Comprehensive Exercise Report

Team Alone of Section 040

Ibrahim Tuncer

NOTE: You will replace all placeholders that are given in <<>>

[**Requirements/Analysis**](#_uwgqwd5ezv2w) **2**

[Journal](#_lsityg2iq9m6) 2

[Software Requirements](#_2h0vru1u2mla) 3

[**Black-Box Testing**](#_prhaxdxmf8n8) **4**

[Journal](#_18f11w613jft) 4

[Black-box Test Cases](#_2xn4jzot820y) 5

[**Design**](#_24fdizefyocn) **6**

[Journal](#_esp2ocs9j6bk) 6

[Software Design](#_aifbl1x6rddt) 7

[**Implementation**](#_hya8f3jqkba6) **8**

[Journal](#_acupzfhai7gz) 8

[Implementation Details](#_ojhtwkms2z3b) 9

[**Testing**](#_3qvya3vi836q) **10**

[Journal](#_ckfs4xbl5pyr) 10

[Testing Details](#_bzt1547yxzxi) 11

[**Presentation**](#_hdjvrbf45b1p) **12**

[Preparation](#_xbiquwtmf36n) 12

[**Grading Rubric**](#_u0hfnmdgusmf) **13**

# Requirements/Analysis

Week 2

## Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* After reading the client’s brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
  + ○ Project Description: The MasterScore application is a web-based platform (also can be implemented as a mobile application) designed to help students achieve higher scores on exams by providing personalized study plans, performance analysis, and progress tracking.
    - \* User accounts for students, teachers (optional), and parents (optional).
    - \* Login functionality with username and password.
    - \* Course enrollment for students.
    - \* Test score input for students.
    - \* Ability for teachers to create courses and add tests (optional).
    - \* Performance analysis with strengths, weaknesses, and recommendations for students.
    - \* Progress tracking towards goals for students (optional).
    - \* Secure data storage for user information and test scores.
* After reading the client’s brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
  + Will the parent user type be included in the initial development phase, or is this planned for a future release?

Yes, we want to involve the parents to this journey.

* If teachers are included, will they have the ability to manage student enrollment in their courses, or will this be handled by another system?

This will be handled by main system

* Will teachers be able to upload study materials for students to access within the platform?

Yes, exactly. Maybe the most interactive part of our project.

* + How detailed should the performance analysis be for students (e.g., breakdowns by topic, comparison to class averages)?

It will not be complicated. Every parent and of course every student must understand the improvement.

* + Will there be pre-defined goals students can choose from, or will they need to create their own custom goals?

It will manage by teachers because they can know what is the level of their students.

* + How will student progress be visualized within the application (e.g., progress bars, charts)?

We are planning to making a simple application that students can understand their level and improvement stage easily.

* Are there any other functionalities envisioned for the initial development phase besides those mentioned (course scheduling, communication tools)?
  + No, maybe we can add something in the proccess.
* Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  + I do not have a lot of experiences about developing applications.
* Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  + Primary, Middle and High school students.
* Describe how each user would interact with the software
  + Teachers will give some tasks to students and they will view of the students level, also they can warn them.
  + Also parents can follow the improvement of students.
  + Students can follow their improvement and also they can reliase what they are doing wrong.
* What features must the software have? What should the users be able to do?
  + The Software must be simple, so students, parents and teachers understand the situation easily.
  + It must be interactive, then the application can work effective.

## Software Requirements

**Project Description: MasterScore Application**

The MasterScore application is a web-based platform designed to empower students of all ages to achieve academic success. By leveraging personalized study plans, in-depth performance analysis, and comprehensive progress tracking, MasterScore equips students with the tools and insights they need to excel on exams. The application caters to students, with optional functionalities for teachers and parents, fostering a collaborative learning environment.

**Requirements:**

* **User Management:**
  + **Functional Requirement (FR-1):** The system shall create and manage user accounts for students.
  + **FR-2 (Optional):** The system shall create and manage user accounts for teachers.
  + **FR-3 (Optional):** The system shall create and manage user accounts for parents.
  + **FR-4:** All user accounts shall require login credentials (username and password) for secure access.
  + **FR-5:** The system shall implement appropriate security measures to protect user data (e.g., password encryption).
* **Course Management:**
  + **FR-6:** Students shall be able to enroll in existing courses.
  + **FR-7 (Optional):** Teachers shall be able to create new courses. (This functionality may be deferred to a future release based on client priorities.)
* **Test Management:**
  + **FR-8:** Students shall be able to enter their scores for exams and tests.
  + **FR-9 (Optional):** Teachers shall be able to create and manage exams/tests within their courses (including adding questions and setting scoring criteria). (This functionality may be deferred to a future release based on client priorities.)
* **Performance Analysis:**
  + **FR-10:** The system shall analyze student performance data to identify strengths and weaknesses for each student.
  + **FR-11:** The system shall generate personalized recommendations for students based on their performance analysis.
* **Progress Tracking:**
  + **FR-12:** The system shall track student progress towards their academic goals.
  + **FR-13:** Student progress shall be visualized in a user-friendly format (e.g., progress bars, charts). (Optional: FR-14 - Allow students to set and manage their own goals.)
* **Additional Considerations:**
  + **Non-Functional Requirement (NFR-1):** The system shall be responsive and user-friendly across various devices (desktops, tablets, smartphones).
  + **NFR-2:** The system shall ensure data privacy and security in accordance with relevant regulations.
  + **NFR-3:** The system shall be scalable to accommodate a growing user base.

