# *AiVault - AI VISIBILITY PLATFORM FOR BUSINESSES*

**Submitted By:**

- Yuvraj Singh (1471)
- Santosh Kumar Sah (1441)
- Rahul Tanwar (1461)
- Ayush Paswan (1479)

**Course:** BSc Computer Science (H)

**Submission Date:**

**Submitted To:** Dr. Kamlesh kumar Raghuvanshi

# CERTIFICATE

This is to certify that **Yuvraj Singh (1471), Santosh Kumar Sah (1441), Rahul Tanwar (1461), and Ayush Paswan (1479)**, the students of B.Sc. (H) Computer Science,

**Ramanujan College, Delhi University**, have successfully completed the project titled **"AI VISIBILITY PLATFORM FOR BUSINESSES"** in their individual capacities under my supervision for the fulfillment of the V Semester practical examination.

I hereby acknowledge their sincere efforts and certify that the work carried out is original and done under my guidance.

Dr. Kamlesh kr. Raghuvanshi

# ACKNOWLEDGEMENT

We extend our profound gratitude to our project guide **Dr. Kamlesh Kumar Raghuvanshi** for his guidance, interest, and valuable suggestions throughout the course of this project. We feel honored and privileged to work under his supervision. His vast knowledge and constant support helped us overcome difficulties and complete the project successfully. This project would not have been possible without his mentorship, and we sincerely thank him for everything he has done for us.

# TABLE OF CONTENTS

# 1. Preliminary Analysis

## Existing System

In today's digital world, most small and medium-sized businesses rely on a mix of websites, directories, and random online platforms to share their service info.
This keeps them visible online, sure, but it also creates a scattered mess of inconsistent, outdated, and sometimes totally mismatched data. A lot of this info is entered manually, with no standard format or structure. Humans can usually make sense of it, but AI systems and automated tools really struggle.

As models like ChatGPT, Gemini, and Claude start shaping how people search for and choose services, this lack of structured, machine-readable info has become a big problem.
AI models don't read the web like people do—they rely on clear schemas like JSON-LD to understand business details properly. The issue is that small business owners don't have an easy or affordable way to check how their business actually shows up to these systems. Current SEO or structured-data tools are super technical, expensive, or split across multiple platforms, usually requiring a developer or expert to even get started.

Basically, business info all over the internet ends up inconsistent, hard for AI to understand, and annoying for owners to manage.
This means a lot of legit businesses become nearly invisible in AI-powered search—even if they provide great services—simply because their data isn't machine-friendly.

## Problem Statement

User behavior is shifting fast from traditional searches to AI assistants.
These assistants need clean, structured data to correctly identify, interpret, and recommend businesses. When a company's online info lacks proper schema formatting—like schema.org JSON-LD—its content can get ignored, misread, or skipped entirely in AI results.

Right now, the ways to fix this—manual SEO, adding structured-data markup, or trying to get large models to produce clean schemas—are expensive, slow, or just not realistic for most small business owners.
There isn't a simple, user-friendly platform that can:

- Help businesses generate structured data
- Show how an AI assistant "sees" their service

Because of this missing piece, tons of local businesses, freelancers, and small service providers lose visibility in AI-driven environments—even when they're actually doing excellent work.

## Scope of the Project

To tackle these issues, the proposed project—**AiVault**—will offer a clean, easy-to-use web platform that lets businesses create AI-readable profiles, preview how they appear to AI assistants, and get simple, practical feedback without needing technical knowledge.

The first version (Phase 1 / MVP) will focus on the most important features:

- Secure onboarding through Google OAuth, with an optional email/password login
- Automatic creation of schema.org-compliant JSON-LD
- Public, crawlable profile pages with embedded structured data
- A built-in simulation engine that shows how an AI assistant might respond to certain queries
- A visibility-scoring tool that checks data completeness and gives prioritized improvements
- A dashboard showing score history, core metrics, and missing schema fields
- A read-only REST API (OpenAPI-compliant) to fetch profiles and run simulations

Some advanced features won't be part of Phase 1—like live integrations with external AI APIs, multilingual or enterprise support, or deep analytics. Those will come later once the basic system is proven stable, useful, and actually valuable to businesses.

# 2. Detailed Analysis

## Proposed System

AiVault is a web platform built to make AI visibility super accessible for small businesses and non-technical owners. It helps them make sure their services are represented correctly in AI-powered systems by automating structured-data creation, simulating how they show up in AI-driven search, and offering simple, practical suggestions for improvement.

The whole idea is to keep things clean, simple, and transparent. A business owner can go from creating an account to publishing an AI-readable profile in just a few minutes. There's no coding, no SEO knowledge needed—just straightforward steps that follow modern standards like schema.org JSON-LD.

