National Teachers College
629 J Nepomuceno, Quiapo, Manila, 1001 Metro Manila
Bachelor of Science in Information Technology

Final Project Documentation in Data Stucture

QuizPoP: SDG 4 Quality Education

A Final Project
Presented to the Faculty of the
College of Information Technology

In partial fulfillment
of the Course Requirements for the degree
of Bachelor of Science in Information Technology

Submitted by:
Prodigo, Evander B.
2.2 BSIT
National Teachers College

Instructor:
Ms. Justin Loiuse R. Neypes

Date:
December 14, 2025

Table of Contents

I. Introduction

1.1. Project Overview & UN SDG Target

This student quiz and assessment project is an educational system developed to enhance learners' academic proficiency through structured, subject-focused evaluations. Its core purpose is to provide students with an accessible and systematic way to strengthen their understanding across major academic areas. The platform presents three levels of difficulty—basic, intermediate, and the Level 3 challenge—each designed to support a gradual progression of learning. This tiered format enables students to move from foundational knowledge to more complex and analytical tasks, encouraging not only memorization but also the development of higher-order thinking skills. By integrating quizzes in essential subjects such as Math, Science, English, Filipino, and Araling Panlipunan, the system ensures broad and well-rounded academic reinforcement.

A key feature of the platform is its ability to deliver immediate feedback and allow students to track their scores after completing each quiz. This score-tracking mechanism helps learners monitor their academic growth, recognize strengths, and identify areas requiring improvement. Such features promote self-awareness and encourage students to take an active role in their learning process. Alongside student-centered functionalities, the system also provides administrative capabilities, enabling administrators to manage accounts, oversee records, and evaluate overall student performance. These administrative tools support organized and transparent academic monitoring, helping maintain the system's effectiveness and reliability. Combined with its interactive and self-paced structure, the system fosters consistent academic engagement and builds confidence among learners.

Furthermore, this project strongly supports the United Nations Sustainable Development Goal (SDG) 4: Quality Education, which advocates for inclusive, equitable, and meaningful learning opportunities. By offering structured quizzes—including advanced Level 3 challenges—and tools that allow learners to self-assess, the system contributes to improved learning outcomes and promotes lifelong learning habits. Its digital and flexible design enhances accessibility, allowing students to learn at their own pace and adapt to their individual needs. Overall, the project contributes to a more engaging, effective, and inclusive educational environment, aligning with global standards aimed at improving educational quality and ensuring that every learner has the opportunity to succeed academically.

1.2. Problem Statement (What real-world problem does the app solve?)

Students today need accessible ways to practice and test their knowledge, yet many struggle because traditional study methods lack instant feedback, different difficulty levels, and tools to track progress. This makes it hard for learners to identify their weaknesses and build confidence. The quiz-based system solves this problem by offering an easy-to-use platform where students can take subject-based quizzes at multiple levels—including advanced Level 3 challenges—to practice anytime and support their academic improvement.

II. Requirements & Analysis

Functional Requirements (FR)

ID	Requirement	Alignment
FR1	Our system includes user authentication: the system must allow users (Admin/Student) to log in using their ID/Number and password.	Prelim DSA
FR2	Subject Progress & Quiz (Core Algorithm): System must present a lesson, conduct a 5-question quiz, record the score, and advance the student to the next difficulty level (max 3 levels).	Core DSA
FR3	Admin Data Analysis & Sorting: The Admin module must load all student records and progress, and then allow for sorting the student list based on their performance (percentage score) in a chosen subject.	Finals DSA
FR4	Student Report Card: System must generate and display a student's individual report card, showing scores, level status, and overall average.	User Interface

Non-Functional Requirements (NFR)

ID	Requirement Type	Metric
NFR1	Performance & Efficiency	Time Complexity: Achieved by using efficient O(N \log N)Merge Sort (mergeSort) for bulk data organization and O(\log N) Binary Search (binarySearchByID) for quick lookup, meeting the < 1 second target for 50+ records.
NFR2	Robustness & Error Handling	Input Validation: Ensured by checks like password minimum length (if (s.password.length() < 4)) and file opening checks (if (!infile)) to prevent crashes from bad input. Error messages are printed when issues occur (e.g., "X Invalid password.").
NFR3	Maintainability & Code Quality	Modular Structure: The system is divided into logical components (login.cpp, menu.cpp, admin.cpp). Data Abstraction is used extensively via structs (Student, Admin,

		Subject, ProgressRow) and helper functions (clearConsole, trim, convertProgress).

