Cruise Employee English Assessment Platform

## AI-Powered English Proficiency Testing System

**Project Summary & Snowflake Data Platform Integration Strategy**Generated: October 08, 2025  
Version: 1.0

# 1. Executive Summary

**1.1 Project Background**

The Cruise Employee English Assessment Platform is a comprehensive AI-driven English proficiency evaluation system designed specifically for the cruise industry. It provides standardized, intelligent English proficiency testing across 16 specific positions in three major divisions: Hotel Operations, Marine Operations, and Casino Operations, ensuring cruise employees meet international maritime communication requirements.

**1.2 Core Features**

* 6 Assessment Modules: Listening, Time & Numbers, Grammar, Vocabulary, Reading, Speaking
* 288 department-specific questions covering 16 positions across professional scenarios
* AI-powered intelligent scoring system with speech recognition and NLP
* 100-point scoring system with mandatory pass requirements for safety questions
* Real-time performance analysis and detailed feedback reports
* Anti-cheating mechanisms: session management, IP tracking, time limits

**1.3 Project Statistics**

|  |  |
| --- | --- |
| **Metric** | **Value/Description** |
| Total Assessment Questions | 288 (96 per division) |
| Speaking Scenarios | 160 workplace situations |
| Departments Covered | 16 (8 Hotel, 3 Marine, 3 Casino, 2 Safety) |
| Assessment Modules | 6 core modules |
| Maximum Score | 100 points (20 from speaking) |
| Passing Criteria | 70% overall + all safety questions + 60% speaking minimum |

# 2. Technical Architecture

**2.1 Current Technology Stack**

**Backend Framework & Database:**

* FastAPI 0.104.1 - High-performance async web framework
* Python 3.10+ - Primary development language
* SQLAlchemy 2.0.23 - Async ORM framework
* PostgreSQL - Relational database (current)
* asyncpg & psycopg2-binary - Database drivers
* Alembic 1.12.1 - Database migration tool

**AI & Machine Learning:**

* OpenAI API 1.3.7 - GPT model integration
* Anthropic Claude 0.7.7 - Claude AI integration
* librosa 0.10.1 - Audio processing
* scikit-learn 1.3.2 - Machine learning algorithms
* numpy & pandas - Data processing

**Security & Session Management:**

* python-jose - JWT authentication
* passlib[bcrypt] - Password hashing
* Redis 5.0.1 - Session storage and caching
* Celery 5.3.4 - Background task queue

**2.2 Current Data Models**

* User - Candidate information (name, email, nationality, department)
* Question - Question bank (288 questions with module type, difficulty, correct answers)
* Assessment - Assessment session (status, scores, timestamps, anti-cheating info)
* AssessmentResponse - Answer records (user answers, scores, time spent, speech analysis)
* DivisionDepartment - Department mapping
* AssessmentConfig - System configuration

**2.3 Current System Architecture**

* Three-tier architecture: API Layer → Business Logic Layer → Data Access Layer
* RESTful API design with CRUD operations
* Asynchronous processing for improved concurrency
* Jinja2 template engine for frontend rendering
* Docker containerization support
* Nginx reverse proxy

# 3. Project Structure Analysis

**3.1 Core Modules**

|  |  |  |
| --- | --- | --- |
| **Module Path** | **Function** | **Key Files** |
| api/routes/ | API Routes | assessment.py, admin.py, analytics.py, ui.py |
| core/ | Core Logic | config.py, database.py, assessment\_engine.py |
| models/ | Data Models | base.py, assessment.py |
| services/ | External Services | ai\_service.py (AI integration) |
| utils/ | Utilities | scoring.py, anti\_cheating.py |
| middleware/ | Middleware | security.py, session.py |
| data/ | Data Management | question\_bank\_loader.py, generate\_question\_bank.py |
| training/ | ML Training | speech\_trainer.py |
| evaluation/ | Model Evaluation | model\_evaluator.py |
| inference/ | Inference Services | speech\_inference.py |

