A PROJECT REPORT

on

**Quiz App**

*Submitted by*

**Mr. Ganesh Umesh Tiwari**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF SCIENCE**

in

**COMPUTER SCIENCE**

*under the guidance of*

**Prof. Vipul Saluja**

**Department of Computer Science**



**R. D. & S.H. National College & S. W. A. Science College**

**Bandra, Mumbai – 400050.**

**(Sem V)**

**(2018 – 2019)**

**DECLARATION**

I, Mr. Ganesh Umesh Tiwari, hereby declare that the project entitled “**Quiz App**” submitted in the partial fulfillment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2018 – 2019** is my original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

**Signature of the Student:**

**Place:**

Bandra West Mumbai 400050

**Date:**

26/10/2018

Success is my only option, failures not.

-Marshall Bruce Mathers III

|  |
| --- |
| **PRELIMINARY INVESTIGATION** |

1.1 SYNOPSIS

**Introduction: -**

Quiz is a Educational app which provides users with educational true or false or Multiple Choice Questions based upon their choice. They also have an option to select their favourite subjects from a list of preincluded subjects.

**Modules: -**

1. MCQ Quiz
2. Boolean Quiz
3. Settings
4. Send Suggestions

**Database Modules: -**

1. mcqQuestions
2. booleanQuestions
3. users
4. scores

1.2 Organizational Overview

QuizApp is a Educational app which provides users with easy to use ui and a very user friendly interface so that they can test their knowledge on a platform on which they would love to do that. Quiz App not only provides users with mcq as well as boolean questions for their thirst of knowledge they can choose from a range of ever growing subjects which currently include 7 subjects namely: - science, history, geography, films, sports, mythology and computer science.

* Users can play quiz
* Users can see detailed history about their performance in earlier games
* Users can compare themselves from players all around the world.

1.3 Proposed System

* The proposed system allows user to play quiz games in mcq as well as boolean format.
* Works on REST API framework so it modular and database can be changed without affecting application codebase.
* Has a Nice, Catchy UI as well as a easy to use interface so that even a newcomer will understand the flow of Application.

1.4 Tools and Technologies to be used.

1. Flutter SDK.
2. Visual Studio Code.
3. Android SDK.
4. Dart Language.
5. Eclipse.
6. Jersey Rest Framework.
7. Heroku Web Platform.
8. MySQL Server.
9. Php.
10. OpenJdk 11.

**Flutter SDK: -**

Flutter is an open-source mobile application development SDK created by Google. It is used to develop applications for Android and iOS, as well as being the primary method of creating applications for Google Fuchsia.

**Visual Studio Code: -**

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. It is free and open-source, although the official download is under a proprietary license.

**Android SDK: -**

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later.

**Dart Language: -**

Dart is a general-purpose programming language originally developed by Google and later approved as a standard by Ecma (ECMA-408). It is used to build web, server, and mobile applications. Dart is an object-oriented, class defined language using a C-style syntax that transcompiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics, static typing, and a sound type system.

**Eclipse: -**

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE.[6] It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Julia,[7] Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang.

**Jersey Rest Framework: -**

Jersey RESTful Web Services framework is an open source framework for developing RESTful Web Services in Java. It provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation.

**Heroku Web Platform: -**

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages. Heroku was acquired by Salesforce.com in 2010 for $212 million.

**MySQL Server: -**

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

**PHP: -**

PHP (recursive acronym for PHP: Hypertext Pre-processor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

**OpenJDK 11**: -

OpenJDK (Open Java Development Kit) is a free and open-source implementation of the Java Platform, Standard Edition (Java SE). It is the result of an effort Sun Microsystems began in 2006. The implementation is licensed under the GNU General Public License (GNU GPL) version 2 with a linking exception. Were it not for the GPL linking exception, components that linked to the Java class library would be subject to the terms of the GPL license. OpenJDK is the official reference implementation of Java SE since version 7.

|  |
| --- |
| 1.5 Feasibility Study. |

A feasibility study is an analysis of how successfully a project can be completed.

It is the initial design stage of any project, which brings together the elements of knowledge.

All activities of feasibility study are directed towards helping answer the question

"Should we proceed with the proposed project idea?"

"Does the proposed system contribute to the overall objectives for which the system was proposed for?"