2.2. Data Requirements (Description of input data structure and size)

Requirement	Implementation and Justification
Input Data: Must handle an input file of at least 50 records.	Data is stored in dynamic C++ containers: std::vector<Student> and std::unordered_map<string, ProgressRow>. These structures efficiently load and manage the required 50+ records from students.txt and student_progress.txt.
Data Integrity: Input must be validated to prevent crashes.	File Handling: Includes checks (if (!infile)) for graceful error handling if data files are missing. Input Validation: registerStudent checks for valid password length and spaces. Login Checks: Login functions ensure input matches stored credentials before proceeding.

2.3. Complexity Analysis: Expected Time/Space complexity of the Core Algorithm (justify using Big O notation).

The core of the system is the quiz and progress-tracking feature.

- **Time Complexity:** O(1) per quiz question, because each subject has a fixed number of levels (3) and questions per level (5). Even for all subjects, the total number of operations is constant, so the core quiz algorithm runs in **O(1)** time.

- **Space Complexity:** O(1), as only the current quiz data and scores are stored in memory during execution.

- **Justification:** Since the number of subjects, levels, and questions is fixed, the core quiz operations do not grow with input size. Storing scores and progress requires minimal, constant memory.

III. Design Specification

1. Array (Static & Dynamic Arrays)

Arrays were chosen because student subjects and lesson levels have a **fixed and predictable size**. Each student takes a predefined number of subjects, and each subject contains exactly three levels with five questions per level. Arrays provide **fast indexed access**, which is ideal for quiz navigation and score computation.

Implementation Details:

- Static arrays are used inside structures such as:
    - LessonLevel.questions[5]
    - LessonLevel.options[5][4]
    - Subject.levels[3]
    - Subject.scores[3]
- These arrays allow direct access using indices during quiz execution and score calculation.
- Dynamic arrays (std::vector) are used when the number of students is unknown at runtime.

2. Structure (Struct) for Data Modeling

Justification:

Structures were used to **group related data together** and represent real-world entities such as Students, Subjects, Lessons, and Admin records. This improves readability, modularity, and maintainability of the system.

Implementation Details:

- struct Student stores student ID, name, and password.
- struct Subject stores subject name, levels, scores, and current progress.
- struct LessonLevel encapsulates lesson text, questions, answer choices, and correct answers.
- struct StudentRecord (admin side) aggregates student information with their progress data.
- Structs are passed by reference to functions to avoid unnecessary copying.

3. File Handling with Sequential Data Storage

Text files were selected as the persistent storage mechanism to simulate database behavior without requiring external libraries. This approach ensures **portability**, **human-readable data**, and **easy version control** for collaborative development.

Implementation Details:

- students.txt stores login credentials.
- admins.txt stores administrator credentials.

- student_progress.txt stores progress for all students in a single sequential file.

Each line in student_progress.txt follows the format:

- StudentID level s0 s1 s2 level s0 s1 s2 ...
- File streams (ifstream, ofstream) are used to read and write data line-by-line.
- String streams (stringstream) parse progress records efficiently.

4. Divide and Conquer Sorting (Merge Sort)

Justification:

Merge Sort was selected because it guarantees **O(n log n)** time complexity and works efficiently even when the dataset grows. This is ideal for sorting students based on subject completion percentage in the admin module.

Implementation Details:

- The student records are recursively divided into smaller sublists.
- Each sublist is sorted based on **subject completion percentage**.
- The sorted lists are merged in descending order.
- Implemented in the mergeSortRecords() function using vectors.
- Sorting is performed dynamically based on the subject chosen by the admin.

5. Binary Search Algorithm

Justification:

Binary Search was chosen to provide **fast and efficient searching** when locating a student by ID. Since student records are sorted beforehand, binary search reduces search time from linear to logarithmic complexity.

