UNIVERSITI MALAYSIA TERENGGANU
FACULTY OF COMPUTER SCIENCE AND MATHEMATICS

CSM3023 (K1)
WEB-BASED APPLICATION DEVELOPMENT

SYSTEM REPORT

UMT ACADEMIC PROGRAM DEVELOPMENT TRACKING SYSTEM (MODULE 4 – FULL ACCREDITATION APPLICATION)

PREPARED BY :

NO	MATRIC NO	NAME
1	S67911	LUTFIL HAZIQ BIN ADNAN
2	S67604	HARINATUL MUFLIHUN BINTI HASNUL MUNAWAR
3	S67978	MUHAMMAD HAZIQ AIMAN BIN MUSTAFA

PREPARED FOR :

DR. MOHAMAD NOR BIN HASSAN

BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONOURS
SEMESTER II 2023/2024

TABLE OF CONTENT

1.0 EXECUTIVE SUMMARY 1
2.0 USECASE DIAGRAM 3
3.0 CLASS DIAGRAM 5
5.0 ENTITY RELATIONSHIP DIAGRAM 8
6.0 MVC MODEL 11
7.0 SAMPLE OF INTERFACE 13
8.0 SAMPLE PORTION OF CODING FOR DATABASE CONNECTION, CRUD PROCESS, AND JAVABEANS 17
9.0 INDIVIDUAL CONTRIBUTION 21
10.0 CONCLUSION 23
11.0 REFERENCES 24

1.0 EXECUTIVE SUMMARY

UMT Academic Program Development Tracking System is a platform designed to enhance the management of academic program development. This system provides a centralized repository for uploading all relevant documents and monitor the progress of academic program development through various stage. The system is structured into four modules. Each module has specific aspects of academic program development lifecycle. This report focuses on last model, Module 4 full accreditation application.

Module 4 : Full Accreditation Application

Module 4 is an important part in the academic program development process, as it ensures that the program meets all necessary standards for full accreditation. All the uploaded document may view, update and delete. This model includes several functions :

1. **Manage Documents**
    - Module 4 handles the management of documents required to apply full accreditation by registering the document with specific program code. This function allows to upload a document referring their program code to stored according to the program name.
2. **Manage MQA-02 Documents**.
    - Module 4 handles the management of MQA-02 documents, which essential for the accreditation process. This function will retrieve the registered program code and set the current status of the program.
3. **Review Panel Member (App)**
    - The system management of review panel members, ensuring that the appropriate experts are involved in evaluating the program. This review panel can be consolidated by selecting the program code to be evaluated.
4. **Internal Reviews**
    - Internal reviews ensures that the program undergoes rigorous scrutiny and meets internal standards before it is submitted for full accreditation. The system tracks the status of documents.

5. **Manage Full Accreditation Application**
    - The final step involves the comprehensive management of all documents related to the full accreditation application. The system ensures that all required documents are compiled by retrieve the ID of the document and state the status.

By integrates these functionalities, module 4 ensure efficient pathway to full accreditation. The completed full accreditation show that the academic program documents has been thoroughly reviewed and successfully registered.

2.0 USECASE DIAGRAM

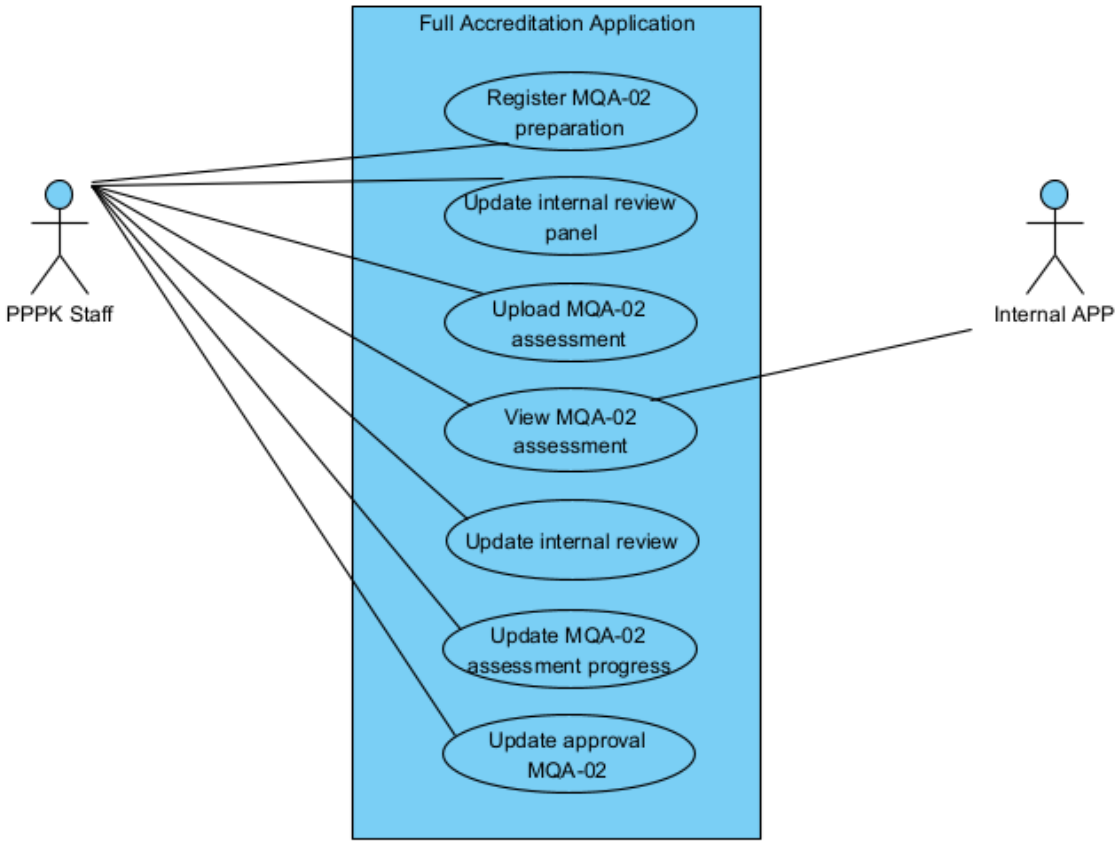

Figure 1 show usecase diagram of full accreditation application that involves multiple functionalities for achieving full accreditation application. This system is utilized primarily by PPPK Staff and Internal APP personnel, ensuring a collaborative approach to the accreditation workflow.

