Equity Smart

Front-Facing Web Applications

This document provides external-facing information for deployed instances of Equity Smart's web applications. All sensitive internal details have been deliberately excluded.

TABLE OF CONTENTS

OVERVIEW

Purpose

Equity Smart is composed of two integrated, public-facing websites:

- *Equity Smart Home Loans* – Focused on mortgage guidance, prequalification, and rate comparisons
- *Equity Smart Real Estate* – Focused on property listings and agent discovery

Both websites display tailored users from the users Mongo database, utilize APIs like Google ReCaptcha, and share a common AI assistant chatbot, capable of answering natural language questions by dynamically accessing structured data from MongoDB and the CoreLogic Trestle API using Retool and AWS services.

Structure

Equity Smart is built as a serverless, full-stack system. Static assets (HTML, CSS, JS, SASS) are deployed to AWS S3 and served via CloudFront for secure HTTPS delivery. The backend is powered by AWS Lambda, API Gateway, and Retool Workflows which access a centralized MongoDB database and the IDX listings API.

Frontend: HTML, CSS, JavaScript, SASS (based on the Homy template)

Hosting: AWS S3 + CloudFront

Backend: API Gateway + Lambda

Database: MongoDB accessed through Retool

Listings: Queried dynamically via CoreLogic Trestle API

AI Assistant: Shared RAG-based chatbot using GPT-4o-mini + structured query execution

Search and filtering logic for users and listings is performed client-side based on initial API data fetched and cached locally.

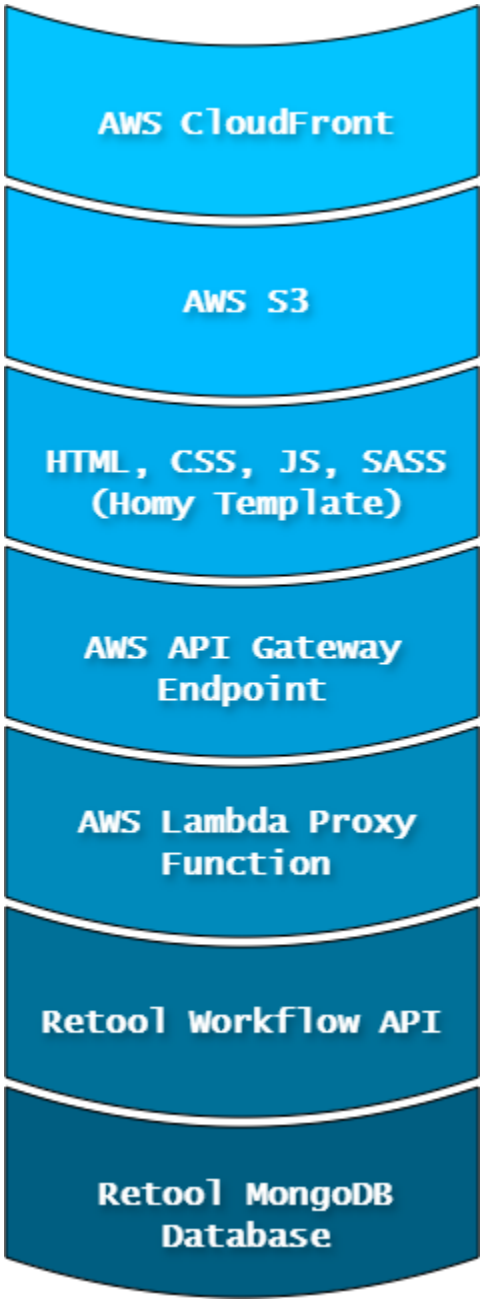


Figure 1: Stack Diagram

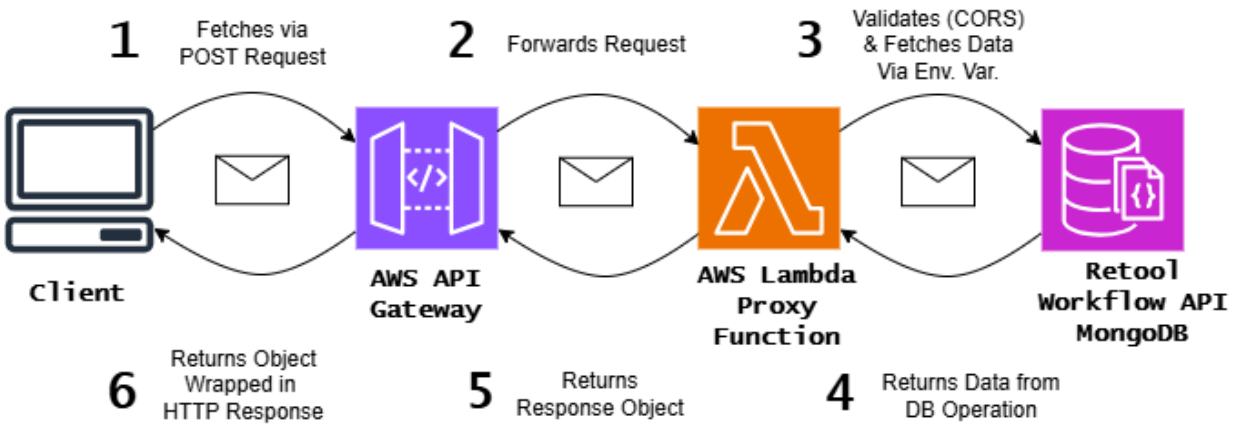
This diagram illustrates the ESRE Web App's architecture stack, showing a comprehensive overview of the component structure.