Some of AiVault's main characteristics include:

- **Simplicity:** The entire process from sign-up to publishing can be done in under five minutes.
- **Lightweight design:** The MVP keeps things efficient by skipping expensive external AI APIs and using internal simulation logic instead.
- **Vendor-neutral:** AiVault doesn't depend on any single search engine or AI assistant, which keeps it broadly compatible and neutral.

By automating structured-data creation and checking it for issues, AiVault fills the gap between normal websites and AI-driven discovery engines.

## Key User Flows

### 1. Business Owner Journey

A typical user experience in AiVault follows a simple, guided path:

- The business owner signs up using Google OAuth or a regular email-and-password account.
- They build their business profile and add services—either manually or by uploading a CSV.
- AiVault generates schema.org-compliant JSON-LD automatically.

- The user can then simulate how an AI assistant might describe or interpret their business.
- Finally, they check their visibility score, read the suggested improvements, and publish their optimized public profile.

### 2. Public and AI Assistant Access

Once published, the profiles are publicly crawlable.
When an AI assistant or search crawler accesses a page, AiVault returns an SEO-friendly HTML page with the correct JSON-LD embedded, making it readable for both humans and machines.

### 3. Admin Operations

Admins can monitor system health, check simulation logs, watch user activity, and manage usage quotas.
This helps keep everything reliable, secure, and fair for all users.

## Core System Components

AiVault is made up of several connected components that cover authentication, data creation, validation, analysis, and monitoring.

### 1. Authentication and Onboarding

Users can sign up through Google OAuth or with traditional email/password credentials.
A guided onboarding flow walks them through profile completion step by step, helping reduce friction, especially for first-time users.

### 2. Business Profile and Service Catalog

This module allows users to create, update, and remove business profiles and services.
It supports bulk CSV imports and separates fixed information (like addresses) from frequently changing info (like hours).

### 3. Schema Generator

This part automatically transforms business details into schema.org JSON-LD.
It includes built-in validation to catch missing fields or errors, making sure the structured data is always compliant and ready for publishing.

### 4. Public Profile Renderer

Once a profile is generated, the system makes it publicly available as an SEO-friendly HTML page.
The JSON-LD is embedded directly, enabling AI assistants and search engines to index the information properly.

### 5. Visibility Simulation Engine

This is the standout feature of AiVault. It reads natural-language queries and simulates how an AI assistant might respond.
It calculates a visibility score and produces clear recommendations—like missing contact info, incomplete hours, or weak service descriptions.

### 6. Dashboard and Reporting

Users can track their performance over time, including visibility scores, schema coverage, and usage stats.
The dashboard also highlights areas that need attention and visualizes progress.

### 7. REST API

AiVault includes a secure, OpenAPI-compliant REST API that lets external apps fetch business data, run simulations, or retrieve scores.
This makes it easy to integrate with other systems or tools.

## High-Level Data Flow

The flow starts when a business owner enters their profile and service details through the dashboard or uploads everything at once using a CSV file. After the data is submitted, the system automatically transforms it into clean, schema.org-compliant JSON-LD without the user needing to deal with any technical formatting. Once that's done, a template-based engine runs AI-style query simulations to check how an AI assistant might interpret the business. Finally, the system outputs crawlable HTML, a visibility score, validation details, and updates the dashboard metrics. Overall, this smooth workflow helps even non-technical users maintain an AI-friendly business profile without stressing over standards or formatting rules.

# 3. Requirements

## Functional Requirements

### A. Authentication and Onboarding

During onboarding, users can log in using Google OAuth or a regular email-and-password setup. The platform must also store important business details like the owner's name, contact information, address, and operating hours. All of this needs to be accessible and easy to update so the business data stays accurate.

### B. Business Profile and Service Management

Users should be able to manage their services by adding, updating, or removing items such as service names, descriptions, tags, and pricing. For businesses with a large catalog, the system also supports bulk CSV uploads, which makes organizing services way faster than entering everything manually.

### C. Schema Generation and Exposure

Once business data is saved, the system should automatically generate valid JSON-LD based on the user's inputs. Each business profile must also be publicly crawable, with the structured data embedded directly into the page. In addition to that, the REST API needs to offer endpoints for retrieving business profiles and starting visibility simulations, and all of this should be documented clearly through OpenAPI.

### D. AI Plugin and Simulation

To support integrations with AI systems, AiVault needs to expose an **ai-plugin.json** file. This file describes what endpoints exist and how external assistants or tools can access them. It basically acts like a guide for AI platforms trying to connect with AiVault.

### E. Dashboard and Analytics

The platform should also give users insights into their usage, including API statistics and the total number of visibility simulations they've run. This helps them understand how often their data is being tested and what areas might still need improvement.