List of modules.

Module 1 : Register MQA-02 preparation

- This use case involves the initial registration of MQA-02 document, which is essential for the accreditation process. This allows PPPK staff to register new program code.

Module 2 : Update internal review panel

- This use case allows the PPPK staff to update the information regarding the review panel members. This can include adding new review panel.

Module 3 : Upload MQA-02 assessment

- This use case involves the uploading of the MQA-02 documents to the system by PPPK staff. These documents are show the status of program.

Module 4 : View MQA-02 assessment

- This use case allows the Internal APP to view the MQA-02 assessment documents. It indicates that the internal review has download to access the documents.

Module 5 : Update internal review

- This use case involves updating the internal review process. The PPPK staff can update the status of the document and necessary changes to the academic program.

Module 6 : Update MQA-02 assessment progress

- This use case allows the PPPK staff to update the progress of the MQA-02 assessment. This can include updating the documents related to the program.

Module 7 : Update approval MQA-02

- This use case involves updating the approval status of the MQA-02 document. The PPPK staff can record the approval or rejection of the document and any related comments or conditions.

3.0 CLASS DIAGRAM

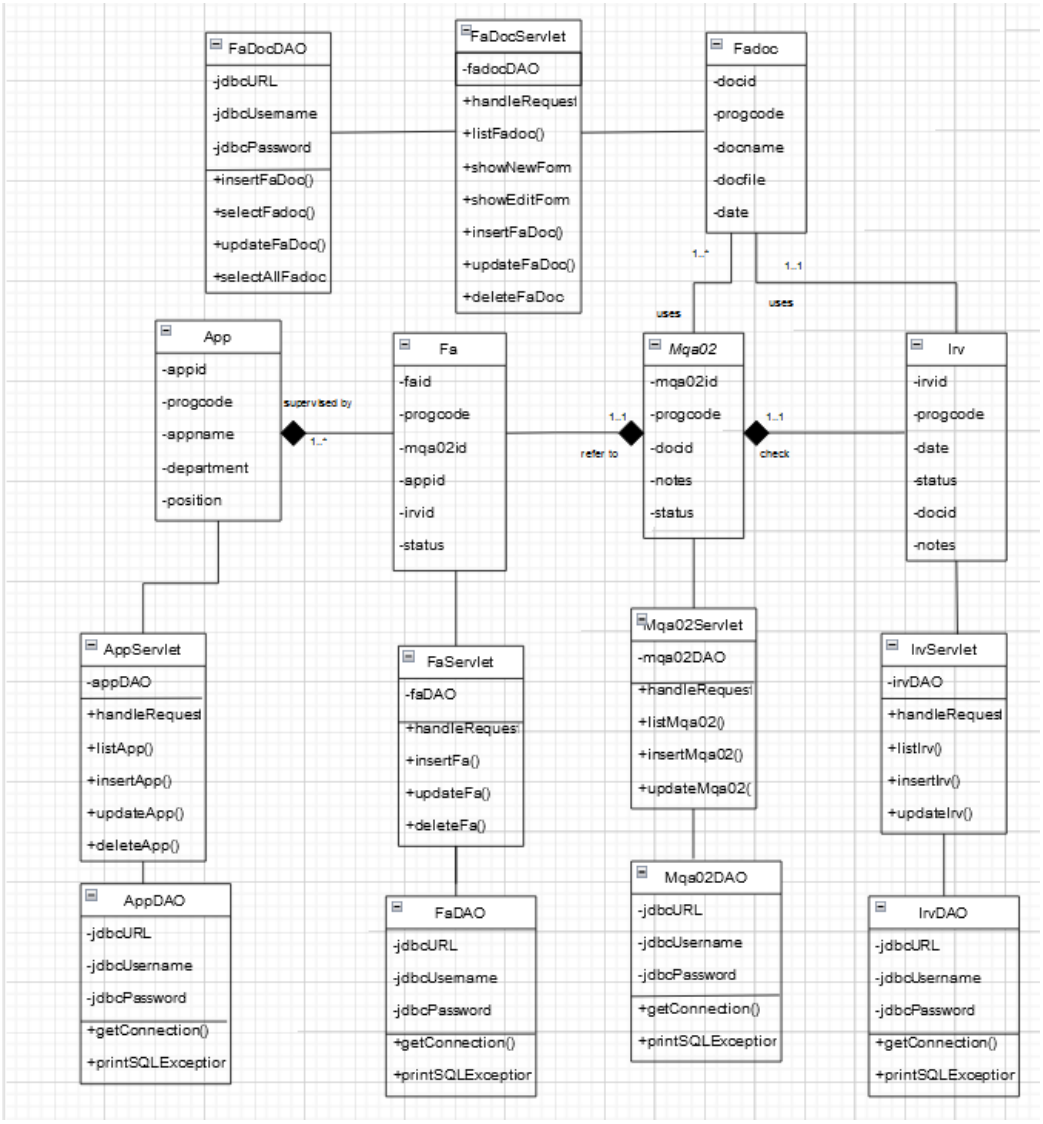

Figure 2 shows the class diagram that outlines module 4, Full Accreditation Application for UMT Academic Program Development Tracking System. It features of 5 classes; App, Fadoc, Mqa02, Irv, and Fa. Next, a Fadoc(Full Accreditation documents) can be assigned to 1 or many Mqa02 application and can be assigned to an Irv (internal review). A Mqa02 application required to be evaluated by an Irv. Lastly, Fa (Full Accreditation) is supervised by 1 or many APP (Review Panel Member) and refers to Mqa02 status. These classes also can be considered as model

component in the MVC architecture that had been applied throughout the development of this module.

