BITP 3123 Distributed Application Development | FTMK, UTeM

Project Artefact Declaration

Sem 2 2022/2023

PROJECT TITLE: 2. EXAMINATION ATTENDANCE SYSTEM IN PHYSICAL SPACE

GROUP NO: 10

LIST OF GROUP MEMBERS

Matric No	Name
B032110132	MOHD HAFIZ SUHAIZAL BIN ISMAIL
B032110201	NG WEI HEN
B032110304	WAFIR DZIHNI BIN ROZUKI

Table of Contents

Description of Document

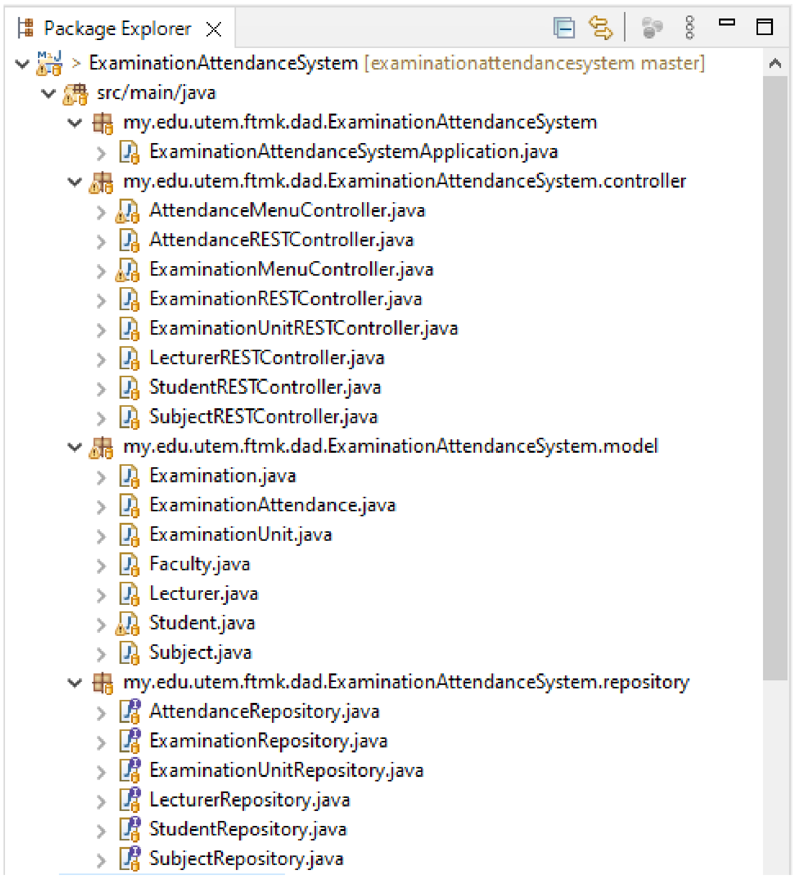
This document describes the development of web services for Examination Attendance System using Spring Boot as the framework. In this project, Maven will be used as a tool for implementing the web service for both provider and consumer. For the front-end at consumer side, languages such as Thymeleaf, Hypertext Markup Language (HTML) and Cascading Style Sheet (CSS) are used to design the webpage.

This document contains several sections including project structure, data structure, implementation of data layer, implementation of web services, implementation of front-end controller and implementation of the front-end.

Project Structure

The project structure is shown as following:

1. Web service implementation:



```
Package Explorer
ExaminationAttendanceSystem [examinationattendancesystem master]
  src/main/java
    my.edu.utem.ftmk.dad.ExaminationAttendanceSystem
      ExaminationAttendanceSystemApplication.java
    my.edu.utem.ftmk.dad.ExaminationAttendanceSystem.controller
      AttendanceMenuController.java
      AttendanceRESTController.java
      ExaminationMenuController.java
      ExaminationRESTController.java
      ExaminationUnitRESTController.java
      LecturerRESTController.java
      StudentRESTController.java
      SubjectRESTController.java
    my.edu.utem.ftmk.dad.ExaminationAttendanceSystem.model
      Examination.java
      ExaminationAttendance.java
      ExaminationUnit.java
      Faculty.java
      Lecturer.java
      Student.java
      Subject.java
    my.edu.utem.ftmk.dad.ExaminationAttendanceSystem.repository
      AttendanceRepository.java
      ExaminationRepository.java
      ExaminationUnitRepository.java
      LecturerRepository.java
      StudentRepository.java
      SubjectRepository.java
```

These are the files used for developing the system at back-end. The back-end of the system consists of 3 main parts, which are model, controller and repository.

