected backend endpoints include the JWT in the Authorization: Bearer <token> header.
    - Backend auth.middleware.ts validates the JWT on each protected request. If invalid/expired, returns a 401 error.
  + **Token Expiry Handling:** Frontend should gracefully handle token expiry (e.g., detected via 401 error from API) by logging out the user and redirecting to login.
* **4.1.5. User Profile Retrieval:**
  + **Functionality:** Allows the application to fetch details of the currently logged-in user.
  + **API:** GET /api/auth/profile (protected) returns user details (id, name, email, createdAt, etc., excluding password).
  + **Usage:** Used by AuthContext on application load (if a token exists) to re-establish user state, and potentially by a user profile page.
* **(Future - User Management):**
  + User ability to change their password.
  + User ability to update their profile information (name, etc.).
  + User ability to delete their account (considering data retention policies).
  + Admin panel for user management (if applicable).

**4.2. Category & Subcategory Management**

* **4.2.1. Core Concepts:**
  + **Categories (Category Entity):** High-level financial groupings. Each category has a type ('expense' or 'income'). Categories can be global (system-defined) or user-specific (linked via userId, userId is nullable for global ones). Each category has name\_he and name\_en fields for localization. Categories can be archived (soft delete).
  + **Subcategories (Subcategory Entity):** More granular classification, belonging to an 'expense' type Category. Subcategories also have name\_he, name\_en, and an archived flag.
* **4.2.2. Seeding:**
  + The system is pre-populated with a comprehensive list of global expense and income categories, and expense subcategories (see Appendix A), with names in both Hebrew and English.
* **4.2.3. Data Retrieval (for UI):**
  + **API:** GET /api/categories?type=<expense|income>[&lang=<he|en>]
  + **Functionality:** Fetches all non-archived global categories AND non-archived user-specific categories for the authenticated user, filtered by the specified type. If lang is provided, it might influence which name field is prioritized or if both are returned. For expenses, associated non-archived subcategories are included.
  + **Usage:** Populating dropdowns in transaction entry forms and budget allocation screens.
* **4.2.4. User-Defined Category/Subcategory Management (Frontend UI - Future, API Backend - Planned):**
  + Users will be able to create new personal categories (type 'expense' or 'income') and subcategories (for their expense categories).
  + Users will be able to edit the names of their personal categories/subcategories.
  + Users will be able to archive/unarchive their personal categories/subcategories. Archiving hides them from selection but preserves historical data linkage.
  + **API (Backend to be extended/implemented):**
    - POST /api/categories: Create user-specific category.
    - PUT /api/categories/:id: Update user-specific category (name, archive status).
    - POST /api/subcategories: Create user-specific subcategory (linked to a user's category).
    - PUT /api/subcategories/:id: Update user-specific subcategory.

**4.3. Income Management**

* **4.3.1. Single Income Entry:**
  + **Functionality:** Allows users to record individual income transactions.
  + **UI:** Via a dedicated section in the AddTransactionModal or a specific income form.
  + **Fields:**
    - Amount (סכום) (Mandatory, positive number)
    - Date of Receipt (תאריך קבלה) (Mandatory, date picker)
    - Income Category (קטגוריית הכנסה) (Mandatory, selection from user's available 'income' type categories via GET /api/categories?type=income)
    - Description/Source (תיאור/מקור ספציפי) (Optional, text)
    - Means of Receipt (אמצעי קבלה) (Optional, text, e.g., "Bank Transfer," "Cash")
  + **API:** POST /api/incomes (protected) saves the new Income record, linked to userId and categoryId.
* **4.3.2. Viewing & Managing Incomes:**
  + **List View (TransactionsListPage.jsx):** Incomes are displayed alongside expenses, distinguishable by type.
  + **API:**
    - GET /api/incomes: Fetches all incomes for the authenticated user, sortable (e.g., by date).
    - GET /api/incomes/:id: Fetches a specific income record.
    - PUT /api/incomes/:id: Updates an existing income record.
    - DELETE /api/incomes/:id: Soft-deletes an income record.
    - PATCH /api/incomes/:id/restore: Restores a soft-deleted income record.
* **4.3.3. Recurring Income (Future - Backend entity RecurringIncomeDefinition similar to expenses):**
  + **Definition:** Users define recurring income sources (e.g., salary, rent received).
    - Fields: Amount, Category, Description, Frequency, Interval, Start Date, End Date/Occurrences (optional).
  + **Instance Generation:** System (potentially via scheduler or upfront generation) creates individual Income records based on the definition, marking future ones appropriately (e.g., using a field like isProcessed or relying on date).
  + **User Interaction:** Option for automatic recording or user confirmation for variable recurring income.
  + **Management UI:** Screen to view, create, edit, and (soft) delete RecurringIncomeDefinitions.
  + **API:** CRUD endpoints for RecurringIncomeDefinition.