There are also data access object (DAO) classes such as AppDAO, FaDAO ,IrvDAO and many more to manage the CRUD process with database.These classes also establish the connection to the database by declaring the driver url, username and password.

Lastly they are also servlet classes such as FaServlet, Appservlet and many more which play important roles to process the user request and generate response back. The servlet is the controller component in MVC centric architecture.It benefits the business logic provided my the models and call the dao classes if the data from database is required.

4.0 USER TRACEABILITY MATRIX

Req. ID	Requirement Description	Justification	Test Case ID	Test Result	Notes
1	Landing Page	Starting page,give user option to manage all application	Test01	Pass	
2	View informations	Necessary for managing documents,MQA02 submissions,APP members, internal reviews submissions and FA applications	Test02	Pass	All information displayed correctly
3	Add new informations	Allows users to upload new documents, submit new MQA-02 applications, new APP member registrations, new internal review reports and new FA applications	Test03	Pass	All information added successfully
4	Edit existing informations	Allows for document,MQA02,APP members, internal review and FA application updates	Test04	Pass	Information edited and saved

Figure 3: User Traceability Matrix

Figure 3 shows The User Traceability Matrix which outlines the key features and corresponding justifications for a system meant to handle multiple sorts of information, such as documents, MQA-02 submissions, APP member information, internal review reports, and Full Accreditation (FA) applications. It starts with the landing page, which acts as a primary portal for users to explore and administer all application sections. Viewing information is important for managing all entries in the system; adding new information ensures that users may submit necessary documents and applications; and editing existing information allows for changes and

corrections. Each need is assigned to a distinct test case, which have all succeeded, indicating the system's capacity to execute these activities swiftly and effectively.

5.0 ENTITY RELATIONSHIP DIAGRAM

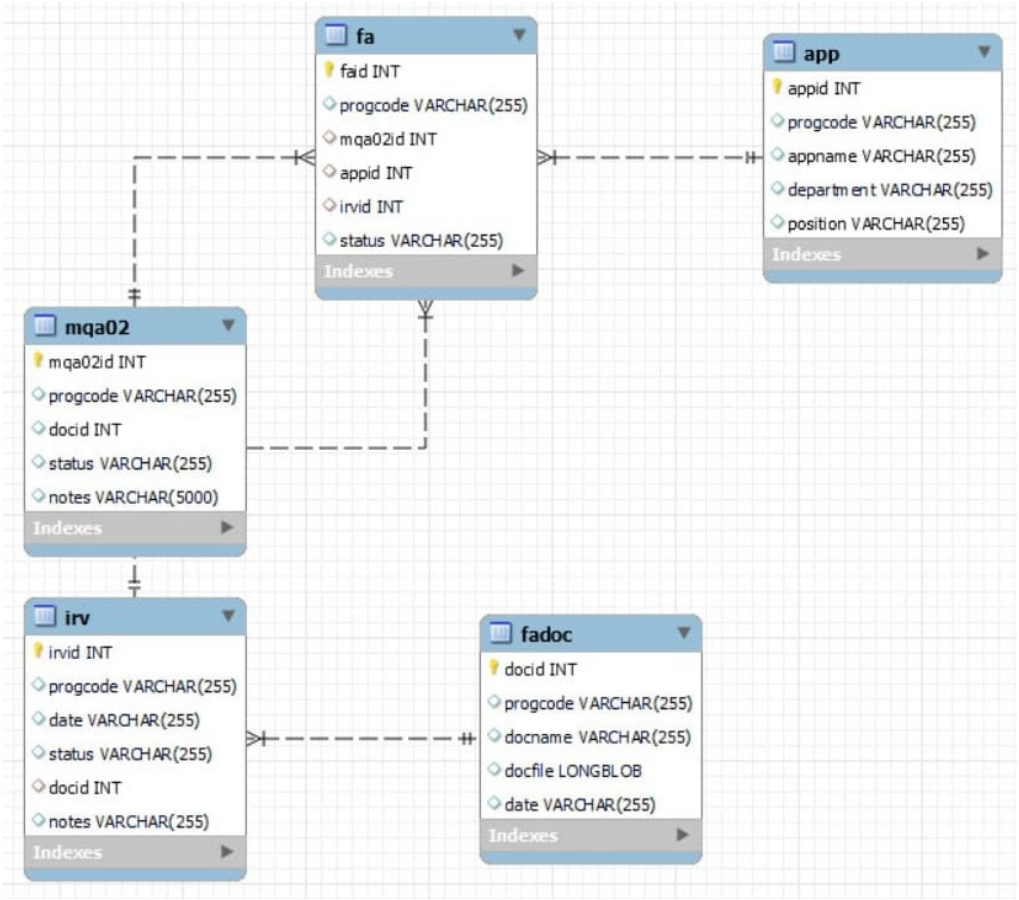

Figure 4 : ER diagram

Figure 4 shows the portion of the Entity Relationship Diagram (ERD) for the system Full Accreditation. Below are detailed explanation of each entity and relationships :

1. Fa
    - faid (INT): Unique identifier for the full accreditation record.
    - progcode (VARCHAR(255)): Program code associated with the accreditation.
    - mqa02id (INT): Foreign key referencing the MQA-02 document.
    - appid (INT): Foreign key referencing the review panel member.

- irvid (INT): Foreign key referencing the internal review.
- status (VARCHAR(255)): Status of the full accreditation application.

