QBank v10 - Enterprise Question Bank System

🚀 Overview

QBank is a production-ready, enterprise-grade question bank platform designed for high-stakes educational assessment. Built with modern architecture principles, it provides advanced psychometric capabilities, adaptive testing, and comprehensive analytics.

Key Features

- **Advanced IRT Calibration**: Support for 1PL, 2PL, 3PL models with Bayesian estimation
- **Sympson-Hetter Exposure Control**: Iterative exposure control with probabilistic item selection
- **Adaptive Testing Engine**: Real-time ability estimation using EAP/MLE methods
- **Multi-tenant Architecture**: Complete tenant isolation with row-level security
- **Semantic Search**: Vector embeddings for intelligent question discovery
- **Real-time Analytics**: Streaming analytics with Kafka and ClickHouse
- **Enterprise Security**: OAuth2/SAML SSO, audit logging, RBAC
- **High Performance**: Async Python, Redis caching, horizontal scaling

🏗️ Architecture

mermaid

```mermaid
graph TB
    subgraph "Frontend Layer"
        A[Admin UI - Next.js]
        B[Student Portal]
        C[Author Portal]
    end

    subgraph "API Gateway"
        D[FastAPI Backend]
        E[GraphQL Gateway]
    end

    subgraph "Application Services"
        F[Quiz Engine]
        G[Adaptive Selector]
        H[Calibration Service]
        I[Analytics Service]
    end

    subgraph "Data Layer"
        J[(PostgreSQL)]
        K[(Redis Cache)]
        L[(Elasticsearch)]
        M[(ClickHouse)]
    end

    subgraph "Streaming"
        N[Kafka]
        O[Flink]
    end

    subgraph "ML Services"
        P[IRT Engine]
        Q[Recommender]
        R[Embeddings]
    end

    A --> D
    B --> D
    C --> D
    D --> F
    D --> G
    D --> H
```

```
D --> I
F --> J
F --> K
G --> P
H --> P
I --> M
F --> N
N --> O
O --> M
D --> L
G --> R
```

📋 Prerequisites

- Docker & Docker Compose (v2.20+)
- Python 3.11+
- Node.js 18+
- PostgreSQL 15+ with pgvector extension
- Redis 7+
- Kafka 3.5+
- Elasticsearch 8+

🚀 Quick Start

1. Clone and Setup

bash

```bash
git clone https://github.com/yourorg/qbank.git
cd qbank
cp .env.example .env
# Edit .env with your configuration
```

2. Start Infrastructure

bash

```
# Start all services
docker-compose up -d

# Wait for services to be healthy
docker-compose ps

# Initialize database
docker-compose exec postgres psql -U qbank -d qbank -f /docker-entrypoint-initdb.d/01_core_schema.sql
docker-compose exec postgres psql -U qbank -d qbank -f /docker-entrypoint-initdb.d/02_content_ddl.sql
docker-compose exec postgres psql -U qbank -d qbank -f /docker-entrypoint-initdb.d/03_delivery_ddl.sql
docker-compose exec postgres psql -U qbank -d qbank -f /docker-entrypoint-initdb.d/04_analytics_ddl.sql
```

3. Run Migrations

bash

```
# Run Alembic migrations
docker-compose exec backend alembic upgrade head

# Seed initial data
docker-compose exec backend python scripts/seed_data.py
```

4. Access Services

- **API Documentation**: http://localhost:8000/docs
- **Admin UI**: http://localhost:3000
- **Kafka UI**: http://localhost:8080
- **Flower (Celery)**: http://localhost:5555
- **Grafana**: http://localhost:3001 (admin/admin)
- **Jaeger Tracing**: http://localhost:16686

📚 API Documentation

Authentication

bash

```
# Get access token
curl -X POST http://localhost:8000/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email": "admin@qbank.com", "password": "secure_password"}'

# Use token in requests
curl -H "Authorization: Bearer <token>" \
  http://localhost:8000/v1/admin/dashboard/metrics
```