Process Flow

When a user interacts with either website:

1. The client sends a POST request (e.g., to fetch data).
2. The request is forwarded to the API Gateway.
3. The API Gateway routes the request to either / or /listings
4. The appropriate Lambda function processes and validates it, processing the request and forwarding it to the appropriate backend workflow.
5. Lambda securely calls the appropriate Retool Workflow API,
6. The workflow returns structured data (e.g., a list of users or listings).
7. Lambda wraps and returns the response to the client as an HTTP response.
8. The client then displays and/or caches the information.

Figure 2: Process Flow Diagram



This diagram illustrates the data flow from the frontend to the backend. A client initiates a POST request to an AWS API Gateway endpoint. This endpoint then proxies the request to an AWS Lambda function. The Lambda function is responsible for validating the request and retrieving necessary credentials from its environment variables. Subsequently, it makes a secure call to a Retool Workflow, which directly interacts with a MongoDB database. After data retrieval, the information is passed back through Lambda and returned to the client as a structured HTTP response.

SYSTEM ARCHITECTURE

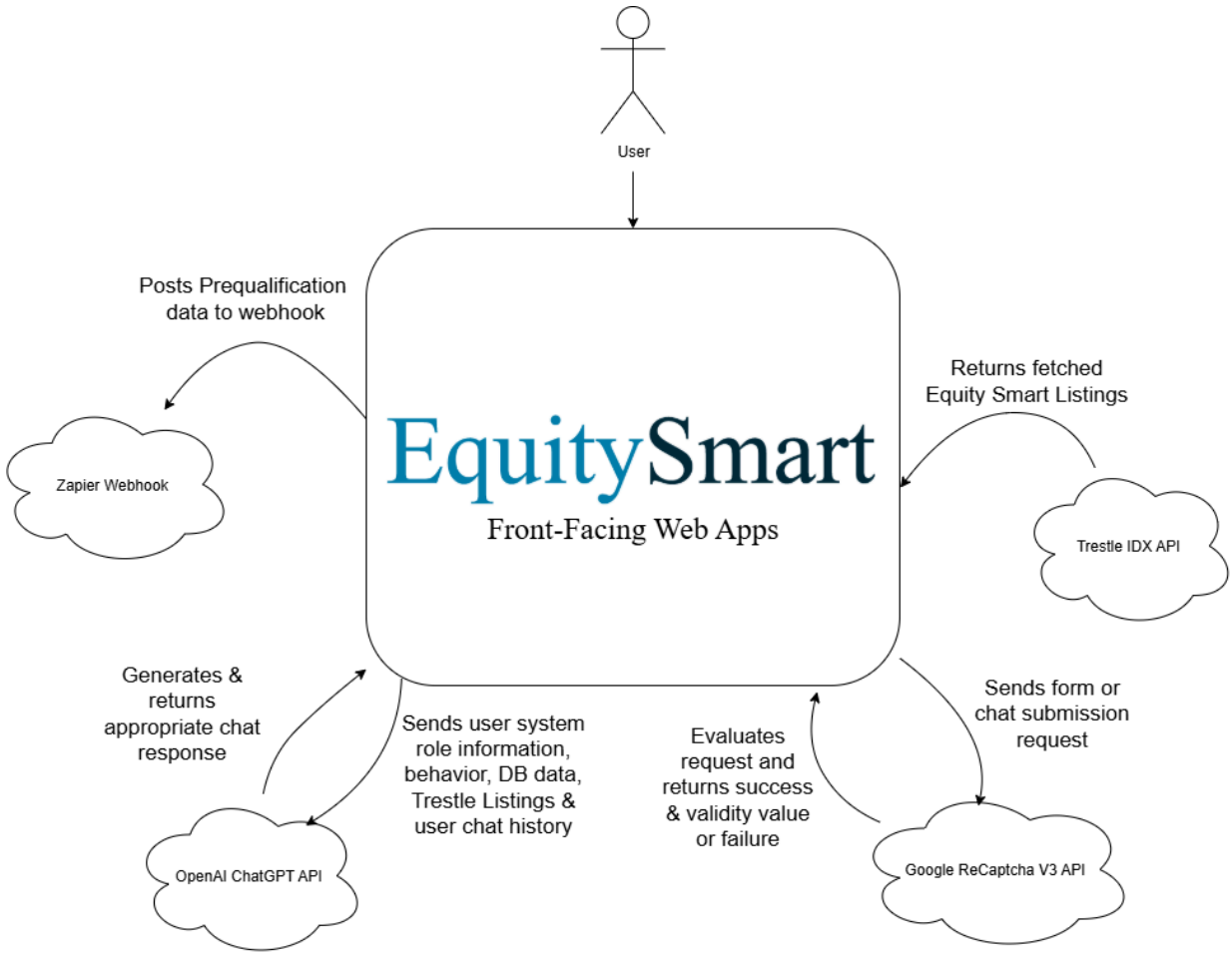
Context (Level 0)

A context diagram visually defines the boundaries of a system and its interactions with external elements, offering a high-level overview. This tool is invaluable for clarifying what falls under the system's control and identifying external services or actors.