2. Mqa02
    - mqa02id (INT): Unique identifier for the MQA-02 document.
    - progcode (VARCHAR(255)): Program code associated with the MQA-02 document.
    - docid (INT): Foreign key referencing the document.
    - status (VARCHAR(255)): Status of the MQA-02 document.
    - notes (VARCHAR(5000)): Additional notes or comments about the MQA-02 document.
3. App
    - appid (INT): Unique identifier for the review panel member.
    - progcode (VARCHAR(255)): Program code associated with the review panel member.
    - appname (VARCHAR(255)): Name of the review panel member.
    - department (VARCHAR(255)): Department of the review panel member.
    - position (VARCHAR(255)): Position of the review panel member.
4. Irv
    - irvid (INT): Unique identifier for the internal review.
    - progcode (VARCHAR(255)): Program code associated with the internal review.
    - date (VARCHAR(255)): Date of the internal review.
    - status (VARCHAR(255)): Status of the internal review.
    - docid (INT): Foreign key referencing the document.
    - notes (VARCHAR(255)): Additional notes or comments about the internal review.
5. Fadoc
    - docid (INT): Unique identifier for the document.
    - progcode (VARCHAR(255)): Program code associated with the document.
    - docname (VARCHAR(255)): Name of the document.
    - docfile (LONGBLOB): The actual document file.
    - date (VARCHAR(255)): Date the document was created or uploaded.

Relationships :

1.**fa to mqa02**

- The fa entity has a foreign key mqa02id that references mqa02id in the mqa02 entity. This indicates that each full accreditation application is associated with a specific MQA-02 document.

2.**fa to app**

- The fa entity has a foreign key appid that references appid in the app entity. This signifies that each full accreditation application involves a specific review panel member.

3.**fa to irv**

- The fa entity has a foreign key irvid that references irvid in the irv entity. This indicates that each full accreditation application is linked to a specific internal review.

4.**mqa02 to fadoc**

- The mqa02 entity has a foreign key docid that references docid in the fadoc entity. This indicates that each MQA-02 document is associated with a specific document file.

5.**irv to fadoc**

- The irv entity has a foreign key docid that references docid in the fadoc entity. This signifies that each internal review is linked to a specific document file.

This ERD effectively depicts the comprehensive structure of the full accreditation application process, highlighting how the different components and documents are connected to ensure a streamlined and organized accreditation workflow.

6.0 MVC MODEL

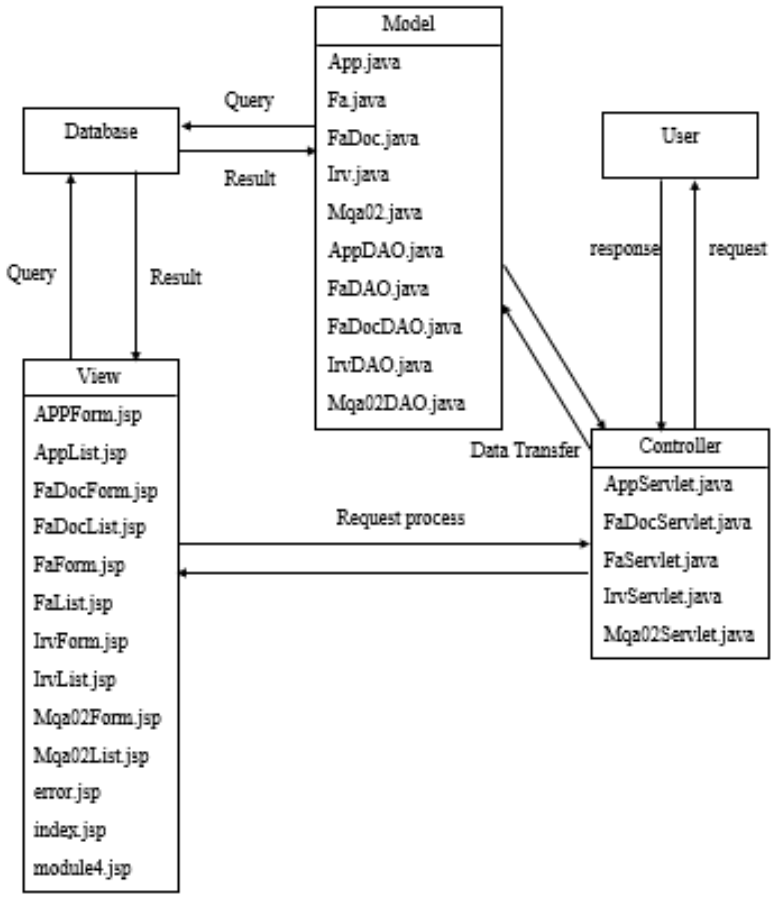

Figure 5 : MVC Model

Figure 5 show a Model-View-Controller (MVC) architecture that applied to a full accreditation system.

1. **Model**

- Model represents the data and the business logic of the full accreditation application. The model includes classes like App.java, Fa.java, FaDoc.java, Irv.java, Mqa02.java, AppDAO.java, FaDAO.java, FaDocDAO.java, IrvDAO.java. These classes represent different entities or components related to full accreditation that directly manage the data, logic and rules of the full accreditation application.

2. **View**

- The view is a presentation layer that display data to the user and sends the user command to the controller. Based on the diagram, the view include all the Java Server Page (JSP) files such as APPForm.jsp, AppList.jsp, FaDocForm.jsp and others. These JSP files make the interface for different parts of the full accreditation process. It allowing users to interact with the system.

3. **Controller**

- The controller handles the input from the user, manipulates the model, and updates the view. It acts as an intermediary between the model and the view. The controller includes Servlet classes like AppServlet.java and FaDocServlet.java. The Servlet classes manage the HTTP request and responses also controls the flow between the user interface and the data logic.

