ITI Examination System - Comprehensive Documentation

Team Members:

- Amira Maged Farhat Mohammed
- Ashraqat Mohamed Aboelella Mohamed
- Doha Waleed Kamal Abd El Maksoud
- Sarah Hani Farid Abdo
- Walaa Ahmed Korani Mohamed

Supervisor: Eng. Aliaa Khaled

Submission Date: 8-Nov-2025

Table of Contents

Project Overview

The ITI Examination System is a full-stack educational platform designed as a simulation of Information Technology Institute's (ITI) intensive code camp scholarship. With the primary goal of automating and managing online examinations efficiently while incorporating comprehensive scholarship management features including storing and managing information about students, tracks, intakes, instructors, departments and courses.

Built as a graduation project, the system simulates real-world ITI ecosystem operations including student tracking, instructor management, exam administration, and performance analytics.

The project covers the full cycle of an examination system, starting from building a relational database, integrating data, generating analytical reports, and visualizing insights.

Additionally, a web application was developed to simulate the online exam environment, allowing users to take and submit exams digitally.

This project provides a complete end-to-end solution that mirrors ITI's real operations and supports better data-driven decision-making.

System Objectives

The main objectives of the ITI Examination System are:

1. ***Centralized Data Management***
   Building a unified database for students, instructors, tracks, courses, and exams, in one reliable system.
2. ***Exam Automation***
   Automate exam creation and grading to reduce manual work.
3. ***Performance Tracking***
   Monitor and analyze student progress through reports and dashboards.
4. ***Data Integration***
   Ensure accurate data flow using SSIS.
5. ***Analytical Reporting***
   Generate insightful reports using SSRS and Power BI.
6. ***User-Friendly Interface***
   Provide a secure, easy-to-use exam portal via Flask.
7. ***Scalability & Efficiency***
   Support future expansion while maintaining high performance.

Tools & Technologies Used

1. ***Draw.io***
   Designed the conceptual ERD (entities, attributes, relationships).
2. ***ERD Lab***
   Built the logical model and relational schema with foreign keys.
3. ***SQL Server Management System (SSMS)***
   Created and managed the SQL Server database, tables, constraints, and stored procedures.
4. ***Excel***
   Prepared initial sample datasets for testing and data generation.
5. ***Faker (Python)***
   Automatically generated realistic additional data.
6. ***SQL Server Integration Service (SSIS)***
   Developed ETL pipelines to load data from Excel into SQL Server.
7. ***SQL Server Reporting Service (SSRS)***
   Built analytical and tabular reports for students and exam performance.
8. ***Power BI***
   Created interactive dashboards for insights and visual analysis.
9. ***Flask***
   Developed the web application for online exam taking and management.

These tools together supported the full workflow from database design to reporting and web-based exam delivery.

System Process & Methodology

The system follows a multi-tier architecture with clear separation of concerns:

1. Database Design
2. Database Implementation and logic
3. Data Preparation and Loading
4. Reporting and Visualization
5. Web Application

To explain how the system was developed end-to-end, each phase of the architecture is described separately. The following sections break down the methodology into clear stages, outlining the tasks, tools, and outputs involved in each phase. This structured approach ensures traceability from the early design steps to the final implementation of the web application.

1. Database Design

1.1 Entity–Relationship Diagram (ERD)

The process started by designing the conceptual model using Draw.io, where the ERD represents the database structure for the student management and examination system. It illustrates the main entities, their attributes, and how they are related.