# Non-Functional Requirements

## Performance

The system should be fast and responsive. Ideally, at least 95% of all API requests should finish in under 200 milliseconds, and public JSON-LD profile pages should load in under a second. This keeps things smooth for both users and crawlers.

## Security

All sensitive information has to be encrypted, using AES-256 for data at rest and TLS 1.2 or higher for data in transit. Write access to the API should only be available through secure authentication methods like OAuth or API keys. On top of that, the platform must enforce role-based access so that business owners, admins, and public users each have the correct permissions.

## Usability

New users should be able to complete onboarding and publish their AI-friendly profile within five minutes of signing up. The interface should feel simple and predictable so people don't get stuck or confused.

## Maintainability and Scalability

The system's codebase needs to remain modular, with clean separation for core areas like authentication, the API, schema generation, and UI components. This structure makes the platform easier to maintain and lets it scale as more users and businesses join.

# 4. Modelling

## Model Used

AiVault follows an Iterative Incremental Model based on Agile ideas. This model helps the team release working features quickly, test things early, and gather feedback before moving too far ahead. It works well with modular development too. For example, frontend, backend, and data engineers can all work at the same time without getting in each other's way. It also encourages continuous testing and improvement, which is important when you're building something that depends on real user behavior.

**Phase 1 Sprint Plan**

| Sprint | Focus Area | Status |
|---|---|---|
| Sprint 0 | Project setup, repository creation, CI/CD pipeline, and Google OAuth integration | Done |
| Sprint 1 | Onboarding flow and business profile CRUD functionality | Done |
| Sprint 2 | Service catalog management and CSV import | CSV droped |
| Sprint 3 | JSON-LD generator and public profile rendering | Done |
| Sprint 4 | Simulation engine and scoring logic | Pending |
| Sprint 5 | Dashboard implementation and OpenAPI documentation | Done |
| Sprint 6 | Final testing, demo, and submission | Pending |

Each sprint spans one week, allowing fast iteration while maintaining stable progress. A short buffer at the end ensures sufficient time for testing and demonstration.

# 5. Feasibility Study

## Economic Feasibility

From an economic perspective, AiVault's MVP is quite affordable to build and run. It only needs lightweight cloud hosting and uses open-source tools wherever possible, which keeps early costs really low. There are no AI API charges in the first version because simulations rely on internal logic. Later on, the platform can shift into a freemium model, where the core features stay free but paid tiers unlock more advanced analytics or higher simulation limits.

## Technical Feasibility

Technically, the system is built on reliable, widely used technologies like React, Node.js, and PostgreSQL, which makes it easy to maintain and scale as usage grows. JSON-LD validation will be handled using established open-source libraries to avoid reinventing the wheel. All public-facing content will be optimized for CDN delivery, helping the platform stay fast even as more profiles get published.

## Operational Feasibility

Operationally, AiVault is designed to be as simple as possible. The guided onboarding and clean interface allow non-technical users to publish AI-ready business profiles without needing any training. The ongoing workload is fairly small—only a light admin team is required for monitoring and basic support. The project's early success can be measured using practical metrics such as how many profiles get published, how often simulations are run, and how much the average visibility score improves over time.

# 6. Risks and Management

One of the main risks for AiVault is that the simulation engine might not fully match how real AI assistants behave. Since every model has its own quirks, there's always a chance the results won't be perfectly aligned. The plan to handle this is to keep refining the simulation templates over time, using real user feedback and real-world examples to make the system more accurate.

Another challenge is incomplete business data. When users skip important fields—like contact details, hours, or service descriptions—their visibility scores naturally drop. To reduce this problem, the platform will include inline validation and smart field suggestions that guide users as they fill out their profiles, helping them avoid missing anything important.

There's also the risk of low adoption, especially early on. Some users might hesitate to try yet another business tool or may not immediately understand the value of AI visibility. To make the platform more appealing, free tiers and simple "before vs after" previews will be offered so users can see the impact right away.

Scalability can also become an issue if traffic grows quickly. High load might slow down simulations or cause delays in profile rendering. To prevent this, the system will rely on caching and horizontal scaling so performance stays consistent even under heavy usage.

Finally, there's the risk of security issues or misuse, such as unauthorized access or scraping of public data. To address this, the platform will enforce API keys, apply rate limits, and maintain proper access auditing to ensure user data stays protected and the system isn't abused.