**Technical feasibility: -**

Technical feasibility is concerned with specifying the equipment’s and the software to satisfy the user requirements.

The aim of technical feasibility is to support the cost of the company to undertake a technical study into: -

The system is very much feasible with its technical aspect as there is not much computing resource required to build the system.

Making changes in the system regarding updating product details can be easily done as the admin will have a complete understanding of the system's content and the tools which are used for developing the system.

**Operational feasibility: -**

In operational feasibility, we attempt to ensure that every user can access the system easily.

The ease to use the system will help to increase the operational importance of the system, as there will be not much computing expertise required to use the system and a person with minimum computing knowledge can use the system very effectively.

The proposed system will really benefit the organisation as the system could be maintained by the admin itself and there will not be requirement for any additional staff for maintaining the system.

The overall response of the system will also increase as there will be a greater number of users affiliated with the system in the near future.

**Economic feasibility: -**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system.

The proposed system can be developed at a minimum cost and resource.

The system can assure a good beneficial cost to the organisation.

The savings that would arise from the beneficial cost of the system can be used to improve the system's performance in future.

|  |
| --- |
| Chapter 2. System Analysis |

* Event Table
* Entity Relationship Diagram
* Class Diagram
* Object Diagram
* Use Case Diagram
* Activity Diagram
* State Chart Diagram
* Sequence Diagram

|  |
| --- |
| 2.1 Event Table |

Events are objects or messages used when a software component wants to notify a state change to other components.

An Event model is a software architecture (a set of classes and interfaces) that determines how components occur.

On the event source side: -

• create and describe events

• trigger (or fire) events

• distribute events to interested components

On the event listener side: -

• subscribe to event sources

• react to events when received

• remove the subscription to event sources when desired

Terminology often used refers to: -

• Event Source or Provider: - the sender of events

• Event: - the object sent

• Event Listener or Event Sink or Consumer: - the receiver of events

User: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Trigger** | **Source** | **Response** | **Destination** |
| User Plays Game | Chooses subject after choosing to play | User | Display Score | Score Page |
| User Changes Audio Setting | Click on Settings in Options Page | User | Change Audio Setting | Audio Setting Page |
| User Views Personal Score | Click on Personal Score in Options Page | User | See Personal Scores | Personal Score Page |
| User Views Leader board. | Click on Leader Board in Options Page | User | Display Leader boards | Leader boards Page |
| User Suggests Question | Click on suggest question in Options Page | User | Submitted Snackbar Shown | Suggest Question Page |

Admin: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Trigger** | **Source** | **Response** | **Destination** |
| Admin Logs In | Submit Login Form | Admin | Confirmed as admin | Admin |
| Adds/ Removes/ Updates Question | Add/ Delete/ Update Button | Admin | Added/ Removed/ Updated Question | Admin |
| Add/ Remove/ Update User Details | Add/ Delete/ Update Button | Admin | Added/ Removed/ Updated User Details | Admin |
| Add/ Remove Suggestion | Add/ Delete Suggestion | Admin | Added/ Removed Suggestion | Admin |
| Add/ Remove/ Update Score | Add/ Remove/ Update Score | Admin | Added/ Removed/ Updated User Score | Admin |

|  |
| --- |
| 2.2 Entity Relationship Diagram |

In software engineering, an entity relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that leads itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

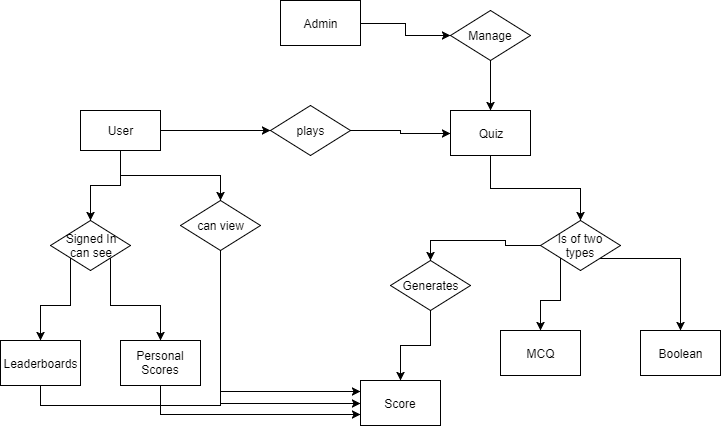
An entity- relationship model is a systematic way of describing and defining a business process. The process is modelled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagram created to represent these entities, attributes and relationships graphically are called entity -relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationship.