Core Endpoints

Questions & Content

python

```
# Create question
POST /v1/author/questions
{
  "external_ref": "MATH-001",
  "topic_name": "Algebra",
  "exam_code": "SAT-MATH",
  "stem_md": "Solve for x: 2x + 3 = 7",
  "lead_in": "What is the value of x?",
  "rationale_md": "Subtract 3 from both sides...",
  "difficulty_label": "medium",
  "options": [
    {"label": "A", "text_md": "1", "is_correct": false},
    {"label": "B", "text_md": "2", "is_correct": true},
    {"label": "C", "text_md": "3", "is_correct": false},
    {"label": "D", "text_md": "4", "is_correct": false}
  ]
}

# Search questions
GET /v1/questions/search?q=algebra&topic_id=1&difficulty=medium
```

Quiz Management

python

```
# Create adaptive quiz
POST /v1/quizzes
{
  "tenant_id": "00000000-0000-0000-0000-000000000001",
  "filters": {
    "topics": ["Algebra", "Geometry"],
    "difficulty": ["medium", "hard"],
    "num_questions": 30,
    "mode": "exam",
    "exam_code": "SAT-MATH"
  },
  "adaptive": true
}

# Get next question (adaptive)
GET /v1/quizzes/{quiz_id}/next

# Submit answer
POST /v1/quizzes/{quiz_id}/answers
{
  "question_id": 123,
  "selected": "B",
  "time_taken_ms": 45000,
  "confidence": 4
}
```

Calibration

python

```
# Start Sympson-Hetter calibration
POST /v1/admin/exposure/calibrate_sh/start
{
  "exam_code": "SAT-MATH",
  "tau": 0.2,
  "n": 1000,
  "test_len": 30,
  "iters": 10,
  "alpha": 0.8,
  "theta_dist": "normal(0,1)",
  "floor": 0.01,
  "ceil": 1.0
}

# Get calibration status
GET /v1/admin/exposure/calibrate_sh/runs/{run_id}
```

🧪 Testing

Unit Tests

bash

```
# Run all tests
docker-compose exec backend pytest

# Run with coverage
docker-compose exec backend pytest --cov=app --cov-report=html

# Run specific test
docker-compose exec backend pytest tests/test_adaptive.py::test_sympson_hetter
```

Load Testing

bash

```
# Install locust
pip install locust

# Run load test
locust -f tests/load/locustfile.py --host=http://localhost:8000
```

📊 Analytics & Monitoring

Item Analysis

Access the analytics dashboard to view:

- P-values (difficulty indices)
- Discrimination indices
- Distractor analysis
- Response time distributions
- DIF (Differential Item Functioning) analysis

Real-time Metrics

Grafana dashboards provide:

- API response times
- Quiz completion rates
- User engagement metrics
- System health indicators
- IRT parameter distributions

Query Examples

```sql
```

```sql
-- Item performance analysis
SELECT
    q.external_ref,
    qv.difficulty_label,
    is.p_value,
    is.discrimination,
    is.n_responses
FROM item_statistics is
JOIN question_versions qv ON is.question_id = qv.question_id
    AND is.version = qv.version
JOIN questions q ON qv.question_id = q.id
WHERE is.p_value BETWEEN 0.3 AND 0.7
    AND is.discrimination > 0.3
ORDER BY is.discrimination DESC;

-- Exposure analysis
SELECT
    question_id,
    version,
    sh_p,
    exposure_count,
    exposure_rate
FROM item_exposure_control
WHERE exposure_rate > 0.25
ORDER BY exposure_rate DESC;
```

🔧 Configuration

Environment Variables

Key configuration options in `.env`:

```bash
```

```
# Core Settings
ENVIRONMENT=production
DEBUG=false
SECRET_KEY=your-256-bit-secret-key

# Database
DATABASE_URL=postgresql+asyncpg://user:pass@host:5432/dbname
DATABASE_POOL_SIZE=50
DATABASE_MAX_OVERFLOW=100

# Redis
REDIS_URL=redis://host:6379/0
CACHE_TTL=3600

# IRT Configuration
IRT_MODEL=3PL
IRT_MIN_RESPONSES=200
ADAPTIVE_ENABLED=true

# Exposure Control
EXPOSURE_CONTROL_ENABLED=true
MAX_DAILY_EXPOSURES=500
DEFAULT_SH_P=1.0

# ML Models
EMBEDDINGS_MODEL=sentence-transformers/all-MiniLM-L6-v2
DIFFICULTY_PREDICTION_ENABLED=true
```