**4.4. Expense Management**

* **4.4.1. Unified Expense Entry (AddTransactionModal.jsx):**
  + **Functionality:** A single modal/form to input all types of expenses.
  + **Type Selection:** User selects expenseType: 'single', 'recurring', or 'installment'.
  + **Common Fields:** Amount (for single/recurring definition), Date (serves as expense date, start date, or first payment date), Subcategory (selection from user's 'expense' subcategories), Description (optional), Payment Method (optional).
* **4.4.2. Single Expense (expenseType: 'single'):**
  + Directly creates an Expense record with isProcessed: true.
  + API: POST /api/expenses handles this type.
* **4.4.3. Recurring Expense (expenseType: 'recurring'):**
  + **Definition Input:** User provides additional fields: Frequency, Interval, Occurrences/End Date.
  + **Backend Process (via POST /api/expenses):**
    1. Creates a RecurringExpenseDefinition record.
    2. Pre-generates all future Expense instances based on the definition. These instances have expenseType: 'recurring\_instance', parentId linking to the definition, date set to their future due date, and isProcessed: false.
    3. The first instance might be marked isProcessed: true if its startDate is today or in the past.
  + **Management UI (Future):** Dedicated screen to manage RecurringExpenseDefinitions (view, edit, delete). Editing a definition will regenerate future, unprocessed Expense instances.
  + **API (Definitions):** GET/PUT/DELETE /api/recurring-definitions.
* **4.4.4. Installment Expense (expenseType: 'installment'):**
  + **Definition Input:** User provides additional fields: Total Amount of purchase, Number of Installments. date field serves as firstPaymentDate.
  + **Backend Process (via POST /api/expenses):**
    1. Creates an InstallmentTransaction record (calculates installmentAmount, handles rounding on the last payment).
    2. Pre-generates all Expense instances (payments). These instances have expenseType: 'installment\_instance', parentId linking to the transaction, date set to their future due date, and isProcessed: false.
  + **Management UI (Future):** Dedicated screen to manage InstallmentTransactions (view, limited edits like description, mark as completed, delete).
  + **API (Definitions):** GET/PUT/DELETE /api/installment-transactions.
* **4.4.5. Processing Planned Expenses:**
  + Expense instances with isProcessed: false and a date in the past or present are considered "due."
  + **UI Indication:** The frontend will visually indicate these (e.g., in lists, on the dashboard).
  + **(Future Scheduler):** A backend scheduler will periodically scan for such expenses and can automatically mark isProcessed: true, or trigger user notifications. For now, users might manually "confirm" or edit them, which would set isProcessed: true.
* **4.4.6. Viewing & Managing Expense Instances:**
  + **List View (TransactionsListPage.jsx):** Expenses are displayed alongside incomes. Planned expenses (isProcessed: false) are visually distinct.
  + **API:**
    1. GET /api/expenses: Fetches all expenses for the user (includes planned/unprocessed by default, can be filtered by API).
    2. GET /api/expenses/:id: Fetches a specific expense instance.
    3. PUT /api/expenses/:id: Updates an existing expense instance (e.g., amount if actual differed, description, mark as isProcessed: true). Does *not* alter the parent definition.
    4. DELETE /api/expenses/:id: Soft-deletes an expense instance.
    5. PATCH /api/expenses/:id/restore: Restores a soft-deleted expense instance.

**4.5. Budget Profile & Allocation Management**

* **4.5.1. Budget Profiles (BudgetProfile Entity):**
  + **Concept:** Allows users to create and manage multiple distinct budgeting scenarios or plans (e.g., "Monthly Standard Budget," "Vacation Savings Budget," "Aggressive Debt Paydown Budget").
  + **Fields:** Name, Description, Start Date (optional), End Date (optional), isActive flag.
  + **Activation Logic:** Only one BudgetProfile can be isActive per user at any given time. Activating one profile programmatically deactivates any other currently active profile for that user (handled in backend API).
  + **UI (Future - Dedicated Budget Management Section):**
    - List all budget profiles.
    - Create new profiles.
    - Edit profile details (name, dates).
    - Activate/Deactivate profiles.
    - (Soft) Delete profiles (which will cascade to delete its monthly allocations).
  + **API:** POST/GET/PUT/DELETE /api/budget-profiles.
* **4.5.2. Monthly Budget Allocations (Budget Entity):**
  + **Concept:** Within a specific BudgetProfile, users define monthly spending targets for each expense Subcategory.
  + **Fields:** budgetProfileId, subcategoryId, year, month, allocatedAmount.
  + **Uniqueness:** Enforced for budgetProfileId, subcategoryId, year, month.
  + **UI (Budget Setup/Management Screen & Onboarding):**
    - User selects a BudgetProfile and a month/year.
    - Displays a list of expense subcategories.
    - User inputs allocatedAmount for each desired subcategory.
    - System displays spentAmount (calculated from actual Expense records for that subcategory, profile, month/year) and remainingAmount.
  + **API:**
    - POST /api/budgets: Upserts (creates or updates) a budget allocation.
    - GET /api/budgets?budgetProfileId=X&year=Y&month=M: Fetches allocations for a profile/month, including calculated spentAmount.
    - DELETE /api/budgets/:id: Deletes a specific allocation entry.

**4.6. Dashboard & Financial Overview**

* **4.6.1. Purpose:** Central landing page providing an immediate, actionable financial snapshot for a selected period (default: current calendar month).
* **4.6.2. Period Selection (UI):** Controls (e.g., arrows, dropdown) to navigate to previous/next months or select a specific month/year. This updates the selectedPeriod state, triggering data refetch for period-sensitive widgets.
* **4.6.3. Key Widgets (Data supplied by /api/dashboard/\* endpoints):**
  + **Periodical Summary:**
    - Total Income (for selected period).
    - Total Processed Expenses (for selected period, where isProcessed=true).
    - Current Balance (Income - Processed Expenses).
    - **Budget Comparison:**
      * If UserSettings.monthlyBudgetGoal is set: Compares totalProcessedExpenses against this overall goal.
      * Alternatively/Additionally: Compares totalProcessedExpenses against the sum of all allocatedAmounts from the active BudgetProfile for the selected period.
      * Visualized with a progress bar (color-coded).
  + **Add Action FAB (Floating Action Button):** Opens AddTransactionModal for quick entry of new income or expense.
  + **Recent Transactions:** Lists the last N (e.g., 5) transactions (incomes and *processed* expenses), sorted by date. Shows icon, description/category, amount, and date.
  + **Expense Distribution Chart:** Pie or Bar chart showing processed expenses for the selected period, aggregated by main Category (non-archived). Interactive (hover for details).
* **4.6.4. UX:** Skeleton loaders during data fetch, clear error states, informative empty states. Data refresh button.
* **(Future Dashboard Features):** Widgets for Savings Goals, Debt Tracking, Top Categories Budget vs. Actual, Income vs. Expense trends over time.