Limitations: -

• ER models assume information content that can readily be represented in a relational database. They describe only a relational structure for this information.

• They are inadequate for systems in which the information cannot readily be represented in relational form, such as with semi-structured data



|  |
| --- |
| 2.3 Class Diagram |

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualising, describing and documenting different aspects of a system but also for constructing executable code of the software application

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

**Purpose**: -

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So, it is the most popular UML diagram in the coder community.

So, the purpose of the class diagram can be summarised as: -

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

|  |
| --- |
| 2.4 Object Diagram |

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

**Purpose**: -

The purpose of a diagram should be understood clearly to implement it practically. The purpose of object diagrams are similar to class diagrams.

The difference is that a class diagram represents an abstract model consisting of classes and their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.

It means the object diagram is closer to the actual system behaviour. The purpose is to capture the static view of a system at a particular moment.

So, the purpose of the object diagram can be summarised as: -

* Forward and reverse engineering
* Object relationships of a system
* Static view of an interaction
* Understand object behaviour and their relationship from practical perspective.

|  |
| --- |
| 2.5 Use Case Diagram |

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used.

Purpose: -

The purpose of use case diagram is to capture the Dayna aspect of a system. But this definition is too generic to describe the purpose.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analysed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

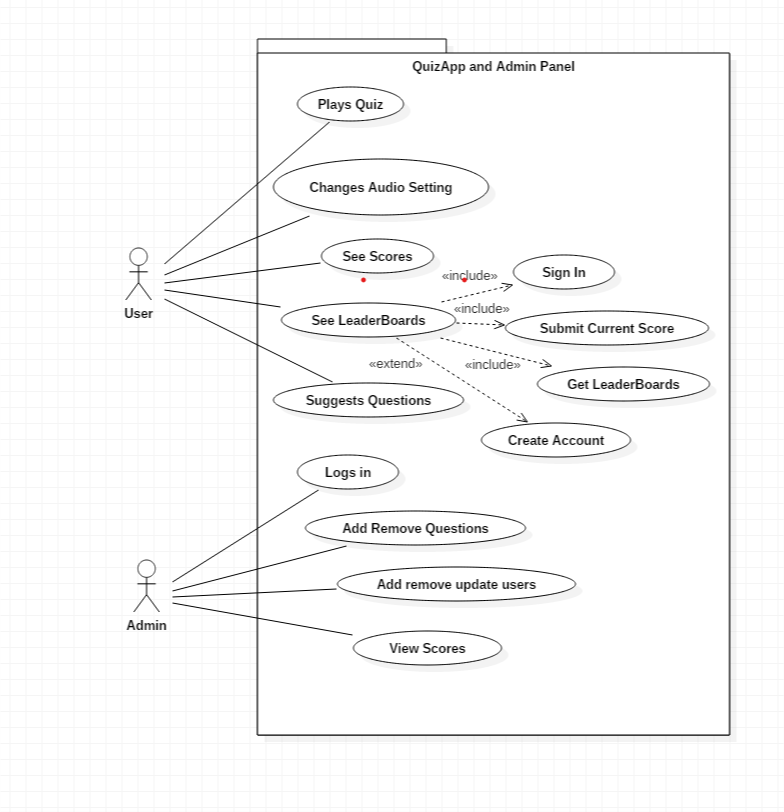
So, in brief the purpose of use case diagram can be as follows: -

• Used to gather requirements of a system.

• Used to get an outside view of a system.

• Identify external and internal factors influencing the system.

• Show the interacting among the requirements are actors.



|  |
| --- |
| 2.6 Activity Diagram |

Activity diagram is another important diagram in UML to describe dynamic aspects of the system.

Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagram deals with all type of flow control by using different elements like fork, join etc.