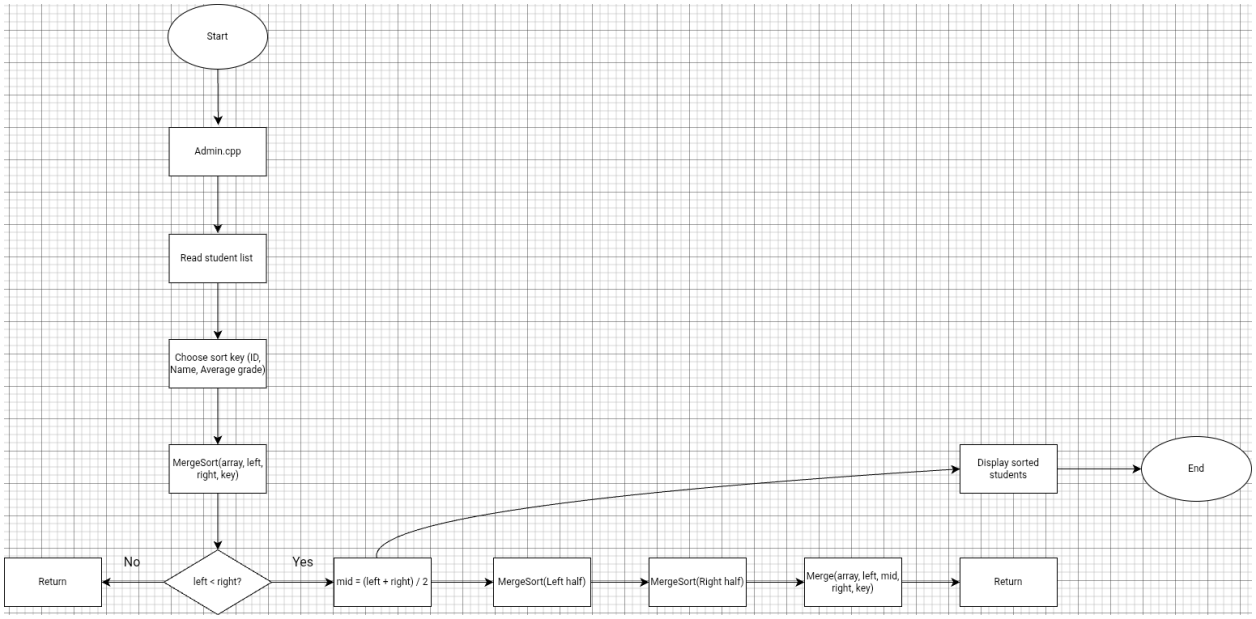
Implementation Details:

- Student records are sorted lexicographically by student ID.
- Binary search is implemented using index pointers (low, high, mid).
- The algorithm returns the index of the student if found, or -1 if not.
- Used in the admin module to retrieve and display a specific student's full report card.

Summary Table

DSA Concept	Purpose in System	Implementation
Array	Fixed lesson & score storage	Static arrays inside structs
Struct	Data modeling	Student, Subject, LessonLevel
File Handling	Persistent storage	Text files with stream parsing
Merge Sort	Student ranking	Divide-and-conquer sorting
Binary Search	Fast student lookup	Sorted ID-based search

3.2 Algorithm Flowchart including the most complicated algorithm.



3.3 Module Breakdown

This system is organized into multiple C++ modules, each responsible for a specific functionality. This modular design improves readability, maintainability, and scalability by separating concerns between authentication, student learning, progress tracking, and administrative reporting.

1. Login Module (login.cpp)

Purpose:

Handles authentication for both students and administrators.

Key Classes / Structures:

- Student
- Admin

Responsibilities:

- Loads student credentials from students.txt
- Loads admin credentials from admins.txt
- Validates login information
- Redirects users to the appropriate executable:
    - menu.exe for students
    - admin.exe for administrators
- Creates a temporary file (current_student_id.tmp) to pass the logged-in student ID to the student menu module

Interactions:

- Launches **Student Menu Module** upon successful student login
- Launches **Admin Module** upon successful admin login
- Reads credential files stored in the input data folder

2. Student Menu & Learning Module (menu.cpp)

Purpose:

Manages student learning sessions, quizzes, and progress tracking.

Key Classes / Structures:

- LessonLevel
- Subject

Responsibilities:

- Displays subject selection menu

- Presents lessons and quizzes per level
- Evaluates answers and updates scores
- Tracks subject progress (levels completed and scores)
- Displays student report card
- Saves and loads progress from student_progress.txt

Interactions:

- Reads student ID from current_student_id.tmp
- Loads and updates progress records in student_progress.txt
- Writes updated progress when student exits
- Operates independently per logged-in student

3. Progress Management Module (student_progress.txt + helper functions)

Purpose:
Provides persistent storage of all student progress data.

Key Data Structures:

- Sequential text-based records
- Parsed using stringstream

Responsibilities:

- Stores progress of all students in one centralized file
- Automatically creates a progress entry for new students
- Ensures data persistence between program executions

Interactions:

- Read and written by menu.cpp
- Read by admin.cpp for report generation
- Does not directly interact with user interfaces

4. Administrative Module (admin.cpp)

Purpose:

Allows administrators to analyze, search, and manage student performance.

