Initial System Specification – DiabRisk

Project: DiabRisk – Diabetes Risk Screener Platform
Date: January 2026
Version: 1.0
Authors: Development Team
Supervisor: Course Instructor

Part I: System Vision

1.1 System Domain and Purpose

DiabRisk is an educational web-based application that estimates an individual's risk of developing Type 2 Diabetes Mellitus (T2DM) using routine, non-invasive health data. The system addresses a critical public health challenge: worldwide, over 400 million adults have diabetes, yet many remain undiagnosed until complications appear.

The primary target users are students, early-career clinicians, and health-conscious individuals seeking a transparent, data-driven risk assessment. Unlike commercial black-box calculators, DiabRisk is open, reproducible, and educational, publishing model details and allowing users to explore feature contributions.

Core Domain Concepts:

- **Risk Assessment**: Binary classification predicting current or near-future T2DM status based on 21 health parameters including age, BMI, blood pressure, cholesterol levels, lifestyle factors (smoking, alcohol consumption, physical activity), and general health indicators.
- **Machine Learning Model**: The system employs interpretable ML models (Logistic Regression, Gradient-Boosted Trees) trained on public datasets (BRFSS 2015 with 250,000+ records). Models are calibrated using Platt or Isotonic regression and exported in ONNX format for efficient serving.
- **Privacy and Transparency**: The system emphasizes user privacy with optional authentication, secure data storage, and GDPR-compliant data deletion. All model metrics (AUROC, AUPRC, Brier score) are publicly documented in model cards.

System Positioning:

Aspect	Description
Target Users	Students, healthcare trainees, individuals tracking metabolic health
Primary Need	Quick estimation of diabetes risk from easily available metrics
Product Type	Web application (Svelte frontend + Go/Python microservices)
Value Proposition	Open, explainable ML model with personal report history
Key Benefit	Early awareness and self-education—not medical diagnosis

Scope and Constraints:

The system is explicitly **educational and non-diagnostic**. It does not integrate with electronic health records, provide medical advice, or claim clinical certification. Phase 1 focuses on core functionality: data entry, ML inference, risk visualization, and optional user accounts for assessment history. Mobile applications and multilingual support are excluded from initial scope.

1.2 Most Significant System Processes

Process 1: Anonymous Risk Assessment

Actors: Unauthenticated User, Risk Prediction Service

Flow:

1. User accesses web interface and encounters health data input form
2. User fills 21 health parameters with real-time validation (e.g., BMI accepts decimals 15.0-60.0, Age 18-100)
3. User submits assessment via frontend
4. API Gateway validates request structure and forwards to ML Risk Service
5. Risk Service loads ONNX model, preprocesses features, executes inference
6. Service returns probability score (0.0-1.0), risk category (low/medium/high), and interpretive message
7. Frontend displays results with color-coded visualization
8. Assessment is ephemeral—not stored without authentication

Technical Details:

- Frontend: HTTP POST to `/api/risk` with JSON payload `{features: {...}}`
- Backend: Go API Gateway forwards to Python ML service (currently deployed at external endpoint, will be migrated to same cluster)
- Performance target: ≤800ms P50 latency
- Rate limiting: Prevents abuse while allowing educational exploration

Current Status: Implemented (Phase 1)
Future Work: Migrate ML service to Kubernetes cluster alongside other microservices

Process 2: Authenticated User Assessment History

Actors: Registered User, Authentication Service, Data Service

Flow:

1. User registers/signs in via Google OAuth or email magic link
2. Authentication Service creates secure HttpOnly session cookie
3. Upon assessment submission, Data Service stores:
    - Input features vector
    - Model version identifier
    - Risk probability and category
    - Timestamp
    - User reference (foreign key)
4. User accesses `/history` route to view past assessments
5. Data Service retrieves assessments filtered by user ID
6. Frontend displays chronological list with sortable columns