For Equity Smart, this helps distinguish the front-facing web applications and their core functions, while also pinpointing third-party systems like APIs, automation endpoints, and users that communicate with the platform. This clear representation is vital for comprehending integration points, data flows, and trust boundaries within a serverless, API-driven architecture.

The Equity Smart platform functions as a unified system, integrating both Home Loans and Real Estate websites. This system interacts with various external entities, with data flowing in and out as needed. Users initiate requests through the front-facing web applications, performing actions such as browsing listings, submitting forms, or interacting with the AI assistant chatbot.

Figure 3: Context Diagram (DFD-0)



This diagram illustrates the context of the Equity Smart Web App platform as a whole. For form submissions and chatbot interactions, the system verifies user authenticity using the Google reCAPTCHA v3 API, which provides a success or failure score to validate the request. Real estate listings are kept current by retrieving property data via the CoreLogic Trestle IDX API.

When users engage with the AI assistant, a structured request is sent to the OpenAI ChatGPT API. This request includes the user's role, behavior, previous queries, and fetched data (such as listings or MongoDB values). The AI assistant then returns a natural language response to the user.

For mortgage-related actions, like prequalification, structured form data is forwarded to a Zapier webhook for external processing and automation. This system is designed to be entirely serverless, securely interfacing with third-party services to deliver dynamic functionality without storing sensitive client-side data.

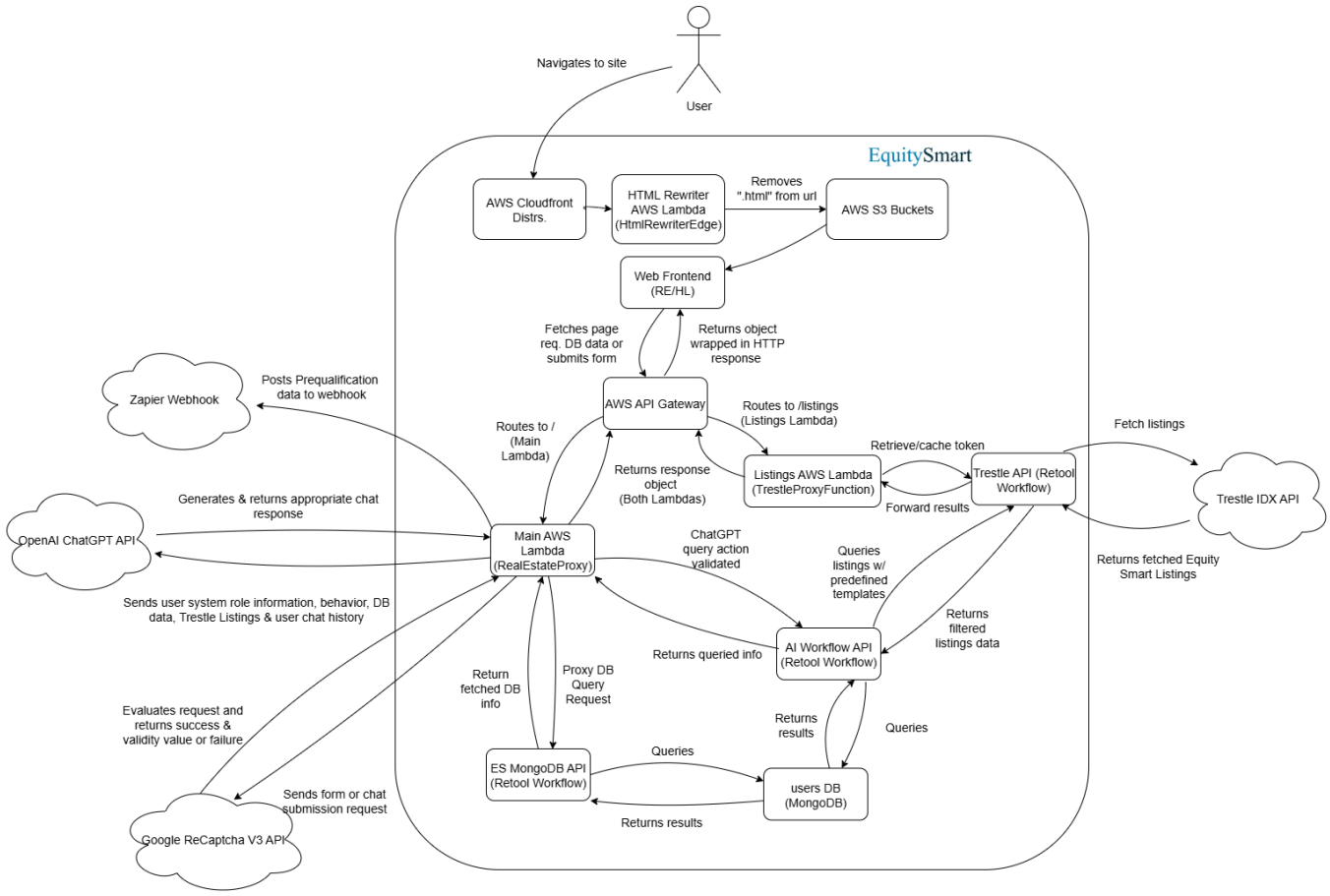
Internal Processes (Level 1)

While a context diagram defines the system's external boundaries, a Data Flow Diagram (DFD) offers a more in-depth look at its internal processes. Specifically, a Level 1 DFD breaks down the system's main components, showing how data moves between internal functions, storage, and external services.

