# Financial Playbook Application: Complete Documentation

This document provides a comprehensive overview of the Financial Playbook application, including wireframes, technical specifications, and integration details.

---

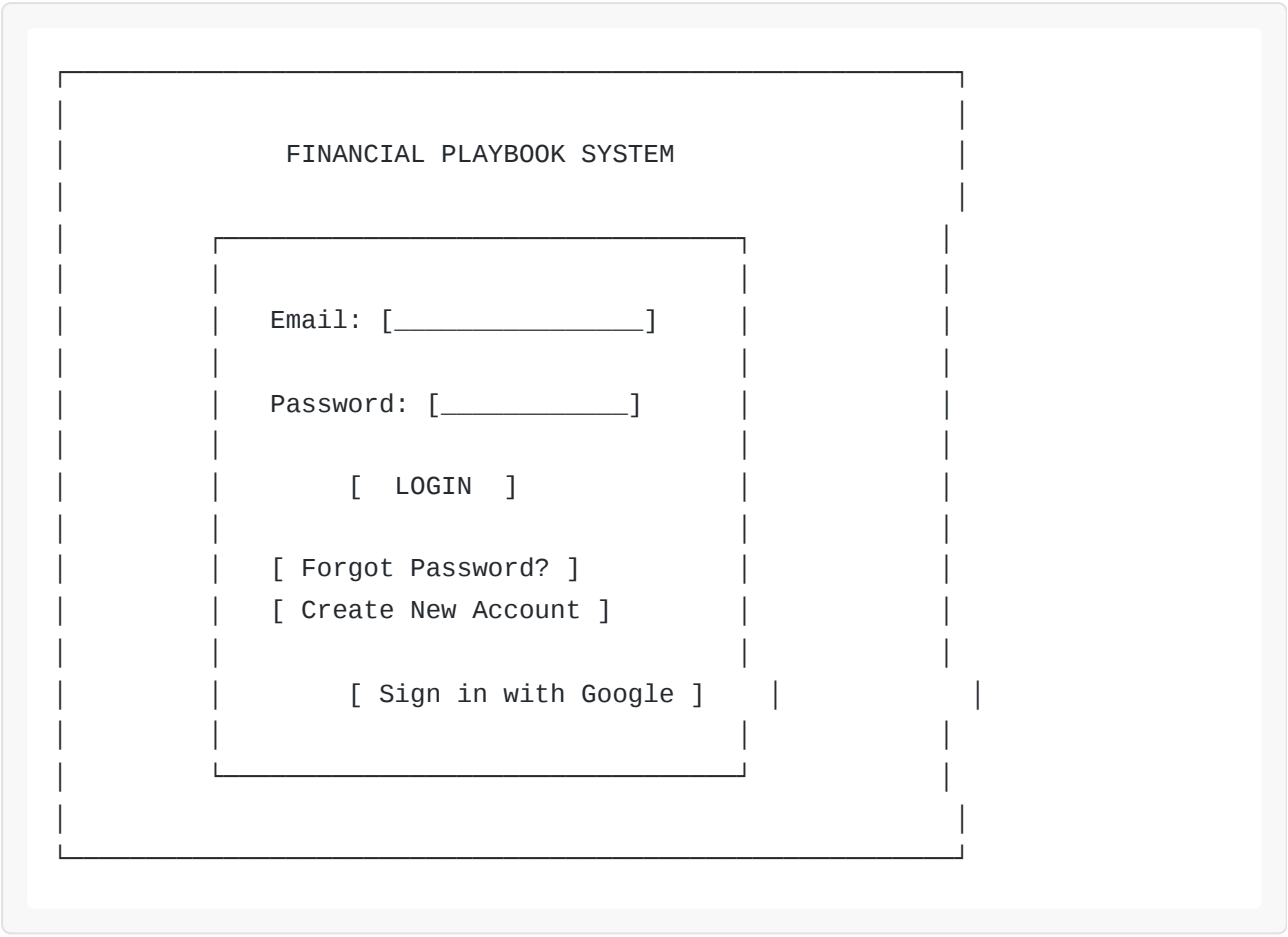
## Wireframes & Component Overview

## Application Overview

**AI-Driven Financial Playbook Website** - A user-authenticated web application that provides comprehensive financial scenario planning, cash flow monitoring, and automated risk analysis with real-time notifications.

---

# Wireframe 1: Login / Authentication Page

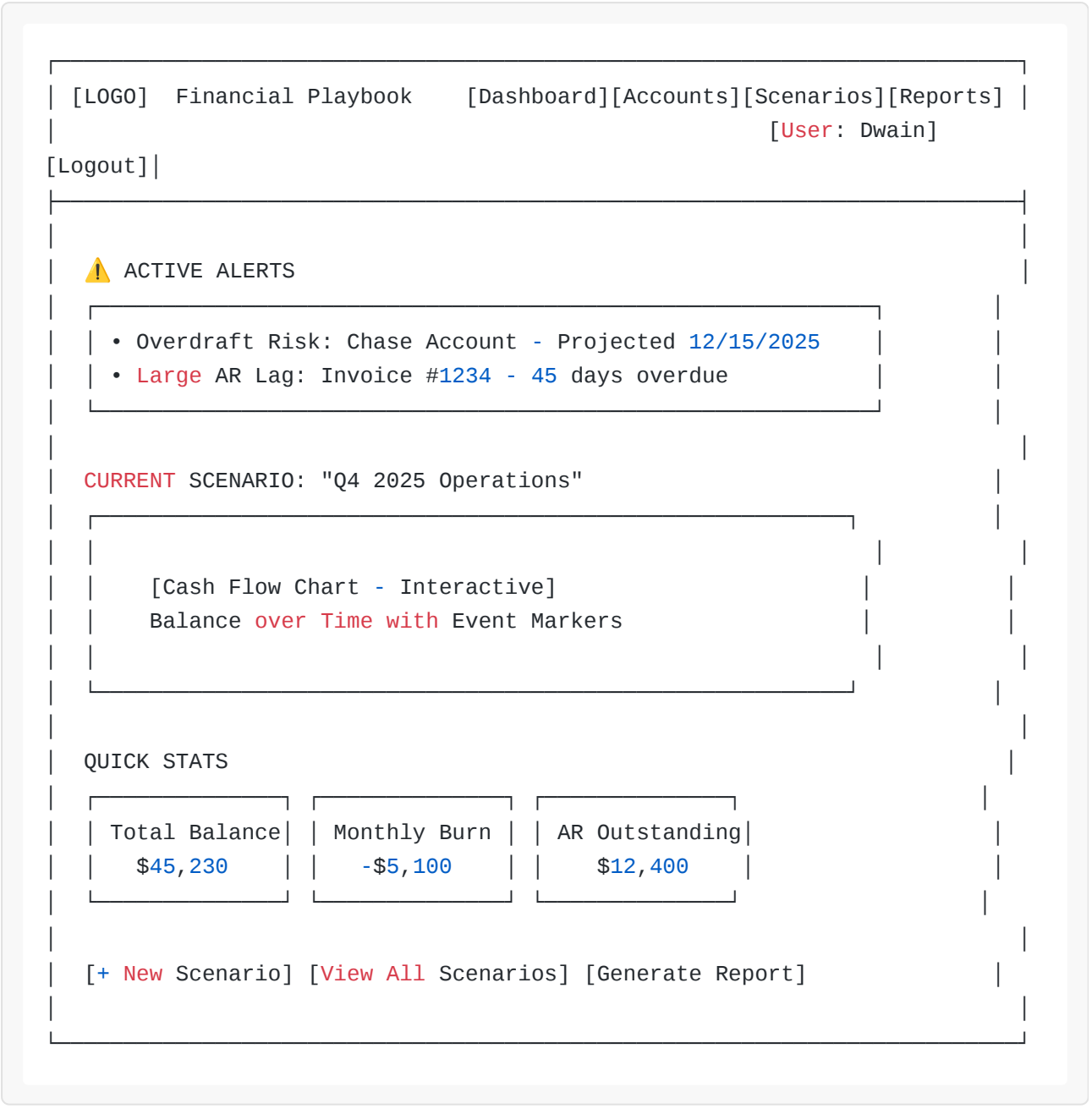



**Components:**

- User authentication form
- Session management
- Password recovery
- New user registration
- Google Authentication (OAuth 2.0)

---

# Wireframe 2: Dashboard / Home Page




**Components:**

- Alert notification system (user-specific)
- Active scenario selector
- Interactive cash flow visualization
- Quick stats dashboard
- Navigation menu

