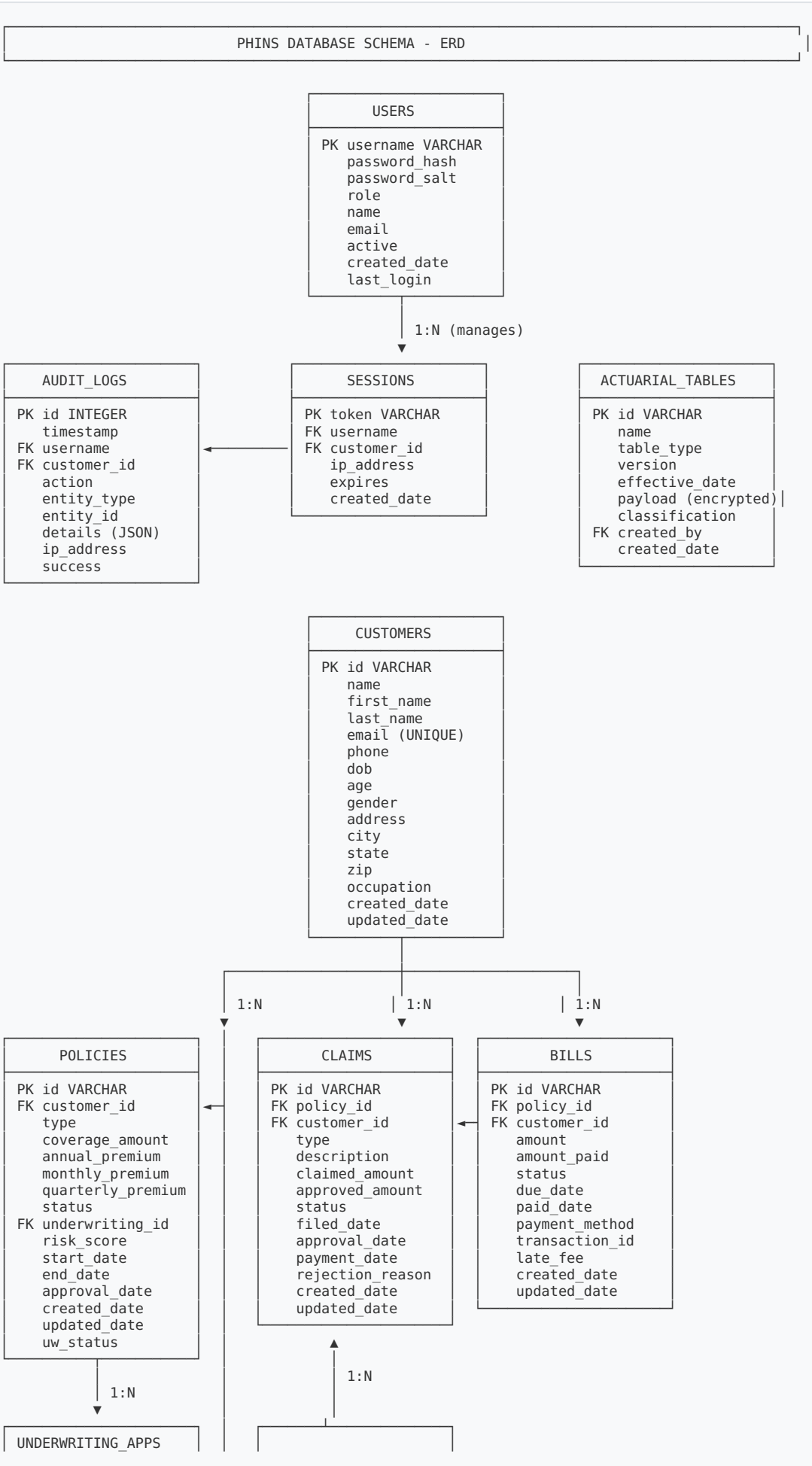
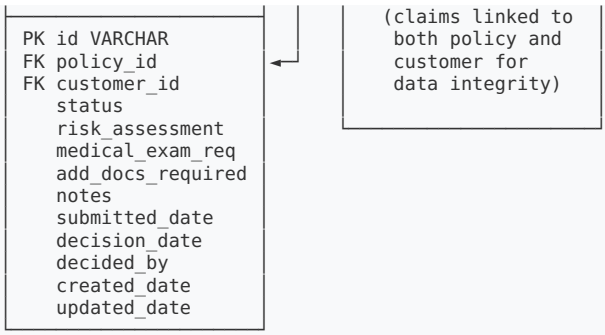
PHINS Insurance Platform - Database Architecture UML

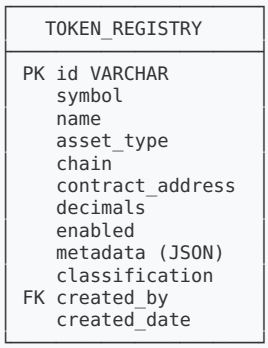
Overview

This document provides a comprehensive DBA UML sketch of the PHINS Insurance Platform's architectural structure, including entity-relationship diagrams, system layer architecture, and data flow patterns.

