Asking (Question and Answer platform)

Software Architecture Document

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 15/03/2017 | 1.0 | Initial software architecture document | P.R.L. Fernando |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 5

2. Architectural Representation 5

3. Architectural Goals and Constraints 5

4. Use-Case View 5

4.1 Use-Case Realizations 6

4.1.1 Use case diagram 6

4.1.2 Use case description 7

5. Logical View 13

5.1 Overview 13

5.2 Architecturally Significant Design Packages 14

6. Process View 14

6.1 Activity diagrams 15

6.2 Sequence diagrams 25

7. Deployment View 28

8. Implementation View 28

8.1 Overview 28

8.2 Layers 29

8.2.1 Model layer 30

8.2.2 Control layer 30

8.2.3 View layer 30

9. Size and Performance 30

10. Quality 30

11. Supporting Information 31

Software Architecture Document

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

## Scope

This software architecture document provide an architectural overview of the Asking question and answer platform is being developed to support academic studies by providing an alternate way to solve problems occurred.

The document was prepared based on the software requirement specification (SRS) document. All entities and requirements identified in it were taken into consideration, ensuring that the current design meets all requirements of the system as well as any implicit requirements identified later.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | Someone who interacts with the mobile phone application |
| Administrator | System administrator who is given specific permission for managing and controlling the system |
| Student | An undergraduate student |
| Lecturer | A lecturer who teaches undergraduate students |
| Stakeholder | Any person who has interaction with the system who is not a developer. |
| UC 1,2,3….. | Use case 1,2,3,……. |
| AD 1,2,3…... | Use case activity diagram 1,2,3….. |
| SD 1,2,3….. | Use case sequence diagram 1,2,3….. |

## References

|  |  |
| --- | --- |
| [1] | MIcrosoft, "Try Visio Professional 2016," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/evalcenter/evaluate-visio-professional-2016. [Accessed 16 03 2017]. |
| [2] | P. Fernando, "Software Requirement Specification," Alternate Solutions, 2017. |
| [3] | IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended," 20 October 1998. |

## Overview

The remainder of this document provides the reader with a deeper understanding of the Asking question and answer platform website. It is organized into several views of the system, the Logical view, which will give an overview of the subsystems and components the system is comprised of, the Deployment view, which will give go into how the system has been deployed and what it's being hosted on, the Implementation view, which will show how the system has been implemented and go into more detail about the different parts of the system, and the Data view which will show how the database is structured and organized

# Architectural Representation

This document describes the architecture as a series of views; use case view, logical view, process view, deployment view and implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model.

# Architectural Goals and Constraints

The following are the key requirements of the system architecture that have a significant impact on the architecture.

**AGC-1:** System usage is controlled by user identification and password control. All the security

details must be protected by unauthorized access.

**AGC-2:** System should be given the accurate search result to increase the user experience

**AGC-3:** Performance and loading requirement must be taken into consideration as the

architecture is developed.

**AGC-4:** Availability of the system should be at least 95%.

**AGC-5:** Mean time to repair must be at most 24 hours and must consider as the architecture is

developed.

# Use-Case View

This section lists use cases from the use case model and the representation of central functionality of the final system. Use case view is to define the main drivers of the system, which are the system requirements that stress or illustrate a specific, delicate point of the architecture or that have a substantial architectural coverage.

## Use-Case Realizations

### Use case diagram



Figure 1: Use case diagram

### Use case description

**User type: Student**

**UC 1:** Sign in

|  |  |  |
| --- | --- | --- |
| **Use case name** | Sign in | |
| **Actor** | Student | |
| **Description** | Student can sign in to the system using email and password | |
| **Pre conditions** | Student must be registered with the system | |
| **Main flow** | **Actor** | **System** |
| 1. User visit login page 2. User enter username and   password   1. User click on login button | 2.1. Validate email address  3.1. Validate information and get  user access type |
| **Successful end/**  **Post condition** | 4. Redirect to student home page | |
| **Fail end/**  **Post condition** | 2.1. If email is not valid notify user  3.1. If user account not found notify user and redirect to login page  3.2. If user account is blacklisted notify user | |
| **Extension** | <N/A> | |