**4.7. Transaction History & Reporting**

* **4.7.1. All Transactions Page (TransactionsListPage.jsx):**
  + **Display:** A comprehensive, filterable, and sortable list/table of all user's income and expense records.
  + **Data includes:** date, description (or category/subcategory name), type (income/expense), amount, isProcessed status (for expenses), deletedAt status.
  + **Filtering:** By date range, type (income/expense), isProcessed status, free-text search (on description/category).
  + **Sorting:** By date, amount.
  + **Actions per transaction:**
    - Edit: Opens AddTransactionModal pre-filled for editing basic fields of the selected transaction instance.
    - Soft Delete: Marks the transaction with a deletedAt timestamp.
    - Restore: Clears the deletedAt timestamp.
* **4.7.2. Advanced Reporting & Analytics (Future):**
  + Customizable report builder (select fields, filters, date ranges, chart types).
  + Trend analysis over extended periods (e.g., year-over-year spending).
  + Detailed Budget vs. Actual reports per category/subcategory.
  + Cash flow statements.
  + Printable reports.

**4.8. User Onboarding Process (Wizard)**

* **4.8.1. Trigger:** Automatically for new users after their first successful login. A flag in UserSettings or localStorage tracks completion.
* **4.8.2. Goal:** Guide users to input essential initial data to make the application immediately useful.
* **4.8.3. Steps:**
  1. **Define Regular Monthly Income:** User lists primary sources of recurring income and their estimated monthly amounts.
     + *Backend Action:* Could create simplified Income records for the current month, or (ideally) create RecurringIncomeDefinitions.
  2. **Define Fixed Essential Monthly Expenses:** User selects key expense subcategories (e.g., Rent/Mortgage, Utilities, Loan Payments) and provides estimated monthly amounts.
     + *Backend Action:* Creates RecurringExpenseDefinition records for these. Corresponding Expense instances (with isProcessed=false) are generated.
  3. **Initial Budget Allocation for Variable Expenses:**
     + System displays (Estimated Monthly Income from Step 1) - (Total Fixed Expenses from Step 2) = Available for Discretionary Spending.
     + User is presented with other (non-fixed) expense subcategories and allocates budget amounts (allocatedAmount) for the current month.
     + *Backend Action:* Creates Budget entries linked to the user's default/active BudgetProfile for the current month.
* **4.8.4. User Experience:** Clear progression, option to skip steps or the entire wizard, save progress. Navigates to Dashboard upon completion.

**4.9. User Settings (UserSettings Entity & Page)**

* **4.9.1. Functionality:** Allows users to configure application preferences.
* **4.9.2. Settings (Initial):**
  + monthlyBudgetGoal: An overall monthly spending target (optional). Used in Dashboard summary.
  + defaultCurrency: Displayed (e.g., "ILS"), not editable in this phase.
* **4.9.3. UI (UserSettingsPage.jsx):** A simple form to view and edit settable fields.
* **API:** GET /api/user-settings and PUT /api/user-settings.
* **(Future):** Language preference, date/number format, notification preferences.

בסדר גמור, הנה המשך המסמך המאוחד, סעיפים 5 עד 9.

**(המשך המסמך המאוחד)**

**5. Workflow Descriptions (Key User Flows)**

This section outlines the primary user journeys within the application.