Data Model:

```
 assessments table:
- id (UUID)
- user_id (FK)
- features (JSONB)
- probability (FLOAT)
- category (VARCHAR)
- model_version (VARCHAR)
- created_at (TIMESTAMP)
```

Current Status: Planned (Phase 1)
Implementation Notes: OAuth integration via Google provider, PostgreSQL database for persistence

Process 3: Report Generation and Export

Actors: Authenticated User, Report Service

Flow:

1. User requests PDF report for specific assessment
2. Report Service retrieves assessment data from Data Service
3. Service generates document containing:
    - Risk score with visual gauge
    - Feature contribution chart (SHAP values)
    - Calibration plot showing model reliability
    - Educational explanation of risk factors
    - Disclaimer and model version metadata
4. PDF rendered using template engine and chart libraries
5. Document stored in object storage (S3-compatible)
6. User downloads report or views in browser

Alternative Flow - Bulk Export:

- User requests CSV/JSON export of all assessments
- Data Service serializes all user assessments
- File generated with complete feature vectors and predictions
- Useful for personal health tracking or clinician review

Current Status: Planned (Phase 1-2)
Dependencies: Requires authenticated user system and data persistence

Process 4: Model Training and Deployment Pipeline

Actors: ML Engineer, CI/CD System

Flow:

1. ML Engineer prepares training script with dataset references
2. Pipeline fetches BRFSS 2015 dataset (or updates)
3. Data preprocessing: handle missing values, normalize features, encode categoricals
4. Train multiple models (Logistic Regression, XGBoost) with cross-validation
5. Evaluate on held-out test set: compute AUROC, AUPRC, ECE, Brier score
6. Calibrate probabilities using isotonic regression
7. Export best model to ONNX format with metadata
8. Generate model card documenting:
    - Training dataset version and hash
    - Hyperparameters and architecture
    - Performance metrics with confidence intervals
    - Subgroup fairness analysis (age, sex, BMI bands)
9. Register model version in artifact repository

10. Deploy to Risk Service via containerized update
11. Health check confirms new model serving correctly

Quality Gates:

- AUROC ≥ 0.80 required for production deployment
- Calibration error (ECE) ≤ 0.10
- No significant fairness violations across subgroups

Current Status: ▯ Partially Implemented
Completed: Basic ML service with trained model serving predictions
Planned: Full CI/CD pipeline with automated model versioning and deployment

Process 5: User Account and Data Management

Actors: Registered User, Data Service

GDPR-Compliant Operations:

Account Creation:

- User provides email, authenticates via OAuth
- System creates user record with UUID
- Session established with secure cookie

Data Access:

- User views all stored assessments
- System provides full data export on request
- Audit log tracks data access events

Account Deletion:

- User initiates deletion request
- System displays confirmation with data summary
- Upon confirmation:
    - All assessments deleted from database
    - Associated reports removed from object storage
    - User record anonymized or purged
    - Audit log entry created (retained per legal requirements)
- Confirmation email sent

1.3 System Architecture

Frontend Layer:

- Svelte SPA with routes: `/`, `/history`, `/login`, `/result/:id`
- Communicates with API Gateway via fetch API
- Responsive design supporting desktop and tablet

Backend Microservices:

- **api-gateway (Go)**: Request routing, CORS, rate limiting
- **auth-svc (Go)**: OAuth integration, session management
- **data-svc (Go)**: Assessment persistence and retrieval
- **risk-svc (Python)**: ML model inference with ONNX runtime
- **report-svc (Go)**: PDF generation and storage

Data Layer:

- PostgreSQL: User accounts, assessments, model metadata
- Object Storage (S3): Generated reports, model artifacts

Event Bus:

- NATS messaging for async operations (report generation)

Deployment:

- Phase 1: Docker Compose for local development
- Phase 2: Kubernetes with Helm charts
- CI/CD: GitHub Actions with automated testing

Observability:

- Prometheus metrics (latency, error rates, model predictions)
- Grafana dashboards for system health
- Structured logging with correlation IDs