---

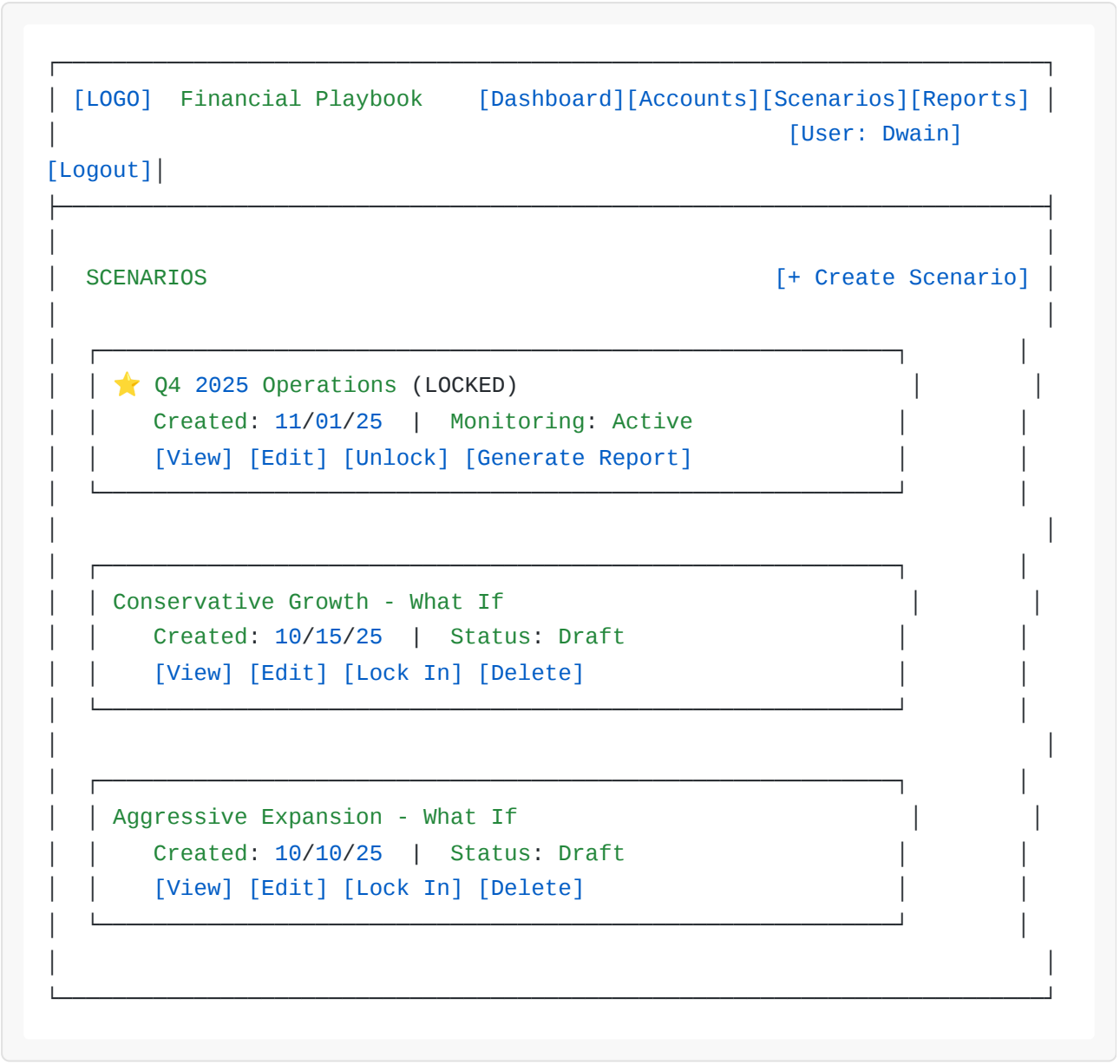
# Wireframe 3: Accounts Management Page

**Components:**

- Bank account connections (API integration)
- Credit card tracking
- QuickBooks Online integration
- Bills and recurring expenses management
- Accounts receivable tracking
- Manual data entry forms
- Data import/upload functionality

---

# Wireframe 4: Scenario Planning Page

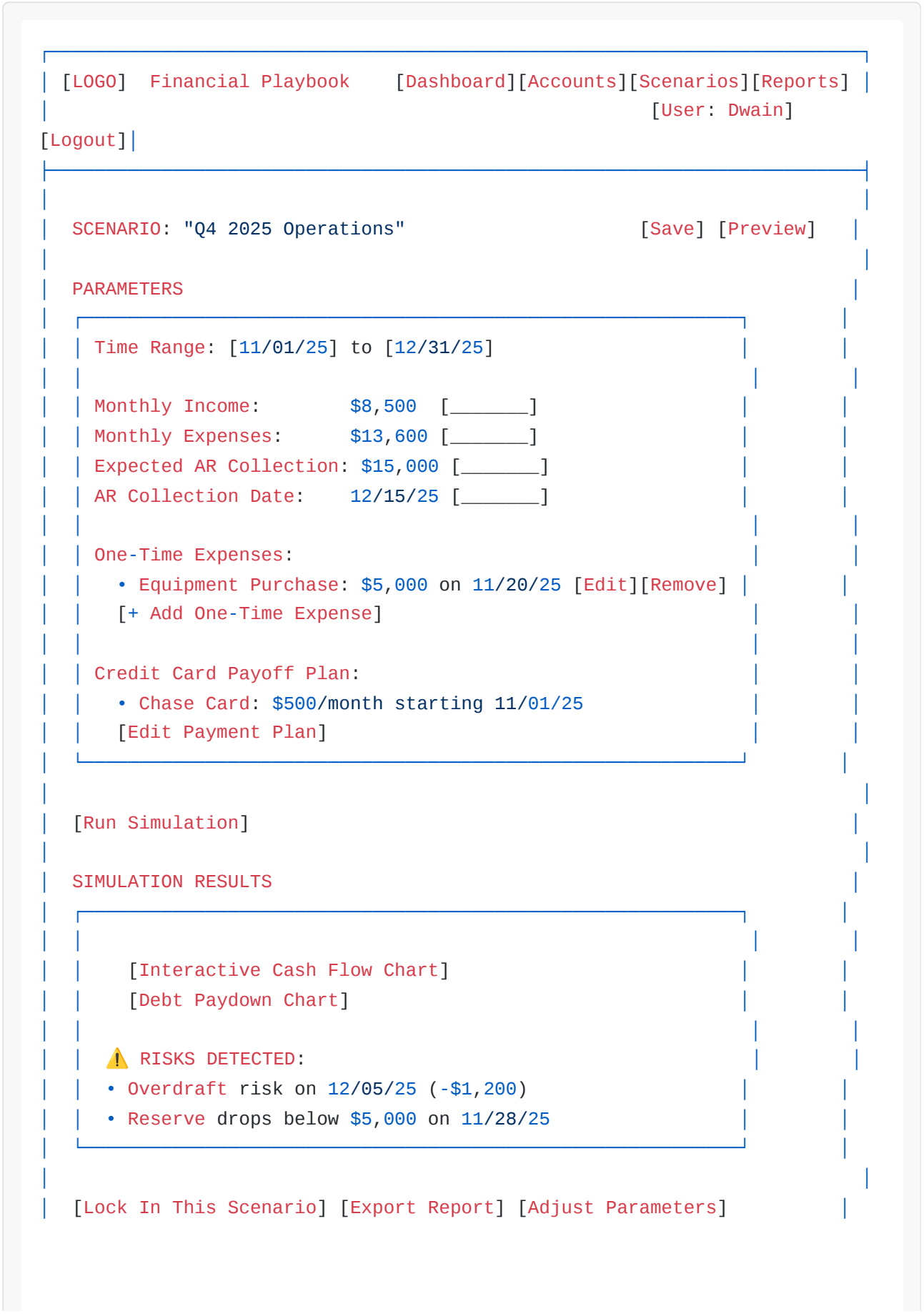



**Components:**

- Scenario list (user-specific)
- Scenario status (Draft, Locked, Active)
- Scenario creation wizard
- “What-if” modeling engine
- Lock-in functionality for active monitoring

