

Learning Report – Applied System Development Life Cycle and Software Testing



L&T Technology Services



GLOBAL
ENGINEERING
ACADEMY

Genesis



Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be approved By	Remarks/Revision Details
		Name/PS No	Name/PS No	Module Owner Name	Comments
1	25/05/21	99004486	99004494	Thiruppathi Seenivasan	
2	25/05/21	99004489	99004486	Thiruppathi Seenivasan	
3	25/05/21	99004492	99004489	Thiruppathi Seenivasan	
4	25/05/21	99004493	99004492	Thiruppathi Seenivasan	
5	25/05/21	99004494	99004493	Thiruppathi Seenivasan	

Table of Contents

TABLE OF FIGURES.....	3
TABLE OF TABLES.....	4
SYSTEM/ SOFTWARE DEVELOPMENT	4
INTRODUCTION	4
MY PRODUCT: “QUIZ GAME LAUNCHER ”	6
SWOT ANALYSIS.....	7
REQUIREMENTS	7
DESIGN	8
HIGH LEVEL DESIGN	8
LOW LEVEL DESIGN	11
TEST PLANS	13
REFERENCES	15
APPENDIX:	17

Table of Figures

Figure 1 CLASS DIAGRAM(HIGH LEVEL)	10
Figure 2 USE CASE DIAGRAM (HIGH LEVEL)	11
Figure 3 ACTIVITY DIAGRAM (HIGH LEVEL)	12
Figure 4 USE CASE DIAGRAM (LOW LEVEL).....	12
Figure 5 ACTIVITY DIAGRAM (LOW LEVEL)	13
Figure 6 BLOCK DIAGRAM.....	13
Figure 7 COMPONENT DIAGRAM (HIGH LEVEL)	22
Figure 8 ACTIVITY DIAGRAM (high level).....	23
Figure 9 ACTIVITY DIAGRAM (LOW LEVEL)	24
Figure 10- ACTIVITY DIAGRAM (LOW LEVEL)	24
Figure 11 TEST PLAN.....	25
Figure 12 GIT	27
Figure 13 GIT ISSUES	28
Figure 14 GIT COMMITS 1.....	28
Figure 15 GIT COMMIT 2.....	29
Figure 16 GIT	30
Figure 17 GIT MAKE.....	31
Figure 18 GIT MAKE 2.....	31

Figure 19 GIT BUILD.....32

Figure 20 GIT CODE QUALITY32

Table of Tables

Table 1 HIGH LEVEL REQUIREMENTS.....6

Table 2 LOW LEVEL REQUIREMENTS.....6

Table 3 HIGH LEVEL TEST PLAN8

Table 4 LOW LEVEL TEST PLAN15

INTRODUCTION

SDLC- >Software Development Life Cycle

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The **SDLC** aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Various steps involved in software development life cycle

1. Planning and requirement analysis
2. Defining Requirements
3. Designing the Software
4. Developing the project
5. Testing
6. Deployment
7. Maintenance

1.Planning and requirement analysis:

It is the most important and fundamental stage in SDLC. In this stage inputs from the customer, sales department and domain experts were taken into consideration to develop the right software. It is very useful to plan the project.

Planning is done to select the right tools and technical approaches and resources which are necessary to do the project with minimum risks.

2.Defining requirements:

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS** document which consists of all the product requirements to be designed and developed during the project life cycle.

3.Designing the product architecture:

In this phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture. This design phase serves as input for the next phase of the model.

4.build:

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

5.Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

6.Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

7.Maintenance:

After the product is released for customers then it is to be maintained for the existing users.

Types of SDLC models

1. Waterfall Model
2. RAD Model
3. Spiral Model
4. V-Model
5. Incremental Model
6. Agile Model
7. Iterative Model

The most used model is agile model where products can be developed at a faster rate.

Agile model

The word agile means swift. Agile model is based on the iterative model. It break down tasks into smaller iterations called sprints. The project scope and requirements were determined at the beginning of the development process. The plan for the number of iterations, the duration and scope of each iteration is clearly defined in advance.

V Model

V model is also called as Verification and Validation Model. The model looks like V shape. It follows the sequential approach. Planning of testing is made parallel with the respective development stage.

Verification involves static analysis method where review is done without executing the code. This will ensure that product meets the required specifications.

Validation involves dynamic analysis where testing for functional and nonfunctional requirements are done by executing the code. Validation will ensure that software meets the customer requirements.

MY PRODUCT: QUIZ GAME LAUNCHER

Quiz Games are designed to test the memory, knowledge, agility, or luck of players competing for cash awards. In this quiz game, questions are asked to the player based on the topics he/she selected. If the player clears all rounds, then he is awarded a price money. The user's general knowledge is tested with quiz questions regarding science, technology, movies, sports, general health, geography and many more.

Here is a fun quizzing game designed using C Programming consisting of different topics. Only requirement is to achieve a minimum score to pass the quiz.

Quizzes to Answers



Steps to test your quizzing knowledge!!

Follow the steps below to download the program, and play the **quiz game launcher**

- Once the repository is public, clone or fork the project
- Open the 3_Implementation folder inside the **sdhc-team-19**
- For Windows Users, type "cmd" in the address bar where the folder is opened in your local machine.
- For Linux Users, Right Click on the "3_Implementation folder" and click on "Open in Terminal"
- This will open the command prompt with the current folder as its working directory
- Type "make" in the command prompt
- Once the files are compiled, type "make run".
- The game will start after the above instructions
- Read the details displayed on the screen and proceed accordingly
- Answer the correct questions and win the price.

SWOT ANALYSIS



Strengths

- The program is scalable in terms of number of topics and questions to be added.
- The scoring system adopted is fairly simple enabling easy calculation of winner.
- Timer addition limits the time to answer the question.

Weaknesses

- Program is not GUI based so every correct answer options needs to be typed.

Opportunities

- Converting program into GUI based game.
- Option of monitoring and keeping the highest score.

Threats

- No Graphic user interface buttons

REQUIREMENTS

Introduction

- Quiz Games are designed to test the memory, knowledge, agility, or luck of players competing for cash awards.
- In this quiz game, questions are asked to the player based on the topics he/she selected. If the player clears all round then he is awarded a price money.
- The user's general knowledge is tested with quiz questions regarding science, technology, movies, sports, general health, geography and many more.

Research

- The quiz game first gained popularity when it was aired on U.S. radio in the 1930s as an audience-participation program.
- Time magazine aired current events quizzes over the radio with The Pop Question Game, which lasted from 1923 to 1926
- American television adopted the quiz show in the early 1950s and further increased its popularity.
- Some popular quiz games are: Jeopardy!, Who Wants to Be a Millionaire, and Are You Smarter Than a 5th Grader? and a few more

Cost and Features

Cost

Since the system uses only open source software, it is free of cost!

Features

The various features of the Quiz Game are:

1. Topic Selection for Questions
2. Difficulty of questions increase as the player progresses
3. Cash Prize awarded for successful completion of the quiz
4. variety of topics available for quiz
5. life chances on incorrect answers to the quiz
6. help option to assist the contestant

Defining Our System

The Quiz comprises of 4 topics

1. LTTS company
2. General Knowledge
3. Sports
4. Science and Technology

Each genre comprises of **5 questions** and he/she has to get minimum of **4 questions correct out of 5** to pass the genre.

4W's and 1'H

Who:

Who are the stakeholders ?

1. Any Individual interested in quizzing.
2. Any Individual looking for fun time activity.
2. The quiz is based on general concepts and not specific to any particular domain.

What:

What does the project do ?

1. The project aims at a fun quiz program which is input/output based depicting the score of the participant.
2. The participant has to qualify a minimum score.

When:

When will it get completed ?

The Project will be completed by 24th May 2021.

Where:

Where is the project done ?

The Project is implemented on C program platform based on Input/Output.

How:

How to achieve the Target ?

To achieve the target

1. A good knowledge in file handling in C is a must.
2. Basic input output based functions
3. Proficiency in Structures and pointers.

Detail requirements

Table 1 - High Level Requirements:

ID	Description	Category	Status
HR01	User shall be able to input onto the terminal	Technical	Implemented
HR02	User shall be able to start the game	Technical	Implemented
HR03	User shall be able to get help	Technical	Implemented
HR04	User should see the questions on the terminal.	Technical	Implemented
HR05	The score should be get calculated correctly.	Technical	Implemented
HR06	Score should be displayed on screen.	Technical	Implemented
HR07	User should be able to exit.	Technical	Implemented

Table 2 - Low level Requirements:

ID	Description	HLR ID	Status (Implemented/Future)
LR01	User should be able to input the name	HR01	Implemented
LR02	User should be able to select start or exit game.	HR02	Implemented
LR03	Function to get help	HR03	Implemented
LR04	User should be able to select the option.	HR04	Implemented
LR05	Score should be calculate for correct answer	HR05	Implemented
LR06	Function to display score	HR06	Implemented
LR07	Function to start game	HR02	Implemented
LR08	Test cases for select answers	HR04	Implemented
LR09	Implementation Testing	HR01,HR02,HR03,HR04,HR05,HR06	Implemented

TEST PLAN

Table 3: High Level Test Plan

Test id	Description	Input	Exp output	Actual ouput	status	Type of test
H_01	Display correct questions based on topic selected	"L"or"S"or"G"or"T"	questions displayed	Question displayed	pass	Requirement based
H_02	working of help option	"H"	Instructions displayed	Instructions displayed	pass	Requirement based
H_03	working of game introduction	"S"	Instructions displayed	Instructions displayed	pass	Scenario based
H_04	correct quiz scoring	"A"or"B"or"C"or"D"	correct or incorrect(life-1) message	correct or incorrect(life-1) message	pass	Requirement based

Table 4: Low level Test Plan

Test id	High level Id	Description	Exp out	Actual output	status	Type of test
L_01	H_01	File IO - Try to read a wrong file	NO_FILE	NO_FILE	pass	Scenario based
L_02	H_04,H_01	File IO - Try to pass an invalid pointer	NULL_PTR	NULL_PTR	pass	Scenario based
L_03	Independent	File IO - reading correct file with valid pointer	SUCCESS	SUCCESS	pass	Scenario based
L_04	Independent	Entering wrong question pointer in the game	NULL_PTR	NULL_PTR	pass	Requirement based