1. Entity-Relationship Diagram (ERD)


PHINS DATABASE SCHEMA - ERD
USERS
PK username VARCHAR
password_hash
password_salt
role
name
email
active
created_date
last_login
1:N (manages)
AUDIT_LOGS
PK id INTEGER
timestamp
FK username
FK customer_id
action
entity_type
entity_id
details (JSON)
ip_address
success
SESSIONS
PK token VARCHAR
FK username
FK customer_id
ip_address
expires
created_date
ACTUARIAL_TABLES
PK id VARCHAR
name
table_type
version
effective_date
payload (encrypted)
classification
FK created_by
created_date
CUSTOMERS
PK id VARCHAR
name
first_name
last_name
email (UNIQUE)
phone
dob
age
gender
address
city
state
zip
occupation
created_date
updated_date
1:N
1:N
1:N
POLICIES
PK id VARCHAR
FK customer_id
type
coverage_amount
annual_premium
monthly_premium
quarterly_premium
status
FK underwriting_id
risk_score
start_date
end_date
approval_date
created_date
updated_date
uw_status
CLAIMS
PK id VARCHAR
FK policy_id
FK customer_id
type
description
claimed_amount
approved_amount
status
filed_date
approval_date
payment_date
rejection_reason
created_date
updated_date
BILLS
PK id VARCHAR
FK policy_id
FK customer_id
amount
amount_paid
status
due_date
paid_date
payment_method
transaction_id
late_fee
created_date
updated_date
1:N
1:N
UNDERWRITING_APPS