---

# Wireframe 5: Scenario Editor / What-If Modeling

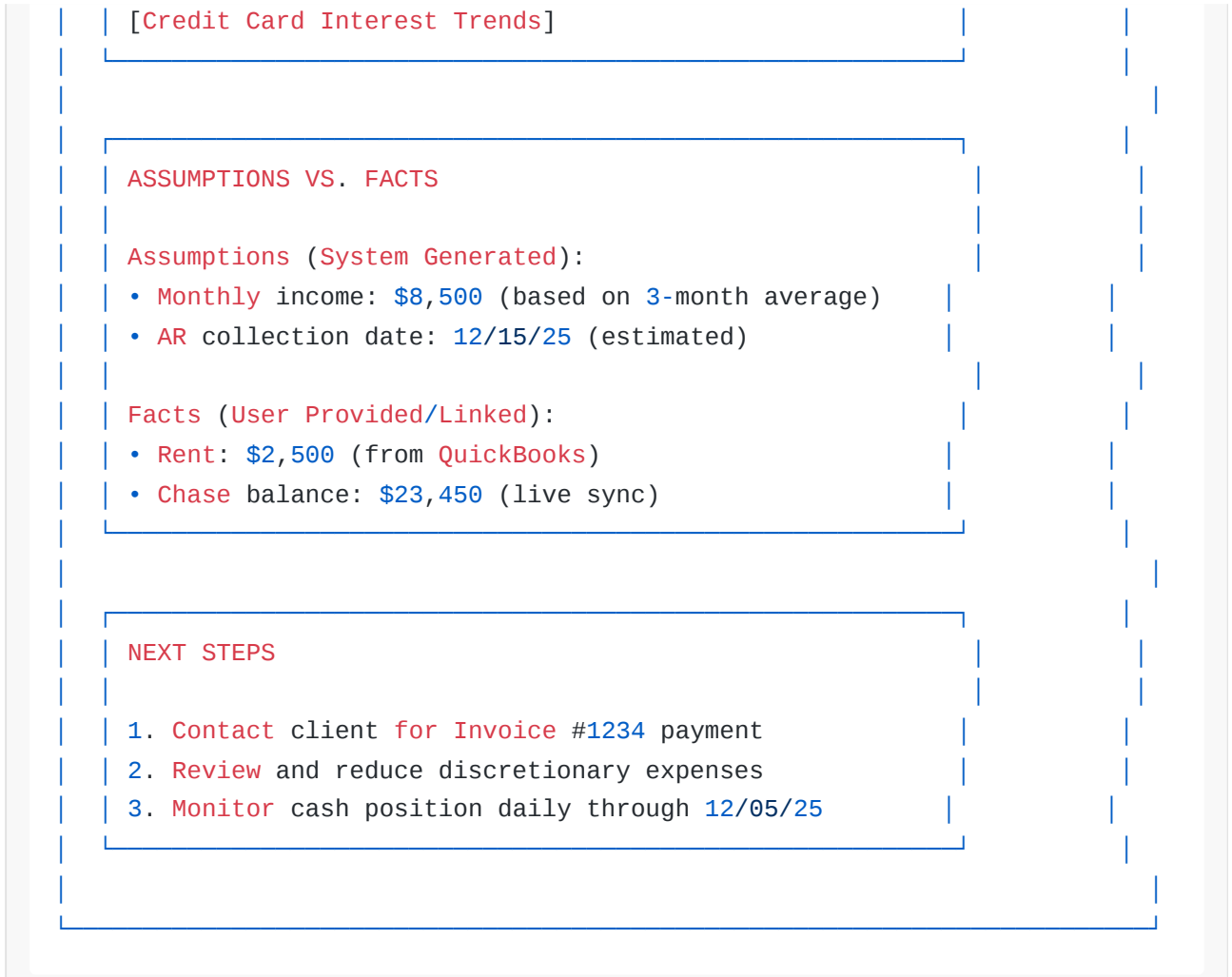


**Components:**

- Parameter input forms
- Variable adjustment sliders/inputs
- Simulation engine
- Real-time chart updates
- Risk detection and display
- Scenario comparison tools

---

# Wireframe 6: Playbook Report View

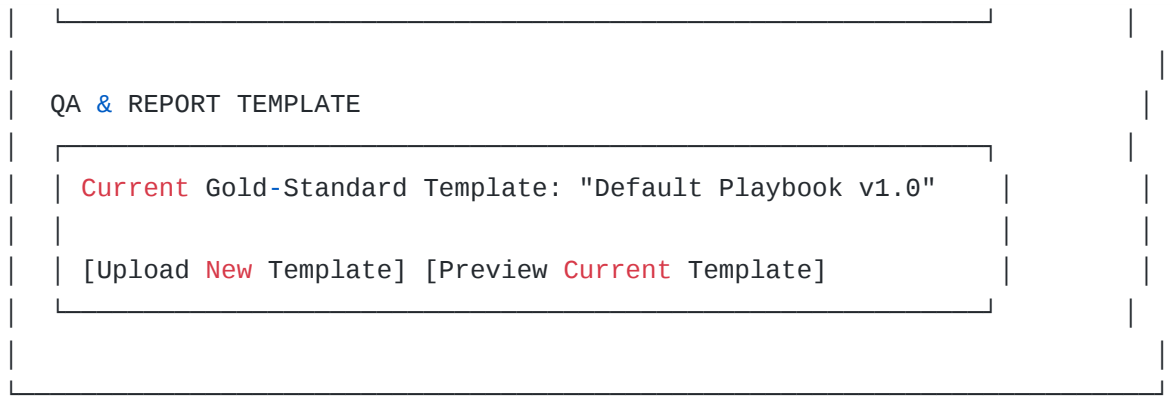



**Components:**

- Executive summary generator
- Register tables (checkbook style) for each account
- Chart rendering (PNG/SVG export)
- Assumptions vs. Facts section
- Recommendations engine
- Multi-format export (PDF, Word, Markdown, CSV)
- Email delivery system

---

# Wireframe 7: Settings & Notifications




**Components:**

- User profile management
- Email notification settings
- Multi-recipient notification system
- Alert threshold configuration
- Template management for QA enforcement
- Timezone settings

---

# System Architecture Components Summary

## 1. Authentication & User Management

- User registration and login
- Session management
- Password recovery
- Multi-user support with data isolation

## 2. Data Integration Layer

- Bank account API connections
- QuickBooks Online integration
- Manual data entry forms

- CSV/Excel import functionality
- Data validation and normalization

### 3. Financial Data Management

- Account tracking (bank accounts, credit cards)
- Bills and recurring expenses
- Accounts receivable tracking
- Transaction history
- Real-time balance updates

### 4. Scenario Engine

- What-if modeling
- Parameter adjustment
- Simulation calculations
- Multi-scenario comparison
- Scenario locking and monitoring

### 5. Analysis & Risk Detection

- Automated financial analysis
- Risk identification (overdrafts, reserve breaches, AR lag)
- Trend analysis
- Recommendations engine
- Assumptions vs. Facts tracking

### 6. Visualization & Reporting

- Interactive charts (cash flow, debt paydown, interest trends)
- Register tables (checkbook style)
- Event markers on charts
- Multi-format export (PDF, Word, Markdown, CSV)

- Chart export (PNG, SVG)

## 7. Quality Assurance System

- Template comparison engine
- Report validation before export
- Preview mode
- Deviation detection and flagging

## 8. Notification System

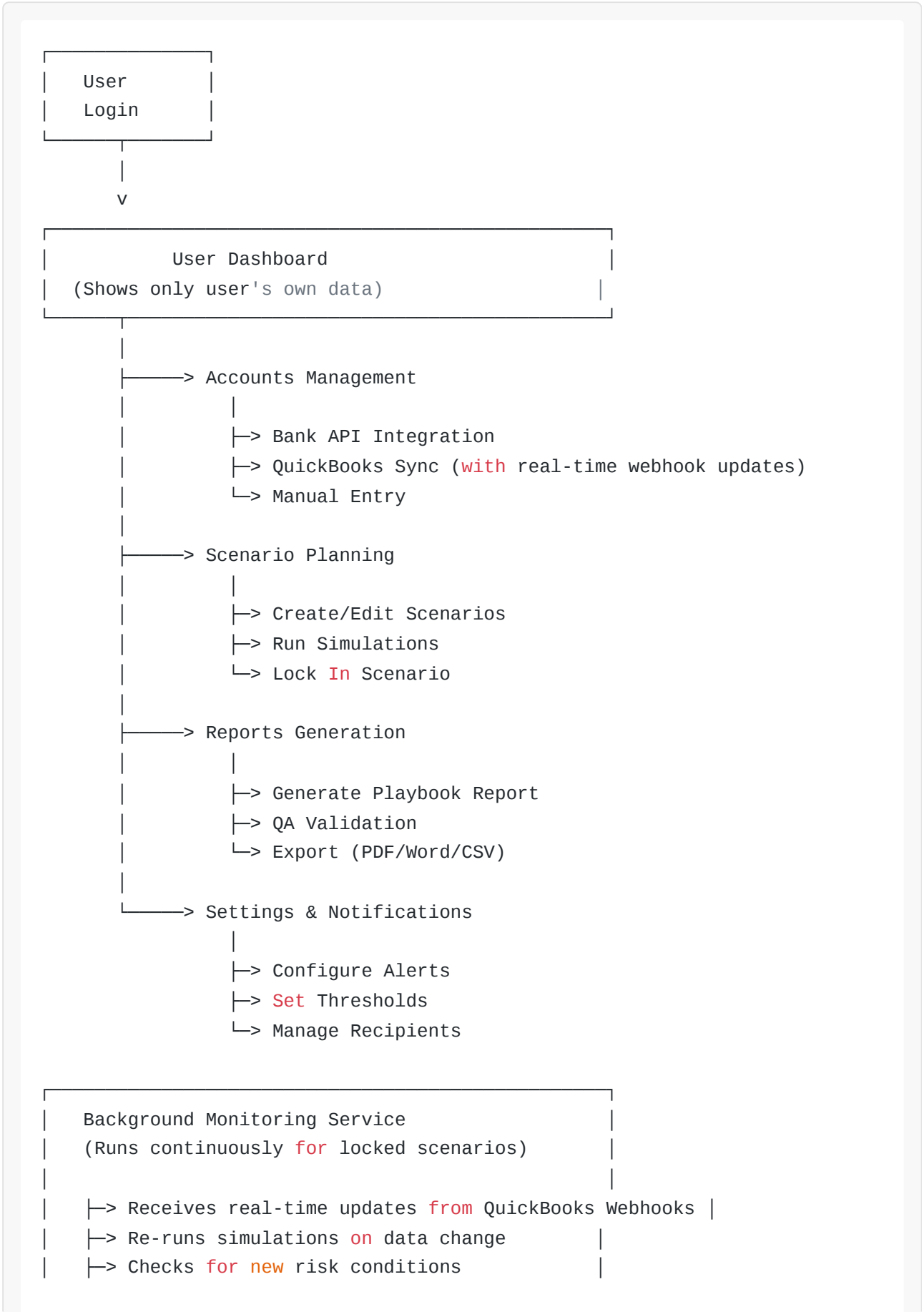
- Real-time risk monitoring
- Email alerts (critical risks)
- Multi-recipient support
- Configurable alert thresholds
- Weekly summary reports

## 9. Export & Collaboration

- PDF generation
- Word document export
- Markdown export
- CSV export for tables
- Email delivery
- Future: Audit logs, collaboration features

---

# Data Flow Diagram



```
|   ├─> Compares against thresholds              |
|   └─> Sends email alerts when triggered        |
└──────────────────────────────────────────────────┘
```

