**1. Introduction**

**1.a Purpose**

This document defines the software requirements for the **Healthcare Insurance Competitor Analytics & Customer Insights Platform**. It specifies what the system must do to ingest, clean, model, and analyze competitor and market/customer data to drive revenue growth via targeted offers, subscriber royalties, and behavioral insights. The SRS will be the baseline for design, implementation, testing, and acceptance.

**1.b Intended Audience and Use**

* **Product Owner / Business Stakeholders (Marketing, Pricing, Finance):** Validate business goals and analytical use cases; accept delivered outcomes.
* **Data Engineering Team (Databricks/EMR, PySpark, Redshift):** Implement pipelines, storage, orchestration, and monitoring per these requirements.
* **Data Analysts / Data Scientists:** Consume curated Redshift data and published result tables; create visualizations and ad‑hoc insights.
* **QA/Test Engineers:** Derive test cases and acceptance criteria; validate data quality and functional outputs.
* **Security/Compliance:** Verify controls for PHI/PII, HIPAA-aligned safeguards, access, and auditability.
* **DevOps/Platform:** Provision environments, CI/CD with GitHub, and integrate with Jira for delivery tracking.

**1.c Product Scope**

The platform will:

* Consolidate **competitor/customer** datasets (scraped and third‑party) in **AWS S3**, cleanse them with **PySpark** on **Databricks or EMR Studio**, and persist curated data in **Amazon Redshift**.
* Produce **repeatable analytical outputs** (one Redshift table per use case) to answer specified business questions (e.g., max claims by disease, subgroup adoption, hospital utilization, rejection rates, profitability, etc.).
* Enable **targeted offers** and **royalties** through measurable insights, improving conversion and revenue while honoring privacy/security.
* Integrate with **GitHub** (versioning/CI) and **Jira** (two-week sprint plan, user stories, and tasks).

**1.d Definitions and Acronyms**

* **SRS:** Software Requirements Specification
* **PHI/PII:** Protected Health Information / Personally Identifiable Information
* **HIPAA:** Health Insurance Portability and Accountability Act (US)
* **ETL/ELT:** Extract‑Transform‑Load / Extract‑Load‑Transform
* **S3:** Amazon Simple Storage Service
* **Redshift:** Amazon Redshift data warehouse
* **EMR Studio:** Managed IDE for EMR
* **Databricks:** Unified analytics platform; Databricks Community Edition is acceptable for dev/testing
* **KMS:** AWS Key Management Service for encryption keys
* **IAM:** AWS Identity and Access Management
* **CDC:** Change Data Capture (if introduced later)
* **Project-Output:** Redshift schema where each use case’s result table will be materialized

**2. Overall Description**

**2.a User Needs**

* **Marketing & Growth:** Identify high‑value cohorts, most-subscribed groups/subgroups, cities with the most claims, and disease trends to personalize offers and optimize campaign spend.
* **Pricing & Finance:** Understand claim rejection counts, profitability by policy group (Government vs Private), average premium behavior, and royalties for loyal subscribers.
* **Operations & Provider Relations:** Learn which hospitals serve the most patients; monitor procedure trends (e.g., knee surgeries) and cashless utilization.
* **Compliance & Security:** Maintain strict controls over PHI/PII; provide audit trails and least‑privilege access.
* **Data & Analytics:** Reliable, deduplicated, schema‑conformed tables with predictable refresh and clearly defined keys.

**2.b Assumptions and Dependencies**

**Assumptions**

* Source data arrives as files (CSV/JSON/Parquet) into s3://<bucket>/input-data/ with stable or documented schemas.
* Minimal data dictionary is provided (or inferred) for **Patients**, **Subscriber**, **Claims**, **Group\_Subgroup**; additional reference dimensions (Hospitals, Cities, Procedures, Policies) may be included or synthesized.
* For development and demo, Databricks Community Edition is acceptable; production targets **Databricks on AWS** or **AWS EMR**.
* Redshift is available with appropriate RA3 capacity and KMS encryption.
* GitHub repo(s) exist and access is granted for CI.
* Jira project is configured for the two‑week sprint and story tracking.

**Dependencies**

* AWS services (S3, Redshift, IAM, KMS, optional Glue/Secrets Manager), Databricks/EMR runtime, PySpark 3.x, Python 3.10+.
* Network/VPC endpoints or gateways for secure service access.
* Third‑party data contracts (licensing, refresh cadence, permissible use) and scraping compliance.