* **5.1. New User Registration & Onboarding:**
  1. **Access:** User navigates to the application URL or clicks "Sign Up."
  2. **Registration Form (SignupPage.jsx):** User enters Name (optional), Email, Password, and Confirm Password. Client-side validation is performed.
  3. **Submission:** Data is sent to POST /api/auth/signup.
  4. **Backend:** Validates data, checks for existing email, hashes password, creates User record.
  5. **Response:** Success message. Frontend may redirect to Login page or prompt user to log in.
  6. **First Login:** User logs in via LoginPage.jsx (POST /api/auth/login).
  7. **Onboarding Check:** Application checks if the user has completed onboarding (e.g., via a flag in UserSettings or localStorage).
  8. **Onboarding Wizard (if not completed):**
     + **Welcome Screen:** Brief introduction.
     + **Step 1: Income Setup:** User inputs details for their primary recurring monthly income sources (description, amount, category). Data is saved (e.g., as simplified income records or RecurringIncomeDefinition).
     + **Step 2: Fixed Expenses Setup:** User identifies and inputs amounts for major fixed monthly expenses (e.g., rent, mortgage, core utilities). Data is saved as RecurringExpenseDefinition and initial Expense instances are generated.
     + **Step 3: Variable Budget Allocation:**
       - System displays (Total Estimated Income) - (Total Fixed Expenses) = Discretionary Amount.
       - User allocates budgets to various expense subcategories for the current month within their default/active BudgetProfile. Data is saved as Budget entries.
     + **Completion:** User is navigated to the main Dashboard. Onboarding flag is set.
  9. **Existing User Login:** User logs in. If onboarding is complete, user is navigated directly to the Dashboard.
* **5.2. Adding a New Transaction (Single Income/Expense):**
  1. **Access:** User clicks the global Add Action FAB (+) from the Dashboard or other relevant screens.
  2. **Modal/Form (AddTransactionModal.jsx):**
     + User selects transaction type: "Income" or "Expense".
     + User selects expense type: "Single" (חד-פעמית).
  3. **Data Entry:** User fills in Amount, Date, Category/Subcategory (selected from dynamic list fetched via GET /api/categories), Description (optional), Payment Method (optional for expenses).
  4. **Submission:** Client-side validation. Data sent to POST /api/incomes or POST /api/expenses (with expenseType: 'single').
  5. **Backend:** Validates data, creates new Income or Expense record (with isProcessed: true for single expenses), links to user and category/subcategory.
  6. **Response:** Success message and created transaction data.
  7. **Frontend Update:** Modal closes. Relevant data views (Dashboard summary, recent transactions, transaction lists) are invalidated and refetched via React Query.
* **5.3. Adding a New Recurring Expense Definition:**
  1. **Access:** Via FAB -> "Add Expense" -> Select "Recurring" type in AddTransactionModal.jsx.
  2. **Data Entry:** User fills common expense fields (Amount for one occurrence, Date as Start Date, Subcategory, etc.) AND specific recurring fields (Frequency, Interval, Occurrences/End Date).
  3. **Submission:** Data sent to POST /api/expenses with expenseType: 'recurring'.
  4. **Backend:**
     + Creates a RecurringExpenseDefinition record.
     + Calculates and pre-generates all future Expense instances (isProcessed: false, expenseType: 'recurring\_instance', parentId set, date as future due date).
     + Updates nextDueDate and isActive on the definition.
  5. **Response:** Success message and created definition data.
  6. **Frontend Update:** Modal closes. Dashboard and transaction lists refresh. (Future: A dedicated "Recurring Expenses" list would also refresh).
* **5.4. Adding a New Installment Transaction:**
  1. **Access:** Via FAB -> "Add Expense" -> Select "Installments" type in AddTransactionModal.jsx.
  2. **Data Entry:** User fills common expense fields (Date as First Payment Date, Subcategory, etc.) AND specific installment fields (Total Amount, Number of Installments).
  3. **Submission:** Data sent to POST /api/expenses with expenseType: 'installment'.
  4. **Backend:**
     + Creates an InstallmentTransaction record (calculates individual installmentAmount).
     + Pre-generates all Expense payment instances (isProcessed: false, expenseType: 'installment\_instance', parentId set, date as future due date).
  5. **Response:** Success message and created transaction definition data.
  6. **Frontend Update:** Modal closes. Dashboard and transaction lists refresh. (Future: An "Installment Plans" list would also refresh).
* **5.5. Viewing Dashboard:**
  1. **Access:** Default page after login, or via main navigation.
  2. **Data Fetching:** DashboardPage.jsx uses React Query hooks to call:
     + GET /api/dashboard/summary?period=<selected\_period>
     + GET /api/dashboard/recent-transactions?limit=5
     + GET /api/dashboard/expense-distribution?period=<selected\_period>
  3. **Display:** Renders widgets with fetched data (Summary, Recent Transactions, Expense Chart).
  4. **Interaction:** User can select different periods (month/year), triggering data refetch for relevant widgets. User can click FAB to add new transactions.
* **5.6. Managing Budgets (Profiles and Allocations):**
  1. **Access:** Via main navigation to "Budget Management" screen(s).
  2. **Profile Management (Future UI):**
     + View list of BudgetProfiles (GET /api/budget-profiles).
     + Create new profile (POST /api/budget-profiles).
     + Edit profile (PUT /api/budget-profiles/:id).
     + Set active profile (PUT /api/budget-profiles/:id with isActive:true).
     + Delete profile (DELETE /api/budget-profiles/:id).
  3. **Monthly Allocation Management (For selected active/chosen profile and month - Future UI):**
     + Fetch allocations and spending (GET /api/budgets?budgetProfileId=X&year=Y&month=M).
     + Display subcategories with allocatedAmount input fields, spentAmount, and remainingAmount.
     + User modifies allocatedAmount.
     + On save, POST /api/budgets (upsert) is called for each modified allocation.
* **5.7. Viewing Transaction List & Filtering (TransactionsListPage.jsx):**
  1. **Access:** Via main navigation.
  2. **Data Fetching:** Calls GET /api/incomes and GET /api/expenses (potentially with filter params if API supports it, otherwise client-side filtering initially).
  3. **Display:** Shows combined list of incomes and expenses, sortable by date.
  4. **Filtering UI:** Controls for date range, type (income/expense), isProcessed status, free-text search.
  5. **Actions per Transaction:**
     + Edit: Opens AddTransactionModal pre-filled, calls PUT on save.
     + Delete: Confirmation dialog, calls DELETE (soft delete).
     + Restore: Confirmation, calls PATCH .../:id/restore.

**6. Data Model (Entities & Relationships)**  
*(This section will list each entity with its fields and relations, as detailed in the "Technical Documentation - Section 5" I provided previously, based on our final entity structure. I will ensure it includes User, Category (with name\_he, name\_en, userId), Subcategory (with name\_he, name\_en), Income, Expense (with expenseType, parentId, isProcessed), UserSettings, RecurringExpenseDefinition, InstallmentTransaction, BudgetProfile, and Budget.)*

**Example Snippet for User.ts (to illustrate format):**

* **User.ts**
  + **Fields:** id (PK, auto-increment), name (String, nullable), email (String, unique, not null), password (String, not null, hashed), createdAt (Timestamp), updatedAt (Timestamp), deletedAt (Timestamp, nullable for soft delete).
  + **Relationships:**
    - settings: OneToOne with UserSettings (User is the inverse side).
    - categories: OneToMany with Category (User can have custom categories, userId in Category is FK).
    - incomes: OneToMany with Income.
    - expenses: OneToMany with Expense.
    - recurringExpenseDefinitions: OneToMany with RecurringExpenseDefinition.
    - installmentTransactions: OneToMany with InstallmentTransaction.
    - budgetProfiles: OneToMany with BudgetProfile.
    - budgets: OneToMany with Budget (direct link for queries, though primarily via BudgetProfile).

