**Quiz game**

---

**ACTIVITY SHEET 1 — Requirements Gathering (SRS Development)**

**Project Title: Quiz game**

**Team Members: 1. Ch. Maneesha**

**2. Uzma begum**

**3. G. Swetha**

**4. K. Tanmayi sree**

**Task 1: Stakeholder Interview**

1. Identify your stakeholders

1..Players / Users

2.Quiz Admin

3.Content Creator (Question setter)

4. System Developer

5. Institute / Organization (Owner)

2. Write 8–10 interview questions.

**Write your questions below:**

1. What type of quiz games do users prefer (MCQs, True/False, Timed quizzes)?

---

2. Should users register or play as guests?

---

3. What subjects or categories should be included in the quiz?
4. Do users want a time limit for each question?
5. Should scores be displayed immediately after the quiz?
6. Do you want difficulty levels (easy, medium, hard)?
7. Should the system maintain a leaderboard?
8. How important is mobile compatibility for users?
9. Should users receive rewards or badges?
10. What security features are required to prevent cheating?

**Task 2: Full the table**

| **Requriment id** | **Task (F/NF)** | **Requirment description** |
|---|---|---|

| | | |
|---|---|---|
| R 1 | F | The system should allow users to register and log in. |
| R 2 | F | The system should display quiz questions and record answers |
| R 3 | NF | The system should be fast and user-friendly with quick response time. |

**Task 3: User Stories**

Write at least 5.

**Format:** *As a [user], I want [feature], so that [reason].*

1. As a user, I want to register and log in, so that my quiz scores can be saved.

---

2. As a player, I want to select quiz categories, so that I can play quizzes of my interest.

---

3. As a user, I want to see my score instantly, so that I know my performance.

---

4. As a player, I want a timer for questions, so that the quiz feels challenging.

---

5. As an admin, I want to add or update questions, so that the quiz content stays fresh

---

**Deliverable: A mini SRS (1–2 pages)**

---

# ACTIVITY SHEET 2 — System Design (High-Level & Low-Level)

## Project: Python Quiz Game System

# Task 1: System Architecture Diagram

## Components

- **User (Player/Admin)**: Uses the quiz application
- **UI Layer**: Quiz screens (questions, options, score)
- **Application Layer (Quiz Logic)**: Handles questions, answers, scoring
- **Database Layer**: Stores users, questions, scores
- **External System**: Optional leaderboard / cloud storage

## Architecture Diagram (3- layer)

```
[ User ]
   |
   v
[ UI Layer ]  →  Question display, answer selection
   |
   v
[ Application Layer ] → Validate answer, calculate score
   |
   v
[ Database Layer ] → Store users, questions, results
   |
   v
[ External System ] → Leaderboard / Backup
```

## Data Flow Explanation

1. User starts quiz from UI
2. UI requests questions from Application Layer
3. Application fetches questions from Database
4. User submits answer
5. Application checks correctness and updates score
6. Final score stored in Database and shown to user

# Task 2: UI Wireframes (Text-Based)

## ☑ Home Screen

```
----------------------
 PYTHON QUIZ GAME
----------------------
 [ Start Quiz ]
```

```
 [ Login ]
 [ View Scores ]
```

## Login / Signup Screen

```
----------------------
 Login / Signup
----------------------
 Username
 Password
 [ Login ]
 [ Register ]
```

## Main Feature Screen (Quiz Screen)

```
----------------------
 Question 1
 What is Python?

 A. Snake
 B. Programming Language
 C. Game
 D. Browser

 [ Submit Answer ]
```

## Settings / Profile Screen

```
----------------------
 Profile
----------------------
 Username
 Highest Score
 Quiz History
 [ Logout ]
```

# Task 3: Database Design

## Entities and Attributes

***1. User***

- user_id (PK)
- username
- password
- role (player/admin)

***2. QuizQuestion***

- question_text
- option_a
- option_b
- option_c
- option_d
- correct_option

***3. QuizResult***

- result_id (PK)
- user_id (FK)
- score
- date_attempted

## ER Diagram (Text Representation)

```
User ----< QuizResult >---- QuizQuestion
```

# Low-Level Explanation (Using Python Quiz Logic)

- **Loops**: Used to display questions repeatedly
- **Functions**: Used to check answers and calculate score
- **Lists**: Store quiz questions and options
- **Conditions (if-else)**: Validate user answers
- **Database / File Handling**: Save sco