**3. System Features and Requirements**

**3.a Functional Requirements**

*All functional requirements (FR) reference curated data persisted in Redshift and results published in Project-Output schema. Each FR lists an output table name and acceptance criteria.*

**Data Foundation**

* **FR‑00 Ingestion & Staging**
  + **Description:** Land raw files to S3 input-data/; catalog folder structure by dataset and load‑date.
  + **Acceptance:** Files discoverable; metadata (source, load date) captured in control table.
* **FR‑01 Data Cleaning (Patients, Subscriber, Claims, Group\_Subgroup)**
  + **Description:** Null scan, per‑column null counts, NA imputation where specified, duplicate detection/removal, type casting, referential integrity checks.
  + **Acceptance:** Data quality report produced per run; curated tables written to Redshift with row counts and DQ metrics stored.
* **FR‑02 Curated Model Load (Redshift)**
  + **Description:** Load cleansed datasets into Redshift in a conformed schema (see 3.b.iv Software—Data Model Overview).
  + **Acceptance:** PK/FK constraints enforced where supported; surrogate keys generated as needed; load is idempotent.

**Analytical Use Cases (each creates a result table in Project-Output)**

* **FR‑10 Disease with Maximum Number of Claims** → Project-Output.max\_claims\_by\_disease
  + Groups claims by disease; ranks by count.
  + **Acceptance:** Top disease and counts reflect underlying curated Claims; ties handled deterministically.
* **FR‑11 Subscribers < 30 with Any Subgroup** → Project-Output.subscribers\_under30\_with\_subgroups
  + Filters subscribers by age < 30 and membership in any subgroup.
  + **Acceptance:** Age computed from DOB or provided age; subgroup membership validated via joins.
* **FR‑12 Group with Maximum Subgroups** → Project-Output.group\_with\_max\_subgroups
  + Aggregates subgroups by group; returns max.
  + **Acceptance:** Includes group id/name, subgroup count, rank.
* **FR‑13 Hospital Serving Most Patients** → Project-Output.top\_hospitals\_by\_patients
  + Counts distinct patients served per hospital.
  + **Acceptance:** Includes hospital id/name, patient count, time window if applicable.
* **FR‑14 Most‑Subscribed Subgroups** → Project-Output.top\_subgroups\_by\_subscriptions
  + Counts how many times each subgroup is subscribed.
  + **Acceptance:** Includes subgroup id/name, subscription count, share.
* **FR‑15 Total Number of Rejected Claims** → Project-Output.claims\_rejected\_summary
  + Filters Claims by status = Rejected; provides totals and rates.
  + **Acceptance:** Includes total rejected, total claims, rejection rate.
* **FR‑16 City with Most Claims** → Project-Output.top\_cities\_by\_claims
  + Aggregates claims by city of service or subscriber city.
  + **Acceptance:** Clear definition of city source column; includes counts and rank.
* **FR‑17 Government vs Private Policy Popularity** → Project-Output.policy\_group\_popularity
  + Splits subscriptions by **Government** vs **Private** groups; counts and shares.
  + **Acceptance:** Totals reconcile with overall subscription count.
* **FR‑18 Average Monthly Premium Paid** → Project-Output.avg\_monthly\_premium
  + Computes subscriber‑level monthly average premium and overall averages.
  + **Acceptance:** Handles missing months; documents calculation basis (billed vs paid).
* **FR‑19 Most Profitable Group** → Project-Output.group\_profitability
  + Profit = Premiums (earned) − Claims (paid) − (optional admin costs if provided/assumed).
  + **Acceptance:** Profit metric defined and consistent; group ranking reproducible.
* **FR‑20 Patients < 18 Admitted for Cancer** → Project-Output.patients\_under18\_cancer
  + Filters Patients/Claims by age < 18 with diagnosis category = Cancer.
  + **Acceptance:** Returns patient identifiers (tokenized if PHI), admission dates, hospital.
* **FR‑21 Cashless Insurance with Total Charges ≥ 50,000** → Project-Output.cashless\_high\_charges
  + Filters by is\_cashless = true and total\_charges >= 50000 (currency documented).
  + **Acceptance:** Includes policy/subscriber linkage and claim identifiers.
* **FR‑22 Female Patients > 40 with Knee Surgery (Past Year)** → Project-Output.female\_over40\_knee\_past\_year
  + Filters by gender = F, age > 40, procedure\_code in Knee Surgery set, and service\_date within last 12 months.
  + **Acceptance:** Procedure dictionary documented; time window parameterized.