Purpose: -

It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

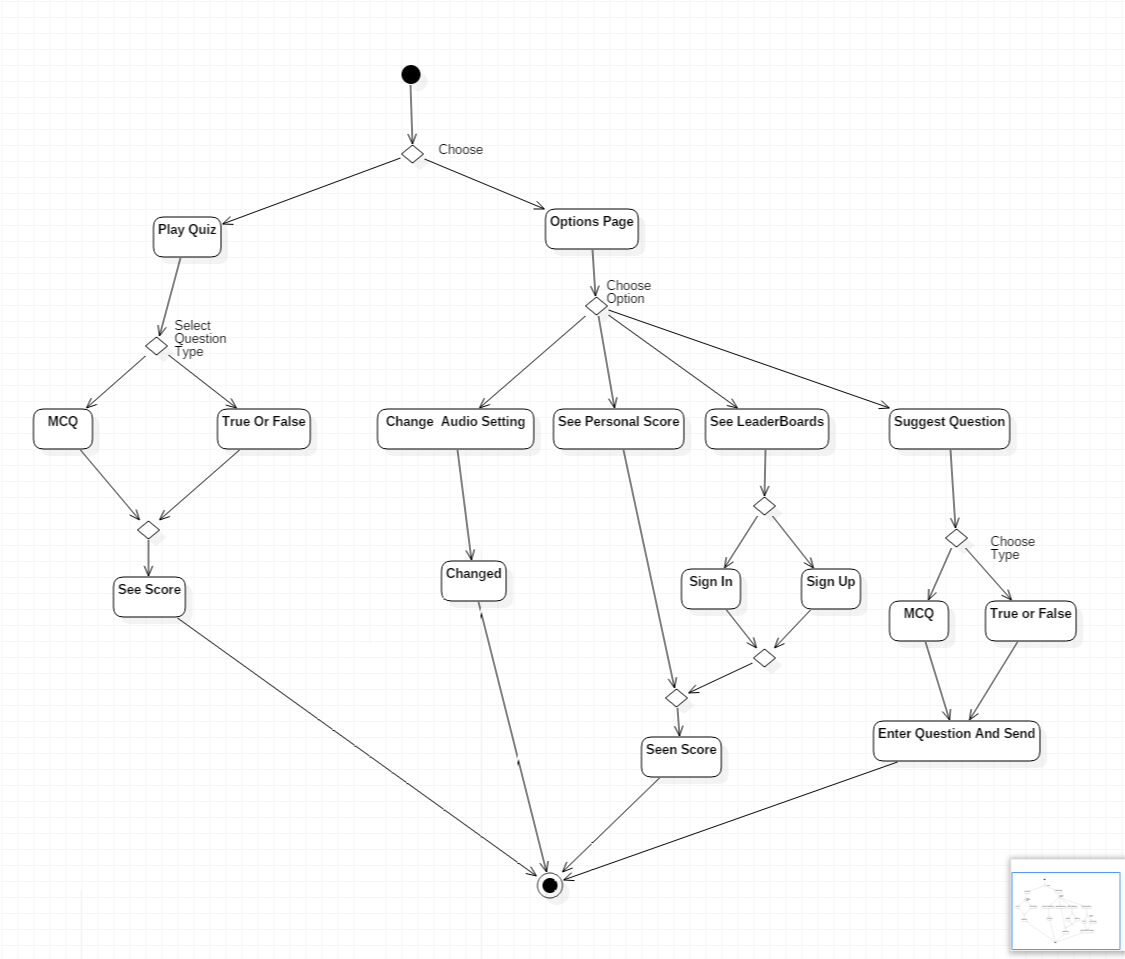
It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagram looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

So, the purpose can be described as:

• Draw the activity flow of system.

• Describe the sequence from one activity to another.

• Describe the parallel, branched and concurrent flow of the system.



|  |
| --- |
| 2.7 State Chart Diagram |

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A State Chart Diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

As State Chart Diagram defines states it is used to model lifetime of an object.