Essentially, user interaction starts when a user submits a request through the View layer in JavaServer Page (JSP), which is captured by the Controller layer (Servlet classes). The Controller processes the request, often invoking methods in the Model layer (Java classes) to handle business logic or data manipulation. The Model layer, consisting of classes like App.java and Fa.java, performs the necessary operations, potentially querying the database via DAO classes, which handle these interactions and return results. The Model processes these results and sends them back to the Controller. The Controller then formulates an appropriate response, which is sent to the View layer. The View generates the final user interface based on this data and presents it to the user.

In the full accreditation system, Model classes such as App.java and Fa.java represent various full accreditation components. DAO classes like AppDAO.java and FaDAO.java manage database operations for these components. Servlet classes act as intermediaries, processing user requests and determining the appropriate View. JSP files in the View layer render forms, lists, and other interfaces for managing and displaying accreditation data.

7.0 SAMPLE OF INTERFACE

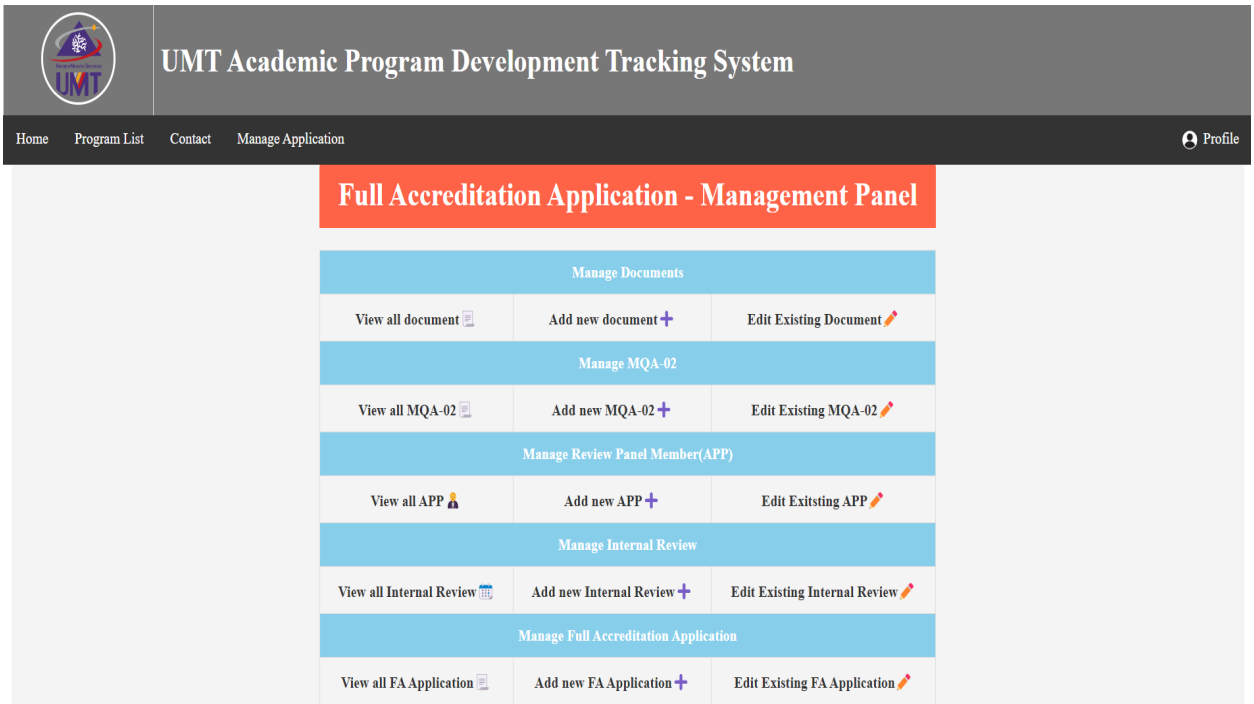

Figure 4:Management Panel

Figure 6 shows the management panel of the UMT Academic Program Development Tracking System's Full Accreditation Application. It provides options to manage documents, MQA-02, review panel members (APP), internal reviews, and the full accreditation application. Users can view, add, or edit entries for each category through the interface.