For Equity Smart, this illustrates how user actions from the Real Estate or Home Loans sites are routed through serverless components like AWS Lambda, API Gateway, and Retool Workflows. It details the querying of listings and user data, the construction and resolution of chatbot prompts, and the secure integration of third-party APIs.

This detailed internal view is essential for understanding how the platform handles data at each stage, supports modular functionality across websites, and manages secure, stateless interactions with both internal services and external APIs.

Figure 4: Data Flow Diagram (DFD-1)



This diagram details the internal structure of the Equity Smart platform, illustrating how user interactions are managed across serverless processes and external APIs. Unlike the high-level context diagram, which views the system as a single black box, this Level 1 perspective reveals the internal workflows and data exchanges among the platform's core components.

When users access either the Real Estate or Home Loans website, static assets are delivered from S3 via AWS CloudFront. Before requests are completed, the HTML Edge Rewriter Lambda modifies URLs by removing `.html` suffixes for cleaner navigation.

Upon user interaction-such as submitting forms, requesting data, or engaging with the AI assistant-the frontend sends API calls through AWS API Gateway. The gateway directs these requests to the appropriate AWS Lambda function: the Main Lambda handles general or chatbot queries, while the Listings Lambda manages `/listings` endpoints and token caching for IDX access.

The platform integrates with three Retool Workflows:

- ***MongoDB Workflow:*** *Retrieves user, review, or branch data from a centralized database.*
- ***Trestle Workflow:*** *Queries the CoreLogic IDX API for real-time property listings and caches tokens as needed.*
- ***AI Workflow:*** *Manages structured query actions initiated by the chatbot.*

The AI assistant constructs prompts using internal context-including user role, chat history, listing results, and Mongo data and sends them to the OpenAI ChatGPT API to generate natural language responses. Concurrently, form submissions requiring reCAPTCHA validation or mortgage prequalification are routed to external services: Google reCAPTCHA v3 for human verification and Zapier for webhook-based data processing.

This diagram demonstrates how the Equity Smart system orchestrates data flow across distributed components, ensuring security, modularity, and external integration at every stage.

Deployment

The deployment architecture for Equity Smart consists of two static websites: Home Loans and Real Estate, each hosted separately using AWS infrastructure.

Both sites are deployed to individual Amazon S3 buckets and served through dedicated CloudFront distributions, enabling secure HTTPS delivery and global caching.

These static sites communicate with backend services through a shared AWS API Gateway, which routes requests to Lambda functions and Retool Workflows.

This setup enables a scalable, serverless deployment with decoupled frontend and backend components.

SECURITY

Introduction

The Equity Smart platform employs a layered, design-driven security model tailored to its fully serverless, public-facing architecture. While the system does not handle authenticated user accounts or private session data, several key measures ensure that data exposure, abuse, and unauthorized access are tightly controlled at every level of the stack.

Secrets & Environment Management

Sensitive credentials, such as API keys, access tokens, and internal URLs, are stored securely in environment variables. Secrets are never exposed to the client or stored in static files. The exact details are not publicly accessible.

Public API & Access Control

The platform exposes two main API Gateway routes (/ and /listings), both of which are public but carefully constrained:

- Access to these endpoints is restricted by CORS and domain whitelisting to ensure requests only originate from approved frontend domains.
- All client requests are routed through Lambda, which performs input

validation and ensures data access is restricted to safe, public-facing content.
- Retool Workflows are used exclusively for database access with additional layers of security in place.

AI Assistant Safeguards

The integrated AI assistant is capable of answering natural language questions about listings, agents, branches, and more. To ensure safe and accurate responses, all AI-generated queries are routed through a controlled backend system that strictly validates the input, applies safe filtering rules, and limits access to approved public data only. The assistant never interacts directly with the database or APIs, instead, it summarizes pre-filtered results returned by secure workflows. This approach ensures a seamless user experience while maintaining strict data security standards.

reCAPTCHA Integration

To mitigate bot spam and automated abuse:

- All form submissions (e.g., reviews, mortgage prequalification) and chatbot messages are gated by **Google reCAPTCHA v3**
- Scores are evaluated server-side before processing requests
- reCAPTCHA keys are stored securely and tokens are never exposed in frontend storage

Data Exposure & Client Caching

All client-side caching is non-sensitive:

- Initial data (users, branches, listings) is cached locally for client-side filtering and sorting
- No sensitive or privileged data is ever included in client-visible responses
- Reviews are public-submittable, but moderated manually before being

shown on-site

While it's possible for a determined user to bypass the per-user review cooldown by clearing local storage, this is considered a low-risk vector due to manual approval and spam detection mechanisms already in place.

WEBSITES

Introduction

The Equity Smart platform consists of two public-facing websites:

- *Equity Smart Real Estate*
- *Equity Smart Home Loans*

Both are built from a shared static template and powered by the same backend infrastructure. While they share layout and architecture, each site presents distinct content, tailored functionality, and dynamic interactions specific to their audience and purpose.