**Purpose: -**

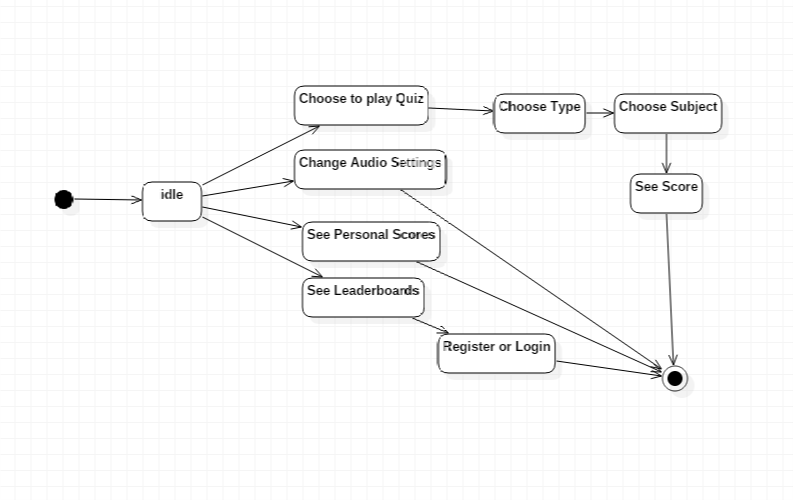
State Chart Diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So, State Chart Diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State Chart Diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So, the most important purpose of State Chart Diagram is to model life time of an object from creation to termination.

State Chart Diagram are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purpose of using State Chart Diagrams: -

* To model dynamic aspect of a system.
* To model life time of a reactive system.
* To describe different states of an object during its life time.
* Define a state machine to model of an object.