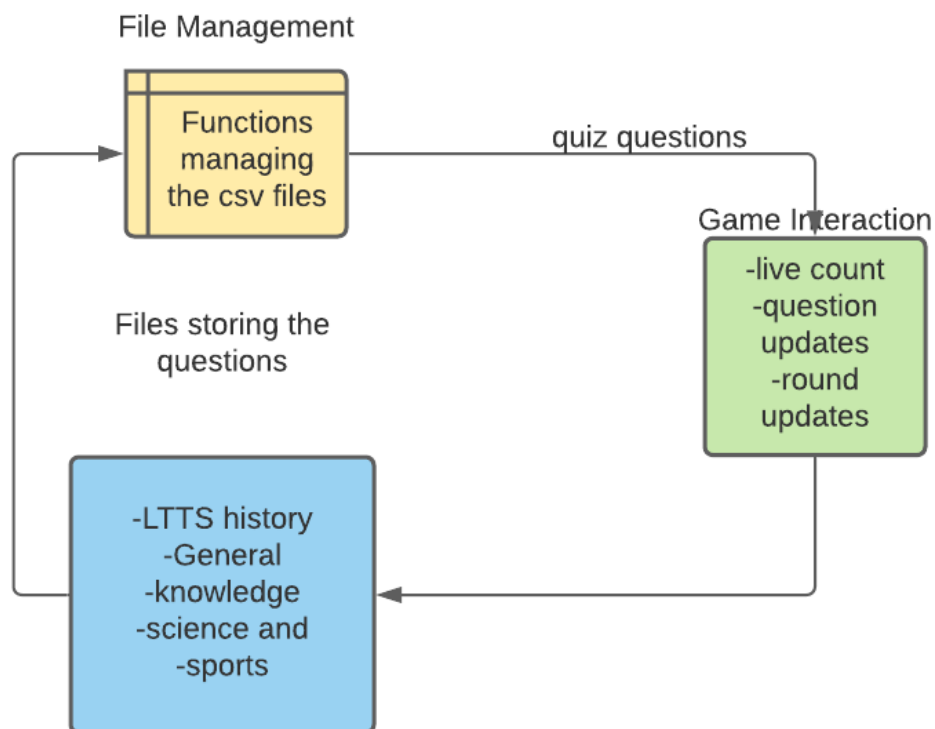
DESIGN

In the design system it is divided into high level and low level design. In the High Level Design, the product is explained in detail with help of Structural and behavioral Diagrams.

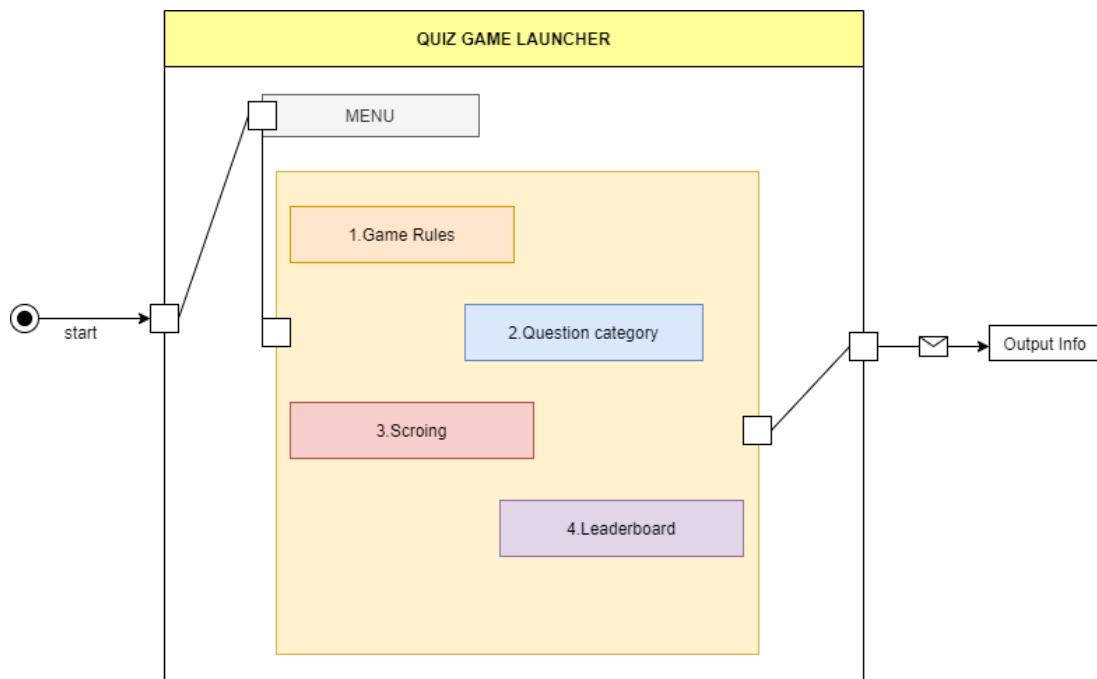
High Level Design

Structural and Behavioral Diagrams

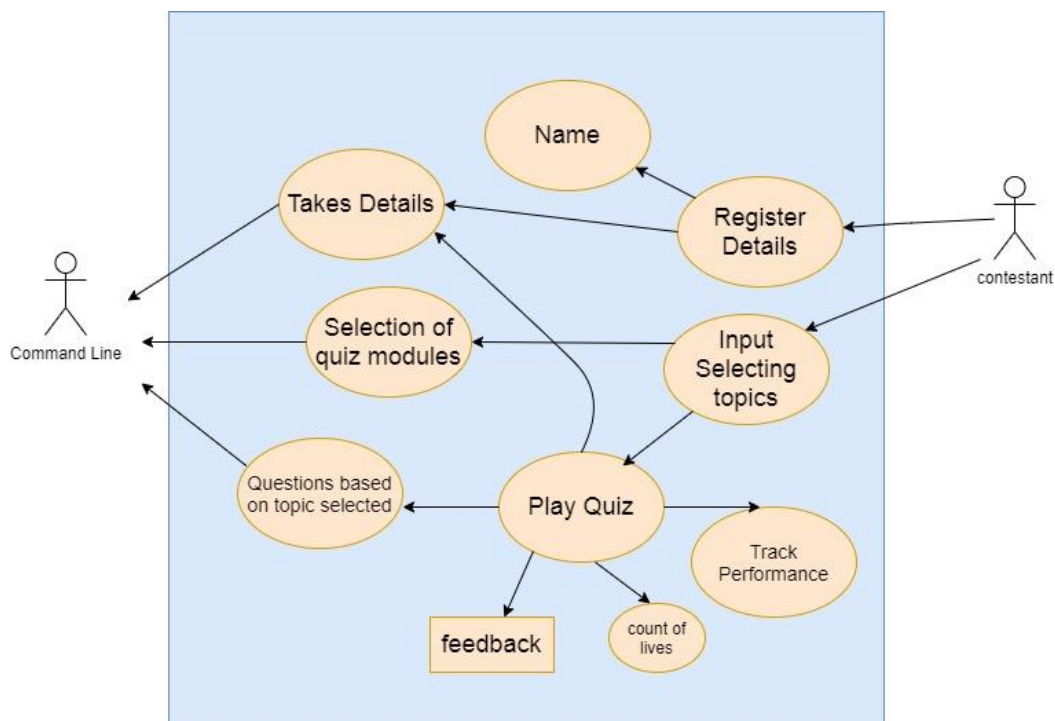
Component Diagram:



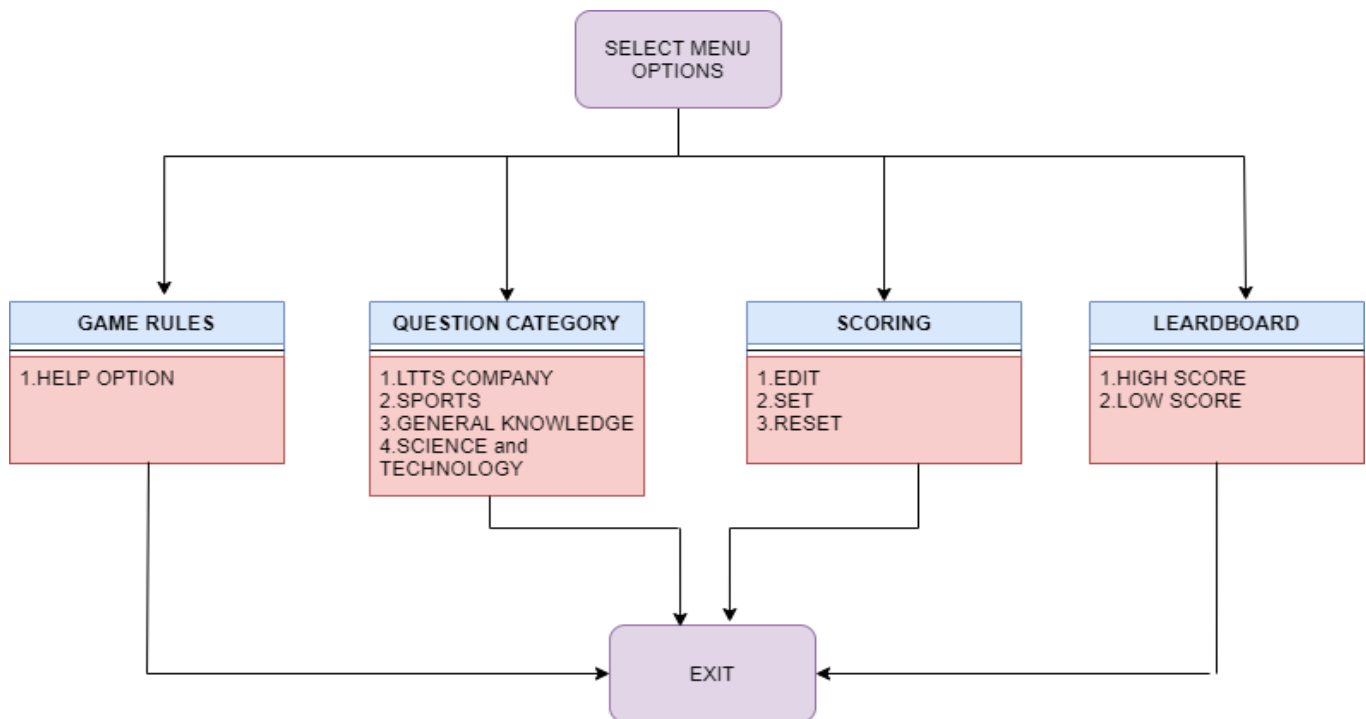
Composite Diagram:



Class diagram:



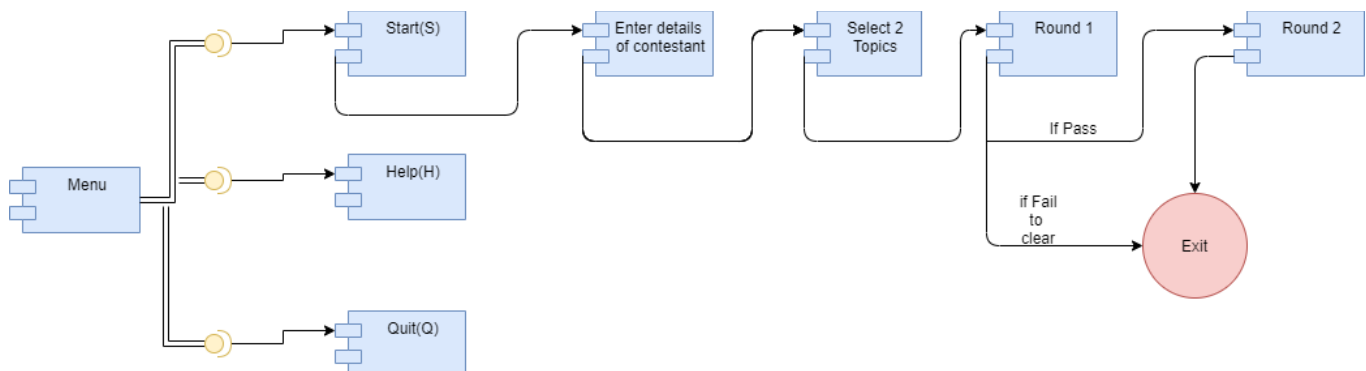
Use Case Diagram:



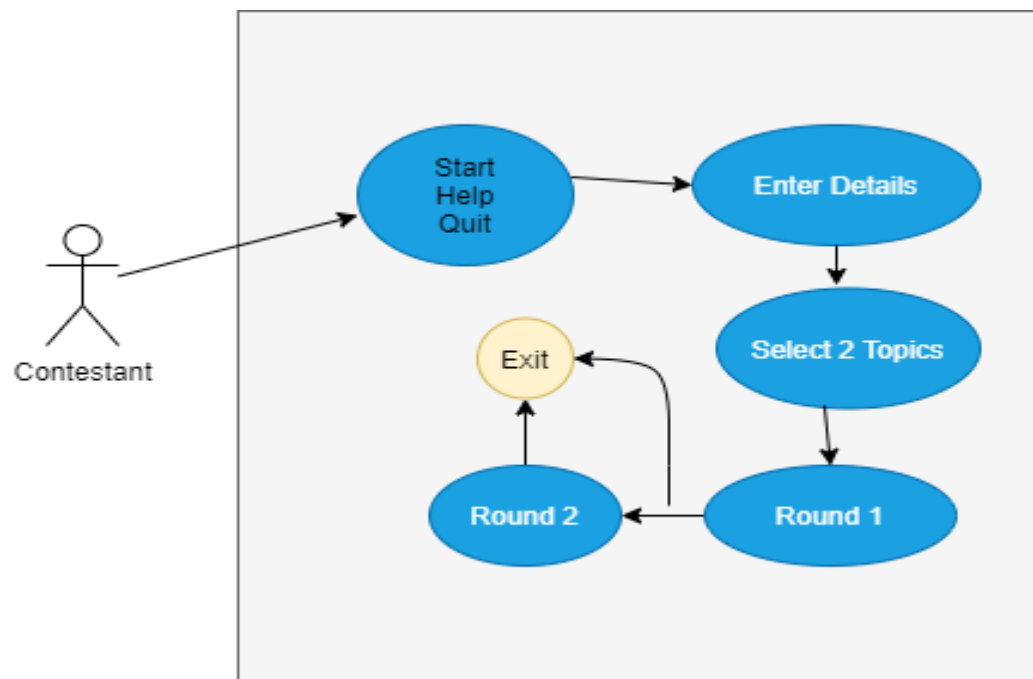
Low Level Design

Life tracking and Game:

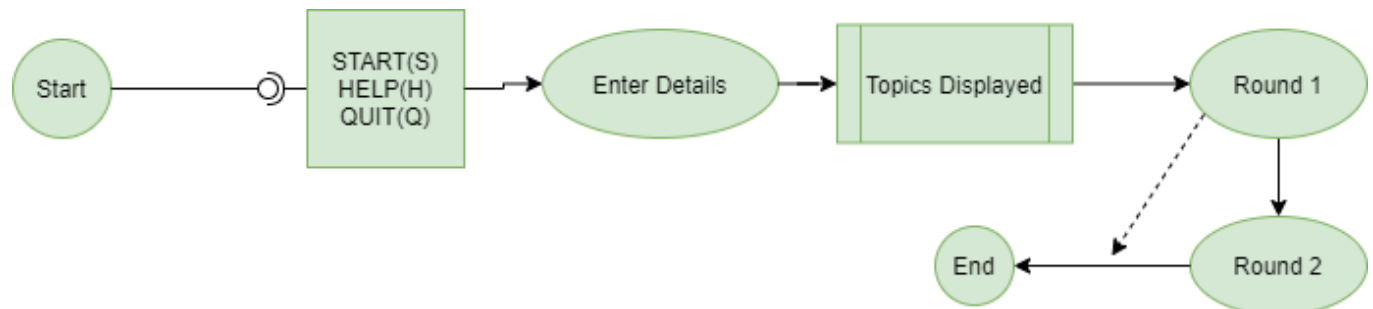
Composite Diagram:



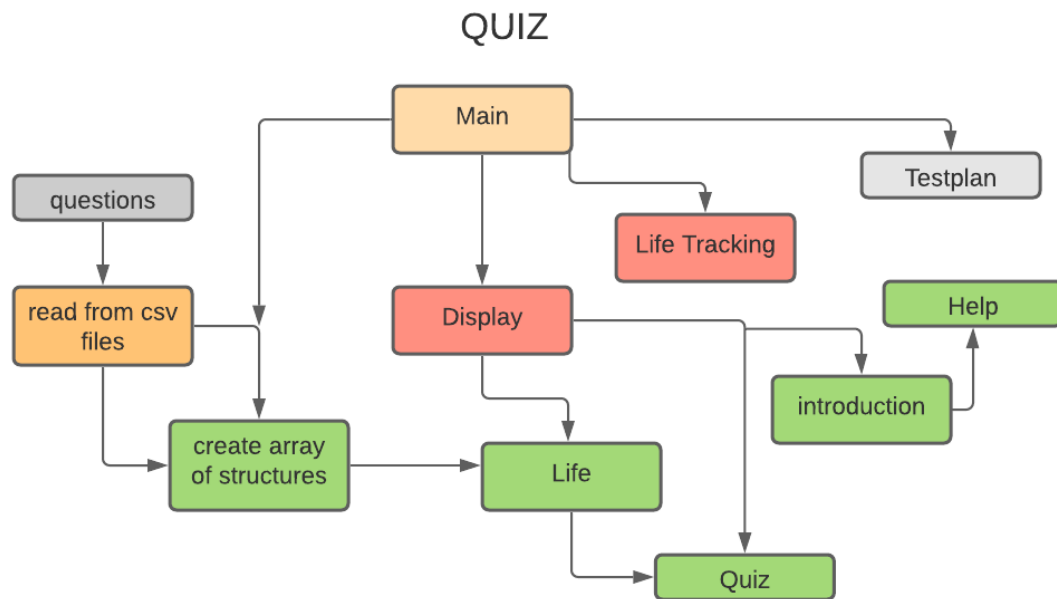
Use case Diagram:



Activity Diagram:

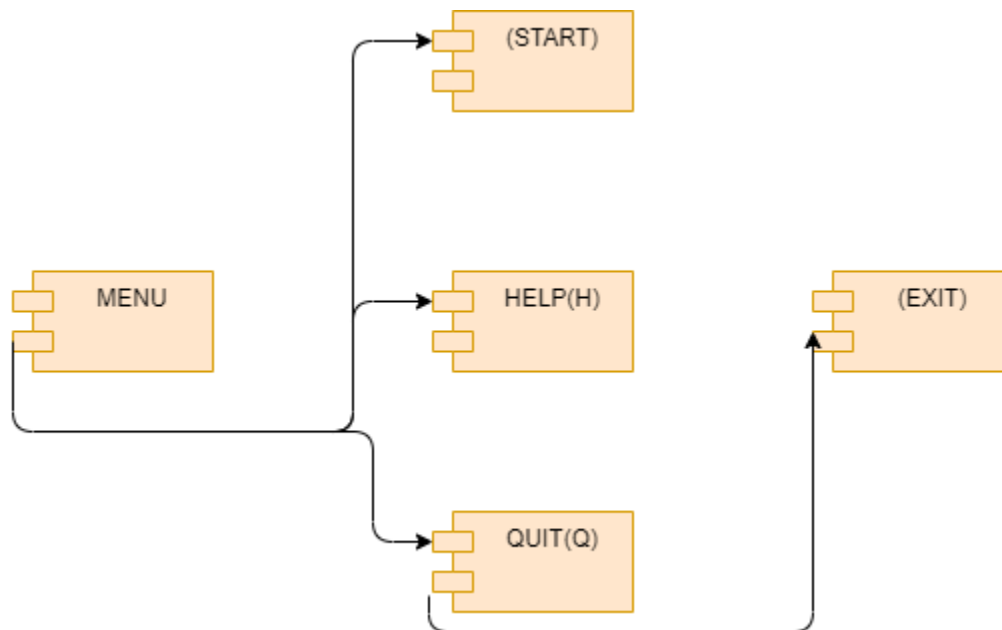


Package Diagram:

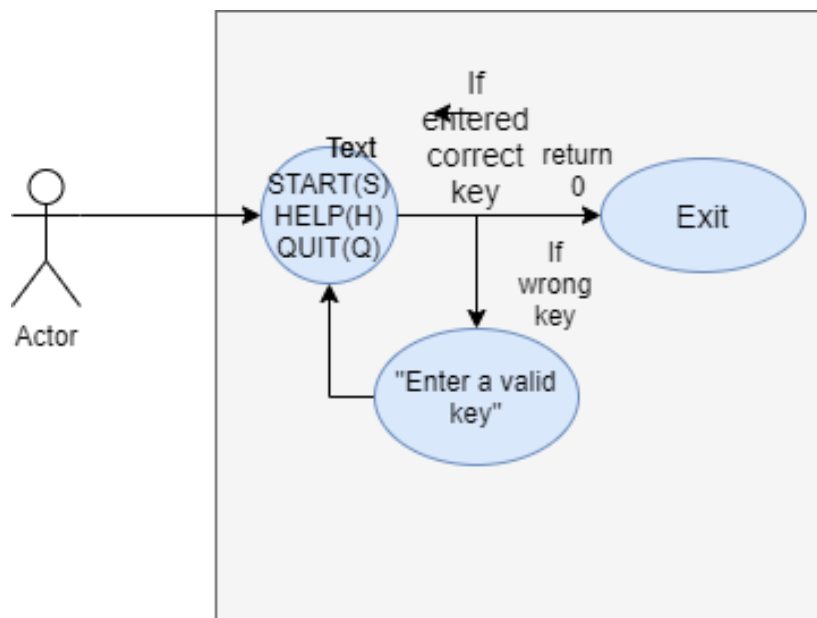


Quit Game:

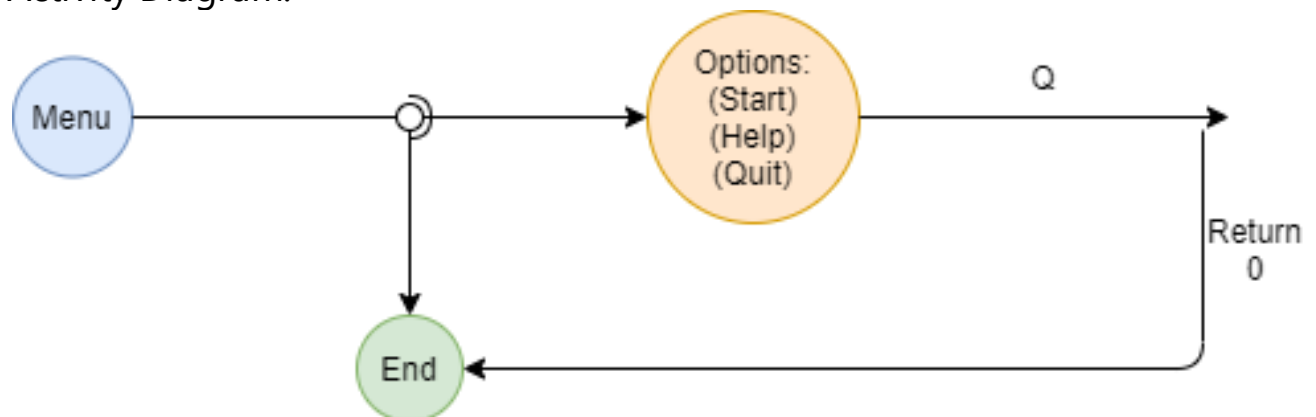
Composite Diagram:



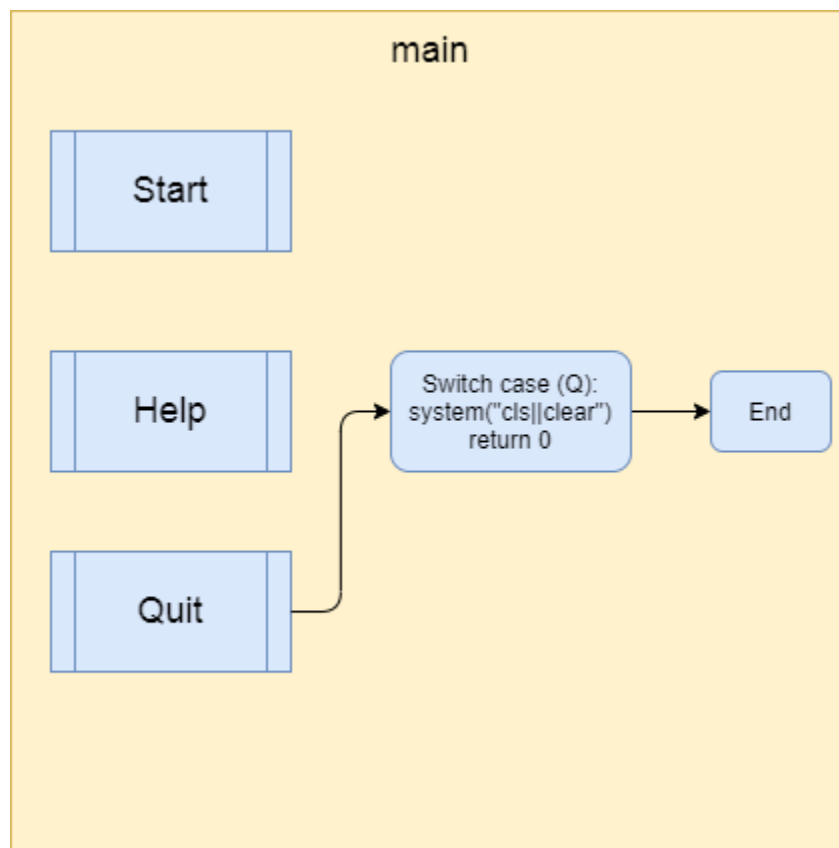
Use case Diagram:



Activity Diagram:

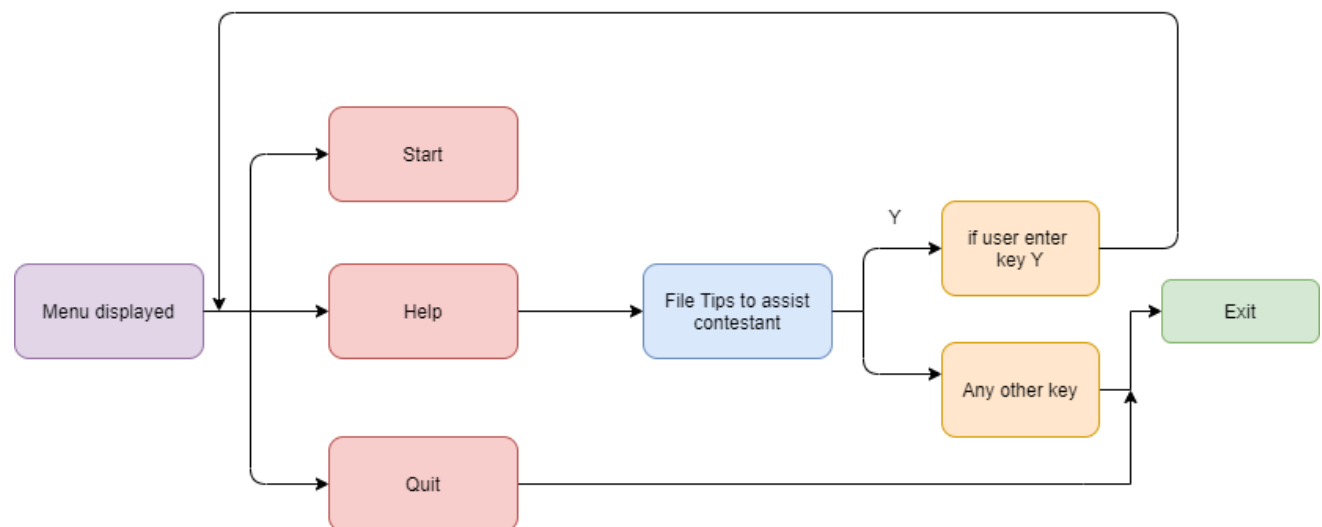


Package Diagram:

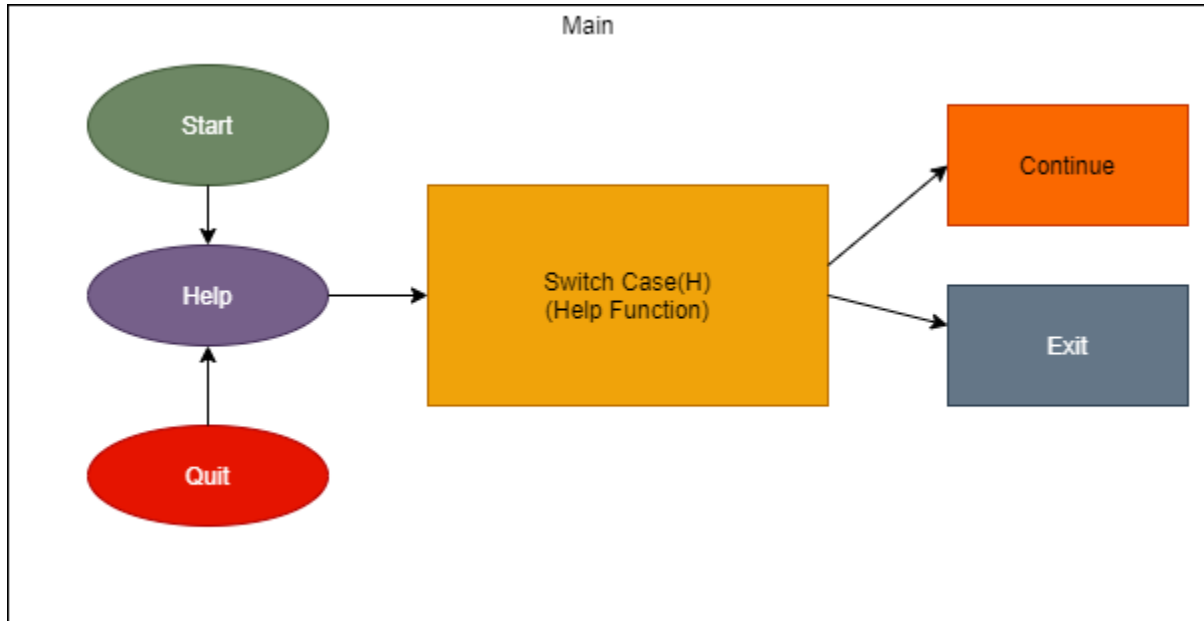


Help Feature:

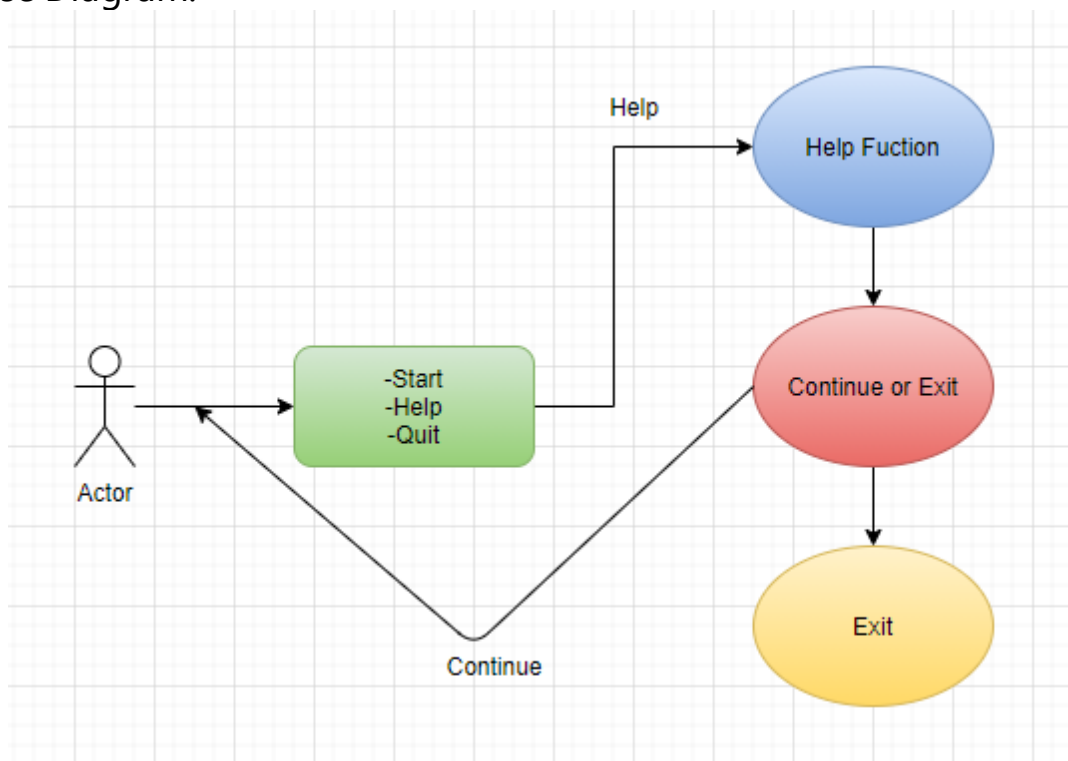
Class Diagram:



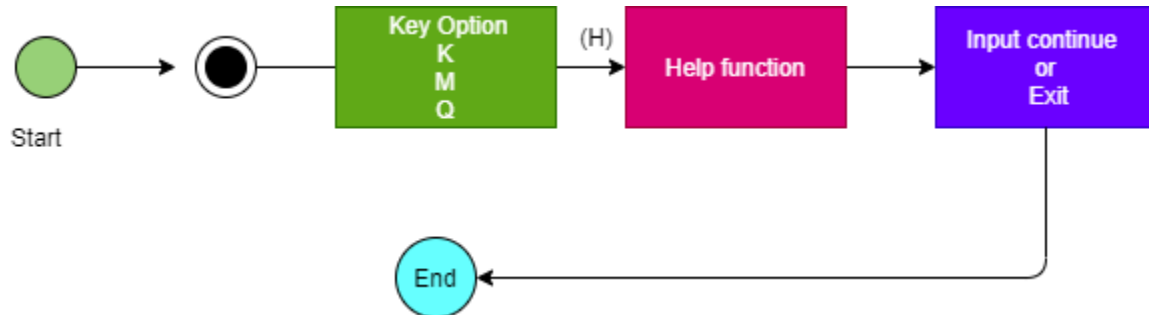
Activity Diagram:



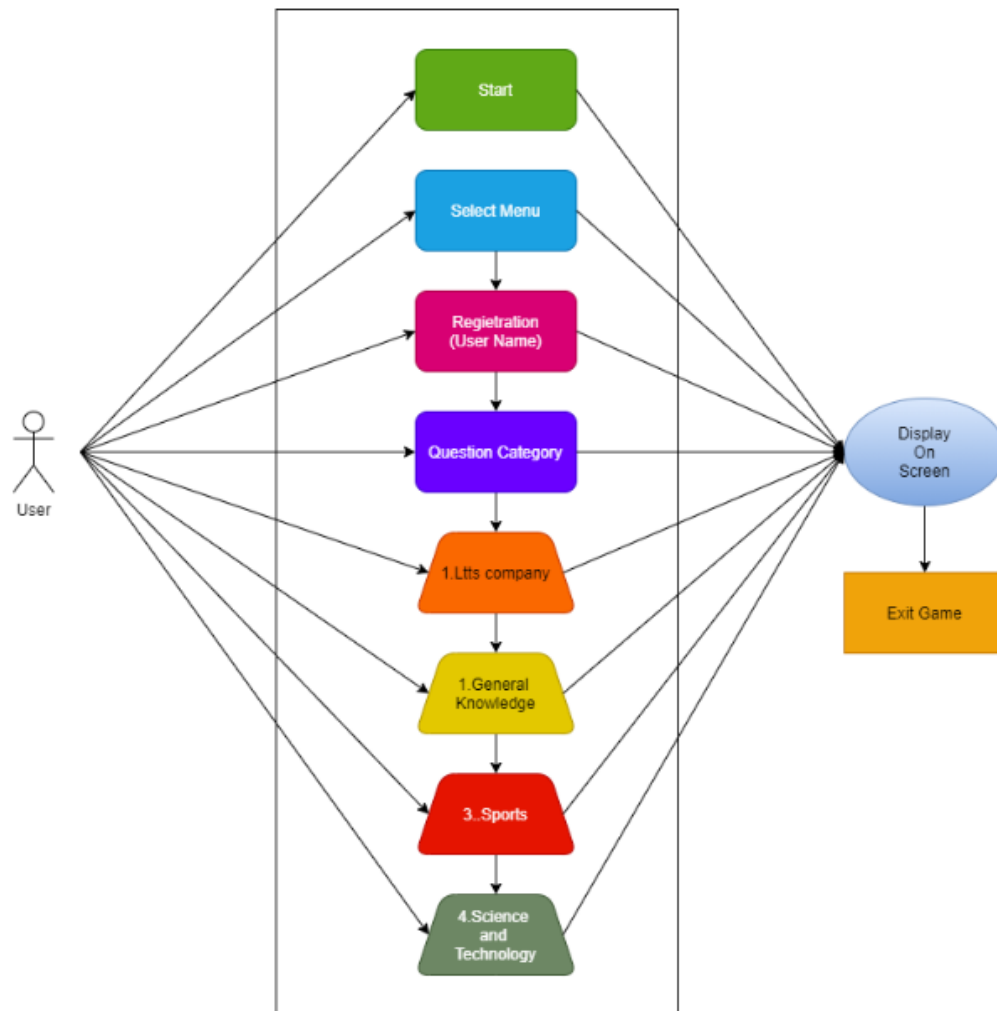
Use case Diagram:



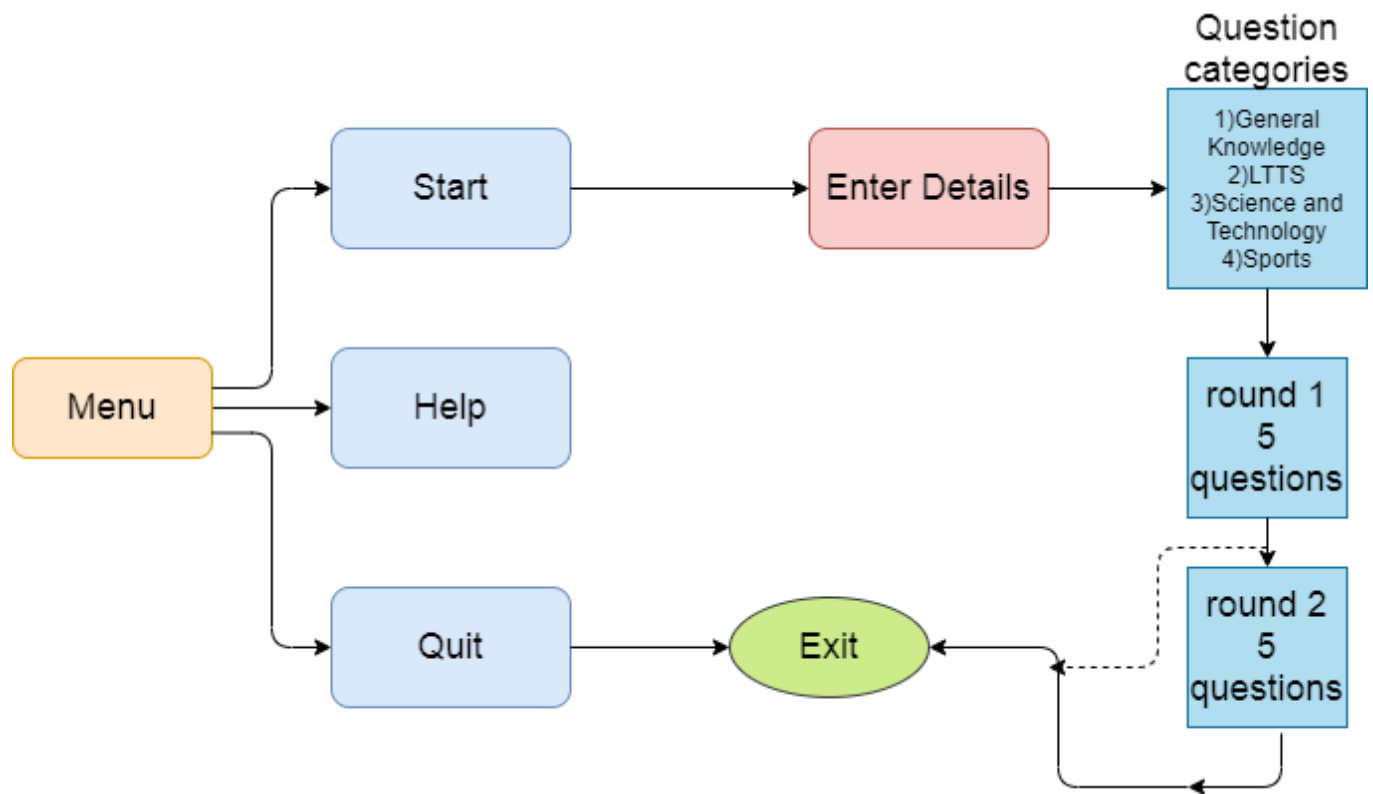
Composite Diagram:



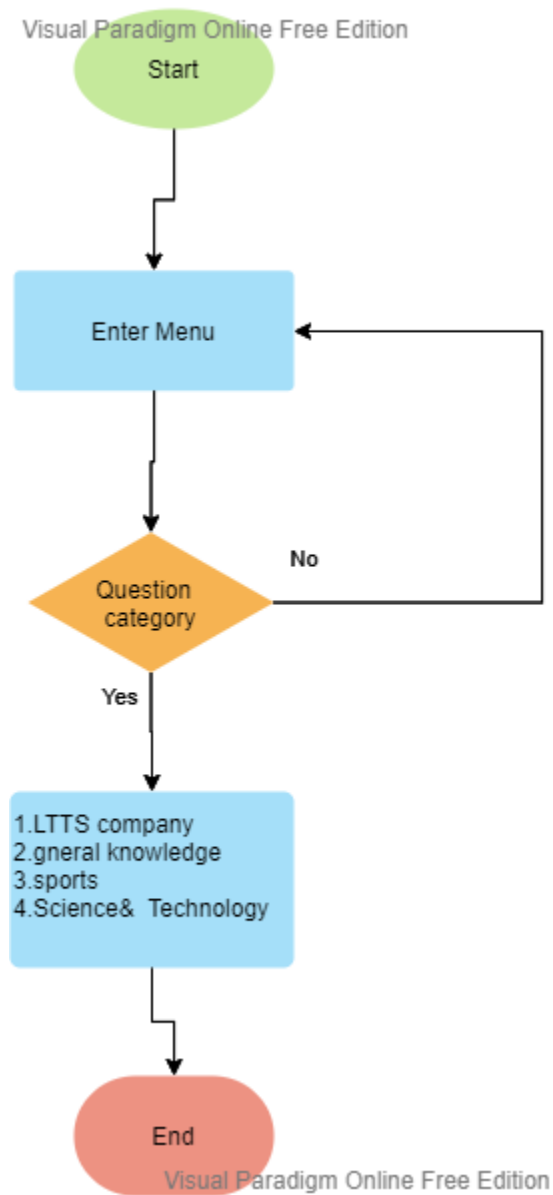
Question Category:



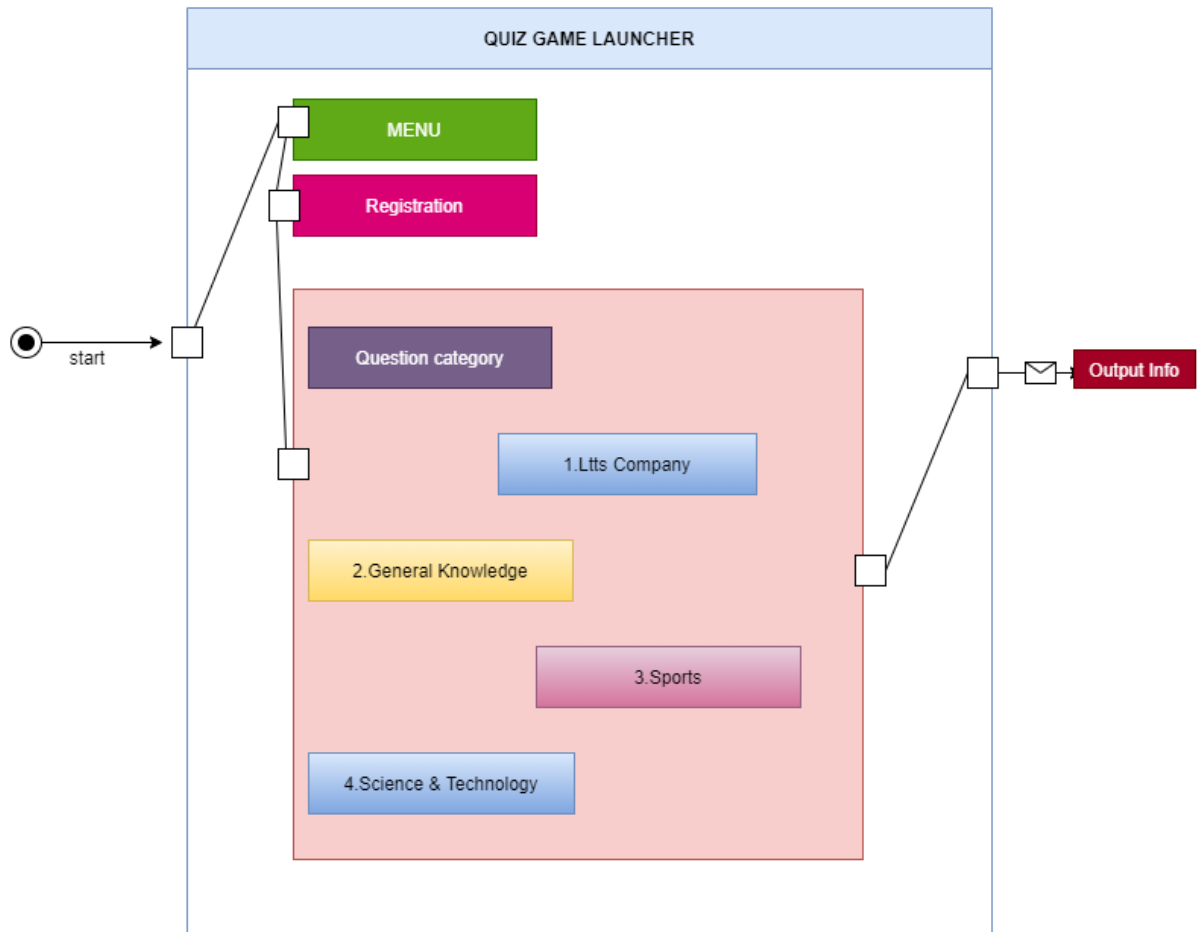
Class Diagram:



Activity Diagram:

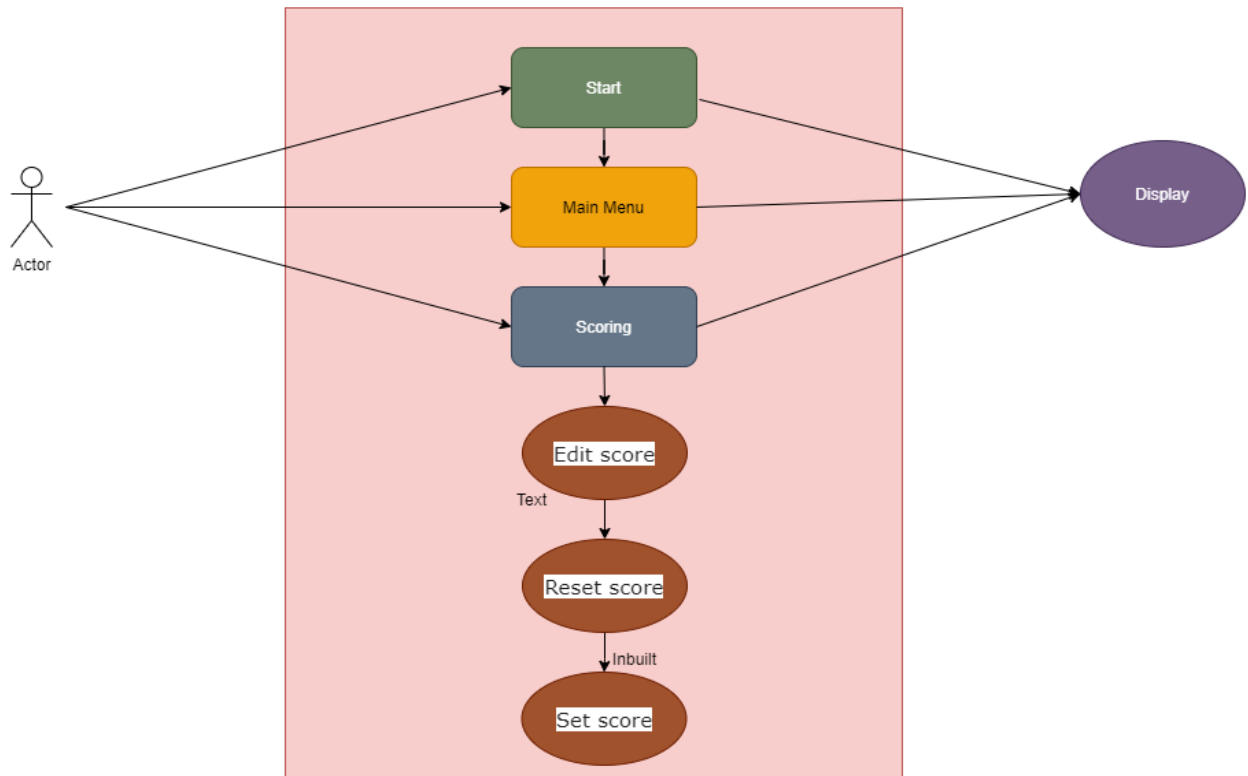


Composite Diagram:

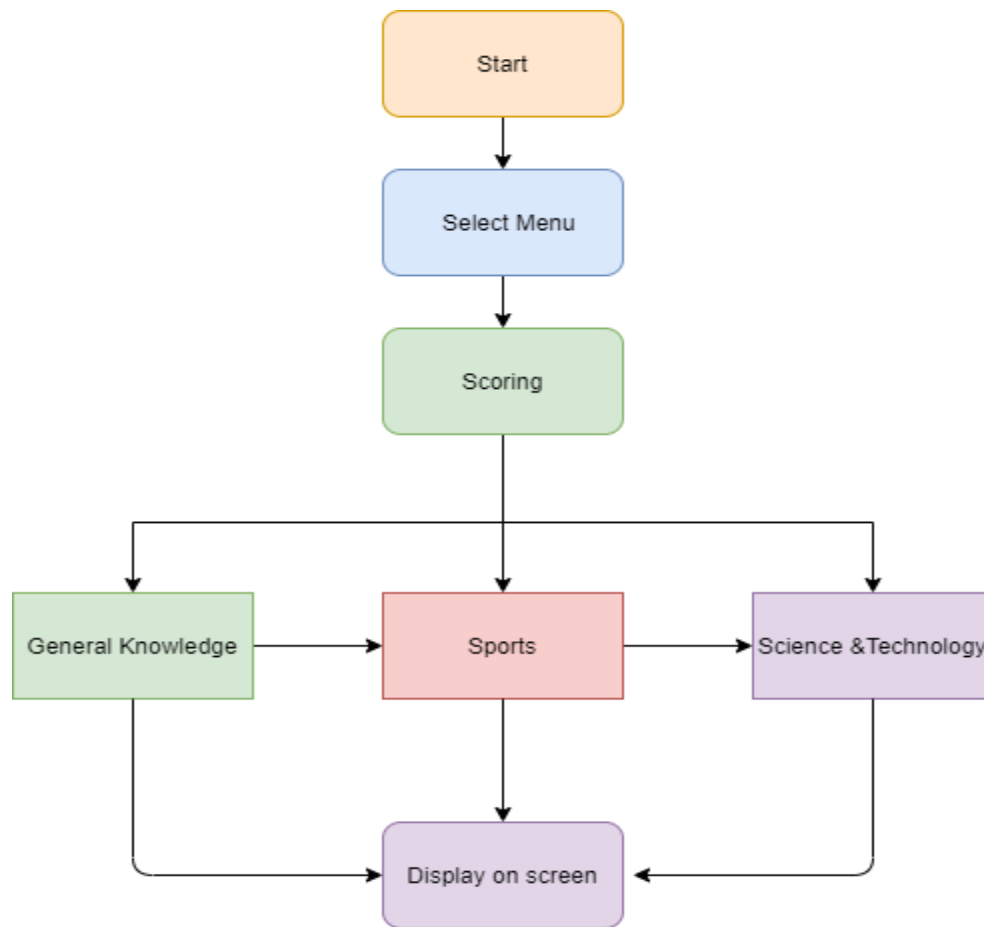


Scoring:

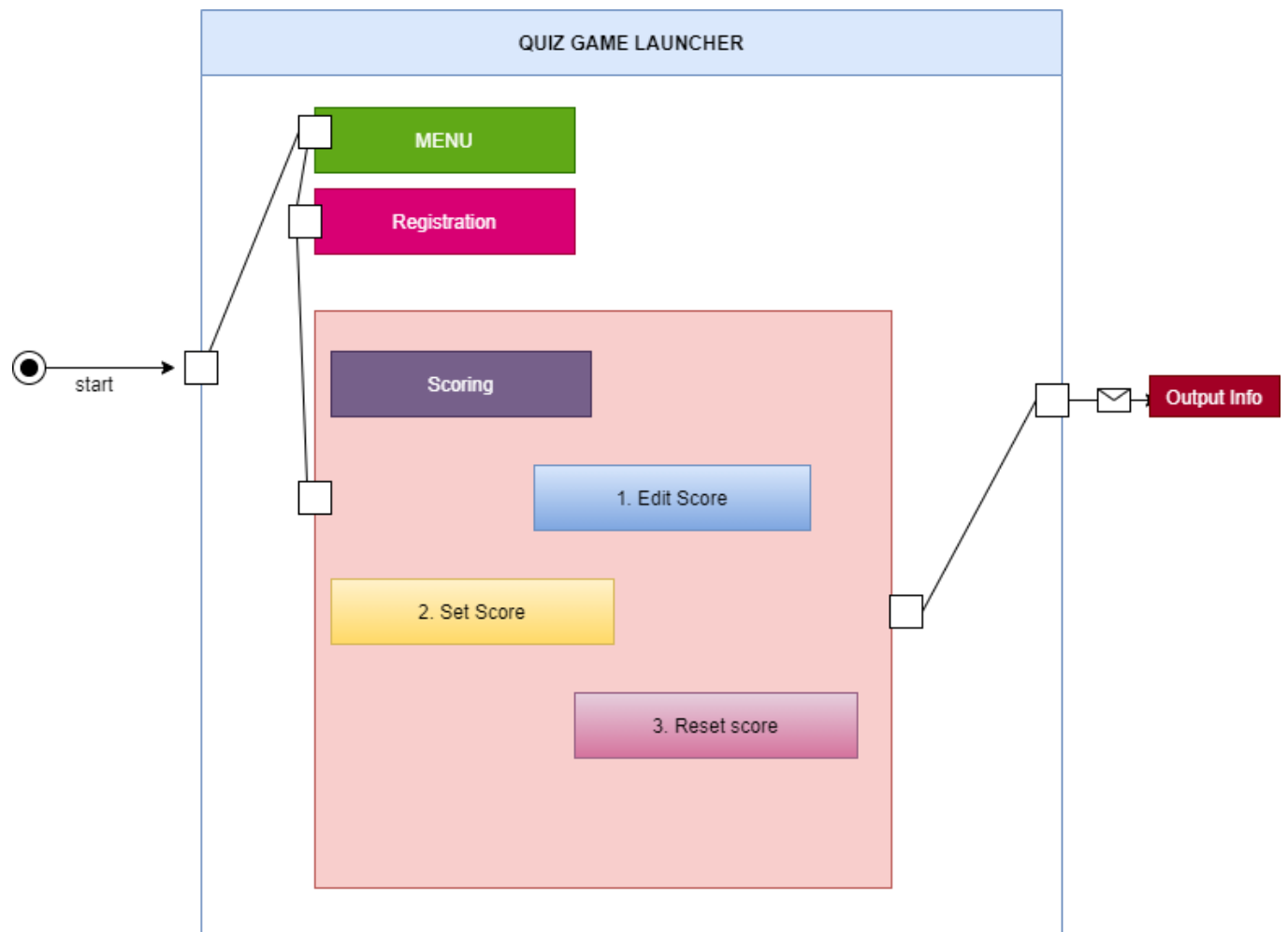
Use case Diagram:



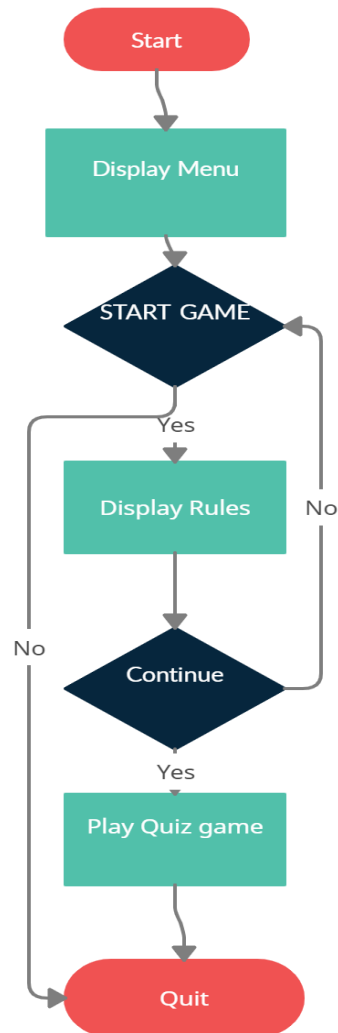
Class Diagram:



Composite Diagram:



Activity Diagram:



IMPLEMENTATION

In Implementation folder, we have subfolders which contains the header file(inc), source files(src), one file containing the documentation, a make file to run the code, a main.c file from where all the functions are called and executed. One folder called question bank is also created which has all the .csv files which contains questions of different topics.

In source files, we have a function help which will assist the user to play the game by displaying some tricks. A file containing function for the introduction of the game is also created.

Main.c file is used to authenticate user by asking for registration or login using credentials. Once the user is registered, he/she gets a option to start the game, quit the game or get help. On selected start, he/she see a display of topics out of which he/she has to choose 2 topics. Quiz will be conducted on those 2 topics. The user gets 2 lifes after which he/she is eliminated.

Implementation was done in the following way:

Step1: Implementation of individual code (header file and source file)

Step2: Integrating the code.

Step3: Setting up of workflows

Important blocks of code are mentioned below:

1) for reading the file:

```
error_t read_file(qna_t *question_array, char *filename)
{
    if(question_array == NULL)                //return null ptr ENUM if the array was not
    initialized.
    {
        return NULL_PTR;
    }
    FILE* fp = fopen(filename, "r");
    if (!fp)                                  // return no file ENUM if the file was unabl
    e to read.
    {
        return NO_FILE;
    }

    else
    {
```



```

    char buffer[200];                //buffer for fetching the file line by line
    int index=0;
    char *token;                    //token used to separate the values in between delimiters

    //this specific loop stores the questions into an array of structures passed by the function.
    while(index <= 4)
    {
        fgets(buffer,200, fp);
        strcpy((question_array+index)->question,buffer);

        fgets(buffer,200, fp);

        token = strtok(buffer, ",");
        strcpy((question_array+index)->option_1,token);

        token = strtok(NULL, ",");
        strcpy((question_array+index)->option_2, token);

        token = strtok(NULL, ",");
        strcpy((question_array+index)->option_3,token);

        token = strtok(NULL, ",");
        strcpy((question_array+index)->option_4, token);

        fgets(buffer,200, fp);
        strcpy((question_array+index)->answer, buffer);

        index++;
    }
}
fclose(fp);
return SUCCESS;
}

```

2) for playing the game:

```

error_t play(qna_t *question_array, char *name_of_participant, const int no_of_questions
)
{
    char choice;

```

```
int life = 2; //initial allocation of life
char ans[2];
if(question_array == NULL)
{
    return NULL_PTR;
}
for (int index = 0; index < no_of_questions; index++) //starting of individual round
{
    strcpy(ans, (question_array+index)->answer);
    system("cls || clear");
    printf("Name : %s\n", name_of_participant);
    printf("Lives : %d", life);

    printf("\n\n");

    printf("%s\n", (question_array+index)->question);

    printf("%s    ", (question_array+index)->option_1);
    printf("    %s\n", (question_array+index)->option_2);
    printf("%s    ", (question_array+index)->option_3);
    printf("    %s\n", (question_array+index)->option_4);

    printf("\n Please enter option : ");
    scanf("%c", &choice);
    getchar();
    choice = toupper(choice);

    //if the answer is correct
    if(choice == ans[0])
    {
        printf("\n congrats!, u r correct! The correct answer is option %s\n", (question_array+index)->answer);
        printf("Press Enter to continue ");
        getchar();
    }
    //if the answer is incorrect and then decrease life
    else
    {
        printf("\nopps! u r wrong!, The correct answer is option %s\n", (question_array+index)->answer);
        life--;
    }
}
```

```
        printf("Press Enter to continue ");
        getchar();

    }

    //checking for life
    if(life == 0)
    {
        system("cls || clear");
        printf("\nsorry u are dead!, try again next time\n");
        time_delay(3);
        return PLAYER_OUT;
    }

}
return SUCCESS;
}
```