Table 1: Use case description - Sign in

**UC 2:** Create new account

|  |  |  |
| --- | --- | --- |
| **Use case name** | Create new account | |
| **Actor** | Student | |
| **Description** | Student can create new account to get functionality of the system | |
| **Pre conditions** | Student must have a active email address | |
| **Main flow** | **Actor** | **System** |
| 1. User visit sign up page 2. User enter email and   password   1. User click on sign up button | 2.1. Validate email address  characters  3.1. Validate information and create  new account |
| **Successful end/**  **Post condition** | 4. Notify user and redirect to login page | |
| **Fail end/**  **Post condition** | 2.1. If email is not valid notify user  3.1. If user already has an account notify and redirect to sign in page | |
| **Extension** | **UC 1:** Sign in | |

Table 2: Use case description - Create new account

**UC 3:** Change account password

|  |  |  |
| --- | --- | --- |
| **Use case name** | Change account password | |
| **Actor** | Student | |
| **Description** | Student can change account password | |
| **Pre conditions** | Student must logged in | |
| **Main flow** | **Actor** | **System** |
| 1. User click user name from the website header 2. Choose change password 3. Insert new password   4. Click on submit button | 2.1. Redirect to change password page  3.1. If old password not match notify user  4.1. Change password |
| **Successful end/**  **Post condition** | 5. Notify user and redirect to previous page | |
| **Fail end/**  **Post condition** | 4.1. If password not changed notify user | |
| **Extension** | <N/A> | |

Table 3: Use case description - Change account password

**UC 4:** Subscribe for module

|  |  |  |
| --- | --- | --- |
| **Use case name** | Subscribe for new module | |
| **Actor** | Student | |
| **Description** | Student can subscribe for module to get notification | |
| **Pre conditions** | Student must be logged in | |
| **Main flow** | **Actor** | **System** |
| 1. Choose add new module from  side bar  2. Search for module  3. Click on subscribe button | 1.1. Display search module page  2.1. Display search result  3.1. Subscribe for new module |
| **Successful end/**  **Post condition** | 4. Notify user and redirect to home page | |
| **Fail end/**  **Post condition** | 2.1. If search result is empty notify user  3.1. If subscription not successful notify user | |
| **Extension** | **UC 7:** View questions | |

Table 4: Use case description - Subscribe for module

**UC 5:** Unsubscribe module

|  |  |  |
| --- | --- | --- |
| **Use case name** | Unsubscribe module | |
| **Actor** | Student | |
| **Description** | Student can unsubscribe module to stop notifications | |
| **Pre conditions** | Student must be logged in | |
| **Main flow** | **Actor** | **System** |
| 1. Student select subscribed  modules from side bar  2. Student select relevant module  3. Click on unsubscribe button | 1.1. Display subscribed modules  2.1. Display module details  3.1. Unsubscribe module from user  account |
| **Successful end/**  **Post condition** | 4. Notify user and redirect to home page | |
| **Fail end/**  **Post condition** | 3.1. If module subscription not successful notify user | |
| **Extension** | **UC 7:** View questions | |

Table 5: Use case description - Unsubscribe module

**UC 6:** Search questions

|  |  |  |
| --- | --- | --- |
| **Use case name** | Search questions | |
| **Actor** | Student | |
| **Description** | Student can search questions by using a keyword | |
| **Pre conditions** | Student must be logged in | |
| **Main flow** | **Actor** | **System** |
| 1. User enter search keyword in the search bar  2. Press <Enter> button from keyboard or click on search button | 2.1. Display search results |
| **Successful end/**  **Post condition** | 3. Display search results | |
| **Fail end/**  **Post condition** | 2.1. If search result is empty notify user | |
| **Extension** | **UC 7:** View questions | |

Table 6: Use case description - Search questions

**UC 7:** View questions

|  |  |  |
| --- | --- | --- |
| **Use case name** | View questions | |
| **Actor** | Student | |
| **Description** | Student can view question buy searching or by navigating through a module | |
| **Pre conditions** | Student must have logged in  Question list must be available to select | |
| **Main flow** | **Actor** | **System** |
| 1. Student select Question | 1.1. Get question details from the server |
| **Successful end/**  **Post condition** | Display question details | |
| **Fail end/**  **Post condition** | <N/A> | |
| **Extension** | **UC 8:**  View answers  **UC 11:** Rate question | |