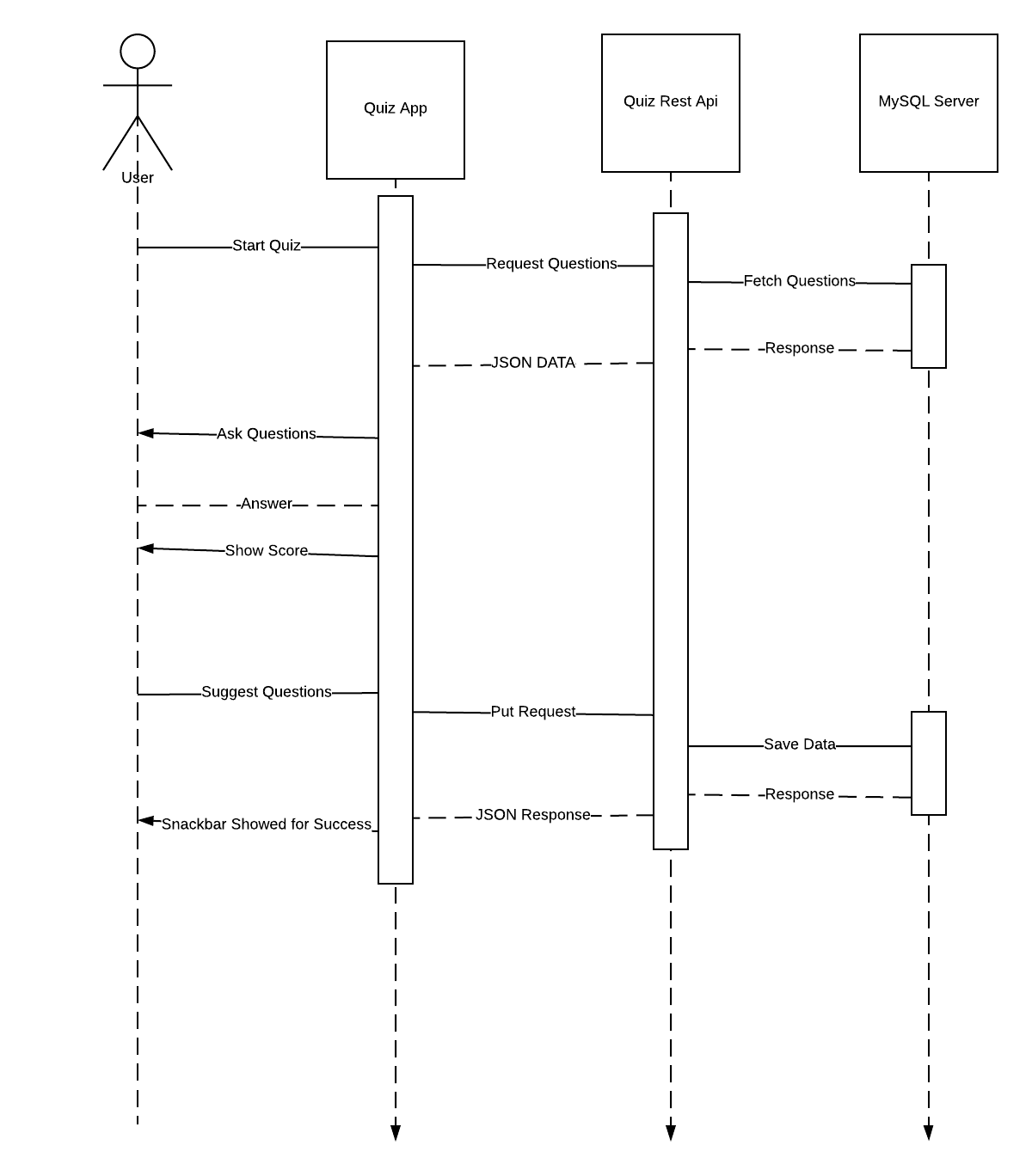


|  |
| --- |
| 2.8 Sequence Diagram |

A Sequence Diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A Sequence Diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence Diagrams are sometimes called event diagrams or event scenarios.

A Sequence Diagram shows parallel vertical lines(lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response. Asynchronous calls are present in multithreaded applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed response to the message (Execution Specification in UML).



|  |
| --- |
| Chapter 3. System Diagram |

* Component Diagram
* Package Diagram
* System Flow Chart
* Structured Chart
* Deployment Diagram

|  |
| --- |
| 3.1 Component Diagram |

Component Diagram are different in terms of nature and behaviour. Component diagrams are used to model physical aspects of a system.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose: -

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the component used to make those functionalities

So, from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of system. Static implementation represents the organization of the components at a particular moment.

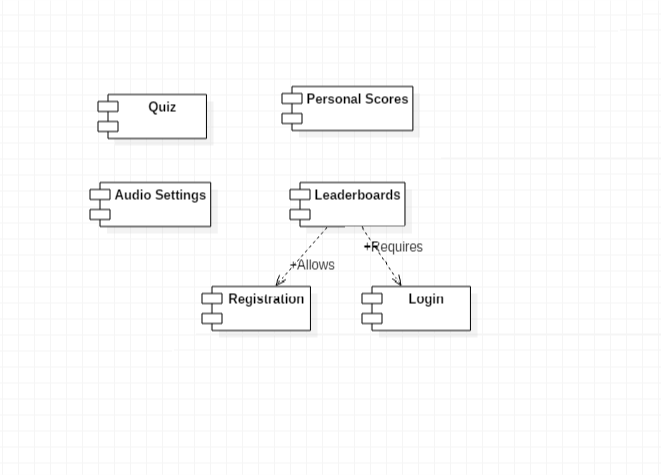
A single Component Diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

So, the purpose of the component diagram can be summarized as:

• Visualize the components of a system

• Construct executable by using forward and reverse engineering

• Describe the organization and relationships of the components.



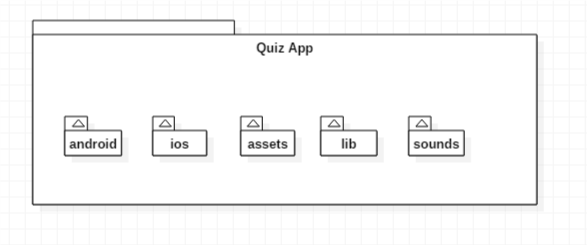
|  |
| --- |
| 3.2 Package Diagram |

When modelling a large-scale system, you would probably be working with a high volume of model elements. They describe a model from different views and different phases, hence are in different types.

UML package helps to organize and arrange model elements and diagrams into logical groups, through which you can manage a chunk of project data together.

You can also use packages to represent different views of the systems architecture. In addition, developers can use package to model the physical package or namespace structure of the application to build.

Package Diagram visualizes packages and depicts the dependency, Import, access, generalization, realization and merge relationships between them. Package diagram enables you to gain a high-level understanding of the collaboration among model elements through analysing the relationships among their parent package. also helps explain the systems architecture from a broad view.



|  |
| --- |
| 3.3 System Flow Chart |

System Flow Charts are a way of displaying how data flows in a system and how decisions are made to control events.

To illustrate this, symbols are use. They are connected together to show what happens to data and where it goes. The basic ones include: symbols used in flow charts.