1.4 Success Criteria

Technical Metrics:

- Model AUROC ≥ 0.80 on validation set
- Prediction latency P50 ≤ 800ms, P95 ≤ 2000ms

- System availability ≥ 99% per semester
- Unit/integration test coverage ≥ 60%

Functional Completeness:

- All 15 prioritized use cases implemented
- Reproducible ML pipeline with version-controlled artifacts
- GDPR-compliant data deletion within 24 hours

User Experience:

- User satisfaction survey ≥ 4/5 average
- < 5% bug escape rate after system testing
- Mobile-responsive interface with accessibility standards (WCAG 2.1 AA)

Part II: Project Dictionary

2.1 Core Entities

Term	Type	Definition
User	Entity	A registered account owner who can log in, submit assessments, and view saved reports. Also referred to as Account or Actor in authentication contexts.
Assessment	Entity	A single request for diabetes-risk estimation containing 21 input features (age, BMI, blood pressure, cholesterol, lifestyle factors), model version identifier, timestamp, and resulting probability. Also called Evaluation or Risk Request.
Feature	Entity	Individual input parameter used by the ML model, such as `Age`, `BMI`, `HighBP`, `HighChol`, `Smoker`, `PhysActivity`. Features are validated for range and type before inference.
Model Version	Entity	Frozen ML artifact including trained weights, preprocessor configuration, and metadata, identified by semantic version (e.g., v0.1.0). Each version has associated performance metrics documented in its Model Card.
Risk Score	Entity	Numerical probability (0.0 to 1.0) representing predicted likelihood of T2DM, along with categorical classification (low: <0.3, medium: 0.3-0.6, high: >0.6) and explanatory message.
Explanation	Entity	Per-feature importance values (SHAP contributions) explaining how each input parameter influenced a specific prediction. Provides transparency and educational value.
Report	Entity	Generated PDF document summarizing an assessment with risk score visualization, calibration plot, feature contribution chart, and educational context. Stored in object storage with UUID reference.
Calibration Curve	Entity	Statistical visualization showing reliability of predicted probabilities by comparing expected vs. observed diabetes rates across probability bins. Demonstrates model trustworthiness.
Auth Session	Entity	Temporary secure state established after successful login, represented by HttpOnly SameSite=Lax cookie containing encrypted session token. Expires after configurable timeout (default 7 days).
Audit Log	Entity	Immutable record of security-relevant operations (login, assessment creation, data export, account deletion) with timestamp, user ID, action type, and IP address. Retained per compliance requirements.

2.2 Functions (System Capabilities)

Function	Description	Input → Output
Register / Sign In	Authenticate user via Google OAuth or email magic link, create session cookie.	`email` → session token (HttpOnly cookie)
Submit Assessment	Accept 21 health parameters, validate ranges (e.g., BMI 15-60, Age 18-100), trigger ML inference pipeline.	`features{}` → `assessment_id`, `probability`, `category`
Predict Risk	Core ML inference executed by risk-svc: load ONNX model, preprocess features, execute forward pass, apply calibration.	`features[]` → `probability`, `confidence_interval`
Explain Prediction	Compute SHAP values for each feature to show contribution to final risk score.	`assessment_id` → `shap_values[]` with per-feature impacts
Generate	Render PDF with risk visualization, calibration plot, SHAP chart, and educational	`assessment_id` → `report.pdf` stored