1.1.1 Entity Descriptions

- ***Student***
  Student_ID (PK), Name, National_ID, Gender, Birthdate, Email, Password, Phone_Number, Faculty, University, Graduation_Year, GPA, Government
- ***Certificates***
  Cer_ID (PK), Name, Provider, Date
- ***Freelancing***
  FL_ID (PK), Platform, Date, Duration, USD
- ***Student Employment***
  Emp_ID (PK), Company, Job_Title, Emp_Type, Date, Salary
- ***Track***
  Track_ID (PK), Track_Name
- ***Intake***
  Code (PK), Name, Start_Date, End_Date
- ***Branch***
  Branch_ID (PK), Branch_Name, Location, Launching_Year
- ***Instructor***
  Instructor_ID (PK), Instructor_Name, Gender, Email, Password , Ins_Phone, Salary, Hire_Date
- ***Department***
  Dept_ID (PK), Dept_Name
- ***Courses***
  Crs_ID (PK), Crs_Name
- ***Topic***
  Topic_ID (PK), Topic_Name

- ***Exam***
  Exam_ID (PK), Title, Total_Marks, Start_Time, End_Time, Number_of_Questions, Date
- ***Questions***
  Question_ID (PK), Q_Head, Q_Type, Q_Choices, Question_Mark, Correct_Answer
- ***Solve relationship***
  Answer, Question_Mark (links Students and Questions)
- ***Has relationship***
  Hours (links Tracks and Courses)

1.1.2 **Relationships Description**

Relationship	Cardinality	Description
Student - Certificates	1:M	A student can have multiple certificates.
Student - Freelancing	1:M	A student can record multiple freelancing experiences.
Student - Employment	1:1	A student can record the first job after graduation.
Student - Track	M:1	Each student joins one track.
Student - Branch	M:1	Each student belongs to one branch.
Student - Intake	M:1	Each student belongs to one intake.
Student - Exam	M:N	Students can take multiple exams, Exams can be taken by multiple students.
Student - Exam - Questions	M:N	Students can solve multiple questions across exams.
Instructor - Department	M:1	Each instructor works in one department.
Instructor - Courses	M:N	An instructor can teach multiple courses, Courses can be taught by multiple instructors.
Department - Track	1:M	A department offers multiple tracks.
Track - Courses	1:M	Each track contains several courses.
Topic - Course	1:M	Each topic covers multiple courses.
Course - Exam	1:M	Each course can include several exams.
Exam - Questions	1:M	Each exam consists of multiple questions.
Intake - Track - Branch	M:N	Each intake open in multiple branches, and each branch offers multiple tracks.

1.2 **Logical Model**

This logical data model defines the structure of the database system. After applying the mapping rules. It organizes the entities and relationships between them while maintaining referential integrity and normalization.

Table	Description
Student	Stores student details including Faculty and Intake_Track_Branch relationships. PK: Student_ID FKs: Faculty_ID, Intake_Track_Branch_ID
Faculty	Contains information about students's faculties and universities. PK: Faculty_ID
Intake	Represents student batch periods. PK: Intake_ID
Track	Represents study specialization. PK: Track_ID FK: Dept_ID
Branch	Represents ITI branch infos and locations. PK: Branch_ID
Intake_Track_Branch	Bridge table linking Intake, Track, and Branch. PK: IntakeTrackBranch_ID
Department	Represents ITI departments that instructors & tracks belong to. PK: Dept_ID
Instructor	Stores instructor details and their assigned departments. PK: Instructor_ID FK: Dept_ID
Instructors_Courses	Bridge between Instructors and Courses. PK: (Instructor_ID, Course_ID)
Topic	Represents higher-level categories that group related courses PK: Topic_ID
Course	Represents individual courses delivered within specific tracks and topics. PK: Course_ID FK: Topic_ID

Track_Courses	Bridge between Track and Course with duration (hours) of each course in different tracks. PK: (Track_ID, Course_ID)
Exams	Stores exam details associated with courses. PK: Exam_ID FK: Course_ID
Questions	Questions bank contains questions for each course that have exams. PK: Question_ID FK: Exam_ID
Exam_Questions	Bridge between Exams and Questions. PK: (Exam_ID, Question_ID)
Question_Choices	Multiple choice options for each question. PK: Choice_ID FK: Question_ID
Student_Exam	Tracks student exam participation with student score in each exam. PK: (Student_ID, Exam_ID)
Student_Exam_Questions	Stores each student's answer for every question in each exam. PK: (Student_ID, Exam_ID, Question_ID)
Certificates	Official certifications obtained by students during the program. PK: Certificate_ID FK: Student_ID
Freelancing_Jobs	Contains students' freelancing or project-based work data. PK: Freelancing_Job_ID FK: Student_ID
Employment	Tracks students' employment data after graduation. PK: Employment_ID FK: Student_ID