---

# Technology Stack Considerations

## Frontend

- Modern responsive UI framework
- Interactive charting library
- Real-time data updates
- Mobile-responsive design

## Backend

- User authentication and authorization
- RESTful API for data operations
- Background job processing for monitoring
- Database with user data isolation

## Database

- User accounts and profiles
- Financial accounts and transactions
- Scenarios and simulations
- Bills, expenses, and AR data
- Notification logs

## External Integrations

- Bank account APIs (Plaid, Yodlee, or similar)
- QuickBooks Online API

- Email service (SendGrid, AWS SES, or similar)
- Future: SMS gateway for text alerts

### Security

- Encrypted data storage
- Secure API connections
- User data isolation
- Session management
- HTTPS/SSL

---

## Next Steps for Discussion

I've created comprehensive wireframes for your Financial Playbook application. Now I need to ask you some key questions to refine the design before we build:

### Critical Questions:

1. **User Roles & Access:**
    - Will you be the only user initially, or do you plan to have multiple users (clients, team members)?
    - Do you need different permission levels (admin, viewer, editor)?
2. **Bank Integration Priority:**
    - Which bank account integration is most important? (Plaid is common for US banks)
    - Do you already have QuickBooks Online, or is this a future integration?
3. **Data Entry Workflow:**
    - For the initial version, would you prefer to start with manual data entry, or is live bank integration essential from day one?
    - How often do you want to sync data (real-time, daily, manual)?

4. **Notification Details:**
    - What email address(es) should receive critical alerts?
    - What time of day do you prefer to receive weekly summary reports?
5. **Report Customization:**
    - Do you have an existing "gold-standard" playbook template you want to match, or should we create a default template?
    - What specific charts are most important to you? (Cash flow, debt paydown, expense breakdown, etc.)
6. **Deployment Preferences:**
    - Where do you want this hosted? (Your own server, cloud service like AWS/Azure, or a simple hosting platform?)
    - Do you need this accessible from mobile devices?
7. **Timeline & MVP:**
    - What features are absolutely essential for version 1.0?
    - What features can be added later as enhancements?
8. **Business Context:**
    - Is this primarily for Superior Networks LLC's internal use, or do you plan to offer it as a service to clients?
    - What's your typical monthly transaction volume?

Please review the wireframes and let me know your thoughts on these questions so we can tailor the application to your exact needs!

## 7. Quality Assurance System

- Template comparison engine
- Report validation before export
- Preview mode
- Deviation detection and flagging

## 8. Notification System

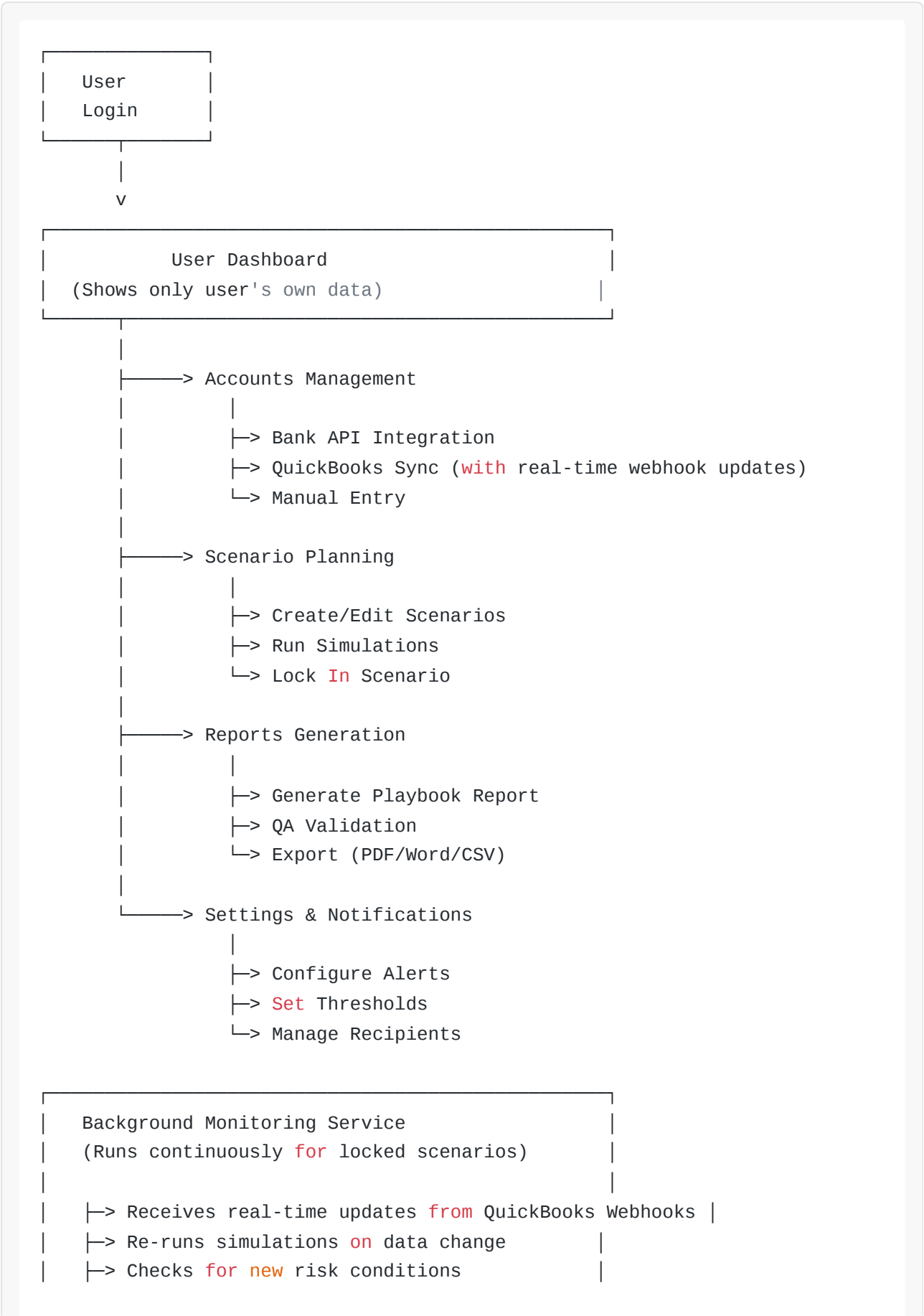
- Real-time risk monitoring
- Email alerts (critical risks)
- Multi-recipient support
- Configurable alert thresholds
- Weekly summary reports

## 9. Export & Collaboration

- PDF generation
- Word document export
- Markdown export
- CSV export for tables
- Email delivery
- Future: Audit logs, collaboration features

---

# Data Flow Diagram



```
|   ├─> Compares against thresholds              |
|   └─> Sends email alerts when triggered        |
└──────────────────────────────────────────────────┘
```

---

# Technology Stack Considerations

## Frontend

- Modern responsive UI framework
- Interactive charting library
- Real-time data updates
- Mobile-responsive design

## Backend

- User authentication and authorization
- RESTful API for data operations
- Background job processing for monitoring
- Database with user data isolation

## Database

- User accounts and profiles
- Financial accounts and transactions
- Scenarios and simulations
- Bills, expenses, and AR data
- Notification logs

## External Integrations

- Bank account APIs (Plaid, Yodlee, or similar)
- QuickBooks Online API

- Email service (SendGrid, AWS SES, or similar)
- Future: SMS gateway for text alerts

### Security

- Encrypted data storage
- Secure API connections
- User data isolation
- Session management
- HTTPS/SSL

---

## Next Steps for Discussion

I’ve created comprehensive wireframes for your Financial Playbook application. Now I need to ask you some key questions to refine the design before we build:

### Critical Questions:

1. **User Roles & Access:**
    - Will you be the only user initially, or do you plan to have multiple users (clients, team members)?
    - Do you need different permission levels (admin, viewer, editor)?
2. **Bank Integration Priority:**
    - Which bank account integration is most important? (Plaid is common for US banks)
    - Do you already have QuickBooks Online, or is this a future integration?
3. **Data Entry Workflow:**
    - For the initial version, would you prefer to start with manual data entry, or is live bank integration essential from day one?
    - How often do you want to sync data (real-time, daily, manual)?

4. **Notification Details:**
    - What email address(es) should receive critical alerts?
    - What time of day do you prefer to receive weekly summary reports?
5. **Report Customization:**
    - Do you have an existing “gold-standard” playbook template you want to match, or should we create a default template?
    - What specific charts are most important to you? (Cash flow, debt paydown, expense breakdown, etc.)
6. **Deployment Preferences:**
    - Where do you want this hosted? (Your own server, cloud service like AWS/Azure, or a simple hosting platform?)
    - Do you need this accessible from mobile devices?
7. **Timeline & MVP:**
    - What features are absolutely essential for version 1.0?
    - What features can be added later as enhancements?
8. **Business Context:**
    - Is this primarily for Superior Networks LLC’s internal use, or do you plan to offer it as a service to clients?
    - What’s your typical monthly transaction volume?

Please review the wireframes and let me know your thoughts on these questions so we can tailor the application to your exact needs!

---

# Integration Feasibility Report: Google Auth, QuickBooks Online & Real-Time Updates

This document outlines the technical feasibility, approach, and implications of integrating Google Authentication, QuickBooks Online, and real-time data synchronization into the Financial Playbook application.