# 7. Designing

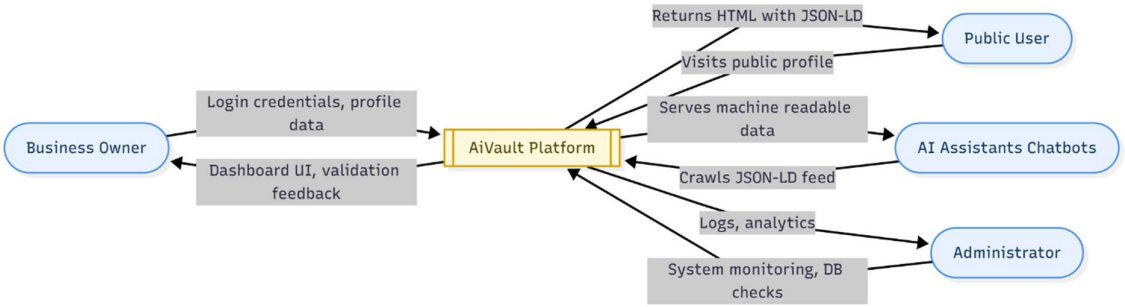
## 7.1 Context Diagram (Level 0)

The Level 0 context diagram shows AiVault at the center and all the main groups that interact with it from the outside. Business owners are the primary users—they log in, fill out their business details, add services, upload photos, create coupons, and update their operational information.

Public users and AI systems access the business pages that AiVault publishes, and they mainly rely on the JSON-LD feeds generated from the business data. This structured information helps AI systems understand the business more clearly.

Administrators also interact with AiVault, mostly for monitoring the system, checking data consistency, and reviewing activity if needed.

Overall, this diagram presents a clear picture of how information flows in and out of the AiVault platform. It highlights the inputs business owners provide, the structured outputs public users retrieve, and the supporting role of admins who help keep the system running smoothly.

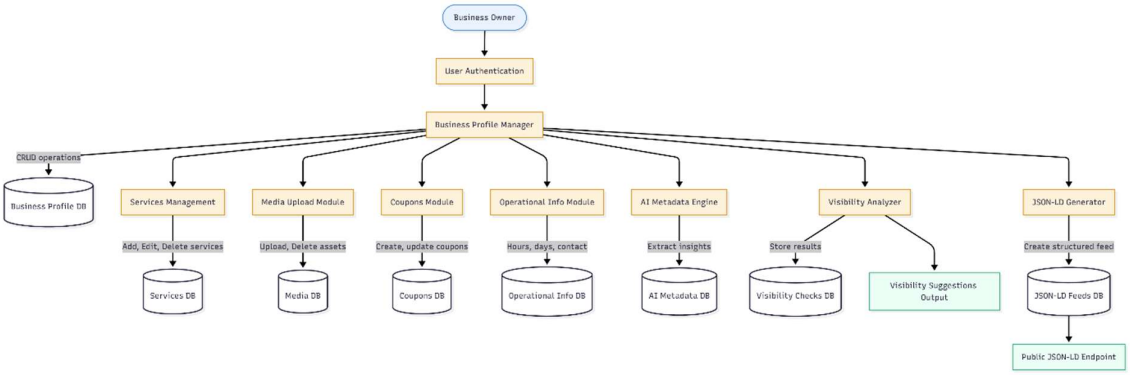



## 7.2 Level 1 DFD — Major Processes

The Level 1 data flow diagram breaks AiVault into its major functional modules and shows how business information moves through the system. Once a user signs in, they land on the dashboard where they can manage every part of their business profile. From here, the user can update their main business details, add services, upload media files, create coupons, and fill out their operational information.

Each of these modules connects to its own backend endpoint and database table, keeping the system clean and modular. AiVault also has a simple AI metadata generator that extracts basic insights from the business description, along with a visibility checker that looks for missing or incomplete information and produces suggestions.

All this data eventually connects to the JSON-LD generator, which creates the structured schema that gets exposed publicly. The Level 1 diagram basically shows how the user's inputs flow into different internal processes and how these processes feed into larger outputs like visibility suggestions and JSON-LD feeds.