Shared Frontend Architecture

Our front-facing web applications are developed using HTML, CSS, JavaScript, and SASS, leveraging the Homy template. They are deployed to S3 and delivered via CloudFront, featuring client-side routing and caching logic. These applications boast a mobile-responsive layout and integrate shared components such as navbars and cards. Additionally, they include chatbot assistant and reviews integrations.

Equity Smart Real Estate

- Focus: Property listings, realtor discovery, location-based browsing
- Dynamic Pages:
    - /listings.html: Dynamically fetches listings via the IDX workflow

    - /listing_details.html: Loads listing details from URL-based listingKey
    - /realtors.html: Shows filtered users (title = “Real Estate Agent”)
- Client-Side Features:
    - Filter by price, city/zipcode, bedroom/bathroom count
    - Listing data cached in-memory for filtering/sorting
    - Chatbot answers real estate related questions

Equity Smart Home Loans

- Focus: Property listings, realtor discovery, location-based browsing
- Dynamic Pages:
    - /loan_officers.html: Filtered user list (title = “LO”)
    - /calculator.html: Multi-tab mortgage calculator (JS only)
    - /preq.html: Prequalification form with reCAPTCHA and Zapier webhook
- Client-Side Features:
    - Client-side logic for form validation and submission
    - reCAPTCHA v3 integrated with form submit
    - Chatbot answers loan and mortgage related questions

Shared Behaviors

- API Gateway routes requests to a shared backend.
- An AI assistant, embedded in both sites, provides context-aware support.
- Client-side filtering and search logic ensure fast performance.
- Only public-safe data is handled, with no authentication or session management.

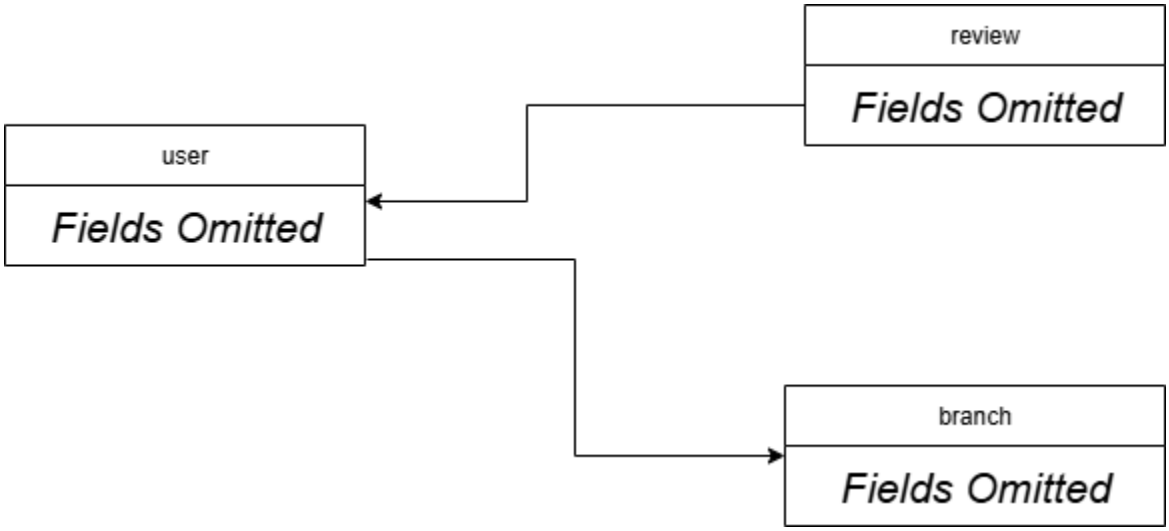
DATABASE STRUCTURE

Overview

The Equity Smart platform uses a centralized MongoDB database to manage public-facing information about real estate professionals, loan officers, branch locations, and user-submitted reviews. This structure supports both the Home Loans and Real Estate websites, enabling features like profile search, branch filtering, and review display.

Each user is linked to a branch and may have multiple associated reviews. Branch records include location and licensing data used for filtering and navigation. Reviews are stored with basic metadata and only appear publicly after manual moderation. Sensitive fields such as home addresses or personal identifiers are excluded from all frontend responses and chatbot queries.

Figure 5: users Schema Diagram



Schema Overview

The database includes three primary collections: user, review, and branch. These collections are relationally connected to support rapid filtering, search, and data display across the websites and AI assistant.

- Users represent realtors, loan officers, and other public-facing staff. Each user is linked to a branch and includes displayable contact information and professional identifiers (e.g., license numbers).
- Reviews are tied to individual users and contain public feedback submitted by clients. Only approved reviews are shown.
- Branches store location information and licensing data used to group and filter staff members.

This schema is designed for efficient querying and is optimized for both client-side performance and AI-assisted workflows. All data returned to the frontend is filtered through secure workflows to ensure only public-safe content is displayed.

Relationships

Users are linked to branches (many users per branch) and can have multiple reviews. This relational structure enables real-time filtering and content generation for both user-facing views and chatbot queries, while maintaining strong data integrity.

RETOOL WORKFLOWS

Introduction

Retool Workflows serve as the primary backend layer for interacting with Equity Smart’s structured data sources. These workflows are triggered by Lambda functions in response to API requests from the websites or chatbot.

Each workflow securely executes parameterized queries, applies projection rules, and returns structured responses to the calling function.

Workflows Overview

Every workflow is designed to gather distinct information from various sources.