# 1. Google Authentication

**Feasibility: High**. Integrating Google Authentication is a standard and well-documented process.

**Approach:** We will use the **OAuth 2.0** protocol, which is the industry standard for secure delegated access [1]. The process involves:

1. **Creating a Google API Console Project:** We will set up a new project in the Google API Console to obtain OAuth 2.0 client credentials (a client ID and client secret) for the application.
2. **Implementing a "Sign in with Google" Button:** A "Sign in with Google" button will be added to the login page. This will redirect users to Google's authentication server.
3. **Handling the OAuth 2.0 Flow:** After the user grants permission, Google will redirect back to our application with an authorization code. Our backend will exchange this code for an access token and a refresh token.
4. **User Profile Creation:** We will use the access token to retrieve the user's profile information (name, email, profile picture) from the Google People API. This information will be used to create a new user account in our application's database or link to an existing one.
5. **Session Management:** Upon successful authentication, a session will be created for the user, granting them access to their personalized dashboard.

**Benefits:**

- **Enhanced Security:** Leverages Google's robust security infrastructure.
- **Improved User Experience:** Simplifies the login process, as users won't need to create and remember a separate password for this application.
- **Faster Registration:** New users can sign up with a single click.

**Wireframe Update:** The login page wireframe will be updated to include a "Sign in with Google" button.

## 2. QuickBooks Online Integration

**Feasibility: High**. The QuickBooks Online API is well-documented and provides the necessary endpoints to access financial data [2].

**Approach:** We will use the **QuickBooks Online API** to establish a live connection to your QuickBooks account. This will also use the OAuth 2.0 protocol for authentication.

1. **Creating an Intuit Developer Account & App:** We will create an Intuit Developer account and register our application to get API keys and credentials.
2. **Implementing the QuickBooks Connection Flow:** In the "Accounts Management" section of the application, there will be an option to "Connect to QuickBooks." This will initiate an OAuth 2.0 flow similar to Google's, where you will be asked to authorize our application to access your QuickBooks data.
3. **Data Synchronization:** Once authorized, the application will be able to make API calls to your QuickBooks Online account to fetch data such as:
    - Chart of Accounts
    - Bank and Credit Card account balances and transactions
    - Invoices (Accounts Receivable)
    - Bills and Expenses (Accounts Payable)
    - Customers and Vendors
4. **Data Mapping:** The data retrieved from QuickBooks will be mapped to the corresponding fields in our application's database, ensuring consistency and accuracy.

**Benefits:**

- **Automation:** Eliminates the need for manual data entry, saving time and reducing errors.
- **Real-time Data:** Ensures that the financial playbook is always based on the most up-to-date information from your accounting system.
- **Comprehensive Financial Picture:** Provides a holistic view of your finances by combining data from bank accounts and your accounting software.

**Wireframe Update:** The "Accounts Management" wireframe will be updated to show the status of the QuickBooks Online connection.

## 3. Real-Time Playbook Updates

**Feasibility: High**. This is a core requirement and will be achieved through a combination of the QuickBooks integration and background processing.

**Approach:**

1. **Webhook Integration (for QuickBooks):** We will utilize QuickBooks Online Webhooks. This means that QuickBooks will proactively send a notification to our application whenever a specific event occurs (e.g., a new invoice is created, a payment is received, a bill is paid). This is more efficient than constantly polling the API for changes.
2. **Background Job Processing:** When our application receives a webhook notification, it will trigger a background job. This job will:
    - Fetch the updated data from QuickBooks.
    - Update the relevant information in our application's database.
    - Re-run the simulation for any "locked-in" playbooks that are affected by the change.
3. **Real-time Adjustments:** This ensures that your active playbook is always current. For example:
    - If a large, unexpected invoice is paid, the cash flow projection will be immediately updated, potentially resolving a previously forecasted overdraft.
    - If a new, significant bill is entered, the playbook will be re-evaluated, and if a new risk is identified (e.g., a projected cash shortfall), a notification will be sent.
4. **Manual Adjustments:** You will still have the ability to make manual adjustments to your playbooks. These manual overrides will be layered on top of the live data from QuickBooks, allowing you to create "what-if" scenarios based on the most current financial reality.

**Benefits:**

- **Proactive Financial Management:** The system doesn't just show you what happened; it shows you what's *about* to happen based on the latest data.

- **Timely Alerts:** You’ ll be notified of potential issues as soon as they arise, giving you more time to react.
- **Dynamic Scenarios:** Your financial forecasts will adapt to the changing conditions of your business in near real-time.

**Wireframe Update:** The data flow diagram will be updated to illustrate the webhook-based real-time updates from QuickBooks.

---

### References

[1] Google Identity. (2025, October 23). *Using OAuth 2.0 for Web Server Applications*. Google for Developers. https://developers.google.com/identity/protocols/oauth2/web-server

[2] Intuit Developer. (n.d.). *QuickBooks API*. Intuit. https://developer.intuit.com/app/developer/qbo/docs/develop

---

## Delegated Access Feature Specification

### Overview

The delegated access feature allows the primary user (Dwain) to grant limited, read-only access to specific financial data and reports to external parties, such as a tax professional or accountant. This ensures that sensitive financial information can be shared securely without compromising the user’ s account security.

---

### Use Case

**Primary User:** Dwain Henderson Jr. (Superior Networks LLC)

**Delegated User:** Tax Professional or Accountant

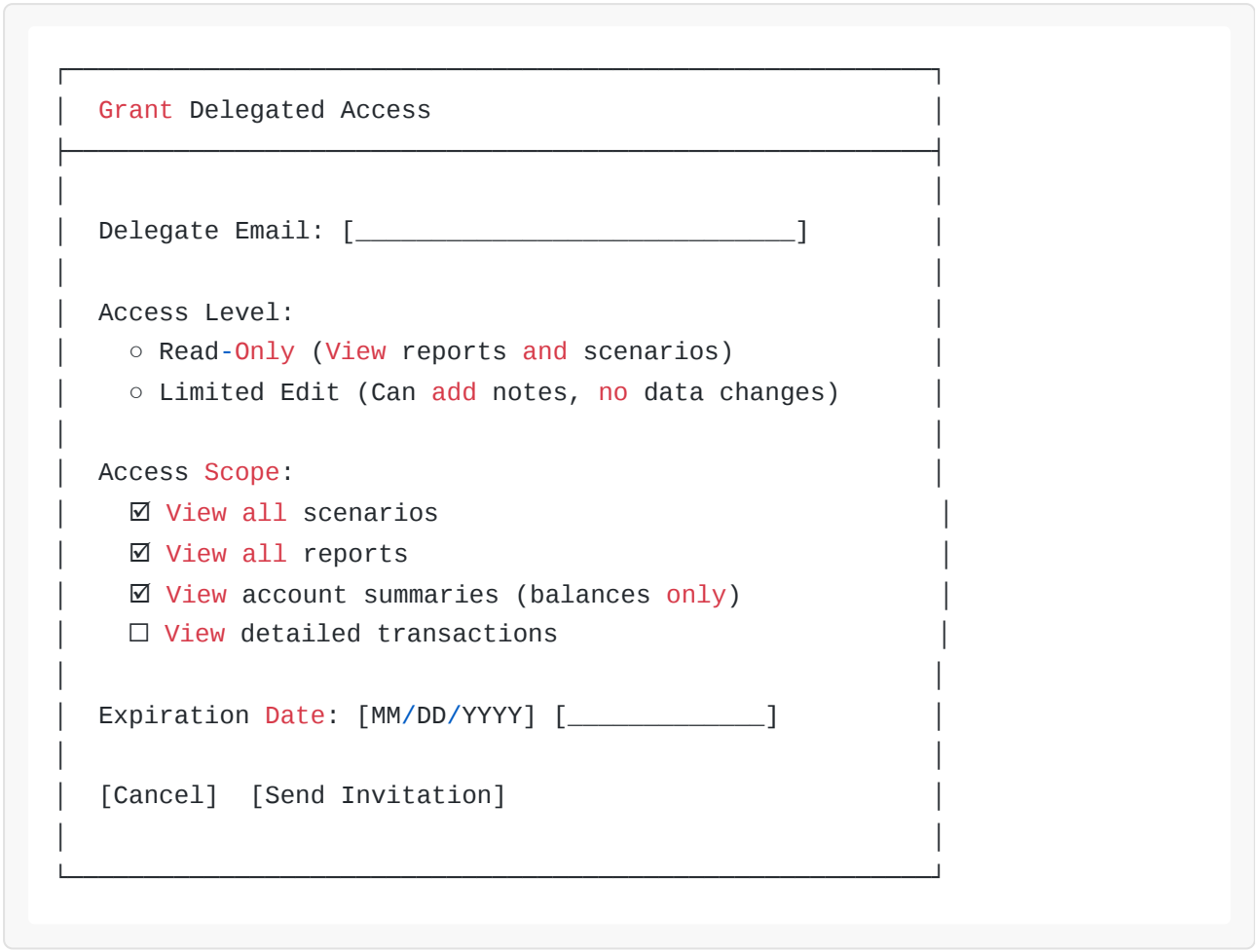
**Goal:** Allow the tax professional to view financial reports, scenarios, and account summaries without the ability to modify data or settings.