**(Full details for all entities will be listed here in the final document.)**

**7. API Specification (Key Endpoints & Data Structures - Summary)**  
*(This section will summarize the key API groups and their main endpoints, as detailed in the "Technical Documentation - Section 7" I provided previously. It will reflect the final API structure supporting all functionalities described above.)*

**Example Snippet (to illustrate format):**

* **Authentication (/api/auth):**
  + POST /signup: { email, password, name? } -> { message }
  + POST /login: { email, password } -> { token, user: {id, email, name} }
  + GET /profile (Protected): -> { id, email, name, createdAt }
* **Expenses (/api/expenses):**
  + POST / (Protected, Unified): { amount, date, subcategoryId, ..., expenseType, [recurring\_fields | installment\_fields] } -> { created\_expense\_instance | definition\_object }
  + GET / (Protected): -> Expense[] (with relations)
  + ... (other CRUD for Expense instances)
* **(Full API endpoint list will be detailed here.)**

**8. Technology Stack & Architecture**

**8.1. Frontend:**

* **Framework/Library:** React (v18+) with Vite as the build tool.
* **Language:** TypeScript (preferred for type safety and scalability) or JavaScript (ES6+).
* **Styling:** Tailwind CSS (utility-first approach, with tailwind.config.js for custom theme). PostCSS with autoprefixer.
* **Routing:** React Router DOM (v6+).
* **State Management:**
  + Server State: @tanstack/react-query (React Query v4/v5) for data fetching, caching, and mutations.
  + Global UI State: React Context API (e.g., AuthContext, potentially SettingsContext, LanguageContext).
  + Local Component State: useState, useReducer.
* **HTTP Client:** Axios, with a configured instance for baseURL and JWT request interceptor.
* **Form Management:** react-hook-form for efficient and performant form handling and validation.
* **Date Utility:** date-fns for date manipulation and formatting.
* **Charting:** recharts for data visualization on the dashboard.
* **Icons:** react-icons or a similar SVG icon library.

**8.2. Backend:**

* **Framework/Runtime:** Node.js (LTS version, e.g., v18 or v20) with Express.js.
* **Language:** TypeScript (compiled to CommonJS for Node.js compatibility).
* **ORM:** TypeORM with PostgreSQL as the database.
* **Authentication:** JWT (JSON Web Tokens) using jsonwebtoken library. Passwords hashed with bcryptjs.
* **Input Validation:** express-validator for API request validation.
* **Database:** PostgreSQL (e.g., v14+).
* **Development Utilities:** nodemon for hot-reloading, concurrently to run multiple scripts.

**8.3. General:**

* **Containerization:** Docker and Docker Compose for consistent development, testing, and deployment environments.
* **Version Control:** Git (with a platform like GitHub, GitLab, or Bitbucket).

**8.4. Architecture Overview:**

* **Client-Server Architecture:** Frontend (React SPA) consumes RESTful APIs provided by the Backend.
* **Backend Layering:**
  + Routes: Define API endpoints.
  + Controllers (or Route Handlers): Handle request/response, call services.
  + Services (Recommended for Business Logic): Encapsulate core business rules and orchestrate repository calls. (To be refactored in if not already).
  + Repositories/Entities: TypeORM handles database interaction.
* **Frontend Structure:** Component-based architecture. Separation of concerns (e.g., container vs. presentational components, custom hooks for logic, contexts for shared state).

**9. Deployment & Environment**

* **Local Development:** Docker Compose (docker-compose.yml) defines services for db (PostgreSQL), backend (Node.js/Express), frontend (Vite dev server), and adminer (DB management UI). Volumes are used for hot-reloading of code.
* **Environment Variables:** Configuration managed via .env files (not committed to Git) for database credentials, JWT secret, API URLs, ports, etc. dotenv library used in backend. Vite handles frontend environment variables (prefixed with VITE\_).
* **(Future) Staging/Production Deployment:**
  + Backend: Containerized Node.js application deployed to a cloud platform (e.g., AWS Elastic Beanstalk, Google Cloud Run, Heroku, DigitalOcean App Platform) or a self-managed server. PostgreSQL managed service (e.g., AWS RDS, Google Cloud SQL) recommended.
  + Frontend: Built into static assets (npm run build in frontend) and deployed to a static hosting service (e.g., Netlify, Vercel, AWS S3/CloudFront, GitHub Pages).
  + CI/CD: Implementation of a CI/CD pipeline (e.g., GitHub Actions, GitLab CI) for automated testing, building, and deployment.

**10. Future Considerations & Roadmap**

This section outlines potential future enhancements and a strategic roadmap beyond the core application, based on initial discussions and user needs. These items are categorized for clarity and can be prioritized based on user feedback and business goals.

**10.1. User Experience & Personalization Enhancements:**

* **Themes & Appearance Customization:** Introduction of user-selectable themes (e.g., light/dark modes) and potentially customizable accent colors to personalize the application's look and feel.
* **User Profile Enhancements:** Allowing users to add avatars and more detailed (optional) personal information to their profiles, fostering a more personalized experience.