Key Classes / Structures:

- Student
- SubjectProgress
- StudentRecord

Responsibilities:

- Loads student list from students.txt
- Reads progress data from student_progress.txt
- Displays individual and batch report cards
- Sorts students by subject completion using Merge Sort
- Searches students by ID using Binary Search
- Displays analytical recommendations based on performance

Interactions:

- Receives control from login.cpp
- Reads shared data files used by the student module
- Does not modify student learning data

5. Data Input Module (students.txt, admins.txt)

Purpose:

Stores static credential and identity information.

Responsibilities:

- students.txt: Student ID, name, and password
- admins.txt: Admin ID, name, and password
- Read-only during runtime (except registration)

Interactions:

- Accessed by login.cpp and admin.cpp
- Supports scalability by allowing new users without code modification

IV. Testing and Results

Test Case ID	Test Scenario ID	Test Scenario Description	Test Case	Test Case Description	Precognitions	Post Cognitions	Expected Results
STU_LOGIN_001	SCN_LOGIN_VALID	Login with correct student number and password	Verify that a registered student can log in successfully	1. Open the system 2. Go to Login page 3. Enter valid student number 4. Enter correct password 5. Click “Student Login” or Login button	Student is already registered in the system	Student is already registered in the system	Active student session created; user is on main student menu (with options for subject study and report card)
STU_LOGIN_002	SCN_LOGIN_INVALID_PASS	Login with valid student number and wrong password	Verify that user cannot log in with an incorrect password	1. Open the system 2. Go to Login page 3. Enter valid student number 4. Enter incorrect password 5. Click “Student Login” or Login button	Student is already registered in the system	No active session created; user stays on Login page	System shows an error message like “Invalid credentials ” and login fails
STU_REG_001	SCN_REG_SUCCESS	Register a new student with complete valid data	Verify that a new student account is created with valid details	1. Open the system 2. From main menu, click “Register” 3. Enter	Entered student number is not yet used in students.txt	New student record is saved to students.txt and can be used for	System confirms registration and allows login with the new credentials

				name, age, sex, address, birthday 4. Enter password 5. Enter unique student number 6. Click “Register” or Save		login	
STU_COURSE_001	SCN_COURSE_SELECT_3	Select exactly 3 different courses after login	Verify that the system accepts exactly three selected subjects	1. Log in with a valid student account 2. From student menu, go to course selection or subject selection 3. Select Math, Science, and English (or any 3 different subjects) 4. Click “Confirm” or “Save”	Student is logged in; list of available subjects is displayed	Three selected subjects are stored for that student	System saves the three chosen subjects and shows options to start lessons for those subjects
STU_STUDY_001	SCN_STUDY_ONE_COURSE	Complete lessons and quizzes for one selected course	Verify that a course can be completed by following the lesson–quiz loop	1. Log in and open one chosen subject from the student menu 2. Read the lesson content for that subject	Student has already selected at least one subject Subject is available in the student’s list	Course status for that subject is set to “Finished” with updated scores/levels	System records scores and marks the subject as completed, then returns the student to the remaining

				3. Take the quiz for the lesson 4. Submit the quiz to save the score 5. Repeat lesson and quiz steps as required until the subject is marked finished			subjects or main menu
STU_PROGRESS_001	SCN_VIEW_REPORT_CARD	View report card when all selected courses are finished	Verify that report card shows complete progress information	1. Log in as a student who has finished all selected subjects 2. From student menu, choose "View Report Card" 3. Review subjects, grades, completion status, and days taken/missed	All selected subjects for the student are marked finished	No data change; only viewing	Report card lists each subject with final grade/score, completion status, and days taken/missed
STU_CERT_001	SCN_ECERT_COMPLETE	Generate E-certificate when all courses are done	Verify that an E-certificate is generated and linked to the student	1. After finishing all selected subjects, go to completion or summary screen 2. Click	All subjects for the student are completed and progress data is available	E-certificate is generated and student record is marked fully completed	System displays or downloads an E-certificate for the student and stores that the student has