Table 7: Use case description - View question

**UC 8:** View answers

|  |  |  |
| --- | --- | --- |
| **Use case name** | View answers | |
| **Actor** | Student | |
| **Description** | Student can view submitted answers buy selecting a question | |
| **Pre conditions** | Student must have logged in  Question list must be available to select | |
| **Main flow** | **Actor** | **System** |
| 1. User click on question and click on show answers button | 1.1. Get answers from database and display |
| **Successful end/**  **Post condition** | 2. Show answers list | |
| **Fail end/**  **Post condition** | 1.1. If there is no answers available for selected question notify user. | |
| **Extension** | **UC 12:** Rate answers | |

Table 8: Use case description - View answer

**UC 9:** Submit new question

|  |  |  |
| --- | --- | --- |
| **Use case name** | Submit new question | |
| **Actor** | Student | |
| **Description** | Student can submit new question by mentioning module. Subscribed user for that module get the notification. | |
| **Pre conditions** | Student must be logged in | |
| **Main flow** | **Actor** | **System** |
| 1. Select add new question from the side bar  2. Add module and title  3. Add question title, description and question tags | 1.1. Display add new question page  2.1. Suggest module code, title  3.1. Add question |
| **Successful end/**  **Post condition** | 4. Notify user and display added question along with other submitted questions | |
| **Fail end/**  **Post condition** | * 1. If module unavailable notify user   2. If user not subscribed for selected module notify and redirect to subscription page   3.1. If question submission is not successful notify user | |
| **Extension** | <N/A> | |

Table 9: Use case description - Submit new question

**UC 10:** Submit new answer

|  |  |  |
| --- | --- | --- |
| **Use case name** | Submit new answer | |
| **Actor** | Student | |
| **Description** | Student can submit new answers for the previously submitted questions | |
| **Pre conditions** | Student must be logged in  Question list must be available to select a question | |
| **Main flow** | **Actor** | **System** |
| 1. Select question from the question list  2. Click add new answer button  3. Add new answer  4. Click on the submit button | 1.1. Display question details with submitted answers  2.1. Display new answer page  4.1. Submit new answer |
| **Successful end/**  **Post condition** | 5. Notify user and display question with submitted answer | |
| **Fail end/**  **Post condition** | 4.1. If answer submission is not successful notify user | |
| **Extension** | <N/A> | |

Table 10: Use case description - Submit new answer

**UC 11:** Rate question

|  |  |  |
| --- | --- | --- |
| **Use case name** | Rate question | |
| **Actor** | Student | |
| **Description** | Student can questions to change its priority | |
| **Pre conditions** | Student must be logged in  Question must be available to select | |
| **Main flow** | **Actor** | **System** |
| 1. Select question  2.Click on rate up or rate down button | 1.1. Display question details  2.1. Change question rating |
| **Successful end/**  **Post condition** | 3. Display question list with new rating | |
| **Fail end/**  **Post condition** | <N/A> | |
| **Extension** | **UC 12:** Rate answer | |

Table 11: Use case description - Rate question

**UC 12:** Rate answer

|  |  |  |
| --- | --- | --- |
| **Use case name** | Rate answers | |
| **Actor** | Student | |
| **Description** | Student can rate answers to change its priority | |
| **Pre conditions** | Student must be logged in  Question must be selected and answers must be available to select | |
| **Main flow** | **Actor** | **System** |
| 1. Select answer  2. Click on rate up or rate down button | 2.1. Change answer rating |
| **Successful end/**  **Post condition** | 3. Display answers with new ratings | |
| **Fail end/**  **Post condition** | <N/A> | |
| **Extension** | <N/A> | |