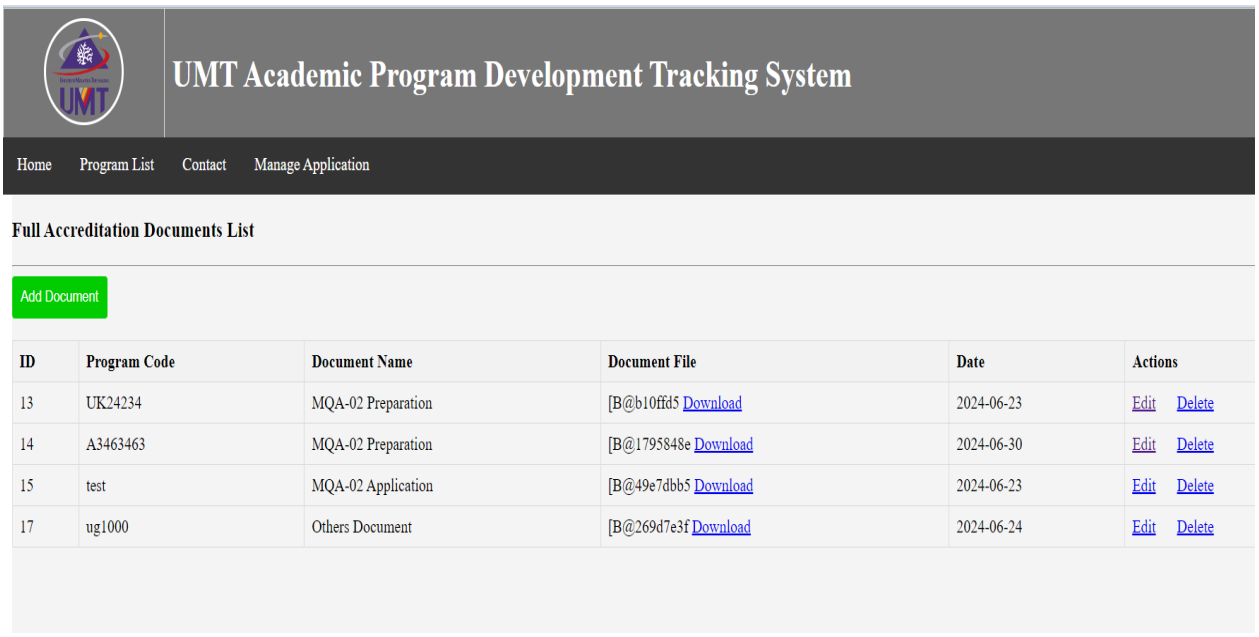

Figure 5:View added information

Figure 7 shows the Full Accreditation Documents List of the UMT Academic Program Development Tracking System. It displays a table with columns for ID, program code, document name, document file (with download links), date, and actions (edit and delete). Users can add new documents using the "Add Document" button and manage existing documents through the edit and delete options.

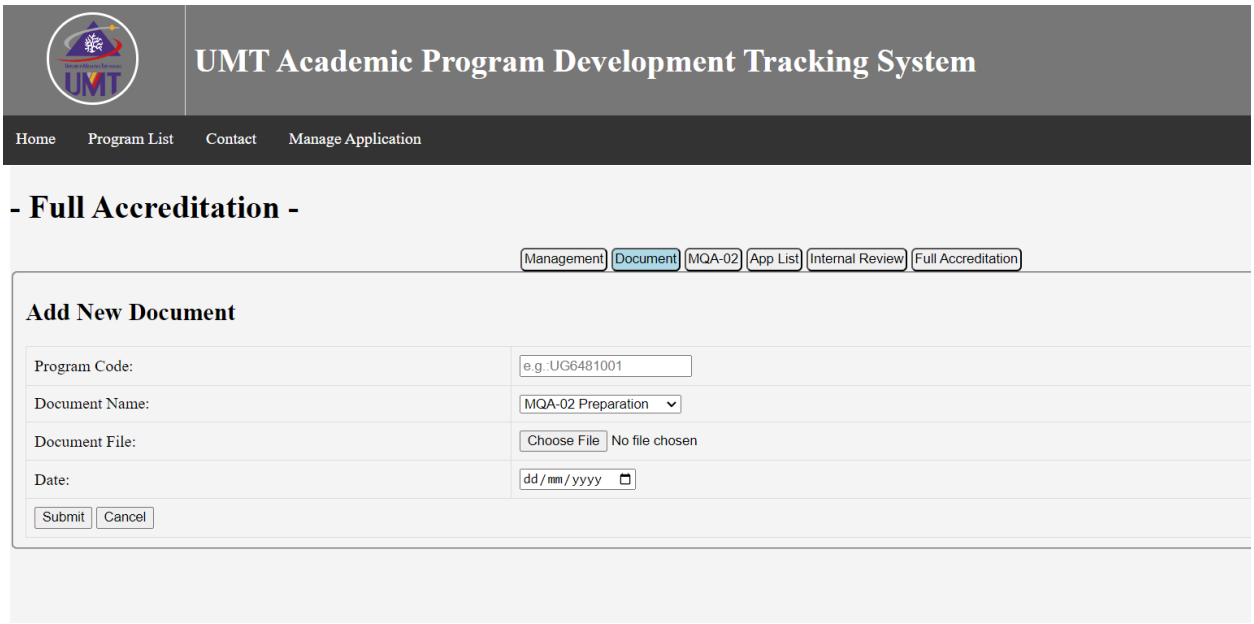

Figure 6:Add new document

Figure 8 shows the interface for adding a new document in the Full Accreditation section of the UMT Academic Program Development Tracking System. Users can input the program code, select the document name from a dropdown menu, choose a file to upload, and enter the date. There are "Submit" and "Cancel" buttons to complete or cancel the action.

