Real Estate Modeling Platform — MVP Development Brief

Overview

The proposed web application is a real estate financial modeling platform built for individual investors, brokers, and eventually institutional users. Inspired by the simplicity of TurboTax, it allows users to build detailed financial models of property investments through a guided, form-based web experience. Models will leverage Excel-based templates on the backend and will eventually support enterprise collaboration, reusable assumption blocks, and shared model libraries.

MVP Goals

The goal of the MVP is to:

- Allow users to create, edit, and version investment models
- Capture all relevant inputs across acquisition, renovation, operation, financing, and exit
- Output Excel files with formulas preserved
- Lay the groundwork for enterprise-grade infrastructure, even if not fully activated in V1

MVP Scope

1. Core Functionality

- **User Authentication**
  - Email/password login (Auth0 or similar)
  - Ability for users to leverage existing Google, Apple, or Microsoft to create account
  - Support for SSO down the line (Okta, Google Workspace, etc.)
- **Model Builder**
  - Step-by-step input capture (click-through style)
  - Input types: text, date, dropdown, number ranges, multi-property support
  - Ability to create multiple scenarios (base, upside, downside)

- **Excel Output**
    - Map inputs to backend Excel templates
    - Export as .xlsx with all formulas intact (no hardcoding)
- **Versioning**
    - Save/load versions of a model
    - Duplicate or fork versions for iteration
- **Project Management Tools**
    - Basic dashboard to view existing models
    - Tags for property type, deal size, geography

Technical Requirements

Frontend

- React (preferred)
- Tailwind (design system consistent with enterprise UX)
- Responsive web layout
- Dynamic form generation (potential future JSON schema-based – not necessary for MVP)

Backend

- Node.js or Python (TBD based on hire's preference)
- API to receive and store inputs, trigger Excel generation, return downloadable file
- Secure endpoints with JWT-based auth

Database

- **PostgreSQL**
    - Multi-tenant schema: isolate user data
    - Tables for users, models, versions, inputs, files

- Scalable design: expectation to have MVP support ~10 users, probably producing 3-5 models per month. Ability to scale (with expected cost structure) as platform increases adoption

Hosting & DevOps

- Looking for developer input here, but considering:
    - **Render** or **Vercel** for frontend
    - **Railway** or **Supabase** for backend and DB
    - AWS or GCP for long-term enterprise readiness
- Must support:
    - TLS encryption in transit
    - AES-256 encryption at rest
    - Field-level encryption for sensitive inputs
    - Logging and audit trails (SOC2 readiness)

Security & Enterprise Readiness (Framework-Level)

These are not fully built in MVP but must be designed for now:

- Role-Based Access Control (RBAC)
- Multi-tenant data isolation
- GDPR/CCPA data policies
- SAML support (future)
- Audit logs
- Compliant error handling and usage tracking

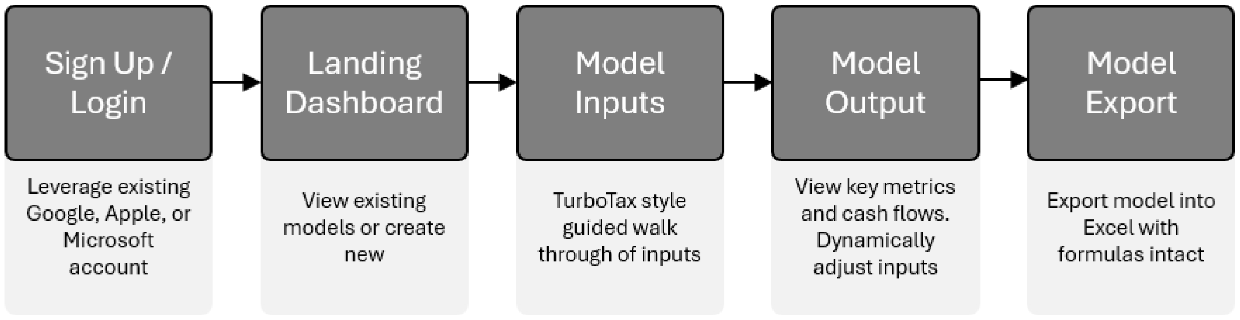
Where We Need Help

We're looking for a full-stack developer who can take ownership of the technical implementation of our MVP and set the foundation for future scaling. Specifically, we need someone who can:

1. Design and implement the backend stack and API endpoints
    - Set up secure endpoints to capture and retrieve model inputs
    - Map user inputs to backend Excel templates for download
2. Structure a scalable and secure PostgreSQL database
    - Design a multi-tenant schema to isolate user data
    - Create tables and relationships for models, versions, users, and assumptions
3. Advise on and configure hosting environments
    - Recommend cost-effective dev/staging/production setup (e.g., Railway, Render, Vercel)
    - Establish secure handling of environment variables and secrets
    - Set up CI/CD pipelines for automated testing and deployment
4. Implement secure user authentication and access controls
    - Use industry-standard tools (e.g., Auth0, Clerk, Supabase Auth)
    - Configure role-based access for future enterprise use cases (RBAC)
5. Support Excel output with formula preservation
    - Ensure all calculations are exported as formulas, not hardcoded values
6. Ensure UI/UX consistency across screens
    - Start from V0-generated wireframes
    - Unify styling, layout, and interactions into a clean, professional design system
    - Implement responsive, accessible, enterprise-grade UI using Tailwind CSS or a similar system
7. Establish a collaborative development workflow

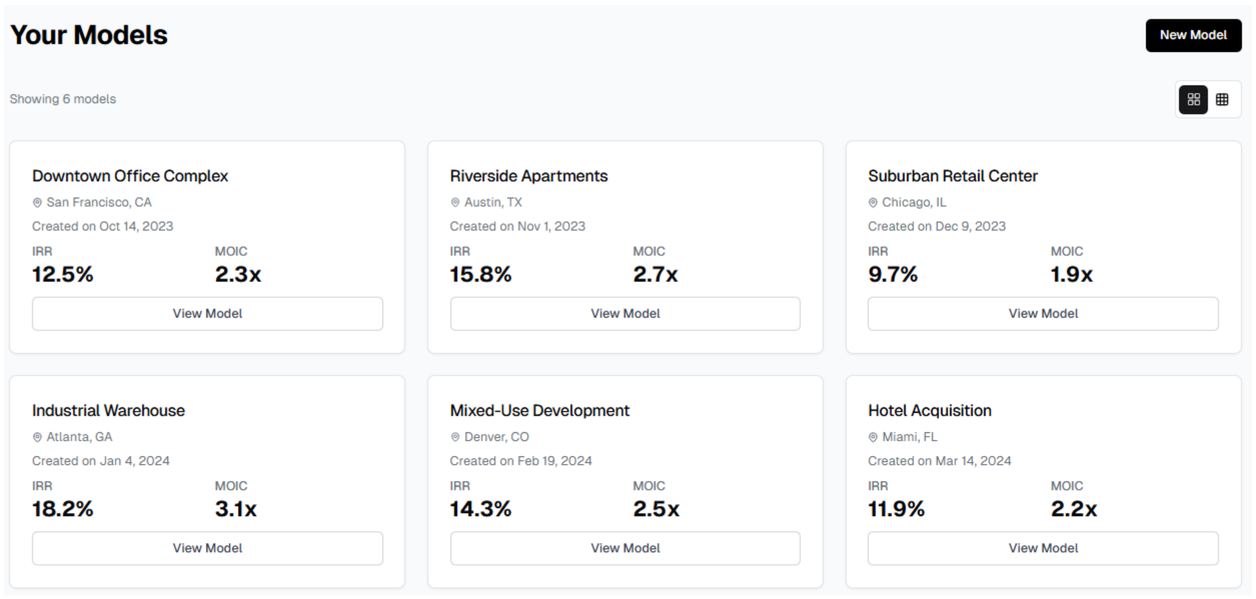
- Set up Git workflows with feature branches and pull request reviews
- Create dev/staging environments for internal testing
- Add basic test coverage and documentation

MVP User Journey:


Sign Up / Login
Leverage existing Google, Apple, or Microsoft account
Landing Dashboard
View existing models or create new
Model Inputs
TurboTax style guided walk through of inputs
Model Output
View key metrics and cash flows. Dynamically adjust inputs
Model Export
Export model into Excel with formulas intact