Make file for implementation is as shown:

```
PROJ_NAME = Team_19
TEST_PROJ_NAME = Test_$(PROJ_NAME)

COVERAGE_TEST_NAME = coverage_$(PROJ_NAME)

SRC = src/game_intro.c src/create_question_array.c src/read_file.c src/please_help.c
src/play.c src/time_delay.c

TEST_SRC = ./test/test.c unity/unity.c

INC = -Iinc -Iunity

#To check if the OS is Windows or Linux and set the executable file extension and
delete command accordingly
ifdef OS
    RM = del /q
    FixPath = $(subst /,\,$1)
    EXEC = exe
    editor = notepad
else
    ifeq ($(shell uname), Linux)
        RM = rm -rf
        FixPath = $1
        EXEC = out
        editor = cat
    endif
endif
```

```
# Makefile will not run target command if the name with file already exists. To
override, use .PHONY
```

```
.PHONY : all test coverage run clean doc
```

```
all:
```

```
    gcc main.c $(SRC) $(INC) -o $(call FixPath,$(PROJ_NAME).$(EXEC))
```

```
run : all
```

```
    ./$(PROJ_NAME).$(EXEC)
```

```
test: $(SRC) $(TEST_SRC)
```

```
    gcc $(TEST_SRC) $(SRC) $(INC) -o $(TEST_PROJ_NAME).$(EXEC)
```

```
    ./$(TEST_PROJ_NAME).$(EXEC)
```

```
coverage:$(SRC)
```

```
    gcc -fprofile-arcs -ftest-coverage main.c -I -c $(SRC) -o
$(COVERAGE_TEST_NAME).$(EXEC)
```

```
    ./$(COVERAGE_TEST_NAME).$(EXEC) < input_redirect.txt
```

```
    gcov -a main.c
```

```
    $(editor) main.c.gcov
```

```
valgrind :
```

```
    ./$(PROJ_NAME).$(EXEC) < input_redirect.txt
```

```
clean:
```

```
    $(RM) *.$(EXEC)
```

```
    $(RM) *.gcno
```

```
    $(RM) *.gcov
```

```
    $(RM) *.gcda
```

Git issue:

Filters Labels 9 Milestones 0 [New issue](#)

☒ Clear current search query, filters, and sorts

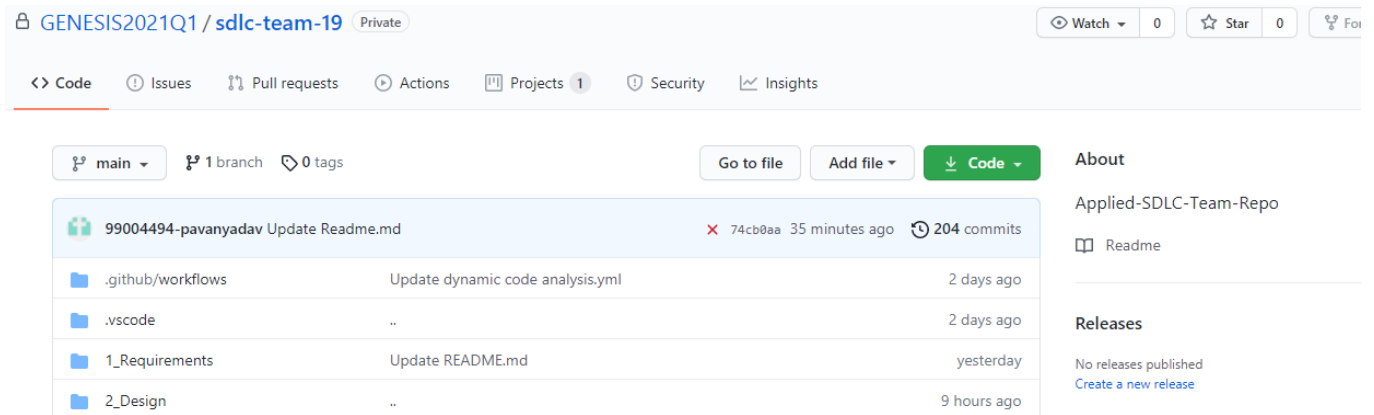
<input type="checkbox"/>	0 Open <input checked="" type="checkbox"/> 5 Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	Scrum stand up call .doc to readme.md #5 by 99004486-Swathi was closed 2 hours ago						
<input type="checkbox"/>	log in issue with taking multiple details from user inset off single user name. #4 by 99004494-pavanyadav was closed 9 hours ago						2
<input type="checkbox"/>	taking a unique input from user instead of a common name #3 by 99004493-PreetPandit was closed 8 hours ago						1
<input type="checkbox"/>	detail in low level composite diagram #2 by 99004493-PreetPandit was closed 10 hours ago						1
<input type="checkbox"/>	difficulty in testcases #1 by 99004493-PreetPandit was closed 10 hours ago						1

Git make:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Preet\real_SDLC\sdlc-team-19>cd 3_Implementation
C:\Users\Preet\real_SDLC\sdlc-team-19\3_Implementation>make
gcc main.c src/game_intro.c src/create_question_array.c src/read_file.c src/ple
se_help.c src/play.c src/time_delay.c -linc -lunity -o Team_19.exe
C:\Users\Preet\real_SDLC\sdlc-team-19\3_Implementation>
```

Git commit:

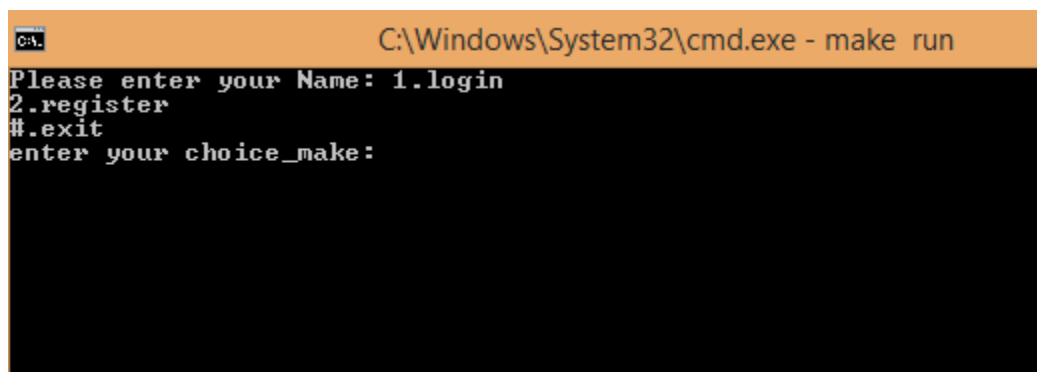


The screenshot shows the GitHub interface for a repository named 'sdhc-team-19' (private). The repository has 204 commits and 0 stars. The main branch is selected. The file list shows:

File	Commit Message	Time
.github/workflows	Update dynamic code analysis.yml	2 days ago
.vscode	..	2 days ago
1_Requirements	Update README.md	yesterday
2_Design	..	9 hours ago

On the right, the 'About' section shows 'Applied-SDLC-Team-Repo' and the 'Releases' section shows 'No releases published' with a link to 'Create a new release'.

Program result images:



```
C:\Windows\System32\cmd.exe - make run
Please enter your Name: 1.login
2.register
#.exit
enter your choice_make:
```

```
TEAM 19 Presents
-----
! QUIZ GAME LAUNCHER__quizes to answer !
-----

! Check out if you are smart enough to answer the questions !

You are provided with four topics and you have to select any two

      Topics                !      key
-----
1. Science and technology  !      <T>
2. LTTS history           !      <L>
3. General Knowledge      !      <G>
4. Sports                 !      <S>

*****
! > Press S to start the game !
! > press H for help         !
! > press Q to quit          !
*****
```

```
please Enter your two choices
1. Science and Technology  ! [T]
2. LTTS history           ! [L]
3. General Knowledge      ! [G]
4. sports                 ! [S]
Enter your First Topic :l
Enter your Second Topic :t
```

```
*****
brilliant, you have cleared 1st round and are appearing for the second round
your score after 1st round is 100
```

```
Press Enter to continue
```

```
! congrats! you are a true champion.. you have cleared the quiz
your score after round 2 is 200
you are a awarded a cash prize of $1million
```

```
click enter to quit the program
```

BADGES:**CI-Build Status:****C/C++ CI - Build Status**

c-cpp.yml

Q	Filter workflow runs	...
12 workflow runs		Event ▾ Status ▾ Branch ▾ Actor ▾
✓	Update unit.yml C/C++ CI - Build Status #12: Commit efd24f6 pushed by padmavathi776	main 36 minutes ago 23s ...

CI-Coverage:**CI-Coverage**

coverage.yml

Q	Filter workflow runs	...
2 workflow runs		Event ▾ Status ▾ Branch ▾ Actor ▾
This workflow has a workflow_dispatch event trigger.		Run workflow ▾
✗	Update unit.yml CI-Coverage #2: Commit efd24f6 pushed by padmavathi776	main 36 minutes ago 43s ...

Code Quality – Static Code Analysis:**Code Quality - Static Code - Cppcheck**

cppcheck.yml

Q	Filter workflow runs	...
11 workflow runs		Event ▾ Status ▾ Branch ▾ Actor ▾
✓	Update unit.yml Code Quality - Static Code - Cppcheck #11: Commit efd24f6 pushed by padmavathi776	main 36 minutes ago 46s ...

Git Inspector:**Contribution Check - Git Inspector**

gitinspector.yml

...

9 workflow runs

Event ▼

Status ▼

Branch ▼

Actor ▼

**Update unit.yml**

Contribution Check - Git Inspector #9: Commit efd24f6 pushed by padmavathi776

main

36 minutes ago

29s

...

Unit Testing:**Unit Testing - Unity**

unit.yml

...

8 workflow runs

Event ▼

Status ▼

Branch ▼

Actor ▼

**Update unit.yml**

Unit Testing - Unity #8: Commit efd24f6 pushed by padmavathi776

main

36 minutes ago

22s

...

Github repository Link:

<https://github.com/GENESIS2021Q1/sdlc-team-19.git>

Project video link:

https://drive.google.com/file/d/1zgU1N3nZT_lhnBhUaEw1JcbYEBNGcPWb/view?usp=sharing