2. Database Implementation and logic

The SQL database was built in SQL Server Management Studio (SSMS) including:

- Creating tables, primary and foreign key constraints ensuring referential integrity, CHECK constraints to validate question types, exam status, and employment types. (Tables & Constraints.sql)

- Adding delete and update cascade rules were applied to maintain referential integrity across related tables. These cascades ensure that when a parent record is updated or removed, all dependent records are automatically adjusted, preventing orphaned data and keeping the database structure consistent and reliable. (Delete & Update CASCADES.sql)

- Adding indexes covering foreign keys and frequently queries columns to enhance query performance. (Indexes.sql)

- Implementing INSTEAD OF DELETE triggers to prevent accidental deletion from critical tables: tracks, intakes, branches, departments, topics and faculties. (Prevent Deletion Triggers.sql)

- Creating comprehensive stored procedures for CRUD (Create, Read, Update, Delete) operations across all tables. Following a consistent structure that ensures reliability and data integrity throughout the system. Following consistent error handling using TRY-CATCH blocks, Input validation checks, transaction management for data consistency and meaningful return messages. (CRUD Procedures Folder)

3. Data Preparation and Loading

3.1 **Data Sample**

Created initial data sample for each entity in the system manually in Excel.

3.2 **Data Generation**

Used Faker python library to generate larger datasets with all relationship/ bridge tables based on the data sample to simulate realistic scenarios, enforcing integrity constraints and consistency between tables.

3.3 **Data Loading**

Used SQL Server Integration Service (SSIS) as the ETL tool to extract data from Excel files, transform and match SQL data types, and loaded into SQL Server tables.

4. Reporting and Visualization

4.1 SQL Server Reporting Service (SSRS)

The system was integrated with SSRS to design analytical reports for tracking and analyzing student performance, exam outcomes, instructor activity, course progress, and overall academic trends, providing clear insights that support monitoring, evaluation, and decision-making.

These reports are powered by a set of specialized stored procedures that retrieve essential information from the database, including detailed student lists by department, course-level grade summaries, instructor course assignments with student counts, topic-based course structures, full exam question banks, and student responses with answer correctness. Together, these stored procedures form the data foundation that enables SSRS to generate accurate, dynamic, and meaningful reports for stakeholders across the institution.
(SSRS Stored Procedures.sql)

4.2 Power BI

Power BI was used to develop the analytical dashboards required for visualizing student performance, exam results, departmental insights, and overall institutional trends. The end-to-end workflow followed a structured approach beginning with data preparation and continuing through modeling, measure creation, visualization, and deployment.

4.2.1 Data Source

The primary data source for the dashboards was the SQL Server database, connected through a dedicated Semantic Model that handled all transformations and modeling before being published to Power BI Service. This ensured a single source of truth and maintained full data governance across all team members reports.

4.2.2 Data Transformation

Data transformation was performed in Power Query, where unnecessary columns were removed, data types were standardized, and a calculated column was created to support specific reporting needs. These transformations helped enhance data quality, improve model performance, and maintain consistency across all relationships.