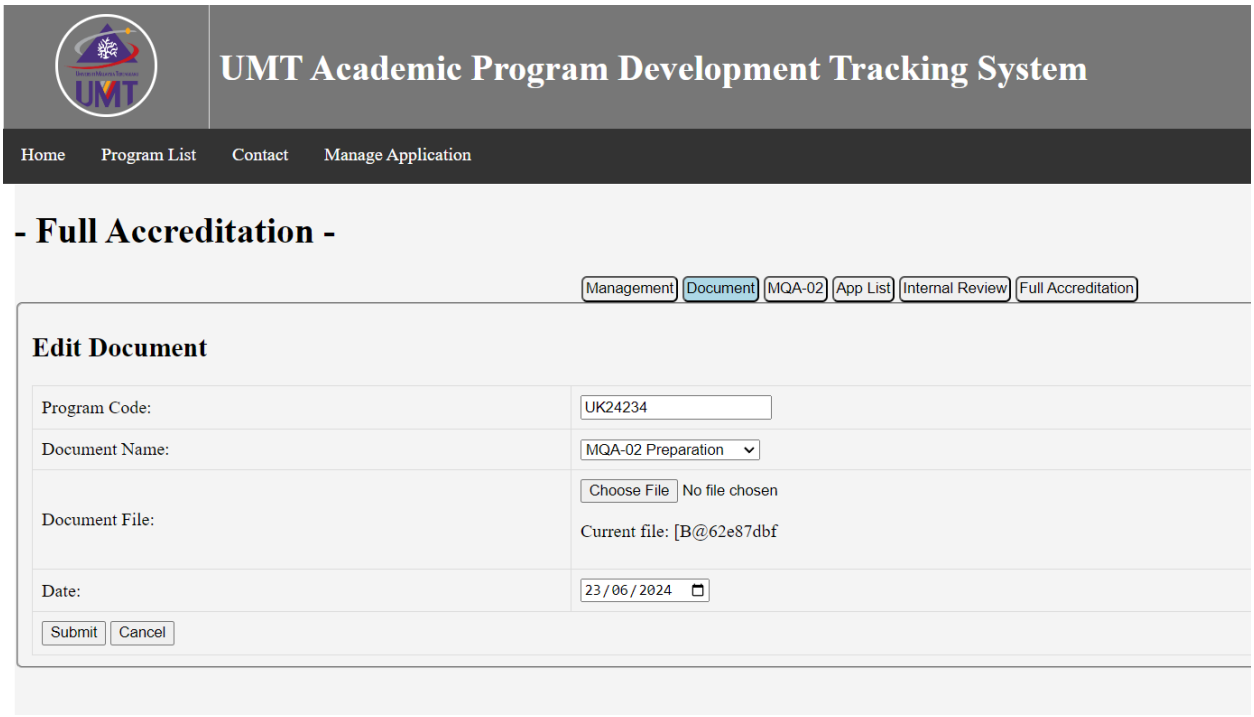

Figure 7:Edit existing document

Figure 9 shows the interface for editing an existing document in the Full Accreditation section of the UMT Academic Program Development Tracking System. Users can modify the program code, select a different document name from a dropdown menu, choose a new file to upload, and update the date. There are "Submit" and "Cancel" buttons to save or cancel the changes.

8.0 SAMPLE PORTION OF CODING FOR DATABASE CONNECTION, CRUD PROCESS, AND JAVABEANS.

a) Database connection

- AppDAO.java

```
public class AppDAO {
    Connection connection = null;
    private String jdbcURL = "jdbc:mysql://localhost:3306/apdtsystem";
    private String jdbcUsername = "root";
    private String jdbcPassword = "admin";

    private static final String INSERT_APP_SQL = "INSERT INTO app(progcode, appname, department, position) VALUES (?, ?, ?, ?);";
    private static final String SELECT_APP_BY_ID = "select appid, progcode, appname, department, position from app where appid=?";
    private static final String SELECT_ALL_APP = "select * from app";
    private static final String DELETE_APP_SQL = "delete from app where appid = ?;";
    private static final String UPDATE_APP_SQL = "update app set progcode = ?,appname= ?, department= ?, position= ? where appid = ?;";
```

b) CRUD process

- Insert data

```
44      public void insertApp(App app) throws SQLException{
45          System.out.println(INSERT_APP_SQL);
            try(Connection connection = getConnection();
47              PreparedStatement preparedStatement = connection.prepareStatement(INSERT_APP_SQL)){
48              preparedStatement.setString(1, app.getProgcode());
49              preparedStatement.setString(2, app.getAppname());
50              preparedStatement.setString(3, app.getDepartment());
51              preparedStatement.setString(4, app.getPosition());
52              System.out.println(preparedStatement);
53              preparedStatement.executeUpdate();
54          }catch(SQLException e){
55              printSQLException(e);
56                  }
57      }
```

- Select data

```
59      public App selectApp(int appid){
60          App app = null;
61          // Step 1: Establishing a Connection
            try(Connection connection = getConnection();
63              // Step 2: Create a statement using connection
64              PreparedStatement preparedStatement = connection.prepareStatement(SELECT_APP_BY_ID);){
65              preparedStatement.setInt(1, appid);
66              System.out.println(preparedStatement);
67              ResultSet rs = preparedStatement.executeQuery();
68
69              while(rs.next()){
70                  String progcode = rs.getString("progcode");
71                  String appname = rs.getString("appname");
72                  String department = rs.getString("department");
73                  String position = rs.getString("position");
74                  app = new App(appid, progcode, appname, department, position);
75              }
76          }catch (SQLException e){
77                  printSQLException(e);
78                  }
79          return app;
80      }
```

- Select all data

```
public List < App > selectAllApp(){
    List <App> apps = new ArrayList <>();
    try (Connection connection = getConnection();

        PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_APP);){
        System.out.println(preparedStatement);
        ResultSet rs = preparedStatement.executeQuery();

        while(rs.next()){
            int appid = rs.getInt("appid");
            String progcode = rs.getString("progcode");
            String appname = rs.getString("appname");
            String department = rs.getString("department");
            String position = rs.getString("position");
            apps.add(new App(appid, progcode, appname, department, position));
        }
    }catch(SQLException e){
        printSQLException(e);
    }
    return apps;
}
```

- Delete data

```
public boolean deleteApp(int appid) throws SQLException{
    boolean rowDeleted;
    try(Connection connection = getConnection(); PreparedStatement statement =
            connection.prepareStatement(DELETE_APP_SQL);){
        statement.setInt(1, appid);
        rowDeleted = statement.executeUpdate() > 0;
    }
    return rowDeleted;
}
```

- Update data

```
public boolean updateApp(App app) throws SQLException{
    boolean rowUpdated;
    try(Connection connection = getConnection(); PreparedStatement statement =
            connection.prepareStatement(UPDATE_APP_SQL);){
        statement.setString(1, app.getProgcode());
        statement.setString(2, app.getAppname());
        statement.setString(3, app.getDepartment());
        statement.setString(4, app.getPosition());
        statement.setInt(5, app.getAppid());

        rowUpdated = statement.executeUpdate() > 0;
    }
    return rowUpdated;
}
```