At **model** side, classes such as Faculty, Lecturer, Student and so on are created to store basic data-related logic to the examination attendance. The attendance will be recorded in ExaminationAttendance based on Examination class which stores about the examination schedule. While handling examination schedule, ExaminationUnit class is included to decide the venue for a certain examination.

At **controller** side, REST controller and Menu controller are included to handle all the information at the model side.

- ***Examination Attendance Controller***
  AttendanceRESTController will be the controller that provides web service to handle students' attendance including the attendance based on venue at provider based on ExaminationAttendance class and its related classes. While AttendanceMenuController interacts with AttendanceRESTController to parse the data retrieved from the provider side to the consumer side which will be displayed at front-end webpage.

- ***Examination Schedule Controller***
  ExaminationRESTController will be the controller that provides web service to handle examination schedule including the respective venue at provider based on Examination class and its related class. While ExaminationMenuController interacts with ExaminationRESTController to parse the data retrieved from the provider side to the consumer side which will be displayed at front-end webpage.

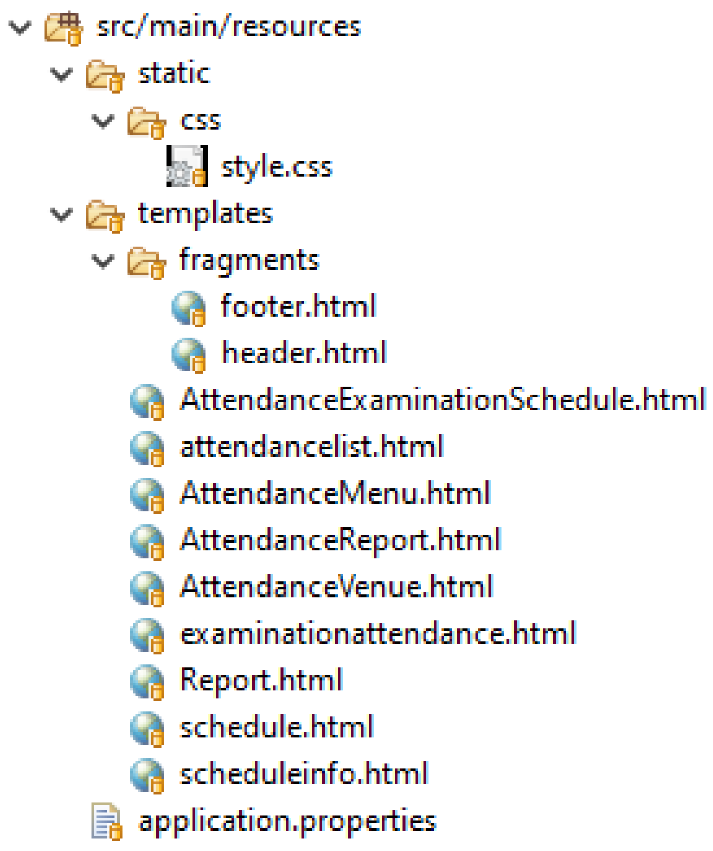
- ***Other Controller***
  Controllers such as ExaminationUnitRESTController, LecturerRESTController, SubjectRESTController and StudentRESTController are used to be displayed

as options such as drop-down menu at front-end based on current record saved in the database to do operation like manipulating examination schedule and make examination attendance records.

At **repository** site, JpaRepository will be used as a main interface to create repositories such as AttendanceRepository, ExaminationUnitRepository, ExaminationRepository, LecturerRepository, StudentRepository and SubjectRepository for data managing in repository. These repositories will inherit JpaRepository interface where generic SQL statements are encapsulated in the repository methods. Customizable SQL queries can be also written in the repository methods to suit the requirements of functionality at the application.

The whole system will be executed as application at ExaminationAttendanceSystemApplication class as it contains main method to execute.

2. Front-end design implementation:



```
src/main/resources
  static
    css
      style.css
  templates
    fragments
      footer.html
      header.html
    AttendanceExaminationSchedule.html
    attendancelist.html
    AttendanceMenu.html
    AttendanceReport.html
    AttendanceVenue.html
    examinationattendance.html
    Report.html
    schedule.html
    scheduleinfo.html
  application.properties
```

The front-end implementation contains following folders:

- ***css***
  This folder contains Cascading Style Sheet (CSS) which is style.css and home.css for designing the front-end of entire Examination Attendance System.