**3.2 Assessment Workflow**

* 1. Candidate Registration → POST /api/v1/assessment/register
* 2. Create Assessment Session → POST /api/v1/assessment/create
* 3. Start Assessment → POST /api/v1/assessment/{id}/start
* 4. Submit Answers → POST /api/v1/assessment/{id}/answer
* 5. Submit Speaking Response → POST /api/v1/assessment/{id}/speaking
* 6. Complete Assessment → POST /api/v1/assessment/{id}/complete
* 7. Retrieve Results → GET /api/v1/assessment/{id}/status

# 4. Snowflake Data Platform Integration Strategy

**4.1 Why Snowflake?**

* Cloud-Native Architecture: Fully managed cloud data warehouse, no infrastructure maintenance
* Elastic Scalability: Scale compute and storage on-demand, support massive concurrent queries
* Data Sharing: Secure cross-organizational data sharing, ideal for multi-department collaboration
* High Performance: Automatic query optimization, supports large-scale data analytics
* Cost Optimization: Separate storage and compute, per-second billing, cost-effective
* Data Governance: Built-in data governance, security, and compliance features
* ML Integration: Supports Snowpark Python for in-database ML model execution

**4.2 Integration Architecture Design**

Recommended Hybrid Architecture: PostgreSQL (Transactional) + Snowflake (Analytical)

**PostgreSQL Layer:**

* - Store real-time transactional data (user registration, assessment sessions, responses)
* - Handle OLTP operations (Online Transaction Processing)
* - Ensure data consistency and ACID properties

**Snowflake Layer:**

* - Store historical and aggregated data
* - Support complex analytical queries (OLAP)
* - Generate BI reports and data visualizations
* - Machine learning model training and inference

**Data Synchronization:**

* - Real-time sync using CDC (Change Data Capture)
* - Scheduled batch ETL jobs
* - Snowpipe streaming data ingestion

**4.3 Technical Implementation Plan**

**Step 1: Install Snowflake Dependencies**

pip install snowflake-sqlalchemy snowflake-connector-python

**Step 2: Configure Connection Strings**

Add Snowflake configuration to .env file:

SNOWFLAKE\_USER=your\_username

SNOWFLAKE\_PASSWORD=your\_password

SNOWFLAKE\_ACCOUNT=your\_account\_identifier

SNOWFLAKE\_WAREHOUSE=COMPUTE\_WH

SNOWFLAKE\_DATABASE=CRUISE\_ASSESSMENT

SNOWFLAKE\_SCHEMA=PUBLIC

**Step 3: Create Snowflake Database Manager**

Create connection manager in src/main/python/core/snowflake\_db.py

**Step 4: Design Snowflake Data Model**

* FACT\_ASSESSMENTS - Assessment fact table
* FACT\_RESPONSES - Response fact table
* DIM\_USERS - User dimension table
* DIM\_QUESTIONS - Question dimension table
* DIM\_DIVISIONS - Division dimension table
* DIM\_DATE - Date dimension table
* AGG\_DAILY\_STATS - Daily statistics aggregate table
* AGG\_DIVISION\_PERFORMANCE - Division performance aggregate table

# 5. Development Recommendations

**5.1 Data Architecture Optimization**

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Recommendation** | **Description** | **Expected Benefits** |
| HIGH | Implement Snowflake Integration | Migrate historical and analytical data to Snowflake, retain PostgreSQL for real-time transactions | 10x+ improvement in analytical query performance, reduced database load |
| HIGH | Implement Data Warehouse Model | Adopt star schema design with fact and dimension tables | Optimize complex queries, enable multi-dimensional analysis |
| MEDIUM | Establish CDC Data Pipeline | Use Debezium or similar tools for real-time data synchronization | Ensure data consistency, reduce sync latency |
| MEDIUM | Implement Snowpipe Streaming Ingestion | Automate data loading processes | Reduce manual ETL workload, enable near real-time analytics |
| LOW | Data Lake Integration | Store raw audio files in S3/Azure Blob, access via Snowflake External Tables | Lower storage costs, support unstructured data analytics |

**5.2 Performance Optimization**

* Use Snowflake clustering keys to optimize large table queries
* Implement automatic clustering to maintain data order
* Use materialized views to cache complex query results
* Enable query result cache to reduce redundant computations
* Properly size virtual warehouses, use dynamic scaling
* Implement partitioning strategy, partition large tables by time or division