**ACTIVITY SHEET 3 — Development Phasec**

- **Solutions attempted / next steps:**
- Checked and corrected database connection settings
- Fixed minor coding errors and tested modules individually
- Plan to complete remaining features and perform integration testing

---

**Deliverable: Updated backlog + coded feature demo**

- ---
  Development backlog updated with current task status
- Core feature implemented and demonstrated successfully

# ACTIVITY SHEET 4 — Testing Phasec

- Major issues found: ___

## Task 1: Create Test Cases (Quiz Game)

| | | | | | **Status (P/F)** |
|---|---|---|---|---|---|
| TC1 | Verify game starts correctly | 1. Open quiz game<br>2. Click **Start Quiz** | First question is displayed | First question displayed | P |
| TC2 | Verify correct answer selection | 1. Start quiz<br>2. Select correct answer | Score increases by 1 | Score increased correctly | P |

| | | | | | |
|---|---|---|---|---|---|
| TC3 | Verify behavior for wrong answer | 1. Start quiz<br>2. Select wrong answer | Score does not increase | Score increased incorrectly | F |

- ## Task 2: Bug Report

| Bug ID | Description | Severity | Steps to Reproduce | Screenshot | Status |
|---|---|---|---|---|---|
| B1 | Score increases even when wrong answer is selected | High | 1. Start quiz<br>2. Choose wrong answer | Screenshot attached | Open |
| B2 | No message shown after quiz completion | Medium | 1. Complete all questions | Screenshot attached | Fixed |

- ## Task 3: Test Summary

- **Total test cases:** 3
- **Passed:** 2
- **Failed : 1**

**Deliverable: Test case sheet + bug report**

# ACTIVITY SHEET 5 — Deployment & Release Notes

## Project: Python Quiz Game System

## Task 1: Deployment Checklist

### Pre-Deployment

- ✓ Source code reviewed and tested
- ✓ Quiz questions verified
- ✓ Input validation implemented
- ✓ Error handling added
- ✓ Database / file storage ready

### Environment Setup

- ✓ Python installed (Python 3.x)

- ✓ Required libraries installed
- ✓ Folder structure organized
- ✓ Configuration files checked

### Testing

- ✓ Unit testing for quiz logic
- ✓ Functional testing (login, quiz, score)
- ✓ Edge case testing (invalid input)

### Deployment

- ✓ Application deployed on local system / server
- ✓ Database connected successfully
- ✓ Initial quiz data loaded

### Post-Deployment

- ✓ Application launch verified
- ✓ Score saving checked
- ✓ User feedback collected

# Task 2: Release Notes

### Release Version: 1.0

### Features Included:

- ✓ User Login and Signup
- ✓ Python-based Quiz Gameplay
- ✓ Multiple Choice Questions
- ✓ Automatic Score Calculation
- ✓ Result Display

### Known Issues:

- ⚠ No timer for questions
- ⚠ Limited number of quiz questions
- ⚠ Basic text-based UI only

## Next Update Goals:

- ➤ Add timer for each question
- ➤ Add leaderboard feature
- ➤ Improve UI (GUI/Web-based)
- ➤ Add more Python quiz questions

# ACTIVITY SHEET 6 — Maintenance & Reflection

**Task 1: Fix & Patch**

Record any issues and fixes.

| Patch id | Issue | Root cause | Fix implemented | Status |
|---|---|---|---|---|
| P1 | Quiz score not updating correctly | Logic error in score calculation | Corrected score calculation algorithm | Done |
| P2 | App loading slowly on start | Unoptimized database queries | Optimized queries and caching | Done |
| P3 | Timer not stopping after quiz ends | Missing condition check | Added proper timer stop condition | Done |

**Task 2: Team Retrospective**

**What worked well?**

1.Clear requirements gathering helped smooth development

2.Team coordination and task division were effective

3.User interface design was simple and user-friendly

4.Core quiz functionalities worked as expected

---

**What needs improvement?**

1.More testing is required before release

2.Performance optimization can be improved further

3.Better error handling for edge cases

---

**What will we change next time?**

1.Add automated testing earlier in development

2.Improve performance monitoring tools

3.Collect user feedback more frequently

4.Plan buffer time for bug fixing

---

**Deliverable: Final reflection + updated patch log**

---