**User Stories (Optional)**

# Black-Box Testing

Instructions: Week 4

## Journal

***Remember:*** Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

* What does input for the software look like (e.g., what type of data, how many pieces of data)?

**Data Types:**

* **Login Credentials:** Text (username/password)
* **User Information:** Text (name, grade level - optional), numbers (unique user ID - generated by the system)
* **Course Information:** Text (course name, code), numbers (credits - optional)
* **Test Information:** Text (test name, subject), numbers (score, maximum score)
* **Study Goals:** Text (descriptive text of the goal)
* **Communication (optional):** Text (messages between teachers and parents)

**Amount of Data:**

This will vary depending on the user's action and the specific functionality being used. Here's a breakdown by user type:

* **Student:**
  + Login: Username and password (2 pieces of text data)
  + Profile (optional): Name and grade level (2 pieces of text data)
  + Enrolling in a course: Course code (1 piece of text data)
  + Entering test scores: Test name, subject, score, and maximum score (4 pieces of data - text and numbers)
  + Setting goals: Descriptive text for each goal (variable amount of text data)
* **Teacher:**
  + Login: Username and password (2 pieces of text data)
  + Creating a course: Course name and code (2 pieces of text data) (Credits - optional: 1 number)
  + Adding tests: Test name, subject, and maximum score (3 pieces of text and number data)
  + Managing scores: Entering scores for each student (number - varies depending on class size)
  + Communication (optional): Text messages to parents (variable amount of text data)
* **Parent (optional):**
  + Login: Username and password (2 pieces of text data)
  + Viewing progress reports: Accessing pre-generated reports (no data input)
  + Communication (optional): Text messages to teachers (variable amount of text data)
* What does output for the software look like (e.g., what type of data, how many pieces of data)?

**Data Types:**

* **Text:** This will be the most common output type, including messages, labels, instructions, performance analysis, course descriptions, etc.
* **Numbers:** Scores, percentages, averages, ranks (optional), completion rates, etc.
* **Visualizations (optional):** Charts, graphs, or other visual representations of data (e.g., progress bars).

**Amount of Data:**

Similar to the input, the amount of output data will depend on the user's action and functionality:

* **Student:**
  + Dashboard: Overview of current courses, upcoming tests, progress towards goals (variable amount of text and numbers).
  + Performance analysis: Detailed breakdown of scores, strengths, weaknesses, personalized recommendations (text and numbers, optional visualizations).
  + Progress reports: Visualizations of progress over time, completion rates for goals (text, numbers, optional visualizations).
  + Learning materials (optional): Text, images, videos depending on the format of the materials.
* **Teacher:**
  + Class roster: List of students with basic information (text).
  + Grade reports: Scores for individual tests and overall performance for each student (text and numbers, optional visualizations).
  + Analytics: Analysis of class performance, identification of struggling students (text, numbers, optional visualizations).
  + Communication (optional): Text messages from parents (variable amount of text data).
* **Parent (optional):**
  + Progress reports: Pre-generated reports with student's performance data, teacher comments (text, numbers, optional visualizations).
  + Communication (optional): Text message threads with teachers (variable amount of text data).
* What equivalence classes can the input be broken into?
* **Login Credentials:**
  + Valid username and password combination.
  + Invalid username (doesn't exist in the system).
  + Invalid password (wrong password for a valid username).
  + Empty username or password field.
* **Text Fields (General):**
  + Empty text field.
  + Text field with only spaces.
  + Text exceeding the maximum character limit (if applicable).

**Students**

* **Course Information:**
  + Existing course code.
  + Non-existent course code.
  + Invalid course code format (e.g., incorrect length, missing characters).
* **Test Scores:**
  + Valid score within the expected range (e.g., 0-100 or based on your grading system).
  + Invalid score (negative value, characters, exceeding the maximum limit).
  + Missing score (empty field).
* **Study Goals:**
  + Specific, measurable, achievable, relevant, and time-bound (SMART) goals.
  + Unrealistic or vague goals.
  + Empty goal fields.