**10.2. Advanced Financial Planning & Analysis Tools:**

* **"What-If" Budget Scenarios:** A feature enabling users to create draft or hypothetical budget profiles to simulate the financial impact of significant life events (e.g., new job, change in family size, major purchase) or to compare different budgeting strategies without affecting their active budget.
* **Debt Paydown Planner:** Tools to help users strategize debt repayment, potentially incorporating methods like the "snowball" or "avalanche" techniques, with calculators and progress tracking.

**10.3. Automation & Integrations:**

* **Enhanced Bill & Recurring Payment Reminders:** A robust notification system (in-app, email, or push notifications for mobile web) alerting users to upcoming recurring expenses or installment payments that are due and haven't been marked as isProcessed. This could include a monthly summary of unprocessed planned items.
* **(Very Long-Term/Complex) Bank Account Aggregation:** Exploration of secure integration with third-party services (e.g., Plaid, Yodlee, or Open Banking APIs, subject to regional availability and strict security/compliance) to allow users to automatically import transactions from their bank accounts and credit cards. This would significantly reduce manual data entry.

**10.4. Reporting & Visualization Improvements:**

* **Advanced Customizable Reports:** A report builder allowing users to generate custom reports by selecting specific data fields, applying multiple filters (date ranges, categories, tags, etc.), choosing chart types, and saving report templates.
* **Long-Term Trend Analysis:** Visualizations and reports showing income, expense, and savings trends over extended periods (e.g., multiple years, year-over-year comparisons).
* **Detailed Budget vs. Actual Reports:** Drill-down capabilities in reports that compare allocated budget amounts against actual spending for all categories and subcategories over selected periods, highlighting variances.
* **Printable Reports:** Option to generate print-friendly versions of key reports and summaries.

**10.5. Platform Expansion & Accessibility:**

* **Optimized Mobile Web Experience:** Continued focus on ensuring the responsive web application provides a seamless and feature-rich experience on mobile browsers, as a primary alternative to native apps initially.
* **(Long-Term) Native Mobile Applications (iOS & Android):** Development of dedicated native mobile applications to offer an enhanced user experience, offline capabilities, and platform-specific features like push notifications and widget support.

**10.6. AI-Powered Features (Exploratory):**

* **Smart Categorization Suggestions:** AI/ML models to suggest categories for new transactions based on user history, transaction descriptions, or common patterns.
* **Spending Anomaly Detection:** AI to identify unusual spending patterns or significant deviations from the norm that might warrant user attention.
* **Personalized Financial Insights & Advice:** AI-driven suggestions for budget optimization, savings opportunities, or financial planning based on the user's data and goals (requires careful consideration of ethics and accuracy).

**11. Appendix A: Initial Category & Subcategory List (Hebrew & English)**

This appendix details the predefined global categories and subcategories available to all users upon system setup. The Category and Subcategory entities will store both Hebrew (name\_he) and English (name\_en) names. The frontend will display the appropriate name based on the user's selected language preference.