Workflow	Purpose	Triggered by
ES MongoDB API	Retrieves users, reviews, and branches from the Equity Smart database	Main Lambda (/ endpoint)
Trestle API	Queries listing data from the CoreLogic IDX API; also handles token caching	Listings Lambda (/listings endpoint)
AI Workflow API	Executes validated AI-generated queries using safe templates or whitelisted logic	Main Lambda (when GPT action is detected)

ES MongoDB API

This workflow handles backend interactions with the Equity Smart MongoDB database, providing secure access to public-facing data such as realtors, loan officers, branch locations, and user-submitted reviews. It supports both website and chatbot features by returning only approved, filtered content. Security is enforced through strict validation rules and field-level filtering to ensure that no private or sensitive data is exposed. All reviews are manually approved before being displayed.

Trestle API

This workflow retrieves real-time property listings and associated media by connecting to the CoreLogic IDX system. It supports filtering by parameters like location or price and ensures that only listings affiliated with Equity Smart are returned. Token-based authorization and media checks are handled internally to guarantee a secure and consistent user experience across both the website and chatbot.

AI Workflow API

This workflow allows the AI assistant to execute structured queries on public data using predefined templates or validated inputs. It supports intelligent, context-aware responses by securely accessing filtered information from both the MongoDB database and property listings. All queries are tightly validated and limited to safe, read-only operations to maintain strict data integrity.

AWS LAMBDA FUNCTIONS

Introduction

The Equity Smart platform uses a modular, serverless backend architecture powered by AWS Lambda. Each function is designed to handle a specific category of frontend interactions, such as retrieving listings, processing form submissions, or supporting AI-assisted queries. These stateless functions act as intermediaries between the public-facing websites and backend workflows, ensuring that all data access is secure, scalable, and limited to public-safe content. This approach enables rapid development and maintenance while keeping the overall system lightweight and efficient.

Main Lambda (RealEstateProxyFunction)

This function acts as the primary backend handler for both Equity Smart websites. It processes data requests, validates user submissions, and

interprets structured responses from the AI assistant. It routes queries to the appropriate workflow based on user intent—whether fetching reviews, user profiles, or listings—and ensures all data is securely processed and filtered before reaching the client.

Listings Lambda (TrestleProxyFunction)

This function handles property listing requests from the Real Estate website and chatbot. It forwards validated queries to the listing workflow, manages access token authorization, and ensures that returned listings are accurate, media-rich, and filtered to include only public listing data for Equity Smart properties. This function helps power real-time search and filtering across the site.

HTML Edge Rewriter Lambda

This lightweight function enables clean, extensionless URLs by rewriting incoming requests to match the correct .html page. It is triggered automatically by the hosting layer and supports static routing for both websites. It does not access any data or perform logic beyond URL transformation.

AWS API GATEWAY

Summary

API Gateway acts as the entry point for all backend requests, routing traffic to the appropriate Lambda function. All routes are secured via origin restrictions and are designed to handle only public-safe POST requests.

CHATBOT INTEGRATION

Summary

The Equity Smart AI Assistant, a RAG-style chatbot powered by GPT-4o-mini, is seamlessly integrated across both the Home Loans and Real Estate websites. It offers natural-language responses to user inquiries on real estate and mortgage topics, leveraging a role-aware context to interpret queries. The assistant securely accesses public and live data through validated workflows and safe queries, ensuring only filtered, relevant information is provided to users. This allows it to answer questions about listings, loan officers, branches, and more.

DEPLOYMENT & CONTINUOUS DELIVERY

Overview

The Equity Smart platform follows a serverless deployment model designed for scalability, speed, and security. Both the Home Loans and Real Estate websites are statically built and deployed to AWS S3, with assets delivered globally via CloudFront for HTTPS delivery and caching.

Changes to the frontend are managed through an automated deployment workflow that pushes updates to S3 and ensures they are live immediately. The backend services—including AWS Lambda functions and Retool Workflows—are modular and

stateless, allowing the platform to scale efficiently while maintaining strict separation between frontend and backend layers.

This setup allows for rapid iteration, minimal manual overhead, and secure, public-facing delivery of real estate and mortgage-related features.

Frontend Deployment

Site	Hosting Stack
Real Estate	S3 + CloudFront
Home Loans	S3 + CloudFront

Backend Services

Component	Deployment Approach
Lambda Functions	Managed using AWS developer tools for deployment
API Gateway	Managed using AWS developer tools for deployment
Retool Workflows	Managed within Retool UI