4.2.3 Data Modelling

The data model was designed using a Snowflake Schema to support analytics scalability while preserving clear separation between fact and dimension tables. Additional supporting structures were created, including a DimDate table for time intelligence and a ToolTip table for interactive visual enhancements. The instructor table remained a standalone dimension and was combined logically with the Department dimension only when required, ensuring the model did not shift toward transactional complexity. (Power BI Model.png)

4.2.4 DAX Measures

A comprehensive set of DAX measures was developed to uncover insights across various analytical domains, including student analytics, exam performance, track comparisons, intake monitoring, department and instructor evaluation, and course-level metrics. These measures served as the foundation for all KPI calculations and comparative visualizations within the dashboards. (DAX Measures.pdf)

4.2.5 Dashboards

Twenty interactive dashboards were designed and built, each focusing on a specific analytical scope. They incorporated drill-through navigation, custom tooltips, comparative charts, trend indicators, and a unified visual theme to ensure clarity and ease of interpretation. The dashboards provided detailed insights for students, exams, tracks, intakes, departments, instructors, and courses. (ITI Dashboards.pbix)

4.2.6 API & Power App

Upon completion, the Semantic Model and all dashboards were published to Power BI Service within a dedicated workspace. An additional API-based report was developed and deployed alongside the project reports. Finally, a Power BI App was created to consolidate all reports into a single, structured, and user-friendly reporting hub, making it easier for stakeholders to navigate, explore insights, and support decision-making. (API.pbix) (Power App link.txt)

5. Web Application

The web application was developed using the Flask framework to provide an interactive and user-friendly interface for both students and instructors. The system supports secure authentication, role-based access management, real-time exam interaction, and full integration with the SQL Server backend and reporting layers.

5.1 Authentication System

A role-based authentication mechanism was implemented to differentiate between student and instructor access. Each user signs in through a dedicated login flow, and session management is used to maintain user state and ensure secure navigation throughout the system.

5.2 Student Interface

The student interface is designed to offer an intuitive experience for exam participation and performance review. Students are provided with a dashboard displaying their available exams and grade summaries. During exam sessions, they interact with a timed examination environment that supports question navigation and automatic submission. After completing an exam, students can view detailed results, including correct and incorrect responses. The interface also includes a dedicated page for ITI branches information and their locations.

5.3 Instructor Interface

The instructor interface enables instructors to manage exams and monitor student progress. It provides a dashboard summarizing courses and enrollment counts, along with tools for creating new exams based on predefined parameters. Instructors can track exam status, review student results, and access analytical insights to support academic decision-making.

5.4 System Integration with Stored Procedures

The Flask application integrates directly with the SQL Server stored procedures that power the system's core functionality. Exams are generated dynamically through sp_Generate_Exam, and access control is managed in real time using sp_Start_Exam. Automated grading is handled through sp_Correct_Exam. This integration ensures accurate, consistent, and efficient system behavior
(Exam Procedures Folder)

5.5 System Workflows and User Journeys

The student journey begins with authentication, followed by a dashboard view of available exams. Once an exam is initiated, the system validates time availability, presents the full question set, records answers, and performs automatic grading before displaying results and performance analytics.

The instructor workflow includes secure authentication, access to course and enrollment data, exam creation using customizable options, monitoring of student submissions, and reviewing aggregated analytical reports.

Data Flow Architecture

The system follows a structured and efficient data pipeline that supports both development and operations. Data begins with Faker-generated synthetic records, which are exported to Excel and loaded into SQL Server through SSIS ETL packages. The web application consumes this data directly, while Power BI provides the analytical layer for visual insights.

Key Features and Innovations

Several technical innovations enhance system performance and reliability, including the use of table-valued parameters for efficient bulk operations, SHA-256 hashing for secure password storage, strict constraint-based validation, automated exam correction, and fair random question distribution algorithms.

Business Logic Enforcement

The system enforces several academic and operational rules, such as validating the four-month intake period, enforcing a maximum exam duration of two hours, limiting exams to 25 questions, checking real-time exam availability, and automatically calculating student scores and grades.

System Performance

Performance optimization was ensured through an indexing strategy tailored to large datasets, efficient stored procedure execution plans, and a responsive web interface that maintains low latency. The architecture is designed to scale effectively, supporting thousands of concurrent users without compromising stability.

Conclusion

The ITI Examination System represents a comprehensive educational technology solution that successfully integrates database management, web application development, and business intelligence analytics. The system demonstrates professional-grade software engineering practices while addressing real-world educational administration challenges.