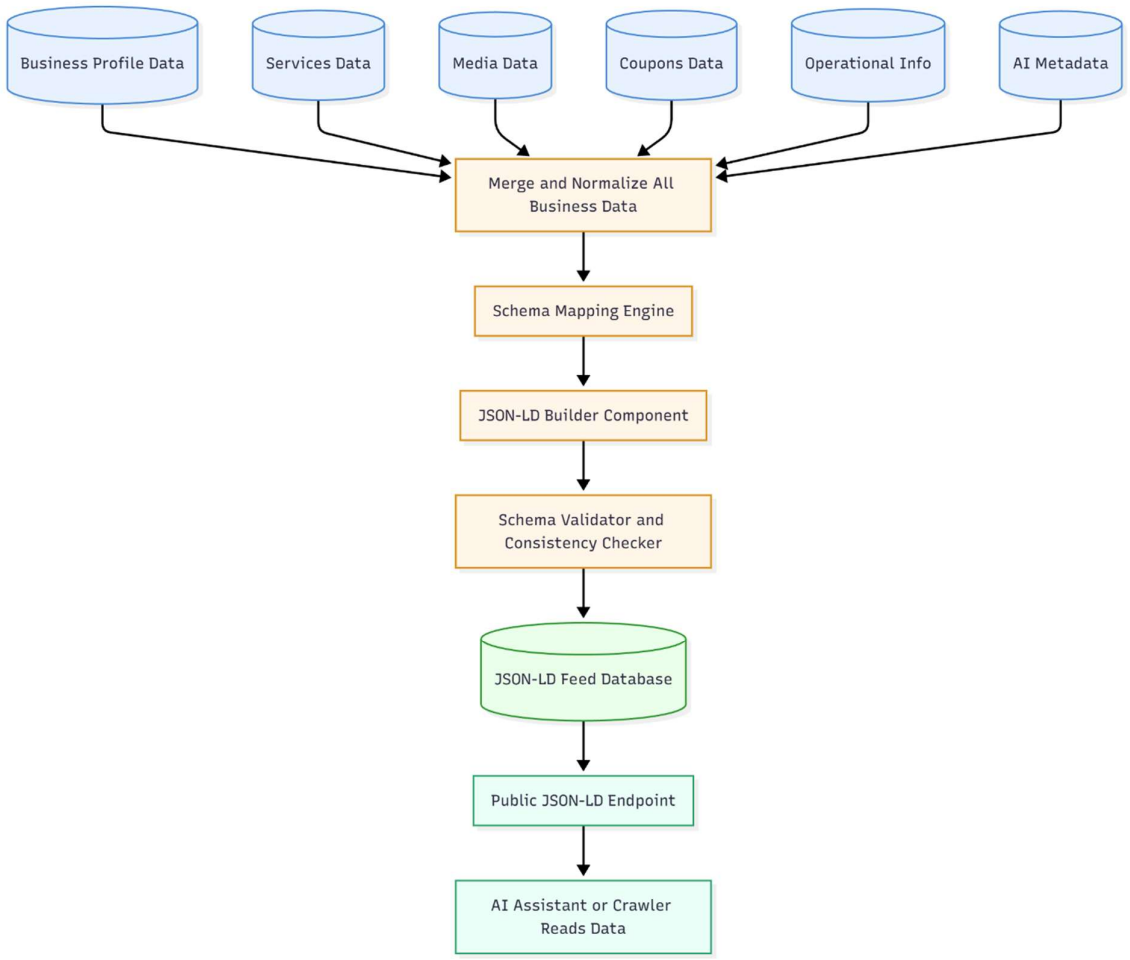



## 7.3 Level 2 DFD — Simulation Engine Detail

The Level 2 diagram focuses specifically on how AiVault creates the JSON-LD feed using all the business information stored in the system. It starts by collecting data from multiple sources: the business profile, its list of services, uploaded media, any active coupons, operational details, and even the AI metadata.

All this information is first merged and cleaned so it follows a consistent structure. Then the system maps each piece of data to a schema.org format depending on the business type. After mapping, the JSON-LD builder assembles the final structured document, making sure every required field is included.

Before storing the JSON-LD, the system runs a basic validation check to ensure the format is correct and the data is complete. Once validated, the JSON-LD feed is saved and published as part of the business's public profile page. This diagram highlights how multiple internal components work together to produce the AI-readable output that public users and crawlers consume.




## 7.4 Data Dictionary

The Data Dictionary defines the main data entities used throughout AiVault, ensuring consistency in naming, data types, and meaning across the system.

**BusinessProfile** represents the core metadata of a published business listing. Each profile includes a unique `business_id` (UUID), an `owner_id` referencing the user who owns the profile, and standard fields like `name`, `description`, `address`, `contact_phone`, and `contact_email`. It also includes `categories` as an array of category names, `opening_hours` as an array of day-wise open and close times, along with `timezone`, `website`, published status, version, and timestamps for creation and last update.

**ServiceRecord** defines each service offered by a business. Each record includes a `service_id` and links to a `business_id`. The service has a `title`, `short_description`, `full_description`, `price` (object containing type and amount), `currency`, `tags`, and `availability`, as well as timestamps for creation and last update.

**User** stores authentication and identity information in a relational/auth store. Key fields include `user_id`, `email`, `name`, `auth_provider`, `password_hash`, roles, active status, and timestamps for creation and last login.

**JSONLD_Feed** contains the generated structured-data metadata for each business profile. It includes `feed_id`, `business_id`, schema version, `payload_url` (object storage pointer), `payload_hash` for version control, public URL, generation timestamp, and status.