**Enablement & Governance**

* **FR‑30 Parameterized Orchestration** (Databricks/EMR)
  + Pipelines can run full or incremental; parameters control time windows and outputs.
  + **Acceptance:** Jobs callable with documented parameters; successful/failed run statuses logged.
* **FR‑31 Version Control & CI (GitHub)**
  + Source code, docs, and IaC (if any) committed; PR checks run unit & DQ tests.
  + **Acceptance:** Tagged releases; branch protection; pipeline badges (optional).
* **FR‑32 Tracking & Delivery (Jira)**
  + Two‑week sprint with epics/user stories/tasks for docs, implementation, and testing.
  + **Acceptance:** Stories linked to PRs; burndown visible.
* **FR‑33 Security & Privacy Controls**
  + PHI/PII masked/tokenized in analytics outputs; least‑privilege IAM; encryption in transit/at rest.
  + **Acceptance:** Access audit log; secrets stored outside code.

**3.b External Interface Requirements**

**3.b.i User Interfaces**

* **Engineering UI:** Databricks/EMR Studio notebooks and jobs; CLI tooling for S3/Redshift.
* **Analytics UI:** Redshift SQL clients; optional Databricks SQL/visualizations for snapshots; clear table/column documentation.
* **Documentation:** Markdown/Docs in GitHub; Jira dashboards for status.

**3.b.ii Hardware Interfaces**

* **AWS Managed:** Redshift RA3 nodes (size to volume), S3 Standard storage, EMR/Databricks clusters with autoscaling. No physical hardware interfaces.

**3.b.iii Software Interfaces**

* **AWS Services:** S3 (staging & curated), Redshift (warehouse), IAM (authZ), KMS (encryption), Secrets Manager (credentials), optional Glue Catalog.
* **Compute Runtimes:** Databricks Runtime or EMR (Spark 3.x, Python 3.10+).
* **Source Formats:** CSV/JSON/Parquet with header; compressed (gzip/snappy) allowed.
* **GitHub:** Repos for **Documents** and **PySpark Source**.
* **Jira:** Scrum project with sprint plan and workflows.

**Data Model Overview (Redshift, high‑level)**

* **patients**(patient\_id PK, dob, gender, city\_id FK, other demographics…)
* **subscribers**(subscriber\_id PK, patient\_id FK, group\_id FK, subgroup\_id FK, start\_date, end\_date…)
* **claims**(claim\_id PK, patient\_id FK, hospital\_id FK, disease\_code, procedure\_code, service\_date, status, total\_charges, paid\_amount, is\_cashless…)
* **groups**(group\_id PK, group\_type ENUM[Government, Private], group\_name…)
* **subgroups**(subgroup\_id PK, group\_id FK, subgroup\_name…)
* **group\_subgroup**(group\_id FK, subgroup\_id FK, active\_flag, effective\_dates…)
* **hospitals**(hospital\_id PK, hospital\_name, city\_id FK, npi\_code…)
* **cities**(city\_id PK, city\_name, state, country)
* **policies**(policy\_id PK, group\_id FK, premium\_amount, premium\_frequency…)
* **subscriber\_policy**(subscriber\_id FK, policy\_id FK, monthly\_premium\_derived…)

**Note:** Exact columns will be finalized during data profiling. Surrogate keys may be introduced where natural keys are unstable.

**3.b.iv Communications Interfaces**

* **Network:** Private connectivity via VPC endpoints to S3/Redshift where applicable; TLS 1.2+ for all traffic.
* **Authentication/Authorization:** IAM roles for compute; Redshift user/role mappings; no static creds in code.
* **Encryption:** S3 and Redshift at rest via KMS; in transit via HTTPS/TLS.

**3.c System Features**

* **Feature A — Data Ingestion & Staging:** Automated discovery of new files in input-data/; metadata capture; schema inference with explicit overrides.
* **Feature B — Data Quality & Cleansing:** Null/duplicate scans, rules-based imputations, type normalization, reference lookups; DQ metrics persisted.
* **Feature C — Conformed Curated Layer:** Redshift conformed schema with enforced PK/FK where feasible; SCD handling if needed.
* **Feature D — Analytical Outputs:** One result table per use case under Project-Output; parameterized refresh.
* **Feature E — Observability & Audit:** Job logs, run history, data volume deltas, schema drift alerts; access auditing.
* **Feature F — Governance & Security:** Data masking/tokenization for PHI in outputs; least‑privilege access; secrets management.