Function	Description	Input → Output
Report	text using template engine.	in object storage
Save Assessment	Persist assessment results (features, probability, model version) to PostgreSQL for authenticated user.	Assessment object → database row with UUID
List Assessments	Retrieve user's assessment history sorted by timestamp, with optional filtering by date range or risk category.	`user_id`, `filters` → `[assessments]` paginated
Export Assessments	Generate CSV or JSON file containing all user assessments with complete feature vectors and metadata.	`user_id`, `format` → downloadable file
Delete Assessment	Permanently remove single assessment from database and associated reports from object storage. Logged in audit trail.	`assessment_id` → confirmation
Delete Account & Data	GDPR-compliant full deletion: remove user, all assessments, reports, and sessions. Anonymize audit logs.	`user_id` → deletion confirmation
List Model Versions	Display available deployed models with training date, AUROC, AUPRC, calibration metrics from model cards.	— → `[model_versions]` with metadata
Health Check	Verify service uptime, database connectivity, model loading status. Returns 200 OK with component statuses.	— → `{status, components[]}`

2.3 Activities (Processes)

Activity	Description	Performed By
Onboarding	First-time user registration with optional tutorial explaining input parameters and interpreting results.	User + System
Data Entry	Interactive form filling with real-time validation (e.g., BMI accepts decimals, age must be positive integer). Tooltip help for complex fields.	User
Risk Evaluation	End-to-end ML inference: request validation → feature preprocessing → ONNX model execution → calibration → response formatting.	System (api-gateway → risk-svc)
Report Generation	Asynchronous PDF creation: retrieve assessment → compute SHAP values → render charts → compile PDF → upload to storage → notify user.	System (report-svc)
Training Pipeline	Offline ML workflow executed periodically: fetch dataset → preprocess → train models → cross-validate → evaluate → calibrate → export ONNX → register version.	ML Engineer + CI/CD
Monitoring & Logging	Continuous collection of metrics (latency, error rates, prediction distribution) and log aggregation for debugging.	DevOps + Prometheus/Grafana
User Feedback Collection	Optional post-assessment survey (1-5 star rating, free text) to improve educational effectiveness.	User

2.4 Persons (Actors and Roles)

Person	Description
End User	Primary actor interacting with web interface to submit assessments, view results, manage account. May be authenticated or anonymous.
Clinician Reviewer	Secondary actor who may review exported assessment history as supplementary context (not primary diagnostic tool).
Product Owner	Responsible for backlog prioritization, scope control (≤20 use cases), acceptance criteria definition.
Go Developer	Implements microservices (api-gateway, auth-svc, data-svc, report-svc), writes integration tests, maintains API documentation.
ML Engineer	Designs experiments, trains models, evaluates performance, writes model cards, deploys models to risk-svc.
Frontend	

Person	Description
Developer	Implements Svelte SPA, designs UI/UX, ensures responsive design and accessibility.
QA Engineer	Performs functional testing, load testing (1000 concurrent users target), security testing (OWASP Top 10).
DevOps Engineer	Manages CI/CD pipelines, Kubernetes configuration, monitoring setup, backup procedures.
Course Instructor	Academic supervisor who reviews documentation, provides domain expertise, evaluates project deliverables.

2.5 Technical Terms

Term	Definition
AUROC	Area Under Receiver Operating Characteristic curve; measures model's ability to discriminate between positive (diabetic) and negative cases. Range 0.5-1.0, higher is better.
AUPRC	Area Under Precision-Recall Curve; more robust than AUROC under class imbalance (diabetes prevalence ~10%).
SHAP	SHapley Additive exPlanations; game-theory-based method for feature attribution. Assigns each feature a contribution value to the prediction.
ONNX	Open Neural Network Exchange format; standard for portable ML models across frameworks and runtimes. Enables Python-trained models to run in Go/C++ inference servers.
Calibration	Process of adjusting model outputs so predicted probabilities match observed frequencies. E.g., if model predicts 30% risk, approximately 30% of such cases should have diabetes.
Expected Calibration Error (ECE)	Metric quantifying calibration quality; average difference between predicted and observed probabilities across bins. Target: ≤0.10.
GDPR	General Data Protection Regulation (EU); mandates user rights including data access, correction, deletion, and portability.
Microservice	Independently deployable component exposing focused API. Each service has dedicated codebase, deployment, and scaling.
API Gateway	Entry point for all external requests; handles cross-cutting concerns like authentication, rate limiting, request routing.