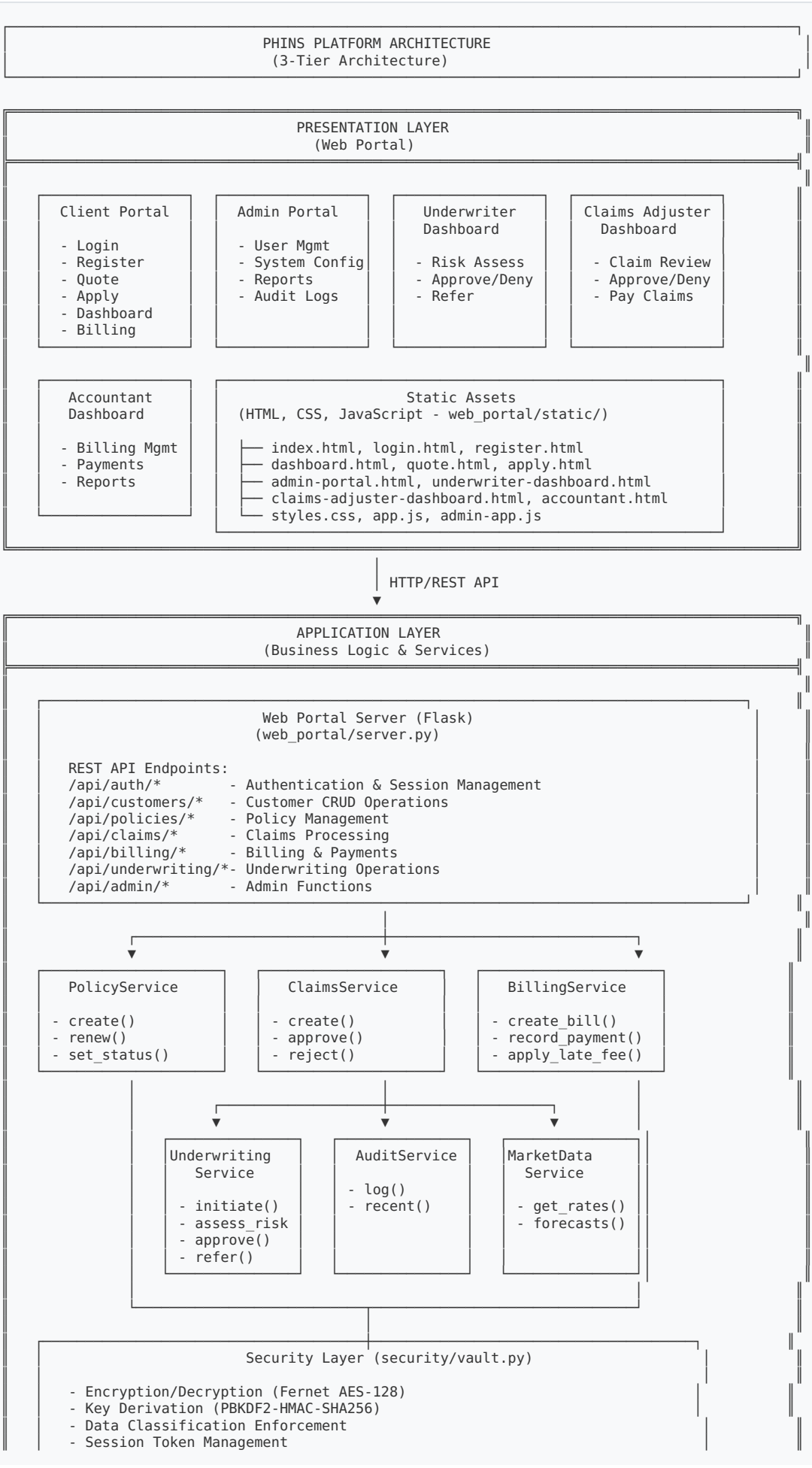
---

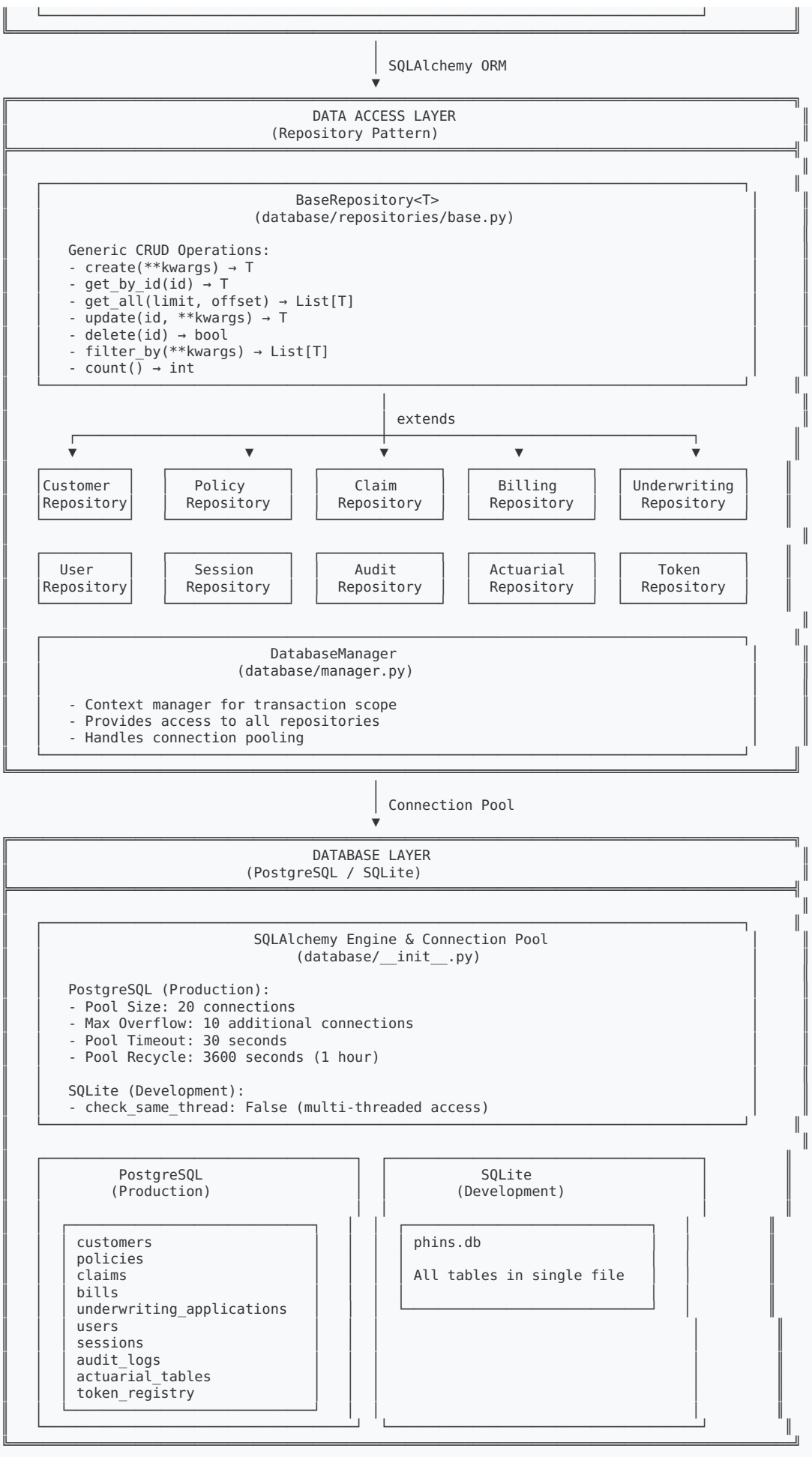
2. Table Relationships Summary

RELATIONSHIP CARDINALITY		
Parent Entity	Child Entity	Relationship
customers	policies	1:N (one customer, many policies)
customers	claims	1:N (one customer, many claims)
policies	claims	1:N (one policy, many claims)
policies	underwriting_apps	1:1 (one policy, one underwriting app)
policies	bills	1:N (one policy, many bills)
customers	bills	1:N (one customer, many bills)
users	sessions	1:N (one user, many sessions)
users	audit_logs	1:N (one user, many audit entries)
customers	sessions	1:N (one customer, many sessions)
users	actuarial_tables	1:N (creator relationship)
users	token_registry	1:N (creator relationship)