**3.d Nonfunctional Requirements**

**3.d.i Performance Requirements**

* **Throughput:** Ability to process daily batch of up to **10M claims** and **1M subscribers** within **< 60 minutes** end‑to‑end on provisioned clusters.
* **Query SLAs:** Each result table should be generated in **< 10 minutes**; common analytics queries on Redshift should return in **< 30 seconds** on curated data.
* **Cost Efficiency:** Use columnar storage/partitioning/sort keys; RA3 managed storage; keep compute idle time minimal via autoscaling and job clusters.

**3.d.ii Safety Requirements**

* **Data Integrity:** All transformations must be deterministic; row counts reconcilable between stages; failed runs do not corrupt curated/output tables (use staging+atomic swaps).
* **Rollback:** Ability to reprocess a given load date; maintain reproducibility via code versioning and parameterized runs.

**3.d.iii Security Requirements**

* **Compliance:** Controls aligned to HIPAA for PHI where applicable.
* **Encryption:** At rest (S3, Redshift with KMS) and in transit (TLS 1.2+).
* **Access Control:** Principle of least privilege via IAM and Redshift; PHI fields masked or tokenized in outputs.
* **Secrets:** Stored in Secrets Manager or Databricks secret scopes; never committed to code.
* **Audit:** Access and query logs retained for ≥ 365 days.

**3.d.iv Usability Requirements**

* **Discoverability:** Data dictionary and ER diagrams documented in the Documents repo.
* **Consistency:** Naming conventions for schemas/tables/columns; parameterized notebooks/jobs with clear README instructions.
* **Visuals:** Databricks visualizations (or Redshift-connected BI) for snapshots included in documentation.

**3.d.v Scalability Requirements**

* **Compute:** Horizontal scaling via EMR/Databricks autoscaling; Redshift RA3 node scaling.
* **Storage:** S3 unlimited; Redshift managed storage; partitioning/sort keys tuned for growth (10× over 12 months).
* **Extensibility:** New datasets (providers, plans, pharmacy) can be added without redesigning core abstractions.

**Appendix A — Result Tables (Naming & Minimal Columns)**

* Project-Output.max\_claims\_by\_disease (disease\_code, disease\_name, claim\_count, rank, as\_of\_date)
* Project-Output.subscribers\_under30\_with\_subgroups (subscriber\_id, age, subgroup\_id, subgroup\_name)
* Project-Output.group\_with\_max\_subgroups (group\_id, group\_name, subgroup\_count, rank)
* Project-Output.top\_hospitals\_by\_patients (hospital\_id, hospital\_name, unique\_patient\_count, rank)
* Project-Output.top\_subgroups\_by\_subscriptions (subgroup\_id, subgroup\_name, subscription\_count, rank)
* Project-Output.claims\_rejected\_summary (as\_of\_date, total\_claims, rejected\_claims, rejection\_rate)
* Project-Output.top\_cities\_by\_claims (city\_id, city\_name, claim\_count, rank)
* Project-Output.policy\_group\_popularity (group\_type, subscription\_count, share\_pct)
* Project-Output.avg\_monthly\_premium (subscriber\_id, month, avg\_monthly\_premium, population\_avg)
* Project-Output.group\_profitability (group\_id, group\_name, total\_premiums, total\_paid\_claims, profit, margin\_pct, rank)
* Project-Output.patients\_under18\_cancer (patient\_token, age, hospital\_id, hospital\_name, admission\_date, disease\_code)
* Project-Output.cashless\_high\_charges (claim\_id, subscriber\_id, is\_cashless, total\_charges, hospital\_id, service\_date)
* Project-Output.female\_over40\_knee\_past\_year (patient\_token, age, hospital\_id, service\_date, procedure\_code)

**Appendix B — Data Cleaning Rules (Minimum)**

* Null scans and per‑column counts; impute NA for specified string fields; leave numeric nulls to business rules unless specified.
* Trim whitespace, standardize casing, remove control characters.
* Type casting strictness with reject/repair logs.
* Duplicate logic: define primary business keys per dataset; drop exact duplicates, resolve fuzzy duplicates where rules are clear.
* Referential integrity checks: orphan detection for FKs; quarantine invalid rows.

**Appendix C — Delivery & Tooling**

* **Repos:**
  + documents/ — SRS, Solution Design, Schema Design, Test Plan, Jira plan export.
  + pyspark-src/ — Ingestion, Cleaning, Result Generation modules; unit tests; job definitions.
* **Jira:** 2‑week sprint (Week 1 docs/design; Week 2 build/test); user stories per FR‑xx and DQ/test tasks.
* **Testing:** Unit tests for transforms; data quality checks; result validation vs known fixtures; performance tests.