**11.1. Expense Categories & Subcategories**

* **Main Category:** Food and Pharma (name\_he: "מזון ופארמה", name\_en: "Food & Pharma")
  + Subcategory: Food and Pharma - General (name\_he: "מזון ופארמה - כללי", name\_en: "General Food & Pharma")
  + Subcategory: Water Bar (name\_he: "בר מים", name\_en: "Water Bar/Cooler")
  + Subcategory: Ready-made food / At work (name\_he: "אוכל מוכן / בעבודה", name\_en: "Takeout/Work Meals")
  + Subcategory: Pharma and Toiletries (name\_he: "פארמה וטואלטיקה", name\_en: "Pharmacy & Toiletries")
  + Subcategory: Groceries (name\_he: "מזון", name\_en: "Groceries")
  + Subcategory: Smoking (name\_he: "עישון", name\_en: "Smoking")
* **Main Category:** Leisure, Entertainment, and Hobbies (name\_he: "פנאי, בילוי ותחביבים", name\_en: "Leisure & Hobbies")
  + Subcategory: Babysitter (name\_he: "בייביסיטר", name\_en: "Babysitter")
  + Subcategory: Leisure - General (name\_he: "פנאי - כללי", name\_en: "General Leisure")
  + Subcategory: Adult Classes/Workshops (name\_he: "חוגי מבוגרים", name\_en: "Adult Classes")
  + Subcategory: Restaurants and Eating Out (name\_he: "מסעדה ואוכל בחוץ", name\_en: "Dining Out")
  + Subcategory: Pets (name\_he: "חיות מחמד", name\_en: "Pets")
  + Subcategory: Lotteries/Gambling (name\_he: "הגרלות", name\_en: "Lotteries/Gambling")
  + Subcategory: Sports & Fitness (name\_he: "ספורט", name\_en: "Sports & Fitness")
  + Subcategory: Vacations & Travel (name\_he: "חופשות", name\_en: "Vacations & Travel")
  + Subcategory: Entertainment and Shows (name\_he: "בילויים ומופעים", name\_en: "Events & Shows")
* **Main Category:** Clothing and Footwear (name\_he: "ביגוד והנעלה", name\_en: "Apparel & Footwear")
  + Subcategory: Parents' Clothing (name\_he: "ביגוד הורים", name\_en: "Adults' Apparel")
  + Subcategory: Children's Clothing (name\_he: "ביגוד ילדים", name\_en: "Children's Apparel")
  + Subcategory: Shoes (name\_he: "נעליים", name\_en: "Shoes")
  + Subcategory: Clothing and Footwear - General (name\_he: "ביגוד והנעלה - כללי", name\_en: "General Apparel & Footwear")
* **Main Category:** Household Goods (name\_he: "תכולת בית", name\_en: "Household Goods")
  + Subcategory: Furniture (name\_he: "ריהוט", name\_en: "Furniture")
  + Subcategory: Appliances & Electronics (name\_he: "מוצרי חשמל ואלקטרוניקה", name\_en: "Appliances & Electronics")
  + Subcategory: Games, Toys, and Books (name\_he: "משחקים, צעצועים וספרים", name\_en: "Games, Toys & Books")
  + Subcategory: Housewares & Decor (name\_he: "כלי בית", name\_en: "Housewares & Decor")
  + Subcategory: Household Goods - General (name\_he: "תכולת בית - כללי", name\_en: "General Household Goods")
* **Main Category:** Home Maintenance (name\_he: "אחזקת בית", name\_en: "Home Maintenance")
  + Subcategory: Garden & Outdoors (name\_he: "גינה", name\_en: "Garden & Outdoors")
  + Subcategory: Water & Sewage (name\_he: "מים וביוב", name\_en: "Water & Sewage")
  + Subcategory: Electricity (name\_he: "חשמל", name\_en: "Electricity")
  + Subcategory: Gas (name\_he: "גז", name\_en: "Gas")
  + Subcategory: Home Maintenance - General (name\_he: "אחזקת בית - כללי", name\_en: "General Home Maintenance")
  + Subcategory: Repairs & Renovations (name\_he: "תיקונים בבית / במכשירים", name\_en: "Repairs & Renovations")
  + Subcategory: Cleaning Supplies & Services (name\_he: "ניקיון", name\_en: "Cleaning")
* **Main Category:** Personal Care (name\_he: "טיפוח", name\_en: "Personal Care")
  + Subcategory: Cosmetics & Skincare (name\_he: "קוסמטיקה", name\_en: "Cosmetics & Skincare")
  + Subcategory: Personal Care - General (name\_he: "טיפוח - כללי", name\_en: "General Personal Care")
  + Subcategory: Hairdresser/Salon (name\_he: "מספרה", name\_en: "Hairdresser/Salon")
* **Main Category:** Education (name\_he: "חינוך", name\_en: "Education")
  + Subcategory: Childcare/Daycare (name\_he: "מסגרות יום", name\_en: "Childcare/Daycare")
  + Subcategory: After-School Programs (name\_he: "מסגרות צהריים", name\_en: "After-School Programs")
  + Subcategory: School Transportation (name\_he: "הסעות", name\_en: "School Transportation")
  + Subcategory: School Fees & Supplies (name\_he: "בית ספר", name\_en: "School Fees & Supplies")
  + Subcategory: Summer Camps/Programs (name\_he: "מסגרות קיץ", name\_en: "Summer Camps/Programs")
  + Subcategory: Nanny/Au Pair (name\_he: "צהרון / מטפלת", name\_en: "Nanny/Au Pair")
  + Subcategory: Private Tutoring (name\_he: "שיעור פרטי", name\_en: "Private Tutoring")
  + Subcategory: Adult Education & Training (name\_he: "לימודים והשתלמות לבוגרים", name\_en: "Adult Education")
  + Subcategory: Education - General (name\_he: "חינוך - כללי", name\_en: "General Education")
  + Subcategory: Extracurricular & Youth Groups (name\_he: "חוגים ותנועת נוער", name\_en: "Extracurricular Activities")
* **Main Category:** Gifts, Donations, Religious (name\_he: "אירועים, תרומות, צרכי דת", name\_en: "Gifts & Donations")
  + Subcategory: Gifts Given (name\_he: "אירוע בעבודה / לחברים", name\_en: "Gifts Given")
  + Subcategory: Donations & Charity (name\_he: "תרומות", name\_en: "Donations & Charity")
  + Subcategory: Religious/Community Events (name\_he: "חגים וצורכי דת", name\_en: "Religious/Community Events")
* **Main Category:** Health & Medical (name\_he: "בריאות", name\_en: "Health & Medical")
  + Subcategory: Health Insurance Premiums (HMO/Kupat Holim) (name\_he: "תשלום לקופ"ח", name\_en: "HMO/Insurance Premiums")
  + Subcategory: Doctor Visits & Specialist Fees (name\_he: "טיפולים פרטיים", name\_en: "Doctor & Specialist Fees")
  + Subcategory: Supplementary Health Insurance (name\_he: "ביטוח רפואי נוסף", name\_en: "Supplementary Insurance")
  + Subcategory: Medications & Prescriptions (name\_he: "תרופות", name\_en: "Medications")
  + Subcategory: Health - General (name\_he: "בריאות - כללי", name\_en: "General Health")
  + Subcategory: Dental & Orthodontics (name\_he: "טיפולי שיניים / אורתודנט", name\_en: "Dental & Orthodontics")
  + Subcategory: Vision Care/Optics (name\_he: "אופטיקה", name\_en: "Vision Care")
* **Main Category:** Transportation (name\_he: "תחבורה", name\_en: "Transportation")
  + Subcategory: Car Lease/Loan Payments (name\_he: "ליסינג", name\_en: "Car Lease/Loan")
  + Subcategory: Ride-Sharing & Taxis (name\_he: "תחבורה שיתופית", name\_en: "Ride-Sharing/Taxis")
  + Subcategory: Vehicle Registration & Licensing (name\_he: "רישוי רכב", name\_en: "Vehicle Licensing")
  + Subcategory: Parking Fees (name\_he: "חניה", name\_en: "Parking")
  + Subcategory: Transportation - General (name\_he: "תחבורה - כללי", name\_en: "General Transportation")
  + Subcategory: Toll Roads (name\_he: "כבישי אגרה", name\_en: "Toll Roads")
  + Subcategory: Car Insurance (name\_he: "ביטוח רכב", name\_en: "Car Insurance")
  + Subcategory: Vehicle Maintenance & Repairs (name\_he: "תחזוקת רכב", name\_en: "Vehicle Maintenance")
  + Subcategory: Public Transportation (name\_he: "תחבורה ציבורית", name\_en: "Public Transport")
  + Subcategory: Fuel/Gas (name\_he: "דלק", name\_en: "Fuel/Gas")
* **Main Category:** Family Obligations (name\_he: "משפחה", name\_en: "Family Obligations")
  + Subcategory: Alimony/Child Support (name\_he: "תשלום מזונות", name\_en: "Alimony/Child Support")
  + Subcategory: Financial Help to Family (name\_he: "עזרה למשפחה", name\_en: "Financial Help to Family")
  + Subcategory: Family - General (name\_he: "משפחה - כללי", name\_en: "General Family")
  + Subcategory: Allowances/Pocket Money (name\_he: "דמי כיס", name\_en: "Allowances")
  + Subcategory: Family Events & Celebrations (name\_he: "אירועי שמחות במשפחה", name\_en: "Family Events")
* **Main Category:** Communication (name\_he: "תקשורת", name\_en: "Communication")
  + Subcategory: Streaming & Content Services (name\_he: "שירותי תוכן", name\_en: "Streaming/Content Services")
  + Subcategory: TV & Internet (Provider and Infrastructure) (name\_he: "טלוויזיה ואינטרנט (ספק ותשתית)", name\_en: "TV & Internet")
  + Subcategory: Communication - General (name\_he: "תקשורת - כללי", name\_en: "General Communication")
  + Subcategory: Phone (Mobile & Landline) (name\_he: "טלפון נייד ונייח", name\_en: "Phone (Mobile & Landline)")
* **Main Category:** Housing (name\_he: "דיור", name\_en: "Housing")
  + Subcategory: Housing - General (name\_he: "דיור - כללי", name\_en: "General Housing")
  + Subcategory: Property & Contents Insurance (name\_he: "ביטוח נכס ותכולה", name\_en: "Property Insurance")
  + Subcategory: Mortgage (name\_he: "משכנתה", name\_en: "Mortgage")
  + Subcategory: Rent (name\_he: "שכר דירה", name\_en: "Rent")
  + Subcategory: Municipal Taxes / Homeowners Association (HOA) (name\_he: "מיסי יישוב / ועד בית", name\_en: "Municipal Taxes/HOA")
  + Subcategory: Arnona (City Property Tax) (name\_he: "ארנונה", name\_en: "Arnona (City Tax)")
* **Main Category:** Debts & Liabilities (name\_he: "התחייבויות", name\_en: "Debts & Liabilities")
  + Subcategory: Debt Repayment (non-mortgage) (name\_he: "החזר חובות חודשי (למעט משכנתה) - כללי", name\_en: "Debt Repayment (non-mortgage)")
  + Subcategory: Overdraft Fees & Interest (name\_he: "ריביות משיכת יתר", name\_en: "Overdraft Fees")
* **Main Category:** Savings & Investments (name\_he: "חסכונות והשקעות", name\_en: "Savings & Investments")
  + Subcategory: Deposits to Savings Accounts (name\_he: "הפקדות לחסכונות - כללי", name\_en: "Savings Deposits")
  + Subcategory: Investment Contributions (name\_he: "הפקדות להשקעות", name\_en: "Investment Contributions")
* **Main Category:** Financial Services (name\_he: "פיננסים", name\_en: "Financial Services")
  + Subcategory: Bank Fees & Charges (name\_he: "עמלות", name\_en: "Bank Fees")
  + Subcategory: Life Insurance (name\_he: "ביטוח חיים", name\_en: "Life Insurance")
  + Subcategory: Financial Services - General (name\_he: "פיננסים - כללי", name\_en: "General Financial Services")
  + Subcategory: National Insurance (for non-workers/self-employed) (name\_he: "ביטוח לאומי (למי שלא עובד)", name\_en: "National Insurance (Self-Paid)")