- ***templates/fragments***
  This folder contains the footer (footer.html) and header (header.html) of the front-end webpage of the system. The header contains a navigation bar to navigate user for every webpage linked to the system. The footer contains the copyright text to show this system is made by our group.

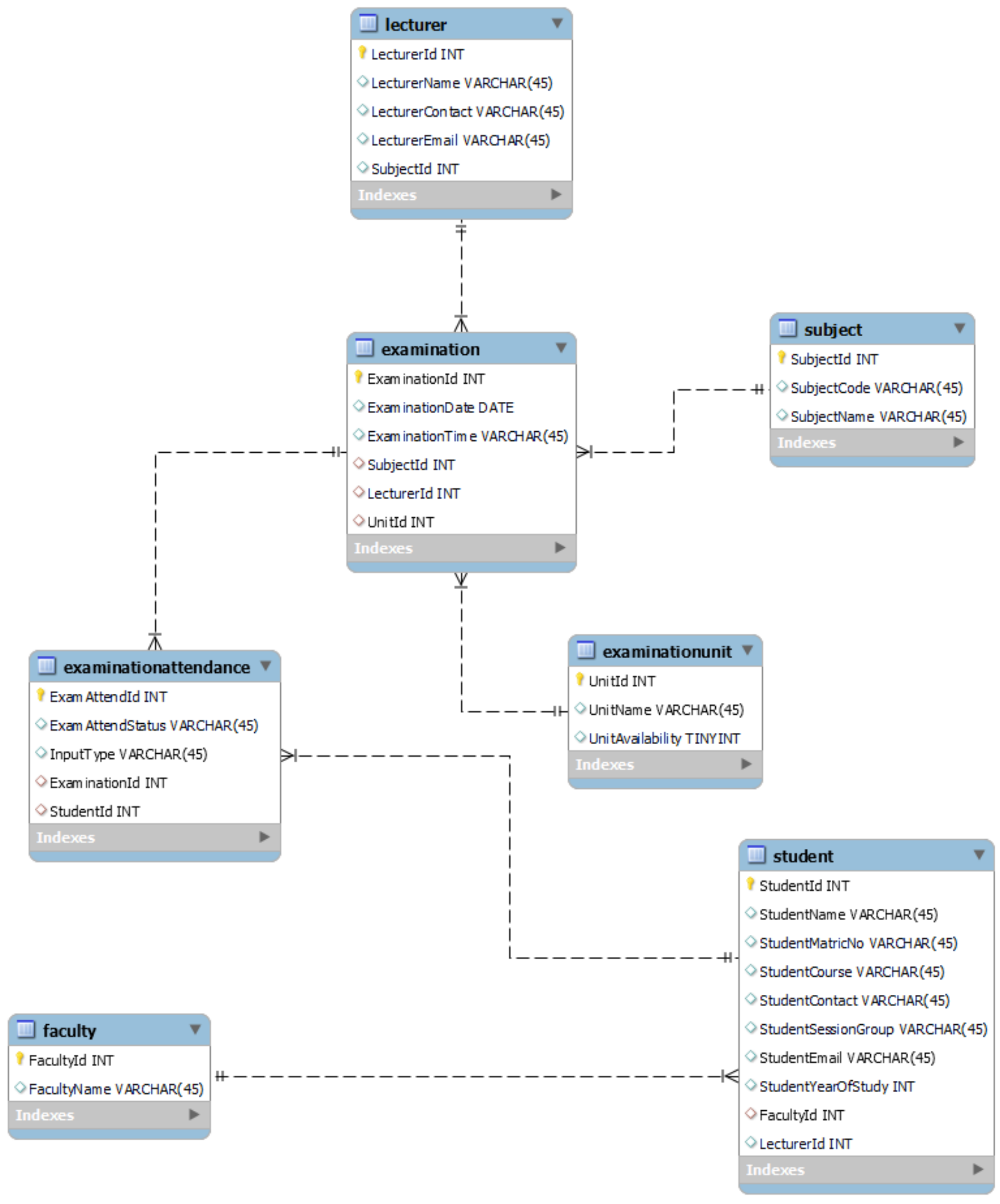
- ***Templates***
  The folder "templates" contains following front-end webpages:

Files for front-end	Description
AttendanceExaminationSchedule	This webpage mainly displays the report of absent students' attendance.
attendancelist	This webpage displays the list of attendance for all students. User can choose to print the list by clicking the "Print" button.
AttendanceMenu	This webpage will display the list of students' attendance for subject.
AttendanceReport	This webpage displays the selection of subjects to view the students' attendance report.
AttendanceVenue	This webpage displays the students' attendance based on examination unit as venue. User can select

	examination unit to filter student’s attendance.
examinationattendance	This is the main menu for front-end to record student attendance. Validation is included to ensure student’s existence is true before attendance is recorded.
Report	This is the report of students’ attendance based on subject selected in AttendanceReport. User can choose to print the report by clicking the “Print” button.
schedule	This front-end page will display the examination schedule with subjects, date and time. It contains buttons to add, update and delete certain examination schedules. User can also click the examination schedule to navigate to “examinationattendance” for recording examination attendance
scheduleinfo	This is a front-end form to add a new examination schedule. After submitting the form, the examination schedule record will be saved to the database.

Data Structure

Entity-Relationship Diagram (ERD)



Data Dictionary

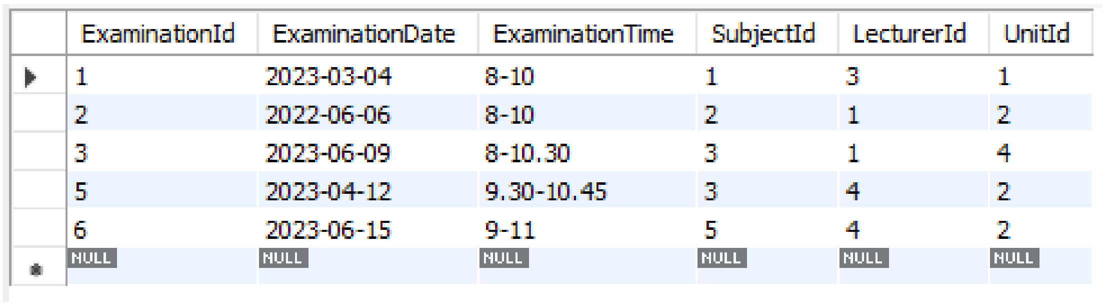
TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	UNIQUE	REQUIRED (Y/N)	PK OR FK	FK REFERENCE TABLE
STUDENT	Student_ID	The Student ID	INT	1234	UNIQUE	Y	PK	
	StudentName	The name of student	VARCHAR(45)	XXXXXXXX		Y		
	StudentMatricNo	The Matric No of the student	VARCHAR(45)	B0XXXXXXXX		Y		
	StudentSessionGroup	The session group of student	VARCHAR(45)	SXGX		Y		
	StudentCourse	The course taken by student	VARCHAR(45)	BXXX		Y		
	StudentContact	The phone number Of student	VARCHAR(45)	0123456789 012		Y		
	StudentEmail	The email of student	VARCHAR(45)	XXXXX@XXX X.COM		Y		
	StudentYearOfStudy	The year of study of student	INT	2XXX		Y		
	FacultyID	The Faculty Id	INT	XXXX		Y	FK	FACULTY
	LecturerID	The Lecturer ID	INT	X000000000		Y	FK	LECTURER
EXAMINATION	ExaminationID	The Examination ID	INT	1234	UNIQUE	Y	PK	
	Time	The time of examination	VARCHAR(45)	0000		Y		

	Date	The date of examination	DATE	YYYY-MM-DD		Y		
	Unit_ID	The Unit ID	INT	1234		Y	FK	EXAMINATION_UNIT
	Subject_ID	The Subject ID	INT	1234		Y	FK	SUBJECT
	Lecturer_ID	The Lecturer ID	INT	1234		Y	FK	LECTURER
LECTURER	Lecturer_ID	The Lecturer ID	INT	1234	UNIQUE	Y	PK	
	LecturerName	The name of lecturer	VARCHAR(45)	XXXXXXXX		Y		
	LecturerContact	The phone number of the lecturer	VARCHAR(45)	0123456789012		Y		
	LecturerEmail	The email of the lecturer	VARCHAR(45)	XXXXX@XXXX.COM		Y		
	Subject_ID	The Subject ID	INT	1234		Y	FK	SUBJECT
FACULTY	Faculty_ID	The Faculty ID	INT	1234	UNIQUE	Y	PK	
	Faculty_Name	The name of faculty	VARCHAR(45)	XXXXXXXXXX		Y		
SUBJECT	Subject_ID	The Subject ID	INT	1234	UNIQUE	Y	PK	
	Subject_Name	The name of the subject	VARCHAR(45)	XXXXXXXXXX		Y		