Example Wireframes:

Landing Page



Downtown Office Complex
San Francisco, CA
Created on Oct 14, 2023
IRR 12.5% MOIC 2.3x

Riverside Apartments
Austin, TX
Created on Nov 1, 2023
IRR 15.8% MOIC 2.7x

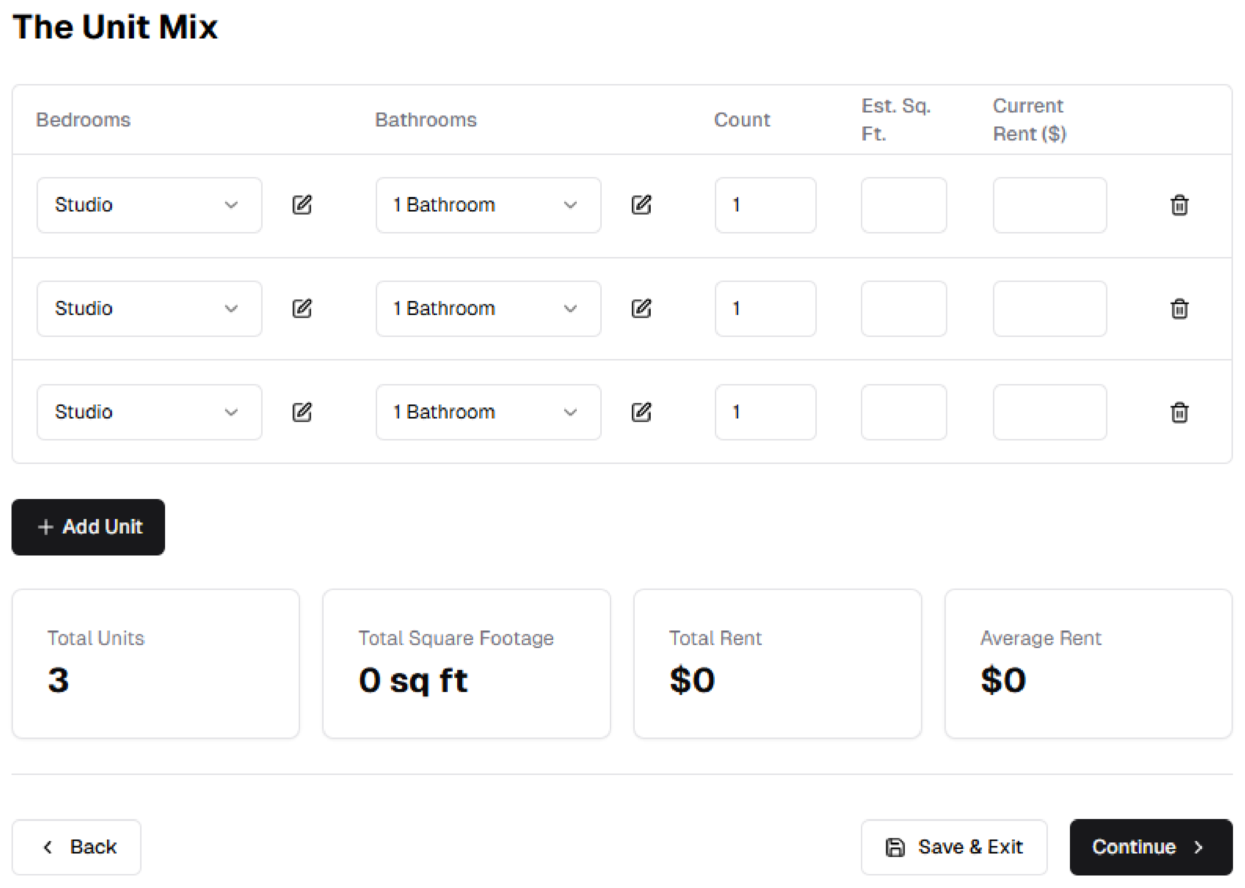
Suburban Retail Center
Chicago, IL
Created on Dec 9, 2023
IRR 9.7% MOIC 1.9x