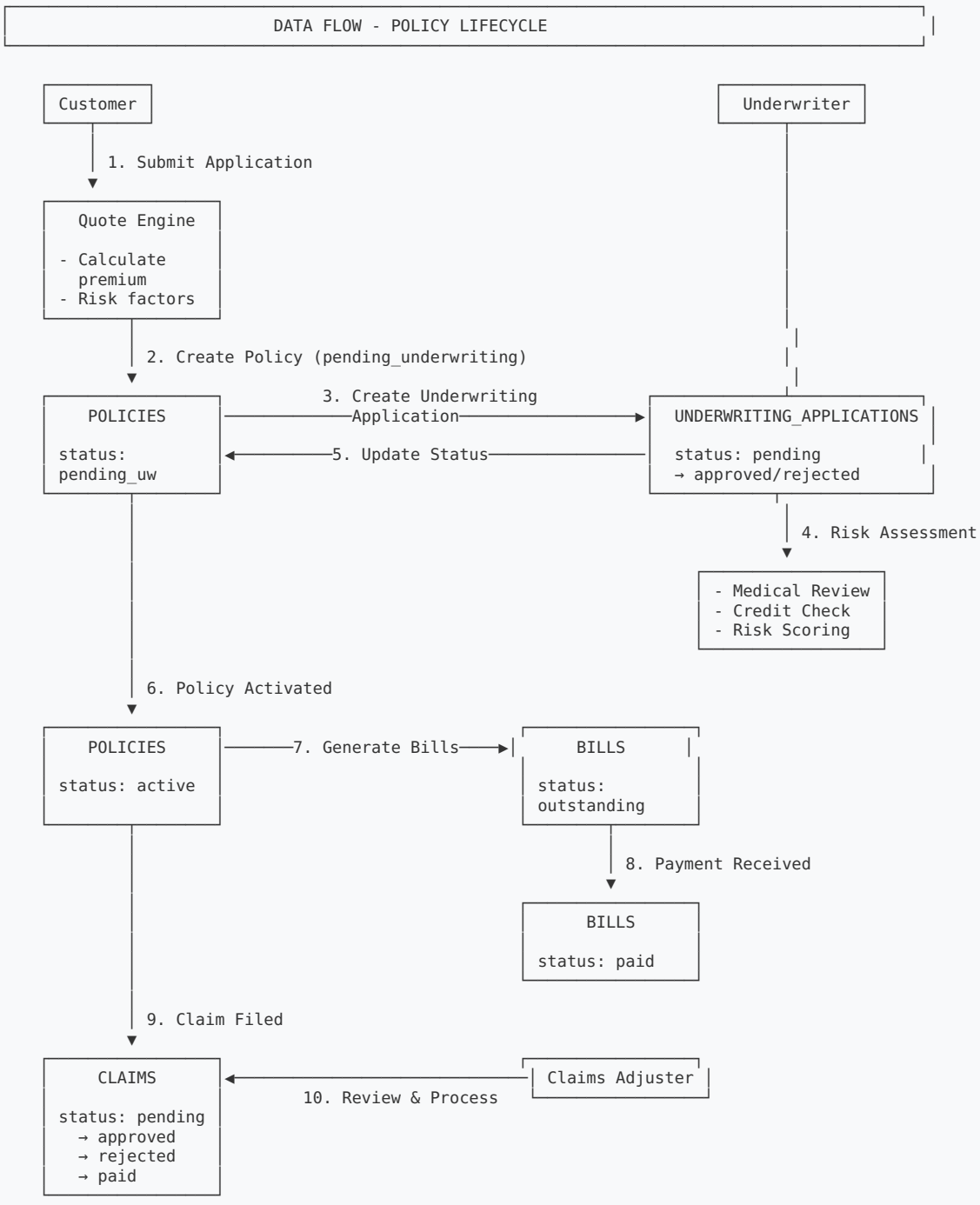
---

3. System Layer Architecture (Component Diagram)

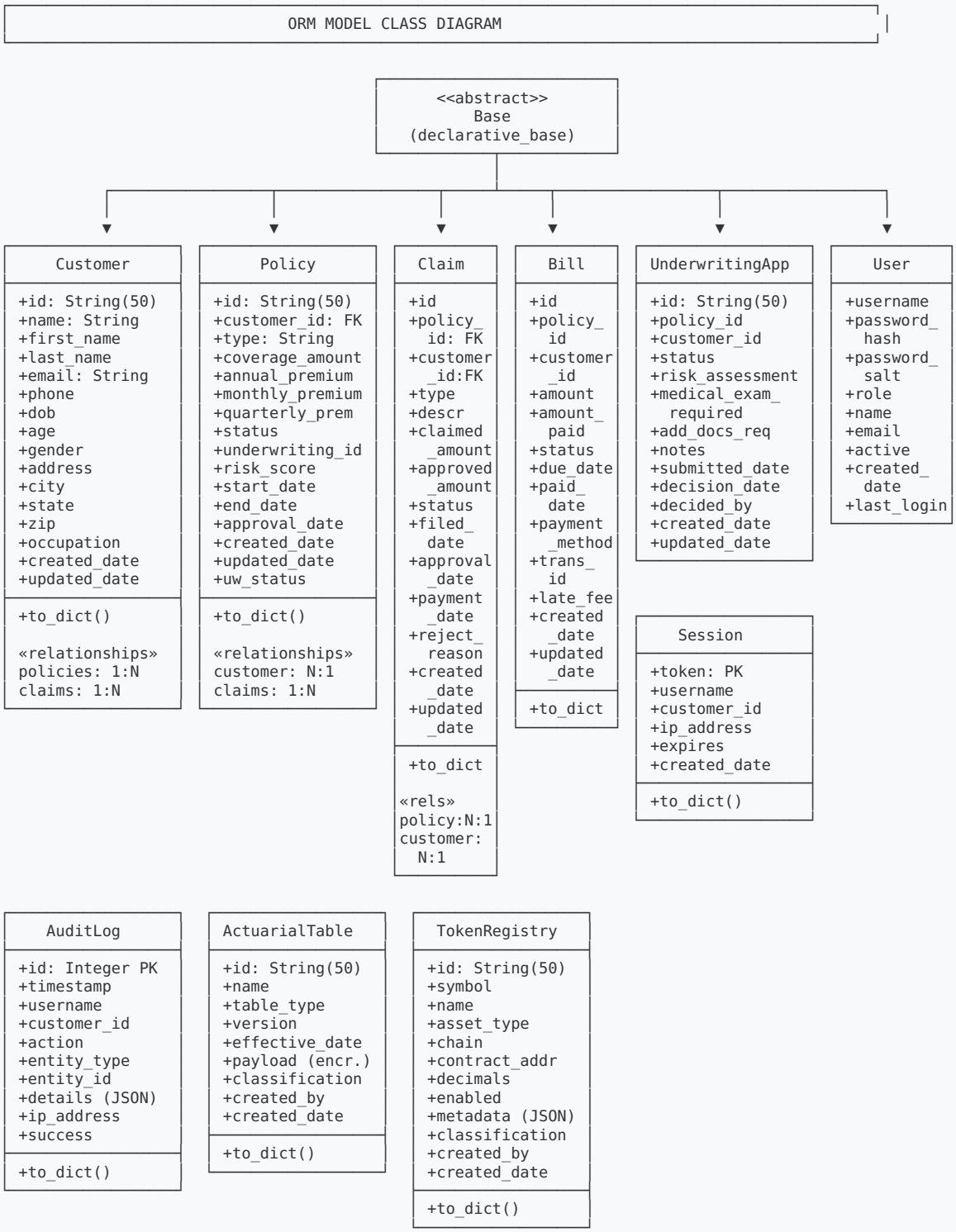

PHINS PLATFORM ARCHITECTURE
(3-Tier Architecture)
PRESENTATION LAYER
(Web Portal)
Client Portal
- Login
- Register
- Quote
- Apply
- Dashboard
- Billing
Admin Portal
- User Mgmt
- System Config
- Reports
- Audit Logs
Underwriter Dashboard
- Risk Assess
- Approve/Deny
- Refer
Claims Adjuster Dashboard
- Claim Review
- Approve/Deny
- Pay Claims
Accountant Dashboard
- Billing Mgmt
- Payments
- Reports
Static Assets
(HTML, CSS, JavaScript - web_portal/static/)
├── index.html, login.html, register.html
├── dashboard.html, quote.html, apply.html
├── admin-portal.html, underwriter-dashboard.html
├── claims-adjuster-dashboard.html, accountant.html
└── styles.css, app.js, admin-app.js
HTTP/REST API
APPLICATION LAYER
(Business Logic & Services)
Web Portal Server (Flask)
(web_portal/server.py)
REST API Endpoints:
/api/auth/* - Authentication & Session Management
/api/customers/* - Customer CRUD Operations
/api/policies/* - Policy Management
/api/claims/* - Claims Processing
/api/billing/* - Billing & Payments
/api/underwriting/*- Underwriting Operations
/api/admin/* - Admin Functions
PolicyService
- create()
- renew()
- set_status()
ClaimsService
- create()
- approve()
- reject()
BillingService
- create_bill()
- record_payment()
- apply_late_fee()
Underwriting Service
- initiate()
- assess_risk
- approve()
- refer()
AuditService
- log()
- recent()
MarketData Service
- get_rates()
- forecasts()
Security Layer (security/vault.py)
- Encryption/Decryption (Fernet AES-128)
- Key Derivation (PBKDF2-HMAC-SHA256)
- Data Classification Enforcement
- Session Token Management


SQLAlchemy ORM
DATA ACCESS LAYER
(Repository Pattern)
BaseRepository<T>
(database/repositories/base.py)
Generic CRUD Operations:
- create(**kwargs) → T
- get_by_id(id) → T
- get_all(limit, offset) → List[T]
- update(id, **kwargs) → T
- delete(id) → bool
- filter_by(**kwargs) → List[T]
- count() → int
extends
Customer Repository
Policy Repository
Claim Repository
Billing Repository
Underwriting Repository
User Repository
Session Repository
Audit Repository
Actuarial Repository
Token Repository
DatabaseManager
(database/manager.py)
- Context manager for transaction scope
- Provides access to all repositories
- Handles connection pooling
Connection Pool
DATABASE LAYER
(PostgreSQL / SQLite)
SQLAlchemy Engine & Connection Pool
(database/__init__.py)
PostgreSQL (Production):
- Pool Size: 20 connections
- Max Overflow: 10 additional connections
- Pool Timeout: 30 seconds
- Pool Recycle: 3600 seconds (1 hour)