2.6 Domain-Specific Terms

Term	Definition
Type 2 Diabetes Mellitus (T2DM)	Chronic metabolic disorder characterized by insulin resistance and high blood glucose. Risk factors include obesity, sedentary lifestyle, family history.
BMI (Body Mass Index)	Weight (kg) / Height² (m²); screening tool for weight categories. Normal: 18.5-24.9, Overweight: 25-29.9, Obese: ≥30.
BRFSS	Behavioral Risk Factor Surveillance System; annual US health survey by CDC. 2015 dataset contains 400,000+ responses with diabetes status and 21 health indicators.
Feature Engineering	Process of transforming raw inputs into model-friendly representations (e.g., binning age into categories, normalizing BMI).
Model Card	Standardized documentation for ML models describing intended use, training data, performance metrics, limitations, fairness analysis.

Part III: System Constraints and Assumptions

3.1 Constraints

1. **Educational Purpose**: System is explicitly non-diagnostic. Prominent disclaimers required on all interfaces and reports.

2. **Dataset Licensing**: Only publicly licensed datasets (BRFSS, Pima, NHANES) may be used for training.
3. **Technology Stack**: Fixed to Svelte (frontend), Go + Python (backend), PostgreSQL (database) per project requirements.
4. **Team Size**: 3-5 students over one semester (~15 weeks).
5. **Scope**: Maximum 20 use cases to maintain feasibility.

3.2 Assumptions

1. **Internet Connectivity**: Users have stable internet access; offline mode not supported.
2. **Browser Compatibility**: Modern browsers (Chrome, Firefox, Safari, Edge) with ES6+ support.
3. **Data Quality**: Users provide honest health data; no validation against medical records.
4. **English Language**: UI and documentation in English only for Phase 1.
5. **Computational Resources**: Sufficient server capacity for 1000 concurrent users (load testing validates).

Part IV: Prioritized Use Cases

ID	Use Case	Priority	Phase
UC-1	Submit Anonymous Assessment	Critical	1
UC-2	View Risk Result with Visualization	Critical	1
UC-3	Download PDF Report	High	1
UC-4	Register / Sign In via OAuth	High	1
UC-5	Save Assessment to Account	High	1
UC-6	List Past Assessments	High	1
UC-7	Delete Individual Assessment	Medium	1
UC-8	Export All Assessments (CSV/JSON)	Medium	2
UC-9	View Model Version and Metrics	Medium	1
UC-10	Generate SHAP Explanations	Medium	2
UC-11	View Calibration Plot	Medium	2
UC-12	Delete Account & All Data (GDPR)	High	1
UC-13	Admin Health Check Endpoint	Low	1
UC-14	Rate Limit Handling	Medium	1
UC-15	Legal Disclaimer Acknowledgement	Critical	1

Part V: Risks and Mitigations

Risk	Impact	Probability	Mitigation
Model Overfitting	High	Medium	Cross-validation, early stopping, separate validation set.
Data Quality Issues	High	Medium	Dataset-specific adapters, automated quality checks in pipeline.
Scope Creep	High	High	Strict 20 use case limit, PO approves all additions.
Regulatory Misunderstanding	High	Low	Prominent disclaimers, legal review of messaging.
Polyglot Complexity	Medium	Medium	Containerization, clear service boundaries, comprehensive docs.
			OWASP testing, dependency scanning, HTTPS-only, HttpOnly

Risk	Impact	Probability	Mitigation
Security Vulnerabilities	High	Medium	cookies.
Performance Degradation	Medium	Low	Load testing, caching, horizontal scaling capability.

Approval and Sign-off

Role	Name	Signature	Date
Product Owner			
Lead Developer			
ML Engineer			
Course Instructor			

Document Version History:

Version	Date	Author	Changes
1.0	2026-01-10	Development Team	Initial specification combining System Vision and Project Dictionary

---

End of Initial System Specification