Industrial Warehouse
Atlanta, GA
Created on Jan 4, 2024
IRR 18.2% MOIC 3.1x

Mixed-Use Development
Denver, CO
Created on Feb 19, 2024
IRR 14.3% MOIC 2.5x

Hotel Acquisition
Miami, FL
Created on Mar 14, 2024
IRR 11.9% MOIC 2.2x

Example Input Page

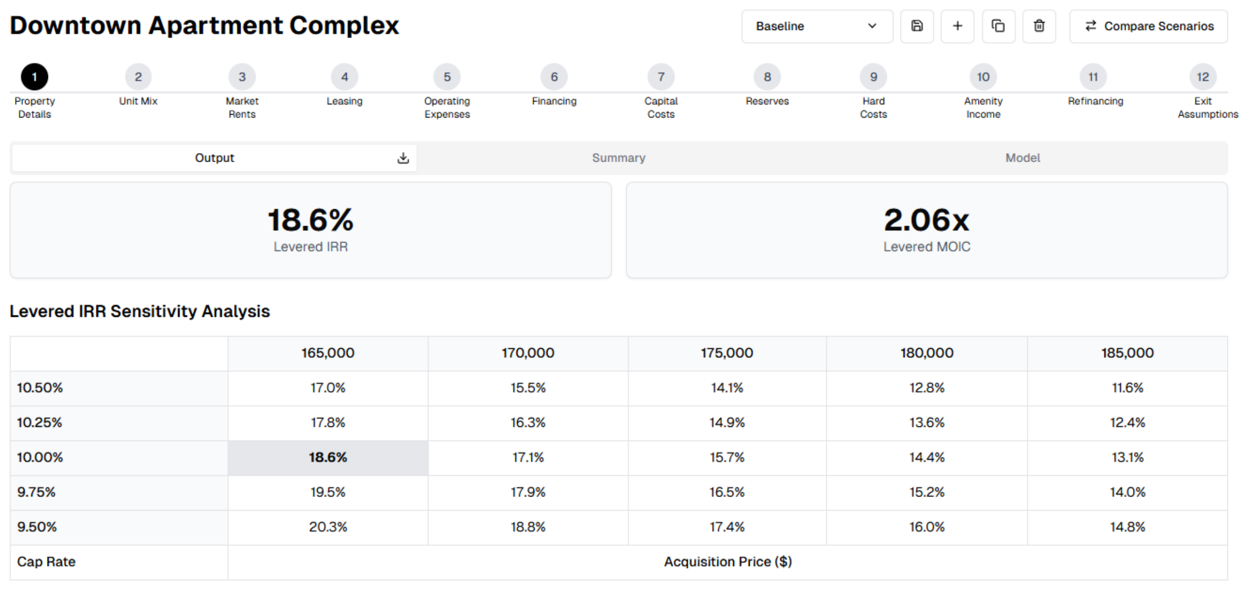


The Unit Mix

Bedrooms	Bathrooms	Count	Est. Sq. Ft.	Current Rent ($)
Studio	1 Bathroom	1		
Studio	1 Bathroom	1		
Studio	1 Bathroom	1		

Total Units	Total Square Footage	Total Rent	Average Rent
3	0 sq ft	$0	$0

Model Output Analysis



Downtown Apartment Complex

18.6% Levered IRR

2.06x Levered MOIC

Levered IRR Sensitivity Analysis

	165,000	170,000	175,000	180,000	185,000
10.50%	17.0%	15.5%	14.1%	12.8%	11.6%
10.25%	17.8%	16.3%	14.9%	13.6%	12.4%
10.00%	**18.6%**	17.1%	15.7%	14.4%	13.1%
9.75%	19.5%	17.9%	16.5%	15.2%	14.0%
9.50%	20.3%	18.8%	17.4%	16.0%	14.8%
Cap Rate	Acquisition Price ($)				

Database Entities & Attributes Strawman:

Entity	Description	Relationships
User	Represents an individual using the app (Investor, Broker)	Has many Models
Model	A full underwriting model created by a user	Belongs to a User, has many Properties, Scenarios, and AssumptionSets
Property	A single building or asset within a model	Belongs to a Model, has many Units, Expenses, etc.
Scenario	A specific financial projection (Base Case, Upside, Downside)	Belongs to a Model, references one or more AssumptionSets
Assumption Set	Grouping of assumptions used in a Scenario	Can be reused across scenarios or models
Unit	A rental unit (used for rent roll, renovations, etc.)	Belongs to a Property
Expense	Any expense (Operating, Closing, Soft/Hard, Legal)	Belongs to an AssumptionSet
Amenity	Revenue-generating amenities like laundry, storage, etc.	Belongs to a Property or AssumptionSet