SQLite (Development):
- check_same_thread: False (multi-threaded access)
PostgreSQL
(Production)
customers
policies
claims
bills
underwriting_applications
users
sessions
audit_logs
actuarial_tables
token_registry
SQLite
(Development)
phins.db
All tables in single file

4. Data Flow Diagram (DFD)



5. Class Diagram (ORM Models)


ORM MODEL CLASS DIAGRAM
<<abstract>>
Base
(declarative_base)
Customer
+id: String(50)
+name: String
+first_name
+last_name
+email: String
+phone
+dob
+age
+gender
+address
+city
+state
+zip
+occupation
+created_date
+updated_date
+to_dict()
«relationships»
policies: 1:N
claims: 1:N
Policy
+id: String(50)
+customer_id: FK
+type: String
+coverage_amount
+annual_premium
+monthly_premium
+quarterly_prem
+status
+underwriting_id
+risk_score
+start_date
+end_date
+approval_date
+created_date
+updated_date
+uw_status
+to_dict()
«relationships»
customer: N:1
claims: 1:N
Claim
+id
+policy_id: FK
+customer_id:FK
+type
+descr
+claimed_amount
+approved_amount
+status
+filed_date
+approval_date
+payment_date
+reject_reason
+created_date
+updated_date
+to_dict
«rels»
policy:N:1
customer: N:1
Bill
+id
+policy_id
+customer_id
+amount
+amount_paid
+status
+due_date
+paid_date
+payment_method
+trans_id
+late_fee
+created_date
+updated_date
+to_dict
UnderwritingApp
+id: String(50)
+policy_id
+customer_id
+status
+risk_assessment
+medical_exam_required
+add_docs_req
+notes
+submitted_date
+decision_date
+decided_by
+created_date
+updated_date
Session
+token: PK
+username
+customer_id
+ip_address
+expires
+created_date
+to_dict()
User
+username
+password_hash
+password_salt
+role
+name
+email
+active
+created_date
+last_login
AuditLog
+id: Integer PK
+timestamp
+username
+customer_id
+action
+entity_type
+entity_id
+details (JSON)
+ip_address
+success
+to_dict()
ActuarialTable
+id: String(50)
+name
+table_type
+version
+effective_date
+payload (encr.)
+classification
+created_by
+created_date
+to_dict()
TokenRegistry
+id: String(50)
+symbol
+name
+asset_type
+chain
+contract_addr
+decimals
+enabled
+metadata (JSON)
+classification
+created_by
+created_date
+to_dict()
ENUMERATIONS
PolicyStatus
ACTIVE
INACTIVE
CANCELLED
LAPSED
SUSPENDED
PENDING_UW
UnderwritingStatus
PENDING
APPROVED
REJECTED
REFERRED
APPROVED_COND.
ClaimStatus
PENDING
UNDER_REVIEW
APPROVED
REJECTED
PAID
CLOSED
BillStatus
OUTSTANDING
PARTIAL
PAID
OVERDUE
CANCELLED
DataClassification
TokenAssetType

```
| PUBLIC              | CURRENCY            |
| INTERNAL            | STABLECOIN          |
| CONFIDENTIAL        | NFT                 |
| RESTRICTED          | INDEX               |
```