**Teachers**

* **Course Information:**
  + Unique course name (not already existing in the system).
  + Duplicate course name (already exists in the system).
  + Empty course name or code field.
  + Invalid course code format (e.g., incorrect length, missing characters).
* **Test Information:**
  + Unique test name within a course (not already existing for that course).
  + Duplicate test name within a course (already exists for that course).
  + Empty test name or subject field.
* What boundary values exist for the input?

Here's a breakdown of possible boundary values for the input data in your MasterScore application, categorized by user type:

**Common to All Users**

* **Text Fields (General):**
  + Minimum character limit (usually 0 or 1 character).
  + Maximum character limit (defined by your application's specifications).

**Login Credentials**

* **Username Length:** Minimum and maximum allowed characters for usernames.

**Password Length:** Minimum and maximum allowed characters for passwords.

**Students**

* **Test Scores:**
  + Lowest possible score (0 or minimum passing grade based on your system).
  + Highest possible score (100 or maximum score based on your system).
  + Values exceeding the maximum score limit (e.g., negative scores).

**Course Enrollment:**

* Minimum and maximum number of courses a student can enroll in (if applicable).

**Study Goals (optional):**

* Minimum and maximum character limit for goal descriptions (if applicable).

**Teachers**

* **Course Names:**
  + Minimum and maximum character limit for course names.

**Test Names:**

* Minimum and maximum character limit for test names.

**Scores Entered by Teachers:**

* Same boundary values as student test scores (lowest possible, highest possible, exceeding limits).
* Are there other cases that must be tested to test all requirements?
* **System Performance:**
  + Test application responsiveness under heavy loads (many concurrent users, large datasets).
  + Simulate network slowdowns or outages and see how the application behaves.
* **Security:**
  + Conduct penetration testing to identify vulnerabilities against unauthorized access attempts.
  + Test how the application handles sensitive data (e.g., passwords) and ensure proper encryption.
* **Error Handling:**
  + Simulate unexpected inputs (e.g., special characters in text fields, nonsensical data formats).
  + Test how the application handles server errors or database connection issues.
* **User Interface (UI):**
  + Verify the UI is intuitive and user-friendly for all user types (students, teachers, parents).
  + Test the application on different devices and screen sizes to ensure proper functionality and responsiveness.
* **Accessibility:**
  + Ensure the application is accessible for users with disabilities (e.g., screen readers, color blindness).
  + Test features like keyboard navigation and alternative text descriptions for images.
* **Business Logic:**
  + Verify calculations for scores, averages, and progress tracking are accurate.
  + Test how the application handles edge cases in grading systems (e.g., curved grading, extra credit).
* Other notes:

The application might also require user selection of options from pre-defined lists (e.g., selecting a course from a list of available courses).

File uploads (e.g., study materials) are a possibility, but depend on your specific functionalities. If your application allows file uploads (e.g., study materials), define equivalence classes for file size, format (e.g., document, image), and invalid/corrupted files

The application might offer downloadable reports (text and numbers) for further analysis.

Notifications (text) could be sent for important events like upcoming tests or low grades.

User IDs (generated by the system): These are typically unique identifiers and shouldn't require specific equivalence classes for testing purposes. However, you might want to consider testing how the application handles unexpected ID formats (e.g., negative values, non-numeric characters).

## Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Expected Results** | **Actual Results** |
| 0 | Inserting Score not between 0-100 | Application will throw a error message |  |
| 1 | Trying to Insert 3 or more parents to a student | Application will not give permission for it |  |
| 2 | Student try to submit after the due date | Application will throw a error message |  |

# Design

Instructions: Week 6

## Journal

***Remember:*** You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* List the nouns from your requirements/analysis documentation.
  + <<Insert answer>>
* Which nouns potentially may represent a class in your design?
  + <<Insert answer>>
* Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
  + <<Insert answer>>
* Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
  + <<List at least two design options with pros and cons of each>>
* Which design do you plan to use? Explain why you have chosen this design.
* List the verbs from your requirements/analysis documentation.
  + <<Insert answer>>
* Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

## Software Design

<<Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.****>>*

# Implementation

Instructions: Week 8

## Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

* What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

## Implementation Details

<<Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.>>

# Testing

Instructions: Week 10

## Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
  + <<Insert answer>>
* List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
  + <<Insert class>>
    - <<Insert needed tests>>
  + <<Insert class and tests for each class>>
* Other notes:
  + <<Insert notes>>

## 

## 

## Testing Details

<<Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.>>

# Presentation

Instructions:Week 12

## Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

* Give a brief description of your final project
  + <<Insert answer>>
* Describe your requirement assumptions/additions.
  + <<Insert answer>>
* Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
  + <<Insert answer>>
* How did the extension affect your design?
  + <<Insert answer>>
* Describe your tests (e.g., what you tested, equivalence classes).
  + <<Insert answer>>
* What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
  + <<Insert answer>>
* What functionalities are you going to demo?
  + <<Insert answer>>
* Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

<<Use your notes from above to complete create your slides and plan your presentation and demo.>>