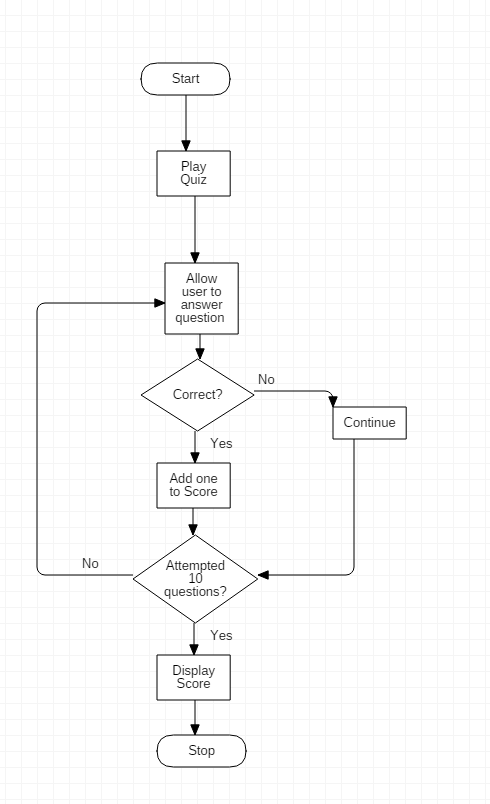
The flow of data generally goes from top to bottom and left to right and depicts the sequence of processing steps along these data lines.

The following are examples of some of the symbols used in system flowcharts: A system flowchart shows the key input and output associated with the program. The shape of symbols indicate the type of input or output devices.

The type of diagram dictates the flow chart symbols that are used. The terminator symbols mark the starting or ending point of the system.

A flow chart is a formalized graphic representation of program logic sequence, work or manufacturing process, organization.

A graphical representation of the sequence of operations in an information system or program. Information system flow charts show how data flows from source



|  |
| --- |
| 3.4 Structured Chart |

A structure chart (SC) in software engineering and organization theory, is a chart which shows the breakdown of a system to its lowest manageable levels.

They are used in structured programming to arrange modules into a tree. Each module is represented by a box, which contains the module’s name. A structured chart(SC) in software engineering and organizational theory ,is chart which shows the breakdown of a system to its lowest manageable levels.

They are used in structured programming to arrange program modules into tree. Each module is represented by a box, which contains the module’s name.

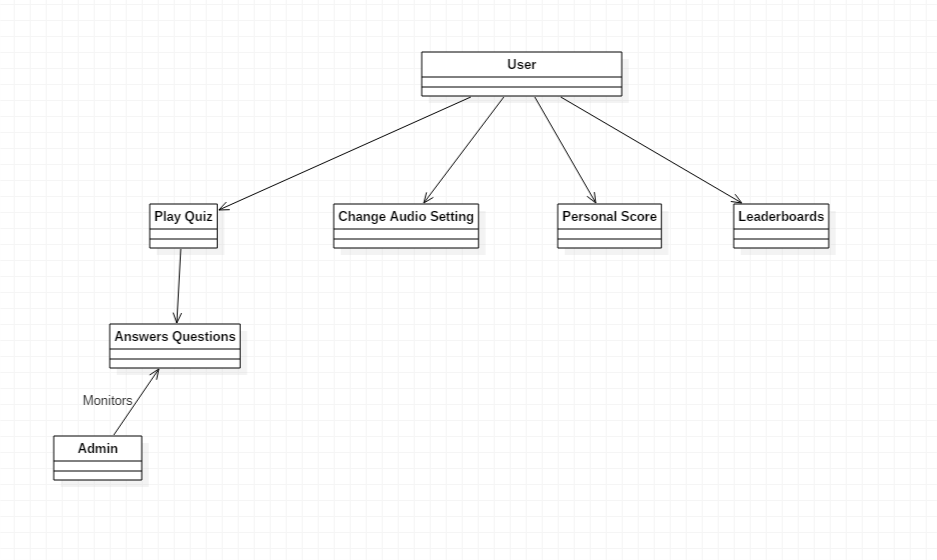
Structure diagram is a chart derived from data flow chart diagram. The system structure chart represents hierarchical structure of modules.

A structure chart depicts -

The size and complexity of the system, and number of readily identifiable functions and modules within each function and whether each identifiable function is a manageable entity or should be broken down into smaller components.