APPENDICES

Figures

Figure 1: Stack Diagram

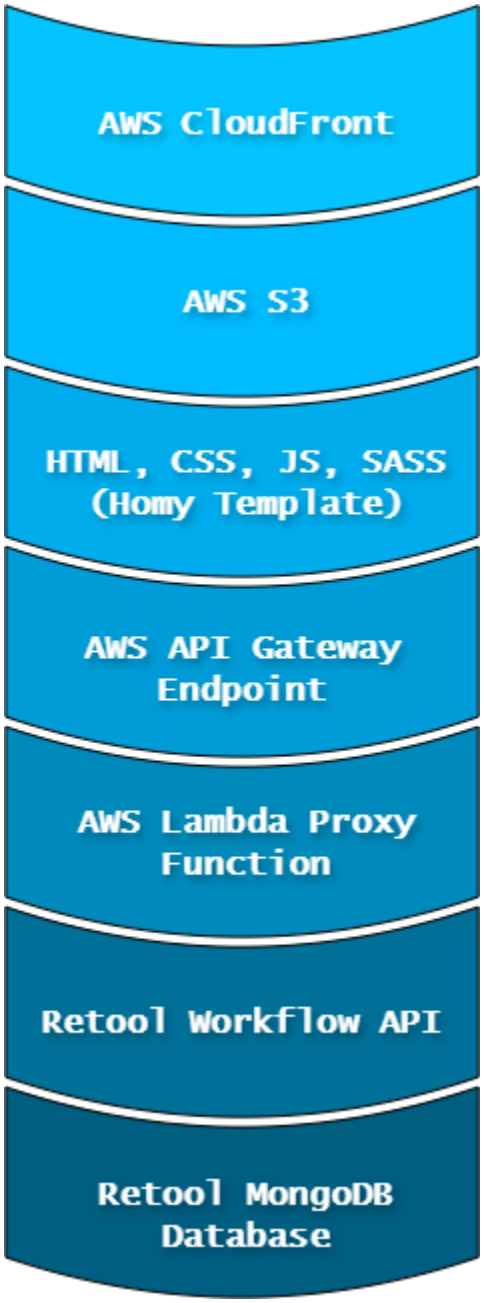


Figure 2: Process Flow Diagram

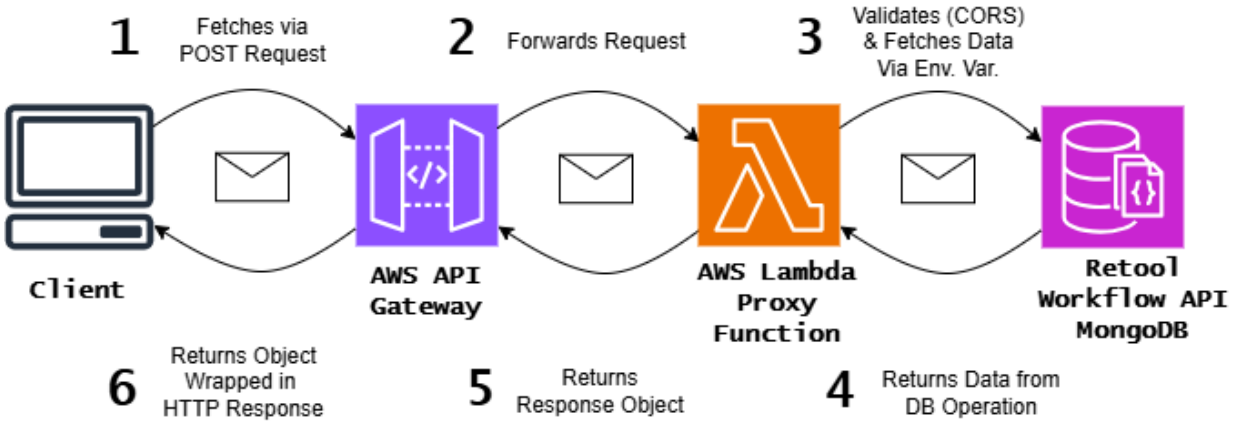


Figure 3: Context Diagram (DFD-0)