Table 12: Use case description - Rate answer

**User type: Lecturer**

**UC 13:** Modify answer

|  |  |  |
| --- | --- | --- |
| **Use case name** | Modify answers | |
| **Actor** | Lecturer | |
| **Description** | Lecturer can modify answers to enhance them | |
| **Pre conditions** | Lecturer must have logged in  Question must have selected and answers must be available | |
| **Main flow** | **Actor** | **System** |
| 1. Select answer  2. Click on modify answer button  3. Modify answer  4. Click on submit button | 2.1. Display modify answer page  4.1. Submit changes |
| **Successful end/**  **Post condition** | 5. Display modified answer | |
| **Fail end/**  **Post condition** | 4.1. If the answer submission unsuccessful notify user and prompt for resubmission | |
| **Extension** | <N/A> | |

Table 13: Use case description - Modify answer

**UC 14:** Delete question

|  |  |  |
| --- | --- | --- |
| **Use case name** | Delete question | |
| **Actor** | Lecturer | |
| **Description** | Lecturer can delete inappropriate questions to maintain the quality of the system | |
| **Pre conditions** | Lecturer must have logged in  Question must have selected and answers must be available | |
| **Main flow** | **Actor** | **System** |
| 1. Select Question  2. Click on remove question button  3. Click on submit button | 2.1. Display notification  4.1. Delete question |
| **Successful end/**  **Post condition** | 4. Display notification and redirect to home page | |
| **Fail end/**  **Post condition** | 4.1. If the question deletion unsuccessful notify user and aks for resubmission | |
| **Extension** | <N/A> | |

Table 14: Use case description - Delete question

**UC 15:** View report

|  |  |  |
| --- | --- | --- |
| **Use case name** | View report | |
| **Actor** | Lecturer | |
| **Description** | View report to get an overall understanding of the teaching process and students | |
| **Pre conditions** | Lecturer must have logged in  Answers must available in the selected module | |
| **Main flow** | **Actor** | **System** |
| 1. Select module from the side menu  2. Click on view report | 2.1. Generate report |
| **Successful end/**  **Post condition** | 3. Display a report which is categorized according to module code | |
| **Fail end/**  **Post condition** | 2.1. If the report generation is unsuccessful notify user | |
| **Extension** | **UC 7:** View questions | |

Table 15: Use case description - View report

**User type: Administrator**

**UC 16:** Blacklist account

|  |  |  |
| --- | --- | --- |
| **Use case name** | Blacklist account | |
| **Actor** | Administrator | |
| **Description** | Administrator can blacklist inappropriate accounts to maintain the quality of the software | |
| **Pre conditions** | Administrator must be logged in | |
| **Main flow** | **Actor** | **System** |
| 1. Visit account management page  2. Select account  3. Click on blacklist account button  4. Input reason to blacklist the account  5. Click on submit button | 1.1. Display account management page  2.1. Display account details  3.1. Notify user and prompt for reason  5.1. Blacklist the account |
| **Successful end/**  **Post condition** | 6. Display notification and redirect to account management page | |
| **Fail end/**  **Post condition** | 5.1. If the process is unsuccessful notify the user | |
| **Extension** | **UC 17:** Delete account | |

Table 16: Use case description - Blacklist account

# Logical View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities.

## Overview

The overall decomposition of the design model in terms of its package hierarchy and layers can be seen as follows.



Figure 2: System overview

## Architecturally Significant Design Packages



Figure 3: Design class diagram

The structure of the system is also defined in terms of classes and their associations and relationships. Class diagrams with the association and relationship diagrams are made in order to generate a logical design for the system. Through the diagrams, connections between classes and their attributes are identified. This enables the thorough analysis and scrutiny of the system

# Process View

This section includes activity diagrams and sequence diagrams to describes the system's decomposition into lightweight processes heavyweight processes and communication between processes such as message parsing.