A structure chart is also used to diagram associated elements that comprise a run stream or thread. It is often developed as a hierarchal

Diagram, but other representations are allowable.



|  |
| --- |
| 3.5 Deployment Diagram |

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

So, deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose: -

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artefacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

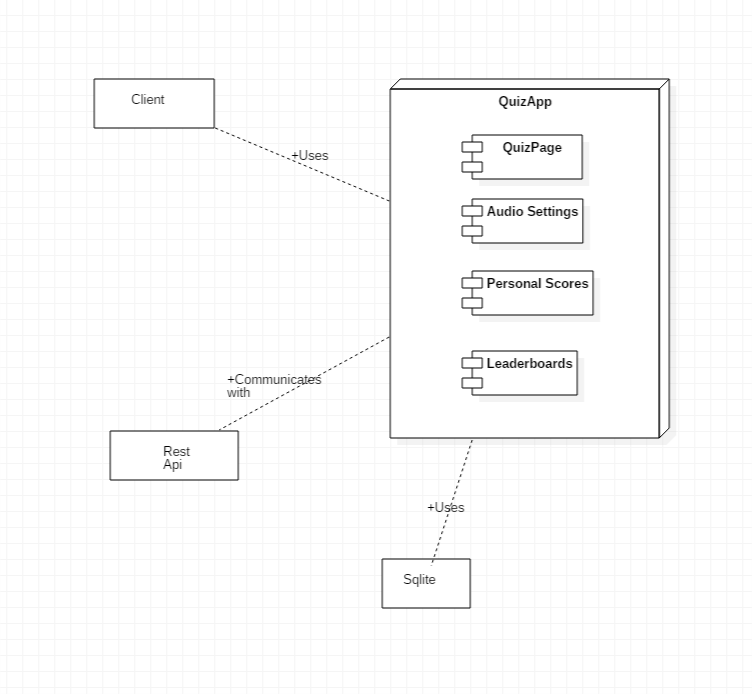
So most of the UML diagrams are used to handle logical components but deployment is made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

It can be described as follows: -

• Visualize hardware topology of a system.

• Describe the hardware components used to deploy software components.

• Describe runtime processing nodes



|  |
| --- |
| Chapter 4. System Coding |

* Site Map
* Data Dictionary
* Source Code

|  |
| --- |
| 4.1 Site Map |

* Quiz Play
* Personal Scores
* Leader Boards
* Settings

**Modules**

* MCQ Quiz
* Boolean Quiz
* Score Page
* Leader Boards

**Database Modules**

* mcqQuestion
* booleanQuestion
* users
* score

|  |
| --- |
| 4.2 Data Dictionary |

MCQ Question

|  |  |  |
| --- | --- | --- |
| Field | Type | Key |
| Id | Int | Primary key |
| Question | Varchar |  |
| Answer | Varchar |  |
| Option1 | Varchar |  |
| Option2 | Varchar |  |
| Option3 | Varchar |  |

Boolean Questions

|  |  |  |
| --- | --- | --- |
| Field | Type | Key |
| Id | Int | Primary key |
| Question | Varchar |  |
| Answer | Varchar |  |

Users

|  |  |  |
| --- | --- | --- |
| Field | Type | Key |
| Username | Varchar | Primary key |
| Password | Varchar |  |
| Email | Varchar |  |

Scores

|  |  |  |
| --- | --- | --- |
| Field | Type | Key |
| username | Varchar | Primary key & Foreign Key |
| Score | int |  |

|  |
| --- |
| 4.3 Source Code |

|  |
| --- |
| Chapter 5: - System Implementation |

* Hardware and Software Requirements
* Screen Layouts

|  |
| --- |
| 5.1 Hardware and Software Requirements |

**Hardware Requirements**: -

The minimum Hardware and system Software requirements for development and using this is:

Processor: Dual Core processor with 1Ghz speed and above

RAM: Minimum requirement is 3 GB

**Software Requirements**: -

Client-Side Requirements

Android Version: - 4.1.x or newer (Android Jellybean)

Tested on: - 6.0.1 and newer (Android Marshmallow)

**Server-Side Requirements**: -

Server running any OS (Windows/Linux) With MySQL installed