**SimulationRequest:** logs every simulation initiated by a user. It includes `sim_id`, `requester_id`, `business_id`, `service_id`, `query_text`, simulation mode, status, start and finish timestamps, and runtime in milliseconds.

**SimulationResult** :stores the output of a simulation job, including `sim_id`, visibility score, simulated response text, rationale, missing fields, improvement suggestions, and creation timestamp.

**BillingRecord:** tracks billing, credit usage, and invoicing. Fields include `billing_id`, linked `business_id`, billing period (`period_start` and `period_end`), number of simulations, credits used, amount due, payment status, and invoice URL.

## 7.5 Process Specifications

Each major process in AiVault follows a clear input-validation-output workflow to ensure reliability and consistency.

**Process 2 – Profile and Service Management** accepts `BusinessProfile` and `ServiceRecord` data entered manually or via CSV upload. Validation rules enforce mandatory fields such as business name and contact information, a complete address with street and city, tag lists limited to ten entries with no duplicates, titles capped at 150 characters, and properly formatted prices. Actions include creating, updating, or deleting records, enforcing ownership through authentication, incrementing the profile version on each change, and triggering automatic JSON-LD regeneration. Successful operations return the resource ID and validation report, while failures provide detailed error messages or partial success summaries.

**Process 3 – Schema Generation and Publication** transforms business profiles and service records into machine-readable JSON-LD. Relevant fields are mapped to schema.org properties such as `name`, `description`, `openingHoursSpecification`, and `Service` or `Offer`. Validation ensures required fields exist and comply with schema.org type definitions. Actions include generating JSON-LD payloads, computing payload hashes for version control, storing payloads in object storage, updating metadata in the database, and

publishing the public profile URL via CDN. Outputs include the public profile link, a validation report, and version ID. If validation fails, the profile is marked as a draft, and the business owner is notified.

**Process 4 – Simulation Engine and Scoring** begins with a `SimulationRequest` from a verified user. The engine checks the user's available quota (linked to `BillingRecord`), selects a template based on business category and tags, executes the simulation, and generates an AI-style response. Visibility scores are calculated based on coverage, formatting, and semantic completeness. Missing or weak fields are identified and ranked by priority. Results are stored, the dashboard is updated, and users are notified. Outputs include the visibility score, simulated assistant response, and prioritized improvement suggestions. Exceptions such as template errors are logged with a failed status, and transient errors are retried once before aborting.

---

## 7.6 Data Store Specifications

AiVault uses PostgreSQL as the single database solution for all data storage, leveraging its relational and JSONB capabilities to handle structured, semi-structured, and time-series data efficiently.

**Users** are stored as relational records. Each user has a `user_id` primary key and a unique index on `email`. Sensitive fields, such as passwords and emails, are encrypted both at rest and in transit.

**BusinessProfiles** are stored in PostgreSQL using JSONB columns to represent flexible, document-like structures. Each profile is indexed by `business_id` (primary key) and `owner_id`. Text indexes on `name` and `categories` enable efficient searching. Historical versions of profiles are kept in a versioned table to allow rollback and auditing.

**Services** are stored in a separate table, also using JSONB where needed for flexible fields. Indexes include `service_id` and `business_id`, with additional text indexes on `title` and `tags`. Queries are optimized for business-based lookups and time-based filtering.

**JSONLD_Feed** payloads are stored directly in PostgreSQL as JSONB fields, with associated metadata for version control and CDN-backed public URLs. Republished payloads automatically trigger cache invalidation.

**Simulations** are recorded in PostgreSQL tables with timestamped entries to support time-series analysis. TTL-style retention policies are implemented via scheduled cleanup jobs or partitioned tables.

**BillingRecords** are fully transactional in PostgreSQL, supporting integrity constraints, audit logging, and invoice tracking. All financial operations are consistent and recoverable.

## 7.7 Validation Rules

Validation ensures that all data in AiVault meets schema, format, and logical consistency requirements.

**Business Profile Validation** requires business names to be 3–200 characters. At least one contact method (email or phone) is mandatory, with emails validated via standard regex and phone numbers following E.164 formatting. Addresses must include street and city, and postal codes are mandatory where applicable. Opening hours require a valid timezone and ISO 8601 time format.

**Service Validation** enforces titles of 3–150 characters and descriptions between 20–300 characters. A maximum of 10 tags is allowed per service, with duplicates prohibited. Prices must include both a numeric amount and a currency code.

**CSV Import Validation** checks each row individually. Invalid rows return explicit error messages referencing line numbers. Both fail-all (atomic) and partial-accept batch modes are supported to accommodate different user needs.

**JSON-LD Validation** requires `@context` and `@type` fields and should include `contactPoint`, `openingHoursSpecification`, and service offerings. All JSON-LD payloads are validated automatically before publication to ensure compliance with schema.org standards.