## Activity diagrams

**AD 1:** Login user



Figure 4: Activity diagram - Login user

**AD 2:** Register new user



Figure 5: Activity diagram - Register new student

**AD 3:** Search question



Figure 6: Activity diagram - Search question

**AD 4:** View answers



Figure 7: Activity diagram - View answers

**AD 5:** Rate question



Figure 8: Activity diagram - Rate question

**AD 6:** Submit new question



Figure 9: Activity diagram - Submit new question

**AD 7:** Submit new answer



Figure 10: Activity diagram - Submit new answer

**AD 8:** Rate question



Figure 11: Activity diagram - Rate question

**AD 9:** View report



Figure 12: Activity diagram - View report

**AD 10:** Blacklist user



Figure 13: Activity diagram - Blacklist user

## Sequence diagrams

**SD 1:** Login user



Figure 14: Sequence diagram - Login user

**SD 2:**  Register new user



Figure 15: Sequence diagram - Register new user

**SD 3:**  Search questions



Figure 16: Sequence diagram - Search question

**SD 4:** Submit new question



Figure 17: Sequence diagram - Submit new question

**SD 5:** Submit new answer



Figure 18: Sequence diagram - Submit new answer

**SD 6:** Rate question



Figure 19: Sequence diagram - Rate question

# Deployment View

Asking question and answer platform is comprised of three physical nodes: the web server, the database server and the user browser. The deployment view of the system is as follows. All the users access the system via internet. Only the registered users will be allowed to use the system. According to the user role, the access privileges will be granted.



Figure 20: Deployment diagram

# Implementation View

The implementation view if focused on the overall structure of the implementation model and the decomposition of the software into layers and sub systems. Asking question and answer platform is basically divided into 4 sub-systems; account management subsystem, question and answer submission system, module management system and notification system. The implementation view uses the MVC architecture in sub dividing the system into three layers; view layer, model layer and control layer.

## Overview

The layers applied to the implementation view are the model layer, control layer and the view layer of MVC architecture.

## Layers



Figure 21: Package diagram

### Model layer

The main function of the model layer is managing the data of the application. It responds to the request from the view layer and also to the instructions from the controller to update itself. The model layer includes model layer components of all four subsystems. Client side model layer includes the model classes and server side model layer include schema classes that use to access and update database.

### Control layer

The control layer is responsible for responding to user input and performs interactions on the data model objects. The controller receives the input; it validates the input and then performs the business operation that modifies the state of the data model. It also includes the controlling components of all the subsystems. Client side control layer includes controllers and services while server side control layer includes REST API and controllers.

### View layer

The view layer consists of the user interfaces of all the subsystems. It includes templates, reusable components and static web pages.

# Size and Performance

**Responsiveness of the UI elements:** The user interface must be responsive to keep the user engaged.

**Searching time:** The time taken to search a question must be considerably low in order to make the product usable to the user.

**Accuracy and consistency:** The system should return accurate and consistent results to make the product useful.

# Quality

**Extensibility:** The application developed must be easily extendable and easily adaptable to add new functions and features.

**Understandable:** The user interfaces must be clutter free and easy to follow for the users.

# Supporting Information

Appendix A: Table of figures

[Figure 1: Use case diagram 6](#_Toc477533509)

[Figure 2: System overview 13](#_Toc477533510)

[Figure 3: Design class diagram 14](#_Toc477533511)

[Figure 4: Activity diagram - Login user 15](#_Toc477533512)

[Figure 5: Activity diagram - Register new student 16](#_Toc477533513)

[Figure 6: Activity diagram - Search question 17](#_Toc477533514)

[Figure 7: Activity diagram - View answers 18](#_Toc477533515)

[Figure 8: Activity diagram - Rate question 19](#_Toc477533516)

[Figure 9: Activity diagram - Submit new question 20](#_Toc477533517)

[Figure 10: Activity diagram - Submit new answer 21](#_Toc477533518)

[Figure 11: Activity diagram - Rate question 22](#_Toc477533519)

[Figure 12: Activity diagram - View report 23](#_Toc477533520)

[Figure 13: Activity diagram - Blacklist user 24](#_Toc477533521)

[Figure 14: Sequence diagram - Login user 25](#_Toc477533522)

[Figure 15: Sequence diagram - Register new user 25](#_Toc477533523)

[Figure 16: Sequence diagram - Search question 26](#_Toc477533524)

[Figure 17: Sequence diagram - Submit new question 26](#_Toc477533525)

[Figure 18: Sequence diagram - Submit new answer 27](#_Toc477533526)

[Figure 19: Sequence diagram - Rate question 27](#_Toc477533527)

[Figure 20: Deployment diagram 28](#_Toc477533528)

[Figure 21: Package diagram 29](#_Toc477533529)