---

## Wireframe: Delegated Access Management (in Settings)


---

## Wireframe: Grant New Delegated Access Dialog



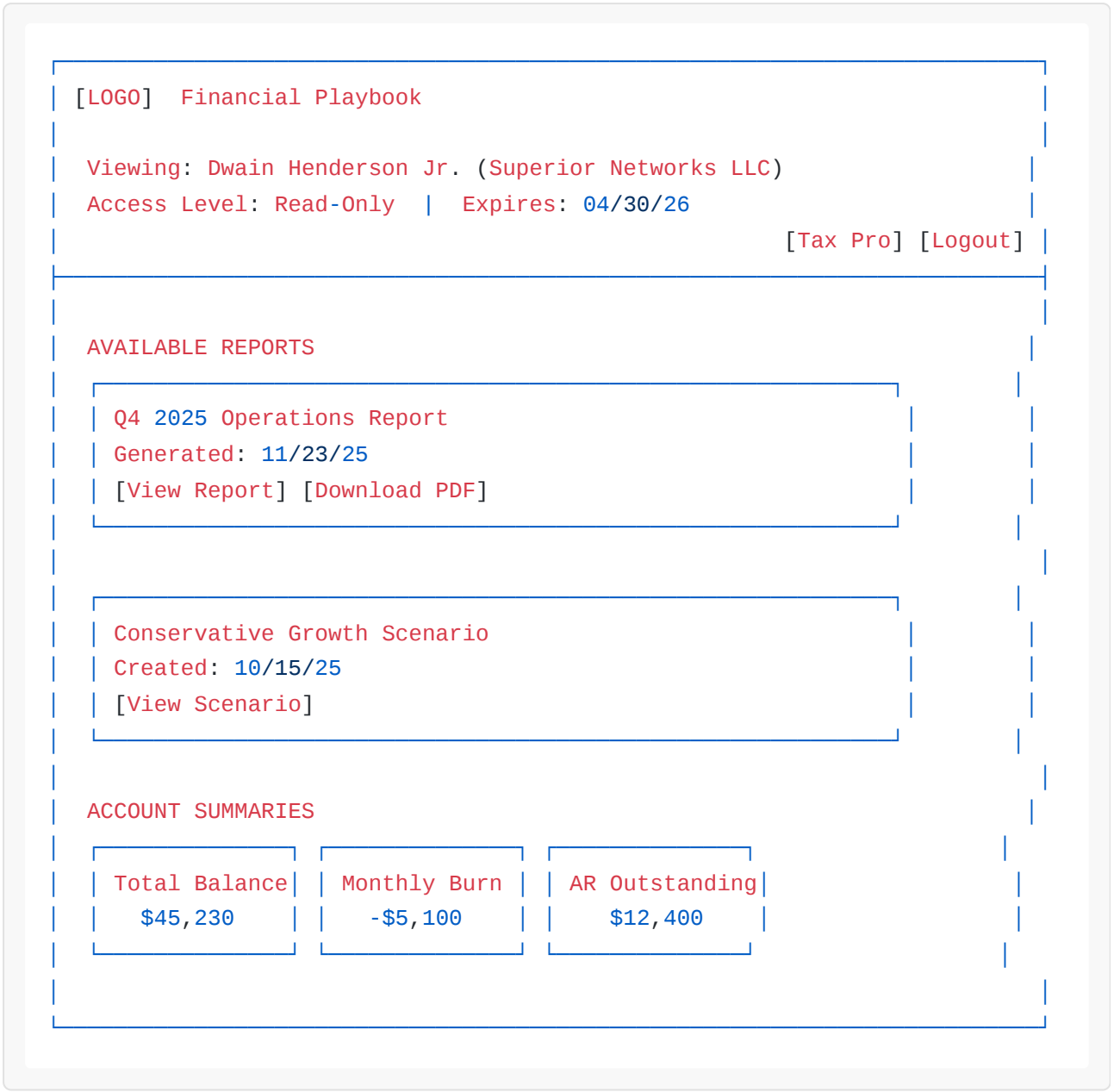

---

## Delegated User Experience

When the tax professional receives the invitation email:

1. **Invitation Email:**
    - Subject: “Dwain Henderson Jr. has invited you to view their Financial Playbook”
    - Body: “Click here to accept the invitation and create your delegated access account.”
2. **Delegated User Login:**
    - The tax professional creates a separate account (or logs in with Google).
    - Upon login, they see a dashboard that clearly indicates they are viewing Dwain’s data.

3. **Delegated Dashboard View:**




---

## Technical Implementation

### Database Schema

### Table: `delegated_access`

| Column | Type | Description |
| --- | --- | --- |
| `id` | UUID | Primary key |
| `owner_user_id` | UUID | Foreign key to the user granting access |
| `delegate_email` | String | Email of the delegated user |
| `delegate_user_id` | UUID | Foreign key to the delegated user (null until accepted) |
| `access_level` | Enum | `read_only`, `limited_edit` |
| `access_scope` | JSON | Permissions object (e.g., `{view_scenarios: true, view_reports: true}`) |
| `granted_at` | Timestamp | When access was granted |
| `expires_at` | Timestamp | When access expires |
| `revoked_at` | Timestamp | When access was revoked (null if active) |
| `invitation_token` | String | Unique token for invitation link |
| `accepted_at` | Timestamp | When the invitation was accepted |

---

### Access Control Logic

1. **When a delegated user logs in:**
    - Check the `delegated_access` table for active delegations.
    - If found, display the owner’s data with the appropriate restrictions.
2. **Permission Checks:**
    - Before rendering any page, check if the current user is a delegated user.
    - If yes, enforce read-only restrictions (disable edit/delete buttons, prevent API calls for modifications).
3. **Audit Logging:**
    - Log all actions taken by delegated users for security and compliance.

---

**Security Considerations**

1. **Expiration:** All delegated access must have an expiration date.
2. **Revocation:** The primary user can revoke access at any time.
3. **Scope Limitation:** Delegated users can only see what the primary user explicitly grants.
4. **No Cascading:** Delegated users cannot grant access to others.
5. **Audit Trail:** All delegated user actions are logged.

---

## User Stories

1. **As a primary user (Dwain), I want to grant my tax professional read-only access to my financial reports so they can prepare my taxes without needing to log into my account.**
2. **As a tax professional, I want to view my client's financial data in a secure, read-only environment so I can provide accurate tax advice without the risk of accidentally modifying their data.**
3. **As a primary user, I want to revoke delegated access at any time so I can control who has visibility into my financial information.**
4. **As a primary user, I want to set an expiration date for delegated access so I don't have to remember to manually revoke it after tax season.**

---

## Future Enhancements

- **Multiple Delegation Levels:** Add more granular permission levels (e.g., "View Only Summaries", "Full Read Access", "Read + Comment").
- **Delegation Notifications:** Notify the primary user when a delegated user logs in or views specific reports.
- **Delegation Activity Log:** Show the primary user a log of what the delegated user has viewed.
- **Delegation Templates:** Create reusable delegation templates for common use cases (e.g., "Tax Professional", "Business Partner").

---

This feature ensures that Dwain can securely share financial information with his tax professional while maintaining full control over his data and account security.

---

# AI-Powered Strategy Builder Wizard Specification

## 1. Overview

The **AI-Powered Strategy Builder Wizard** is a new feature designed to dramatically simplify the onboarding and playbook creation process. Instead of relying solely on manual data entry or live API integrations, users can upload their existing financial documents (such as bank statements or financial statements). An AI model will then analyze these documents to automatically extract data, identify financial patterns, generate initial graphs, and recommend tailored playbook strategies.

This feature acts as an intelligent onboarding assistant, doing the heavy lifting of data aggregation and initial analysis, which allows the user to move directly to the strategic "what-if" planning stage.