# 8 Project Planning

## 8.1 System Architecture

AiVault is built using a **modular, service-oriented architecture (SOA)** that separates responsibilities across layers while ensuring smooth data flow between the frontend, backend, and database. This structure supports scalability, maintainability, and future extensibility.

The **frontend layer** is developed in React.js, providing a responsive, interactive user interface. Users interact with the system through RESTful APIs, fetching business profiles, services, and AI-generated visibility results. Components are organized by feature, such as Dashboard, Profile Editor, and Simulation Screen, ensuring modularity. Form validation and user interactions are handled locally where possible to provide immediate feedback and reduce unnecessary server calls.

The **backend layer** is powered by FastAPI, chosen for its asynchronous capabilities, high performance, and simplicity. Each functional area, including users, business, JSON-LD, and operational_info, is separated into dedicated routers to maintain a clean, scalable codebase. Public and authenticated endpoints are provided, with access controlled via JSON Web Tokens (JWT) and API keys. Proper error handling and logging are implemented for traceability, and endpoints are structured consistently for predictable behavior.

AiVault uses **PostgreSQL** as the primary database for all relational data, including users, business profiles, services, JSON-LD feeds, and billing records. SQLAlchemy ORM is used for schema management, migrations, and query abstraction. JSONB columns allow storage of semi-structured data such as JSON-LD payloads, enabling future flexibility without needing a separate document database.

The **integration layer** handles connections to external services, such as Google OAuth for authentication. Future extensions will also allow real-time schema.org compliance validation, third-party data verification, and additional AI-driven analytics.

This layered architecture ensures that each subsystem can evolve independently, supporting advanced features, analytics, and future growth.

## 8.2 System Design

The system design focuses on balancing simplicity, usability, and technical depth, with each component planned for reusability and extensibility.

The **user interface** is clean and minimal, enabling users to perform key actions such as creating profiles, managing services, running simulations, and reviewing analytics. The Dashboard summarizes profile health, visibility scores, and improvement suggestions. The Profile Editor provides a guided form for step-by-step data collection, while the Simulation Screen displays AI-generated results with explanations. Public-facing profile pages are optimized for search engines and AI crawlers, embedding JSON-LD markup directly into the HTML.

The **backend design** ensures consistency and security across APIs. All endpoints follow REST principles and intuitive structures, such as `/users/register`, `/users/login`, `/business/{id}`, `/jsonld/`, and `/simulation/`. Data validation is enforced via Pydantic schemas, communication is secured with HTTPS, and robust logging supports monitoring and debugging. The backend is fully testable with frameworks like pytest, supporting continuous integration and reliable deployment.

The **database design** supports efficient querying and minimal redundancy. Core tables include Users, BusinessProfiles, Services, JsonLDFeed, OperationalInfo, and AiMetadata for extracted insights and labels. Relationships are clearly defined: one-to-many between users and businesses, one-to-many between businesses and services, and one-to-one between each business and its JSON-LD feed. This structure ensures referential integrity, optimized queries, and easier analytics aggregation.

**Security considerations** include encryption of sensitive fields using AES-256, JWT-based authentication for API access, role-based access control to separate normal users from administrators, and rate limiting to prevent abuse of public APIs.

## 8.3 System Implementation

The development environment uses **React.js with Vite and Tailwind CSS** for the frontend, **FastAPI with Python 3.11+ and SQLAlchemy ORM** for the backend, and **PostgreSQL** as the database for all structured data. Git and GitHub manage version control, while testing uses Pytest for the backend and Jest/React Testing Library for frontend components. Optional deployment leverages Docker containers on platforms like Render or AWS EC2.

Key features include a secure **authentication system** that supports OAuth and email/password login with hashed credentials, automatic **JSON-LD schema generation and validation** for each business profile and service, and a **simulation engine** that generates visibility scores by analyzing schema completeness and relevance. Dashboard analytics are powered through API endpoints, with visualizations rendered using Recharts for real-time insight into profile performance.

This consistent architecture and design ensure that AiVault provides a scalable, secure, and user-friendly experience while maintaining technical rigor for data validation, simulation, and structured data publication.

# 9. System Testing

Comprehensive testing ensures reliability, correctness, and performance across all AiVault modules. The testing strategy combines automated and manual approaches, covering every layer of the system from individual functions to full user workflows.

**Unit Testing** validates each function and endpoint independently, such as `create_business_profile`, ensuring that core logic works as expected. Pytest is used extensively to automate these tests.

**Integration Testing** confirms that the backend interacts correctly with PostgreSQL, verifying proper data flow, relationships, and JSON-LD schema generation. This ensures that adding or updating business profiles, services, and JSON-LD feeds behaves consistently across modules.