				“Generate E-certificate” 3. Confirm the request if prompted			completed the program
ADM_LOGIN_001	SCN_ADM_LOGIN_VALID	Login with correct admin ID and password	Verify that a registered admin can log in successfully	1. Open the system 2. Go to Login page 3. Click Admin Login 4. Enter valid admin ID 5. Enter correct password 6. Click Login	Admin account already exists in the system	Active admin session created; user is on Admin menu screen	System accepts credentials and redirects to Admin menu
ADM_LOGIN_002	SCN_ADM_LOGIN_INVALID	Login with valid admin ID and wrong password	Verify that admin cannot log in with an incorrect password	1. Open the system 2. Go to Login page 3. Click Admin Login 4. Enter valid admin ID 5. Enter incorrect password 6. Click Login	Admin account already exists in the system	No active session created; user stays on Login page	Error message like “Invalid admin credentials ” is displayed and login fails
ADM_STUDENTS_001	SCN_ADM_DISPLAY_ALL	Display list of all students	Verify that admin can view all registered students	1. Log in as admin 2. From Admin menu, choose “Display All Students”	At least one student is registered in the system (students.txt has records)	No data change; only viewing	System shows all student records stored in students.txt

				3. Observe the students list			
ADM_STUDENTS_002	SCN_ADM_SEARCH_FOUND	Search an existing student by ID	Verify that searching an existing student shows the record	1. Log in as admin 2. From Admin menu, choose "Search Student" 3. Enter a valid existing student ID 4. Confirm search	Student with that ID exists in students.txt	No data change; only viewing	System displays the student record and allows printing of the student report card
ADM_STUDENTS_003	SCN_ADM_SEARCH_NOT_FOUND	Search a non-existing student by ID	Verify that searching an unknown student shows "Not Found"	1. Log in as admin 2. From Admin menu, choose "Search Student" 3. Enter a student ID that does not exist 4. Confirm search	Student ID used does not exist in students.txt	No data change; only viewing	System shows a "Student not found" message and returns to Admin menu
STU_SUBJECT_001	SCN_STUDY_SUBJECT_FLOW	Study one subject using lessons and quizzes	Verify that the student can open a subject and update scores and status	1. Log in with a valid student account 2. From the menu, choose a subject (Math, Science, English, Filipino, AP, ESP, or	Student is already registered and logged in The chosen subject is part of the student's selected courses	Subject status for that student is updated (score, level, and completion flag)	System records the quiz score, updates the subject's status, and shows that the subject is completed when all required lessons/qui

				PE) 3. Read the lesson content shown for that subject 4. Answer the quiz items for that lesson 5. Submit the quiz to save the score 6. Repeat lesson and quiz until the subject is marked finished			zzes are done

V. Conclusion

In summary, this comprehensive student quiz and assessment system exemplifies an innovative and effective approach to enhancing educational accessibility, engagement, and self-directed learning. By incorporating tiered difficulty levels—basic, intermediate, and advanced Level 3 challenges—it allows students to progress gradually from foundational knowledge to more complex and analytical tasks, fostering higher-order thinking skills and deeper understanding. The system's features, such as immediate feedback, real-time score tracking, and detailed progress reports, address the common challenges students face with traditional study methods, enabling learners to identify their weaknesses, monitor their growth, and build confidence in their abilities.

The modular design of the platform promotes maintainability and scalability, with dedicated components for user authentication, subject-specific lessons, quizzes, administrative analysis, and data management. Its use of robust algorithms—such as merge sort for student ranking and binary search for quick data retrieval—ensures efficient performance even with increasing data volume. The choice of persistent storage through human-readable text files guarantees portability, easy updates, and transparency, making it accessible for schools and institutions with limited technical infrastructure.

Furthermore, the system's alignment with global educational standards and its support for inclusive, equitable learning directly contribute to the United Nations Sustainable Development Goal 4, which advocates for quality education for all. Its flexible, self-paced structure encourages lifelong learning habits and adapts to the individual needs of students, whether they are in remote or resource-limited settings. The administrative modules provide educators and administrators with powerful tools to analyze student performance, sort and search records efficiently, and generate comprehensive reports, thereby facilitating data-driven decision-making and targeted interventions.

Overall, this project not only fosters academic growth, confidence, and self-awareness among learners but also empowers educators with valuable insights for continuous improvement. It represents a significant step toward creating a more engaging, inclusive, and effective educational environment—one that leverages technology to bridge gaps, promote equity, and ensure that every learner has the opportunity to succeed academically. As educational challenges evolve, systems like this will be vital in shaping a future where quality education is accessible, personalized, and impactful for all.

Contributions

1. Prodigo, Evander B. - Programmer for Admin.cpp/owner of the main repository, and documentation for Design Specifications
2. Guillido, John Vincent - Programmer for Menu.cpp and the documentation for Requirements & analysis
3. De guia, Christianne Jade - Programmer for Login.cpp and the documentation for Introduction.
4. Agullo, Ralph - Tester, Flowchart Creator and writer for documentation for Algorithm flowchart and Testing & Results.