Feature Flags

Control feature rollout via the database:

sql

```sql
-- Enable/disable features
UPDATE feature_flags
SET enabled = true,
    value_json = '{"model": "3PL", "min_responses": 200}'
WHERE key = 'adaptive_testing';

-- Gradual rollout
UPDATE feature_flags
SET rollout_percentage = 50,
    whitelist_users = '{"user1@example.com", "user2@example.com"}'
WHERE key = 'new_calibration_algorithm';
```

🚢 Production Deployment

Kubernetes Deployment

bash

```bash
# Create namespace
kubectl create namespace qbank-prod

# Apply secrets
kubectl create secret generic qbank-secrets \
  --from-env-file=.env.production \
  -n qbank-prod

# Deploy application
kubectl apply -f k8s/ -n qbank-prod

# Check status
kubectl get pods -n qbank-prod
kubectl get services -n qbank-prod
```

Database Migrations

bash

```bash
# Create migration
docker-compose exec backend alembic revision -m "Add new column"

# Apply migrations (staging)
kubectl exec -it deployment/qbank-backend -n staging -- alembic upgrade head

# Apply migrations (production)
kubectl exec -it deployment/qbank-backend -n production -- alembic upgrade head
```

Backup & Recovery

```bash
# Backup database
docker-compose exec postgres pg_dump -U qbank qbank | gzip > backup_$(date +%Y%m%d).sql.gz

# Restore database
gunzip -c backup_20240115.sql.gz | docker-compose exec -T postgres psql -U qbank qbank

# Backup Redis
docker-compose exec redis redis-cli BGSAVE
docker cp qbank-redis:/data/dump.rdb ./backups/redis_$(date +%Y%m%d).rdb
```

📈 Performance Optimization

Database Optimization

```sql
```

```sql
-- Analyze query performance
EXPLAIN ANALYZE
SELECT * FROM user_responses ur
JOIN question_versions qv ON ur.question_id = qv.question_id
WHERE ur.user_id = 'user123'
ORDER BY ur.created_at DESC
LIMIT 100;

-- Create partial indexes for common queries
CREATE INDEX CONCURRENTLY idx_ur_user_recent
ON user_responses(user_id, created_at DESC)
WHERE created_at > NOW() - INTERVAL '30 days';

-- Partition large tables
CREATE TABLE user_responses_2024_01 PARTITION OF user_responses
FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

Caching Strategy

python

```python
# Cache question pools
@cache.cache_result(prefix="pool", expire=3600)
async def get_question_pool(exam_code: str):
    # Expensive query
    return await db.fetch_questions(exam_code)

# Cache user abilities
@cache.cache_result(
    prefix="ability",
    expire=300,
    key_builder=lambda user_id, topic_id: f"ability:{user_id}:{topic_id}"
)
async def get_user_ability(user_id: str, topic_id: int):
    return await db.fetch_ability(user_id, topic_id)
```

🔒 Security

Authentication & Authorization

- JWT-based authentication with refresh tokens
- Role-based access control (RBAC)

- Row-level security (RLS) for multi-tenancy
- API rate limiting per user/IP

Data Protection

- Encryption at rest (PostgreSQL TDE)
- Encryption in transit (TLS 1.3)
- PII data segregation
- Audit logging for compliance

Security Headers

```python
# Configured in FastAPI middleware
security_headers = {
    "X-Content-Type-Options": "nosniff",
    "X-Frame-Options": "DENY",
    "X-XSS-Protection": "1; mode=block",
    "Strict-Transport-Security": "max-age=31536000; includeSubDomains",
    "Content-Security-Policy": "default-src 'self'"
}
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

Code Style

```bash
```

```
# Format code
black qbank-backend/
ruff check qbank-backend/

# Type checking
mypy qbank-backend/app

# Frontend linting
cd admin-ui && npm run lint
```

📝 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙏 Acknowledgments

- IRT algorithms based on research by Hambleton & Swaminathan
- Sympson-Hetter method from "A New Item Exposure Control Algorithm" (1985)
- FastAPI framework by Sebastián Ramírez
- UI components inspired by Tailwind UI

📞 Support

- Documentation: https://docs.qbank.io
- Issues: https://github.com/yourorg/qbank/issues
- Discord: https://discord.gg/qbank
- Email: support@qbank.io

Built with ❤️ for the education community