---

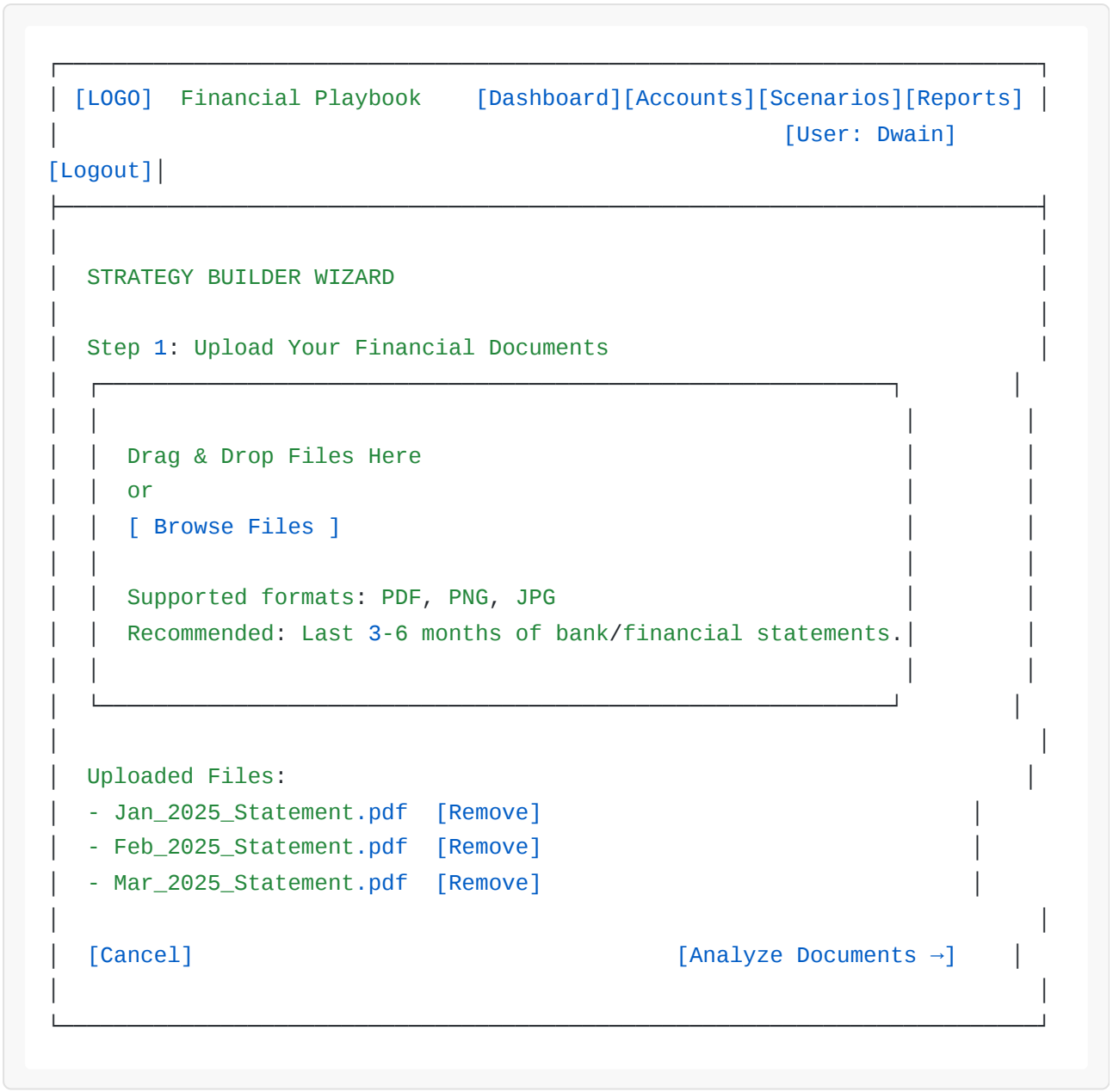
## 2. User Journey & Wireframes

The wizard will guide the user through a simple, three-step process: Upload, Analyze, and Recommend.

### Step 1: Document Upload

The user is prompted to upload their financial documents. The interface will accept multiple files and common formats (PDF, PNG, JPG).

**Wireframe: Document Upload Screen**




## Step 2: AI Analysis & Data Verification

Once the documents are uploaded, the AI model processes them. The user sees a progress screen. After analysis, the user is presented with a summary of the extracted data for verification. This is a crucial step to ensure accuracy.

**Wireframe: AI Analysis & Verification Screen**


## Step 3: Recommendations & Playbook Creation

After the user confirms the data, the AI generates insights and recommends 1-3 potential playbook strategies. Each recommendation comes with a brief explanation and a starting cash flow graph.

**Wireframe: Recommendations Screen**


Once a strategy is selected, the user is taken directly to the **Scenario Editor** (Wireframe 5), with all the extracted data, recurring bills, and the initial cash flow graph pre-populated. The user can then fine-tune the playbook.

---

## 3. Technical Implementation

The AI engine for this wizard involves a multi-step pipeline:

1. **Document Ingestion & OCR:**

- Uploaded files (PDF, PNG, JPG) are processed.
- **Optical Character Recognition (OCR)** is used to extract raw text from the documents. For PDF files, text extraction is direct where possible.

2. **AI-Powered Data Structuring:**

- The raw text is fed into a **Large Language Model (LLM)** with a specific prompt.
- The prompt instructs the model to act as a financial analyst and extract key information into a structured JSON format. This includes:
  - Transactions (date, description, amount, credit/debit)
  - Recurring income and expenses (e.g., payroll, rent, utilities)
  - Account balances (starting and ending)
  - Statement periods

3. **Analysis & Assumption Generation:**

- The structured JSON data is then processed by a backend service.
- This service calculates key metrics: average monthly income, expenses, burn rate, etc.
- It identifies patterns, such as bills that appear consistently across multiple statements.
- It generates a list of **assumptions** (e.g., "Assumed rent of $2,500/month based on consistent payments").

4. **Recommendation Engine:**

- The key metrics and identified patterns are fed into another LLM prompt or a rules-based engine.
- This engine is designed to recognize common financial situations and suggest appropriate strategies. For example:
  - If `(total_debt / total_assets) > 0.5`, recommend a **Debt Reduction** strategy.
  - If `(monthly_income - monthly_expenses)` is highly variable, recommend a **Cash Buffer** strategy.

        - If `AR_days_outstanding > 45`, recommend an **AR Acceleration** strategy.

5. **Graph & Playbook Generation:**
    - Based on the selected strategy, a starting playbook is generated.
    - The extracted transaction data is used to create an initial cash flow projection graph.
    - All this data is pre-populated into the Scenario Editor for the user to begin their detailed planning.

---

## 4. Benefits of this Feature

- **Reduces Friction:** Massively lowers the barrier to entry for new users.
- **Saves Time:** Automates hours of manual data entry.
- **Provides Instant Value:** Users get actionable insights and a working playbook within minutes of signing up.
- **Improves Data Accuracy:** Reduces the chance of human error during manual data input.
- **Creates a "Wow" Factor:** Demonstrates the power of the application's AI capabilities from the very first interaction.

This Strategy Builder Wizard will be a key differentiator, making the Financial Playbook application not just a tool for planning, but an intelligent partner in financial strategy creation.

---

# Comprehensive Audit Trail System Specification

## 1. Overview

To ensure data integrity, security, and accountability, a comprehensive **Audit Trail System** will be integrated into the Financial Playbook application. This system will automatically log all significant actions performed by users, delegated users, and the system itself. The audit log will provide a clear, filterable history of who changed what, and when, making it an essential tool for security reviews, debugging, and maintaining a transparent record of all activities.

---

## 2. Key Requirements

- **Immutability:** Audit logs cannot be altered or deleted by any user.
- **Comprehensiveness:** All critical actions must be logged.
- **Clarity:** Logs must be human-readable and provide sufficient context.
- **Accessibility:** The primary user must be able to easily view, filter, and export the audit trail.

---

## 3. Wireframe: Audit Trail Report Page

This new report will be accessible via the main navigation, likely under a new "Activity" or "Reports" tab.


Clicking **[Details]** would open a modal showing the raw `change_details` JSON, providing a before-and-after snapshot of the data.

---

# 4. Technical Implementation

## Actions to be Logged

The system will log the following events:

| Category | Action | Description |
| --- | --- | --- |
| **User Management** | `LOGIN`, `LOGOUT`, `UPDATE_PROFILE` | User authentication and profile changes. |
| **Data Entities** | `CREATE`, `UPDATE`, `DELETE` | Any change to Scenarios, Bills, Invoices, Accounts. |
| **Delegated Access** | `GRANT_ACCESS`, `REVOKE_ACCESS`, `VIEW_DATA` | Actions related to delegated users. |
| **System Events** | `SYNC_QUICKBOOKS`, `AI_ANALYSIS` | Automated system processes. |
| **Reports** | `GENERATE_REPORT`, `EXPORT_REPORT` | Creation and export of financial reports. |

## Database Schema

A new table, `audit_logs`, will be created.

**Table: `audit_logs`**

| Column | Type | Description |
| --- | --- | --- |
| `id` | UUID | Primary key for the log entry. |
| `timestamp` | Timestamp | The exact date and time the action occurred (UTC). |
| `user_id` | UUID | Foreign key to the `users` table. Identifies who performed the action. Can be null for system actions. |
| `user_name` | String | The name of the user for easy display (e.g., "Dwain Henderson Jr.", "System"). |
| `action_type` | Enum | The type of action performed (e.g., `UPDATE`, `CREATE`, `LOGIN`). |
| `target_entity` | String | The type of object that was affected (e.g., "Scenario", "Bill", "User"). |
| `target_id` | String | The unique ID of the specific object that was affected. |
| `change_details` | JSON | A JSON object containing `before` and `after` keys, showing the state of the data that was changed. |
| `description` | String | A human-readable summary of the event (e.g., "User updated the 'Q4 2025 Operations' scenario."). |

### Example `change_details` JSON for an UPDATE action:

```
{
  "before": {
    "monthly_income": 8200
  },
  "after": {
    "monthly_income": 8500
  }
}
```

### Implementation Logic

- **Middleware/Hooks:** The audit logging logic will be implemented using middleware in the backend API or database hooks. Before any `CREATE`, `UPDATE`, or `DELETE` operation is committed, a pre-save hook will capture the current state (`before`) and the new state (`after`) of the data.
- **Asynchronous Logging:** To avoid impacting application performance, audit log entries will be written to the database asynchronously.

---

## 5. Benefits

- **Enhanced Security:** Provides a clear record of all activities, helping to detect and investigate suspicious behavior, especially from delegated accounts.
- **Accountability:** Creates an undeniable record of who is responsible for every data modification.
- **Change Tracking:** Makes it easy to understand the history of a specific scenario or financial entry, which is invaluable for debugging and analysis.
- **Compliance:** For any future business needs, having a robust audit trail is often a requirement for financial compliance standards.

This audit system provides a powerful layer of security and transparency, making the Financial Playbook application a truly enterprise-ready platform.

---

# Enhanced Synchronization Controls Specification

## 1. Overview

To provide users with greater control over data freshness and system performance, this specification outlines a set of enhanced synchronization controls for the

QuickBooks Online integration. These features will allow for automatic syncing upon login, manual on-demand syncing, and user-configurable background sync frequencies.