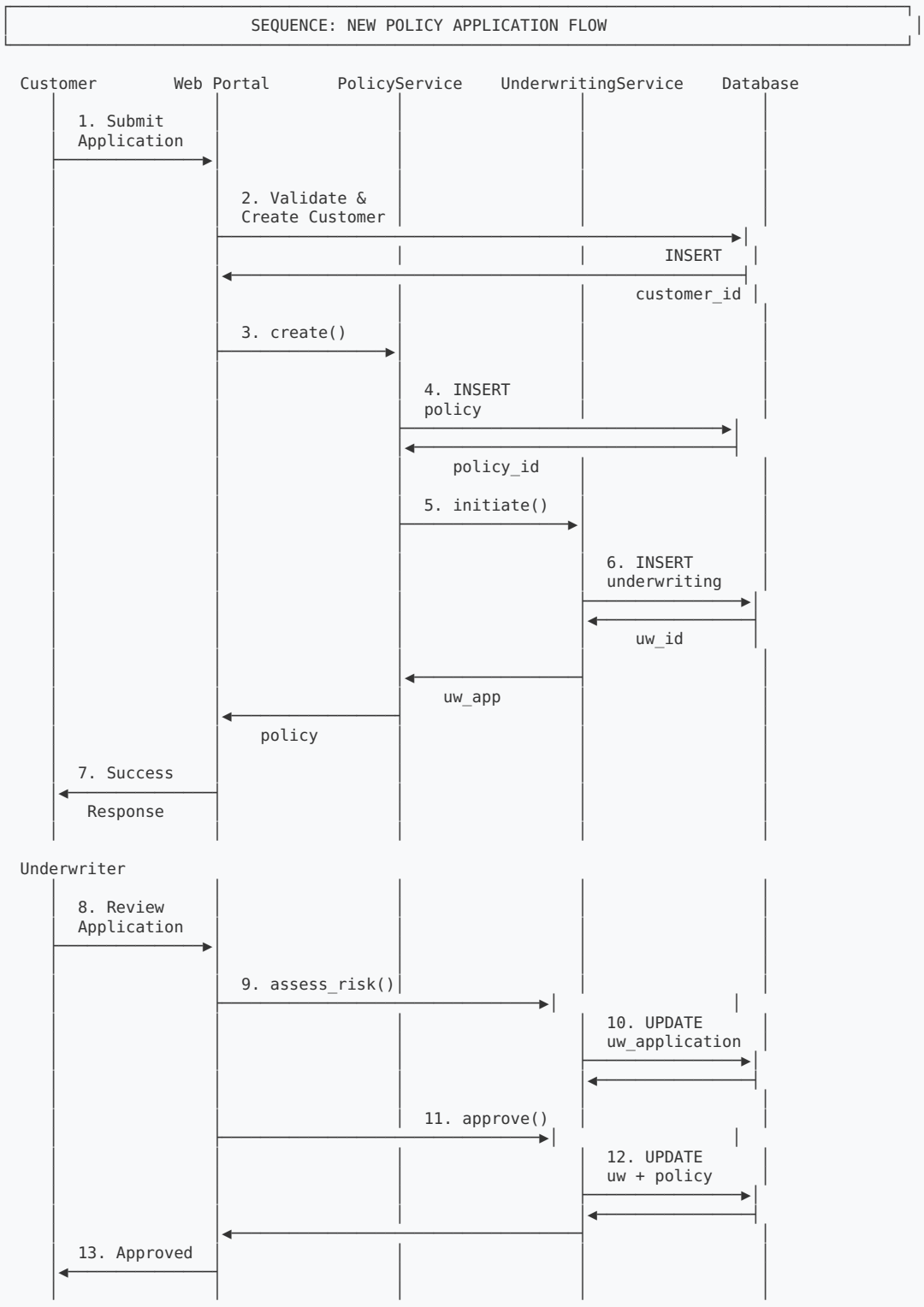
---

6. Database Indexes

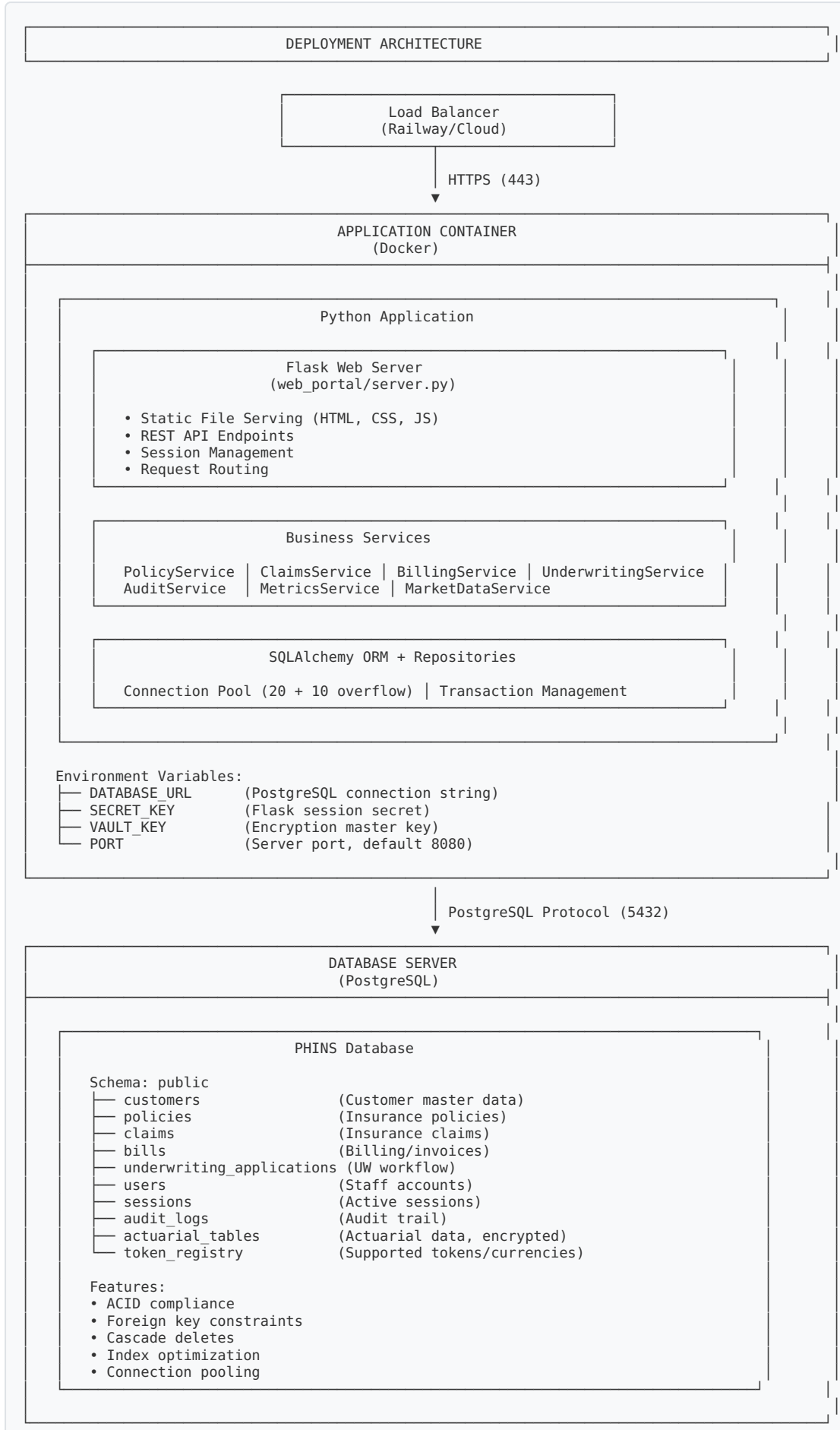
```
┌──────────────────────────────────────────────────────────────────────────────────────────┐
│                                DATABASE INDEX STRATEGY                                   │
└──────────────────────────────────────────────────────────────────────────────────────────┘

Table: customers
├── PRIMARY KEY (id)
├── UNIQUE INDEX (email)
└── INDEX (email) - for login lookups

Table: policies
├── PRIMARY KEY (id)
├── INDEX (customer_id) - FK lookups
└── INDEX (underwriting_id) - status tracking

Table: claims
├── PRIMARY KEY (id)
├── INDEX (policy_id) - FK lookups
└── INDEX (customer_id) - customer history

Table: underwriting_applications
├── PRIMARY KEY (id)
├── INDEX (policy_id) - FK lookups
└── INDEX (customer_id) - customer applications

Table: bills
├── PRIMARY KEY (id)
├── INDEX (policy_id) - policy billing
└── INDEX (customer_id) - customer billing

Table: users
├── PRIMARY KEY (username)
└── (implicit index on PK)

Table: sessions
├── PRIMARY KEY (token)
├── INDEX (username) - user sessions
├── INDEX (customer_id) - customer sessions
└── INDEX (expires) - session cleanup

Table: audit_logs
├── PRIMARY KEY (id) - auto-increment
├── INDEX (timestamp) - time-based queries
├── INDEX (username) - user audit trail
└── INDEX (customer_id) - customer audit trail

Table: actuarial_tables
├── PRIMARY KEY (id)
├── INDEX (name) - name lookups
├── INDEX (table_type) - type filtering
├── INDEX (version) - version control
├── INDEX (effective_date) - date-based queries
├── INDEX (classification) - security filtering
└── INDEX (created_by) - creator tracking

Table: token_registry
├── PRIMARY KEY (id)
├── INDEX (symbol) - symbol lookups
├── INDEX (asset_type) - type filtering
├── INDEX (enabled) - active tokens
├── INDEX (classification) - security filtering
└── INDEX (created_by) - creator tracking
```