**5.3 Security & Compliance**

* Enable end-to-end encryption
* Implement role-based access control (RBAC)
* Use Snowflake dynamic data masking
* Configure network policies to restrict access sources
* Enable audit logging to track all data access
* Implement multi-factor authentication (MFA)
* Conduct regular security audits and vulnerability scans

**5.4 Machine Learning Enhancements**

* Use Snowpark Python to train ML models within Snowflake
* Implement UDFs (User-Defined Functions) for real-time scoring
* Integrate Snowflake ML prediction features to predict candidate performance
* Use Snowflake Cortex for AI-driven data analytics
* Build feature engineering pipelines to extract assessment data features
* Implement A/B testing framework to optimize assessment processes

# 6. Implementation Roadmap

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Tasks** | **Duration** | **Key Deliverables** |
| Phase 1 Preparation | - Assess current data scale - Design Snowflake data model - Set up development environment - Create POC validation | 2-3 weeks | - Technical proposal - POC demonstration - Cost estimation |
| Phase 2 Development | - Develop Snowflake connector - Implement data migration scripts - Create fact and dimension tables - Develop ETL pipeline | 4-6 weeks | - Data pipeline code - Unit tests - Technical documentation |
| Phase 3 Integration | - Integrate analytics API - Develop BI reports - Implement data synchronization - Performance testing | 3-4 weeks | - API endpoints - Report dashboards - Performance benchmarks |
| Phase 4 Optimization | - Performance tuning - Security hardening - Documentation completion - Team training | 2-3 weeks | - Optimization report - User manuals - Training materials |
| Phase 5 Deployment | - Production environment deployment - Data migration - Monitoring configuration - Go-live support | 1-2 weeks | - Production system - Monitoring dashboard - Operations manual |

**Estimated Total Duration: 12-18 weeks (approximately 3-4.5 months)**

# 7. Cost Analysis

**7.1 Snowflake Cost Components**

**Compute Costs (Virtual Warehouse)**

* - Per-second billing, 1-minute minimum
* - X-Small: $2/hour, Small: $4/hour, Medium: $8/hour
* - Recommendation: Use X-Small for dev, Small-Medium for production

**Storage Costs**

* - $23-40/TB/month (varies by region)
* - Automatic compression typically saves 50-70% storage space

**Data Transfer Costs**

* - Free within region
* - Cross-region transfers incur charges

**7.2 Estimated Monthly Costs (Medium Scale)**

|  |  |  |
| --- | --- | --- |
| **Item** | **Scale** | **Estimated Cost (USD/month)** |
| Compute Resources | Small warehouse, 8 hours/day | $960 (30 days x 8 hours x $4) |
| Storage | 5TB data (after compression) | $115-200 |
| Data Transfer | Within region | $0 |
| Snowpipe | Continuous ingestion | $50-100 |
| Total |  | Approximately $1,125-1,260/month |

**7.3 Cost Optimization Strategies**

* Use auto-suspend feature to automatically shut down idle warehouses
* Properly size warehouses to avoid over-provisioning
* Use query tags to track cost attribution
* Enable resource monitors to control budgets
* Regularly clean up expired data and temporary tables
* Use Snowflake Cost Attribution to track costs

# 8. Risks & Mitigation Strategies

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Category** | **Specific Risk** | **Impact** | **Mitigation Strategy** |
| Technical | Data migration failure or data loss | High | - Phased migration - Complete backups - Parallel validation |
| Technical | Performance below expectations | Medium | - POC performance testing - Use Snowflake optimization recommendations - Expert consultation |
| Operational | Team lacks Snowflake experience | Medium | - Official Snowflake training - Hire consultants - Gradual migration with learning |
| Financial | Costs exceed budget | Medium | - Set up resource monitors - Regular cost reviews - Optimize query efficiency |
| Security | Data breach or unauthorized access | High | - Implement RBAC - Enable encryption - Regular security audits |
| Business | Impact on existing operations | High | - Gradual rollout - Rollback plan - 24/7 monitoring |