- Retrieve data

```
129     private void printSQLException(SQLException ex){
130         for(Throwable e: ex){
131             if(e instanceof SQLException){
132                 e.printStackTrace(System.err);
133                 System.err.println("SQLState: " + ((SQLException) e).getSQLState());
134                 System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
135                 System.err.println("Message: " + e.getMessage());
136                 Throwable t = ex.getCause();
137                 while(t != null){
138                     System.out.println("Cause: " + t);
139                     t = t.getCause();
140                 }
141             }
142         }
143     }
```

c) JavaBeans

```
5   package com.model;
6
7   public class App {
8       private int appid;
9       private String progcode;
10      private String appname;
11      private String department;
12      private String position;
13
14      public App() {
15      }
16
17      public App(String progcode, String appname, String department, String position) {
18          super();
19          this.progcode = progcode;
20          this.appname = appname;
21          this.department = department;
22          this.position = position;
23      }
24
25      public App(int appid, String progcode, String appname, String department, String position) {
26          this.appid = appid;
27          this.progcode = progcode;
28          this.appname = appname;
29          this.department = department;
30          this.position = position;
31      }
```

```
33      public int getAppid() {
34          return appid;
35      }
36
37      public void setAppid(int appid) {
38          this.appid = appid;
39      }
40
41      public String getProgcode() {
42          return progcode;
43      }
44
45      public void setProgcode(String progcode) {
46          this.progcode = progcode;
47      }
48
49      public String getAppname() {
50          return appname;
51      }
52
53      public void setAppname(String appname) {
54          this.appname = appname;
55      }
56
57      public String getDepartment() {
58          return department;
59      }
60
61      public void setDepartment(String department) {
62          this.department = department;
63      }
```

```
65      public String getPosition() {
66          return position;
67      }
68
69      public void setPosition(String position) {
70          this.position = position;
71      }
72
73  }
```

9.0 INDIVIDUAL CONTRIBUTION

NO	MATRIC NO	NAME	MODULE
1	S67911	Lutfil Haziq Bin Adnan	• AppDao.java • AppServlet.java • Mqa02DAO.java
2	S67604	Harinatul Muflihun Binti Hasnul Munawar	• FaDAO.java • FaServlet.java • FaDocDAO.java
3	S67978	Muhammad Haziq Aiman Bin Mustafa	• FaDocServlet.java • IrvDAO.java • IrvServlet.java • Mqa02Servlet.java

1. Lutfil Haziq Bin Adnan

Lutfil was responsible for creating and managing the data access object (DAO) for the application through AppDao.java. This involved designing methods to interact with the database, ensuring efficient retrieval, insertion, updating, and deletion of records. His work focused on maintaining a clean and scalable code structure for database operations. He also developed AppServlet.java, the servlet responsible for handling HTTP requests and responses related to the application. This servlet acted as the controller in the MVC (Model-View-Controller) architecture, managing user inputs, invoking the appropriate DAO methods, and forwarding the data to the appropriate view. Additionally, Lutfil worked on Mqa02DAO.java, another DAO specifically for handling MQA02-related data, ensuring data integrity and optimal performance.

2. Harinatul Muflihun Binti Hasnul Munawar

Harinatul was in charge of FaDAO.java, focusing on data operations related to the FA module. She implemented methods to handle complex queries and transactions, ensuring the data layer was robust and could support the application's requirements. She also developed FaServlet.java, which handled web requests for the FA module, creating the logic to process user inputs, interact with the FaDAO, and manage the response sent back to the client, adhering to the principles of MVC. Additionally, Harinatul managed FaDocDAO.java, a specialized DAO for handling document-related operations within the FA module, ensuring that document storage, retrieval, and management were efficient and secure, contributing to the module's overall functionality.

3. Muhammad Haziq Aiman Bin Mustafa

Haziq responsible for developing FaDocServlet.java, the servlet that handled document-related web requests within the FA module. His work involved processing file uploads/downloads, validating user inputs, and ensuring seamless interaction with the FaDocDAO. He also created and managed IrvDAO.java, focusing on data operations related to the IRV module, including designing efficient database queries and maintaining data consistency and integrity. Muhammad developed IrvServlet.java, which processed HTTP requests and responses for the IRV module, implementing the logic to handle user interactions, invoke DAO methods, and forward data to the appropriate views. Additionally, he worked on Mqa02Servlet.java, handling web requests for the MQA02 module, ensuring that the servlet efficiently processed requests, interacted with the Mqa02DAO, and managed the user interface interactions.

10.0 CONCLUSION

UMT Academic Program Development Tracking System is an important tool designed to simplify and improve academic development management. Incorporating a structured approach through its four modules, the system provides a comprehensive solution to address the complex processes of full accreditation.

This report focused on Module 4, Full Accreditation Application, which detailed its functions to ensure that programs meet the standards required for full accreditation. The system's ability to manage documentation, track MQA-02 documentation, convene review boards, conduct internal review , and manage the overall accreditation process ensures a robust and efficient workflow.

Using a Model-View-Controller (MVC) framework makes it easier to clearly separate concerns, increasing maintenance and scalability. Entity-relationship diagrams and class diagrams provide a solid foundation for the data structure of the system, and ensure that all components are properly integrated.

The collaborative efforts of team members in producing specific modules and features ensures system efficiency, with each member contributing significantly to different aspects of the project The user's traceability matrix confirms that the system performs required services efficiently.

In conclusion, the UMT Academic Program Development Tracking System with its comprehensive design and effective implementation contributes significantly to the smooth and systematic management of program accreditation, and helps UMT maintain high standards in his educational offerings.

11.0 REFERENCES

1. *Portal rasmi MQA. (n.d.).* https://www.mqa.gov.my/new/bm/faq_permohonanakr.cfm#gsc.tab=0
2. *Full Accreditation MQA 02*. (n.d.). UNIMAS Main Website. https://www.bjka.unimas.my/faq/full-accreditation-mqa-02
3. GeeksforGeeks. (2024, April 16). *MVC Framework Introduction*. GeeksforGeeks. https://www.geeksforgeeks.org/mvc-framework-introduction/
4. *MySQL :: Connectors and APIs Manual :: 3.3.2 Installing Connector/J using Maven*. (n.d.). https://dev.mysql.com/doc/connectors/en/connector-j-installing-maven.html
5. *Difference between @Path and @WebServlet*. (n.d.). Stack Overflow. https://stackoverflow.com/questions/24758101/difference-between-path-and-webservlet
6. GeeksforGeeks. (2022, February 2). *Servlet form*. GeeksforGeeks. https://www.geeksforgeeks.org/servlet-form/