---

7. Sequence Diagram - New Policy Application



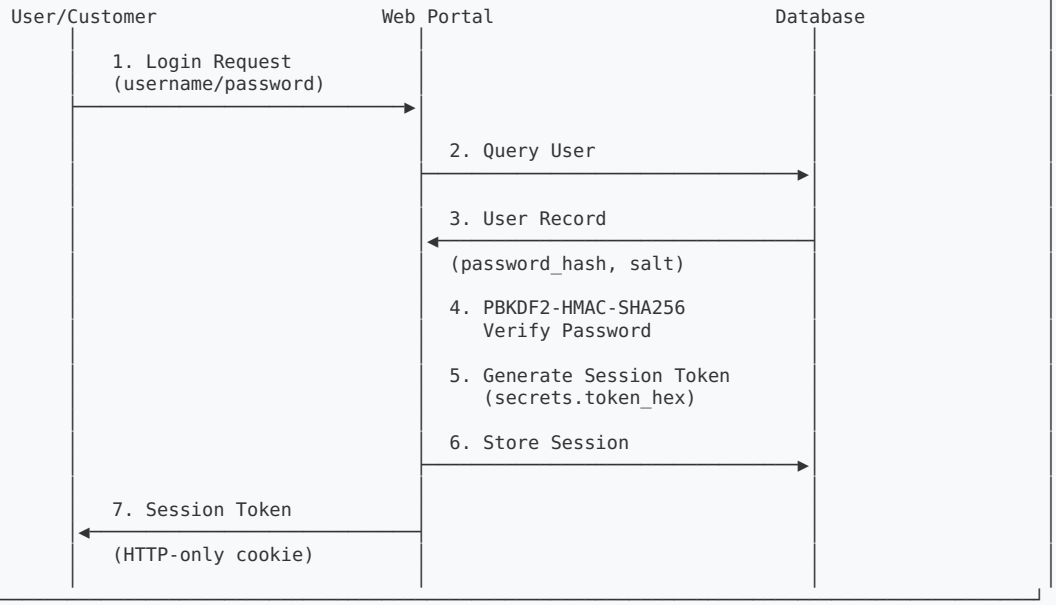
8. Deployment Architecture


DEPLOYMENT ARCHITECTURE
Load Balancer
(Railway/Cloud)
HTTPS (443)
APPLICATION CONTAINER
(Docker)
Python Application
Flask Web Server
(web_portal/server.py)
• Static File Serving (HTML, CSS, JS)
• REST API Endpoints
• Session Management
• Request Routing
Business Services
PolicyService | ClaimsService | BillingService | UnderwritingService
AuditService | MetricsService | MarketDataService
SQLAlchemy ORM + Repositories
Connection Pool (20 + 10 overflow) | Transaction Management
Environment Variables:
├── DATABASE_URL (PostgreSQL connection string)
├── SECRET_KEY (Flask session secret)
├── VAULT_KEY (Encryption master key)
└── PORT (Server port, default 8080)
PostgreSQL Protocol (5432)
DATABASE SERVER
(PostgreSQL)
PHINS Database
Schema: public
├── customers (Customer master data)
├── policies (Insurance policies)
├── claims (Insurance claims)
├── bills (Billing/invoices)
├── underwriting_applications (UW workflow)
├── users (Staff accounts)
├── sessions (Active sessions)
├── audit_logs (Audit trail)
├── actuarial_tables (Actuarial data, encrypted)
└── token_registry (Supported tokens/currencies)
Features:
• ACID compliance
• Foreign key constraints
• Cascade deletes
• Index optimization
• Connection pooling

9. Security Model

SECURITY ARCHITECTURE

AUTHENTICATION FLOW



DATA CLASSIFICATION & ENCRYPTION

```
Classification Levels:
├── PUBLIC        - Non-sensitive, can be shared externally
├── INTERNAL      - Internal use only, not for external sharing
├── CONFIDENTIAL  - Sensitive business data, limited access
└── RESTRICTED    - Highly sensitive (actuarial tables, PII)

Encryption (security/vault.py):
├── Algorithm: Fernet (AES-128-CBC with HMAC-SHA256)
├── Key Derivation: PBKDF2-HMAC-SHA256 (100,000 iterations)
└── Usage: Actuarial table payloads, sensitive configuration

Password Storage:
├── Hash: PBKDF2-HMAC-SHA256
├── Salt: Random per-user (stored in password_salt)
└── Iterations: 100,000+
```

ROLE-BASED ACCESS CONTROL

```
Role: admin
├── Full system access
├── User management
├── System configuration
└── All data access

Role: underwriter
├── View/approve/reject underwriting applications
├── View customer and policy data
└── Risk assessment

Role: claims
├── View/approve/reject claims
├── View customer and policy data
└── Process claim payments

Role: accountant
├── Billing management
├── Payment processing
├── Financial reports
└── View customer data

Role: customer (portal access)
```

```
│    ├── View own policies                                                    │
│    ├── View own claims                                                      │
│    ├── View own bills                                                       │
│    ├── Submit new applications                                              │
│    └── File claims                                                          │
└─────────────────────────────────────────────────────────────────────────────┘
```

---

10. Summary Statistics

Component	Count	Description
Database Tables	10	Core data entities
ORM Models	10	SQLAlchemy model classes
Repositories	10	Data access layer classes
Services	7	Business logic services
Enumerations	6	Status/type definitions
API Endpoints	~30+	REST API routes
Web Portals	5	User interface dashboards

Document Information

- **Platform**: PHINS Insurance Platform
- **Database**: PostgreSQL (production) / SQLite (development)
- **ORM**: SQLAlchemy 2.x
- **Architecture**: 3-Tier (Presentation → Application → Data)
- **Pattern**: Repository Pattern with Service Layer

Generated: December 2024