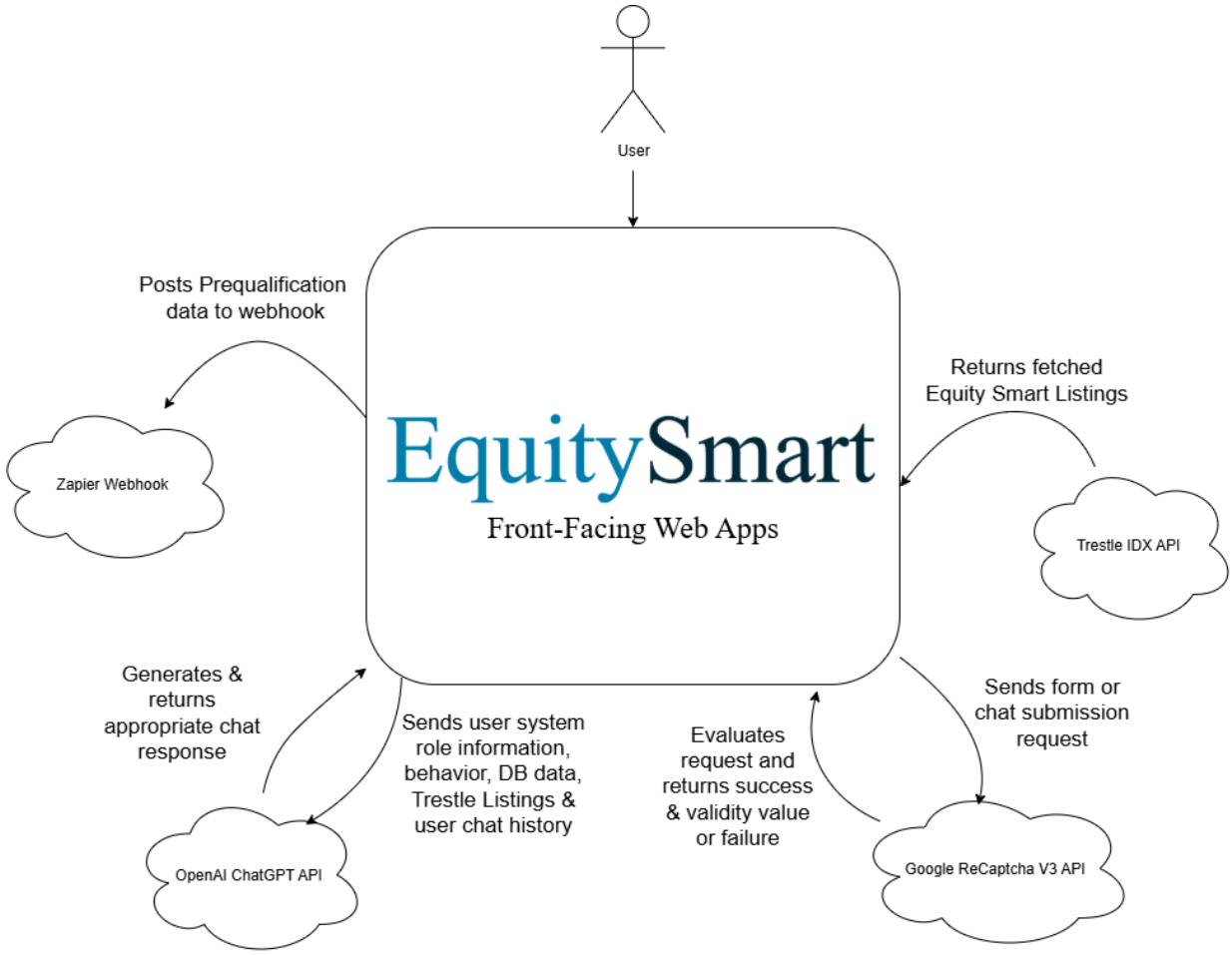


Figure 4: Data Flow Diagram (DFD-1)

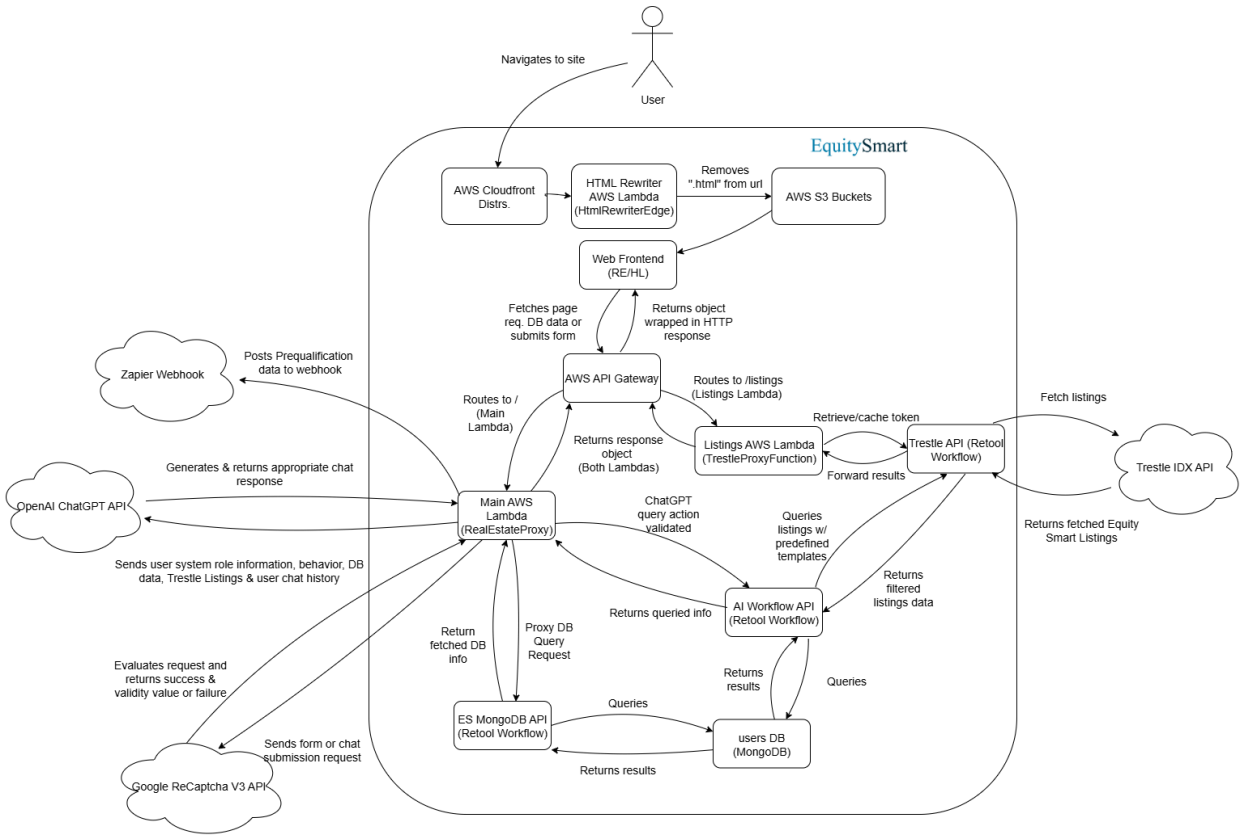


Figure 5: users Schema Diagram