**11.2. Income Categories**

* **Main Category:** Salary/Wages (name\_he: "שכר עבודה", name\_en: "Salary/Wages")
  + (Subcategories optional, e.g., "Salary - Primary," "Salary - Secondary" if multiple earners/jobs)
* **Main Category:** Allowances & Pensions (name\_he: "קצבאות וגמלאות", name\_en: "Allowances & Pensions")
  + (Subcategories can be created by user if needed, or use main category. Examples: "Child Allowance," "Disability Allowance," "Unemployment," "Pension")
* **Main Category:** Investment & Asset Income (name\_he: "הכנסות מנכסים והשקעות", name\_en: "Investment & Asset Income")
  + (Examples: "Rental Income," "Dividends," "Capital Gains")
* **Main Category:** Self-Employment / Freelance (name\_he: "עסק עצמאי / פרילנס", name\_en: "Self-Employment/Freelance")
* **Main Category:** Financial Assistance & Support (name\_he: "סיוע ותמיכות", name\_en: "Financial Assistance")
  + (Examples: "Help from Family," "Alimony Received")
* **Main Category:** Miscellaneous Income (name\_he: "הכנסות שונות", name\_en: "Miscellaneous Income")
  + (Examples: "Gifts (Monetary)," "Refunds," "Sale of Items")