Financing	**Debt structure (Acquisition, Refinance, etc.)**	**Belongs to a Scenario**
Exit	**Exit assumptions (Cap Rate, Exit Month, Selling Costs)**	**Belongs to a Scenario**
Version	**Tracks model state at a given point in time**	**Belongs to a Model, has a snapshot of a Scenario**

Entity	**Attributes**
User	**id, name, email, role**
Model	**id, user_id, title, created_at**
Property	**id, model_id, address, city, state, zip, purchase_price, closing_date**
Scenario	**id, model_id, name, type, assumption_set_id**
AssumptionSet	**id, name, description**
Unit	**id, property_id, bedrooms, bathrooms, square_feet, current_rent, market_rent**
Expense	**id, assumption_set_id, category, title, amount, factor, annual_cost, date_range**
Amenity	**id, assumption_set_id, type, start_date, utilization_rate, unit_count, monthly_fee**
Financing	**id, scenario_id, type, ltv, dscr, loan_amount, interest_rate, amortization, io_period**

Exit	**id, scenario_id, exit_month, exit_cap_rate, selling_costs**
Version	**id, model_id, scenario_id, version_name, timestamp**

Markdown-Style API Reference:

This section outlines the proposed REST API endpoints for the MVP version of our real estate financial modeling tool. Each section includes endpoint purpose, method, route, sample payload, and notes.

Authentication

POST /auth/register

Register a new user.

```
{
  "name": "Charlie",
  "email": "charlie@example.com",
  "password": "securepassword123"
}
```

POST /auth/login

Authenticate a user and return a JWT token.

```
{
  "email": "charlie@example.com",
  "password": "securepassword123"
}
```

Users

GET /users/me

Returns profile data for the currently authenticated user.

Models

GET /models

Get all models created by the current user.

POST /models

Create a new model.

```
{
  "title": "Montclair 4-Unit Model"
}
```

GET /models/:id

Get full details for a single model (including properties, scenarios, etc).

PUT /models/:id

Update model metadata.

DELETE /models/:id

Delete a model.

Properties

POST /models/:id/properties

Add a new property to a model.

```
{
  "address": "123 Main St",
  "city": "Montclair",
  "state": "NJ",
  "zip": "07042",
  "purchase_price": 850000,
  "closing_date": "2025-07-01"
}
```

PUT /properties/:id

Update property details.

Units

POST /properties/:id/units

Add unit mix info to a property.

```
{
  "bedrooms": 2,
  "bathrooms": 1,
  "square_feet": 850,
  "current_rent": 1800,
  "market_rent": 2200
}
```

Scenarios

POST /models/:id/scenarios

Add a financial scenario (e.g. Base Case, Upside).

```
{
  "name": "Upside Case"
}
```

PUT /scenarios/:id

Update scenario metadata.

Assumptions

POST /scenarios/:id/assumptions

Create a new assumption set for a scenario.

PUT /assumptions/:id

Update assumptions (bulk payload).

```
{
 "rent_growth_rate": 0.025,
 "exit_cap_rate": 0.0525,
 "operating_expenses": [
  { "title": "Insurance", "amount": 12000, "factor": "Per Unit" },
  { "title": "Taxes", "amount": 20000, "factor": "Total" }
 ]
}
```

Financing

POST /scenarios/:id/financing

Define acquisition or refinance terms.

```
{
 "type": "Acquisition",
 "ltv": 0.75,
 "loan_amount": 637500,
 "interest_rate": 0.06,
 "amortization": 30
}
```

Exit

POST /scenarios/:id/exit

Define model exit assumptions.

```
{
  "exit_month": 120,
  "exit_cap_rate": 0.055,
  "selling_costs": 0.05
}
```

Export (Optional for MVP)

GET /models/:id/export

Return downloadable Excel file with formulas based on inputs.