**Functional Testing** evaluates end-to-end features including user registration, business and service management, JSON-LD generation, and AI simulation scoring, confirming that the system meets functional requirements.

**Performance Testing** measures API response times and identifies any slow queries or potential bottlenecks in the PostgreSQL database or backend logic.

**User Acceptance Testing (UAT)** is conducted with business-owner workflows to ensure the system is intuitive, complete, and meets the intended use cases.

## 9.1 Sample Test Cases

| Test Case ID | Description | Input | Expected Output | Status |
|---|---|---|---|---|
| TC-01 | User Registration | Valid email, password | Returns `user_id` and JWT token | Passed |
| TC-02 | Add Business Profile | Valid profile data | Profile created, returns `business_id` | Passed |
| TC-03 | Generate JSON-LD | Valid service data | JSON-LD object created and validated | Passed |
| TC-04 | Invalid Schema Data | Missing required field | Returns detailed error message | Passed |

| | | | | |
|---|---|---|---|---|
| TC-05 | Fetch Visibility Score | Valid business ID | Returns visibility score object | Passed |

This testing framework ensures that each module is validated independently while maintaining the integrity of the overall system workflow.

## 9.2 Project Plan and Timeline

AiVault's development follows an **Agile iterative model**, allowing for incremental progress, continuous feedback, and improvements after each sprint. Each phase builds on the previous, ensuring minimal rework and smooth integration.

| Phase | Deliverables | Timeline | Status |
|---|---|---|---|
| Phase 1 | Requirement gathering and system design | Week 1 | Completed |
| Phase 2 | Frontend setup and authentication integration | Week 2 | Completed |
| Phase 3 | Backend setup with FastAPI | Week 3 | Completed |
| Phase 4 | Business and Service CRUD APIs | Week 4 | Completed |
| Phase 5 | JSON-LD generation module | Week 5 | Completed |
| Phase 6 | Simulation engine and scoring system | Week 6 | Dropped |
| Phase 7 | Dashboard | Week 7 | Pending |
| Phase 8 | Final testing, optimization, and documentation | Week 8 | Done |

This timeline ensures a structured approach to development while maintaining flexibility to adapt based on testing outcomes and stakeholder feedback.

## 9.3 Future Enhancements

While AiVault's MVP establishes a robust foundation for AI-driven visibility, several enhancements are planned for future iterations to expand functionality, automation, and insights:

**AI-Assisted Schema Generation**
An integrated large language model (LLM) will analyze website content or uploaded data and automatically generate fully structured JSON-LD schemas, reducing manual effort and improving consistency.

**Marketplace Integration**
Business owners will be able to publish AI-optimized profiles directly to external

marketplaces, virtual assistants, and third-party platforms, broadening visibility and potential reach.

**Custom AI Visibility Reports**

Advanced analytics dashboards will provide deeper insights into performance, competitor benchmarking, and actionable recommendations to optimize AI presence.

**Mobile App**

A companion mobile application will enable real-time notifications, instant validation checks, score updates, and interactive reporting, keeping users informed on the go.

**Blockchain Integration**

Optionally, validation proofs and schema publication logs can be stored on a blockchain, providing transparency, immutability, and trust for critical business data.

These planned features maintain the same principles of automation, usability, and reliability while allowing AiVault to evolve into a comprehensive AI visibility platform.

# 10. Conclusion

AiVault delivers a modern, lightweight solution for a pressing need: ensuring small and medium businesses remain visible in AI-driven ecosystems. By combining structured data management, automated schema generation, and simulation-based scoring, it empowers non-technical users to achieve AI readiness without relying on specialized developers or SEO experts.

The platform's modular architecture, rigorous validation rules, and intuitive interfaces make it practical today and adaptable tomorrow. Designed for scalability and future integrations, AiVault can incorporate emerging AI tools, third-party marketplaces, and advanced analytics to continually enhance business visibility in a data-driven, intelligent manner.

# 11. References

1. *Schema.org Documentation*. "Structured Data for LocalBusiness and related schema types." Retrieved from https://schema.org
2. FastAPI Official Documentation. "Building APIs with FastAPI, Models, Routing, and Validation." Retrieved from https://fastapi.tiangolo.com
3. ReactJS Official Documentation. "Building component-based user interfaces." Retrieved from https://react.dev
4. PostgreSQL Documentation. "Relational database concepts, table schema design, and querying." Retrieved from https://postgresql.org
5. Google Developers. "JSON-LD for structured data and SEO basics." Retrieved from https://developers.google.com/search/docs/appearance/structured-data
6. W3C. "JSON-LD 1.1 Specification." Retrieved from https://www.w3.org/TR/json-ld11