---

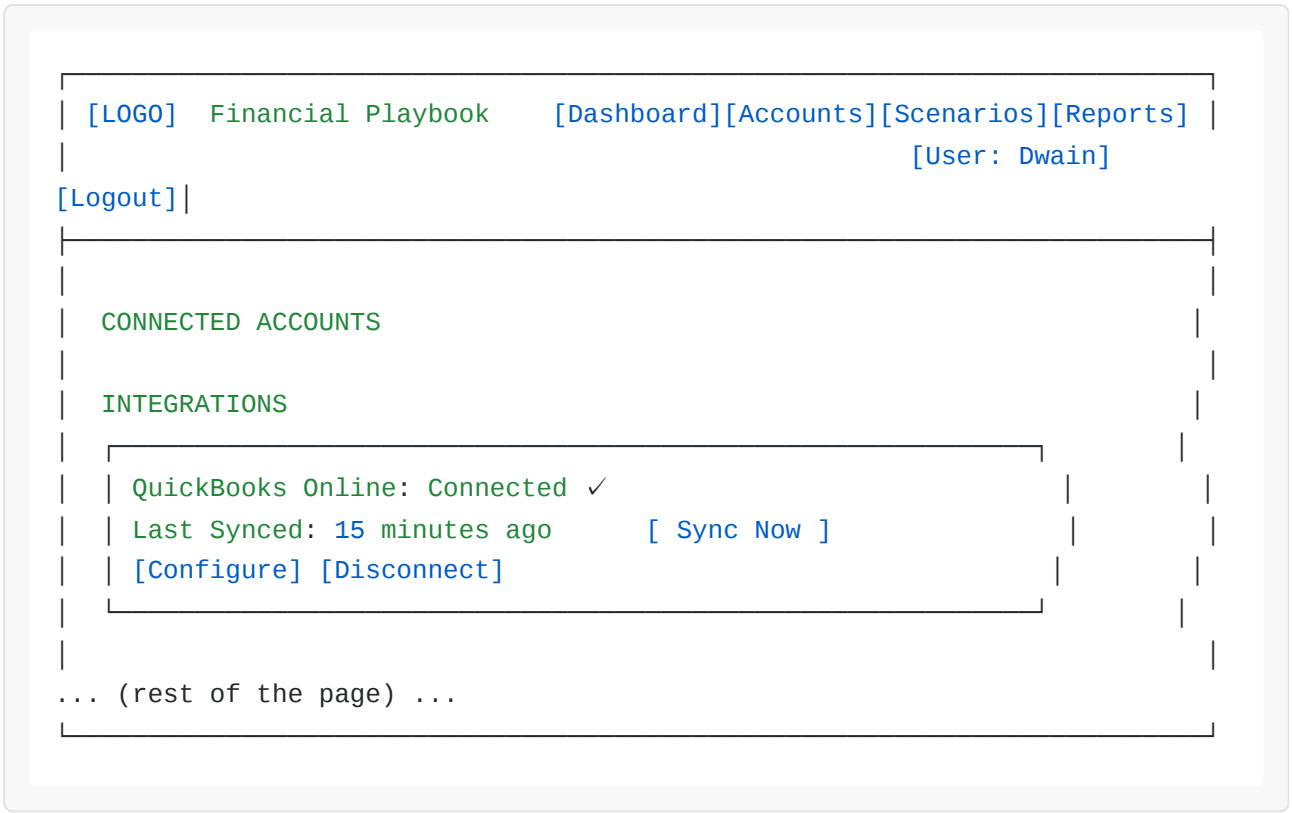
## 2. Feature Breakdown & Wireframe Updates

### a. Automatic Sync on Login

- **Description:** The system will automatically initiate a data synchronization with QuickBooks Online immediately after a user successfully logs in. This ensures that the user is always greeted with the most up-to-date financial data.
- **Implementation:** A backend job will be triggered post-authentication to run the sync process in the background.

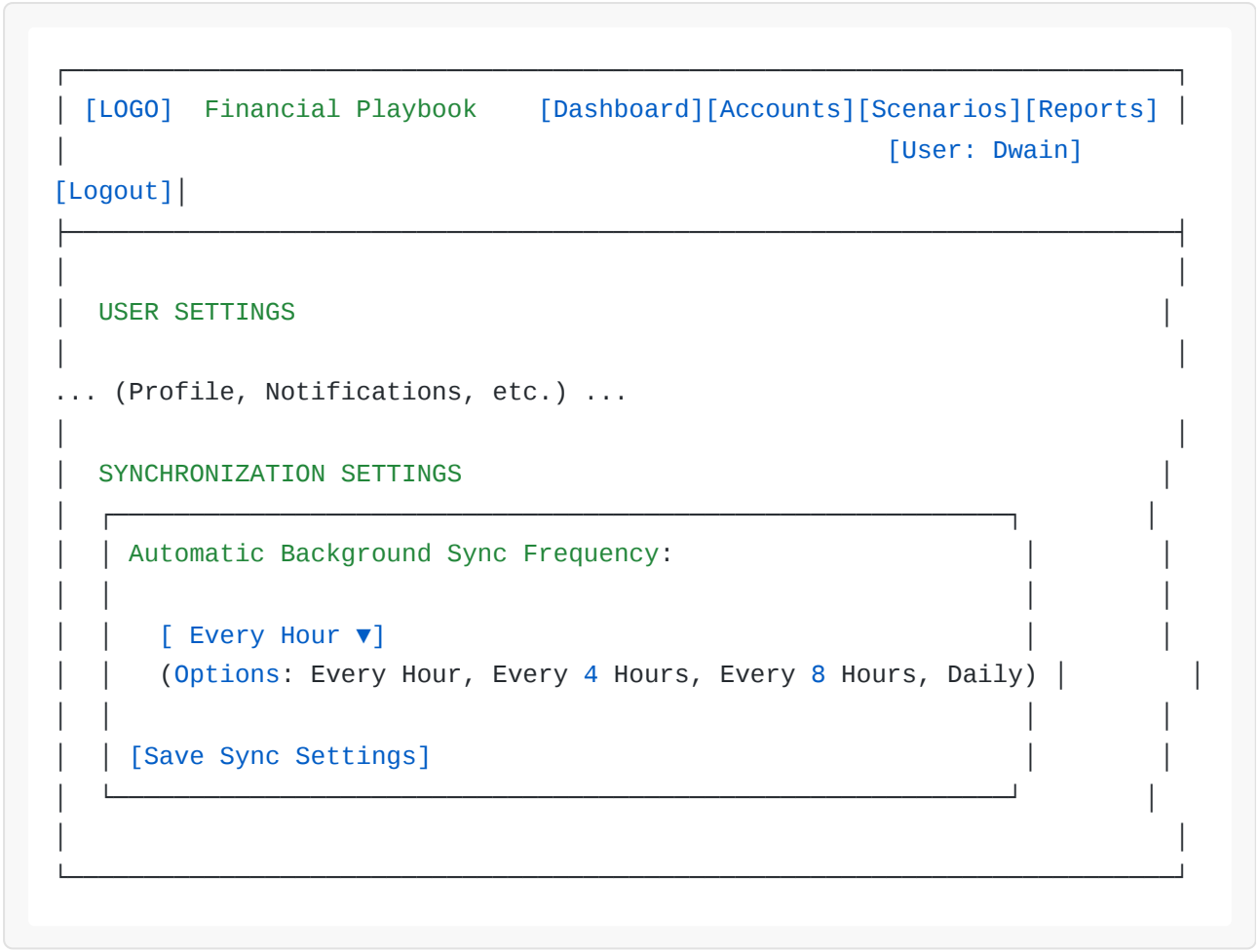
### b. Manual Sync Control & Status Display

- **Description:** The “Accounts Management” page will be updated to display the timestamp of the last successful synchronization and a button to trigger a manual sync at any time.
- **Wireframe Update (Accounts Management Page):**




## c. Configurable Background Sync Frequency

- **Description:** The “Settings” page will include a new section allowing the user to choose how often the system automatically syncs with QuickBooks in the background. This allows users to balance the need for real-time data with potential API rate limits from Intuit.
- **Wireframe Update (Settings & Notifications Page):**




---

## 3. Technical Implementation

- **Database Update:** The `users` table (or a new `user_settings` table) will be updated to include a `sync_frequency` column to store the user's preference (e.g., 1, 4, 8, 24 hours).
- **Backend Scheduler:** A cron job scheduler will be used on the backend. When a user updates their sync frequency, the system will update the schedule for that user's background sync job.
- **Sync Process:**
    1. **Trigger:** The sync process can be triggered in three ways: on-login, manually via the "Sync Now" button, or by the scheduled background job.
    2. **Execution:** The backend will initiate the QuickBooks API data fetch.

3. **Status Update:** The UI will update the “Last Synced” timestamp upon successful completion. If the user triggers a manual sync, the “Sync Now” button should show a “Syncing…” state and become disabled until the process is complete.

---

## 4. Benefits

- **User Control:** Empowers the user to decide how frequently their data is updated, balancing real-time needs with system efficiency.
- **Improved Data Freshness:** The combination of sync-on-login and background syncs ensures data is rarely stale.
- **Transparency:** The “Last Synced” status provides clear visibility into the data’s timeliness.
- **Flexibility:** The manual “Sync Now” button provides an immediate way to pull in the very latest data before making a critical decision or running a new scenario.