# 9. Success Metrics

**9.1 Technical Metrics**

* Query Performance: Complex analytical queries respond in < 5 seconds
* Data Sync Latency: < 5 minutes
* System Availability: > 99.9%
* Data Consistency: 100% (via automated validation)
* API Response Time: P95 < 200ms
* Concurrent Support: > 100 concurrent queries

**9.2 Business Metrics**

* Report generation time reduced by 70%
* Data analyst efficiency improved by 50%
* Data storage costs reduced by 30%
* Concurrent assessment capacity increased 3x
* New analytics requirement delivery time reduced by 60%
* BI dashboard user satisfaction > 4.5/5

**9.3 User Experience Metrics**

* Admin report query satisfaction > 90%
* Data analyst training completion rate > 95%
* Report error rate < 1%
* Self-service analytics adoption rate > 70%

# 10. Next Steps & Action Plan

**10.1 Immediate Actions (Next 1 Month)**

* Form Snowflake integration project team
* Apply for Snowflake trial account
* Complete POC environment setup
* Assess current data scale and growth trends
* Design initial data warehouse model
* Develop detailed project plan and budget

**10.2 Short-Term Actions (1-3 Months)**

* Complete Snowflake environment configuration
* Develop data synchronization pipeline
* Migrate historical data to Snowflake
* Develop analytics API and BI reports
* Conduct performance and stress testing
* Team training and knowledge transfer

**10.3 Medium-Term Actions (3-6 Months)**

* Production environment go-live
* Implement advanced analytics features
* Integrate machine learning models
* Optimize costs and performance
* Expand data source integrations
* Establish data governance framework

# 11. Conclusion

The Cruise Employee English Assessment Platform has established a solid technical foundation. Through the integration of FastAPI, PostgreSQL, and AI services, it successfully delivers intelligent English proficiency assessment. The current system architecture is clear, code quality is good, and functional modules are complete.

**Integrating the Snowflake data platform will bring significant improvements:**

* Analytical Capabilities: Support complex multi-dimensional data analysis and BI reporting
* Performance Boost: Dramatically improve query performance, especially for aggregation scenarios
* Data Insights: Discover assessment trends and optimization opportunities through advanced analytics
* Enterprise-Grade: Enhanced data governance, security, and compliance capabilities
* Cost Optimization: Lower total cost through storage-compute separation and pay-per-use
* Future-Ready: Prepared for AI/ML model training and large-scale expansion

We recommend adopting a phased implementation strategy, starting with POC validation and gradually migrating data and functionality to ensure a smooth transition. Through proper cost control and performance optimization, Snowflake integration will bring long-term value to the Cruise Employee English Assessment Platform, supporting continuous business growth and data-driven decision-making.

**--- End of Report ---**Generated: 2025-10-08 13:29:51

# Appendix A: Technical References

* Snowflake Documentation: https://docs.snowflake.com/
* Snowflake Python Connector: https://docs.snowflake.com/en/developer-guide/python-connector/
* Snowflake SQLAlchemy: https://docs.snowflake.com/en/developer-guide/python-connector/sqlalchemy
* FastAPI Documentation: https://fastapi.tiangolo.com/
* SQLAlchemy Async: https://docs.sqlalchemy.org/en/20/orm/extensions/asyncio.html
* Snowpark Python: https://docs.snowflake.com/en/developer-guide/snowpark/python/index
* Snowflake Cost Optimization: https://www.snowflake.com/en/data-cloud/workloads/cost-optimization/
* Snowflake Security Best Practices: https://docs.snowflake.com/en/user-guide/security-best-practices

# Appendix B: Project Files Inventory

* src/main/python/core/database.py - Database connection management
* src/main/python/models/assessment.py - Data model definitions
* src/main/python/api/routes/assessment.py - Assessment API routes
* src/main/python/api/routes/analytics.py - Analytics API routes
* src/main/python/services/ai\_service.py - AI service integration
* src/main/python/utils/scoring.py - Scoring algorithms
* src/main/python/data/generate\_question\_bank.py - Question bank generation
* requirements.txt - Python dependencies
* README.md - Project documentation
* CLAUDE.md - Development guidelines