	SubjectCode	The code of the subject	VARCHAR(45)	XXXXXXX		Y		
Examinationattendance	ExamAttendId	The Examination Attendance ID	INT	1234		Y	PK	
	ExamAttendStatus	The Examination Attendance Status	VARCHAR(45)	Absent/Hadir		Y		
	InputType	The Input Type	VARCHAR(45)	Input Type		Y		
	ExaminationID	The Examination ID	INT	1234		Y	FK	Examination
	StudentID	The Student ID	INT	1234		Y	FK	Student
EXAMINATION_UNIT	Unit_ID	The unit ID of the building	INT	1234	UNIQUE	Y	PK	
	UnitName	The name of the unit building	VARCHAR(45)	XXXXXXXXXX		Y		
	UnitAvailability	The status availability of the building	VARCHAR(45)	AVAILABLE/NOTAVAILABLE		Y		

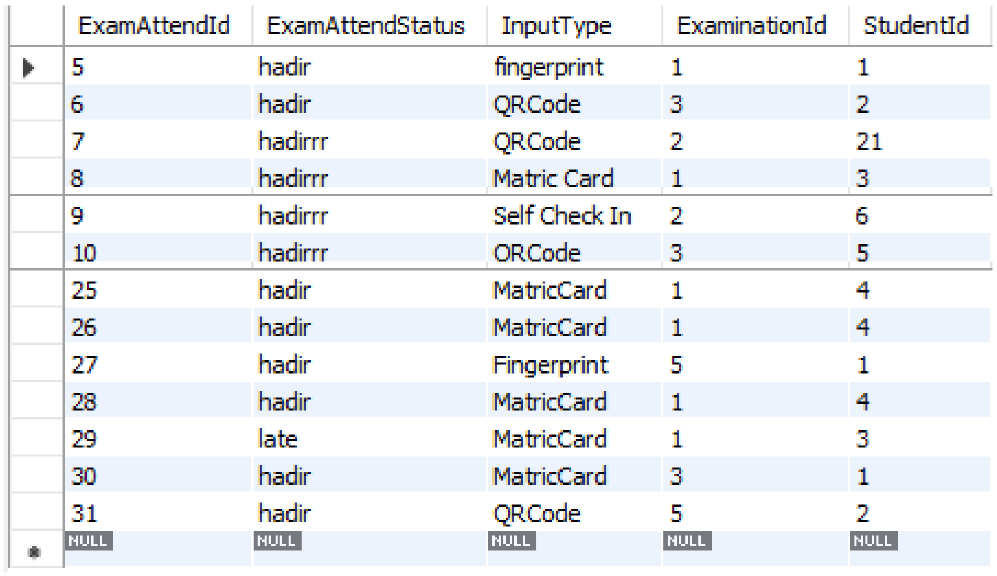
Samples of Data

1. Examination



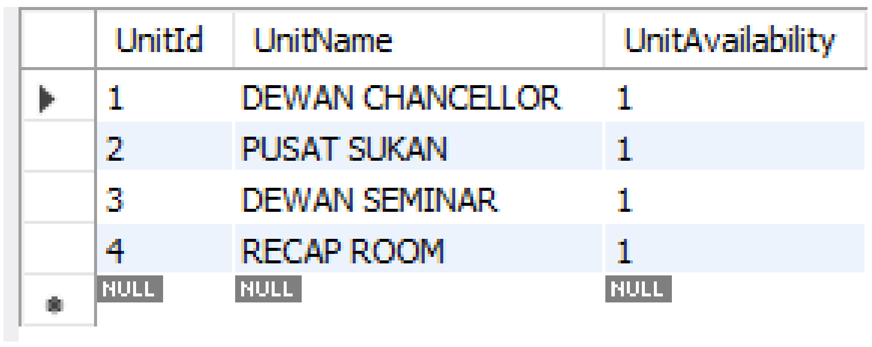
	ExaminationId	ExaminationDate	ExaminationTime	SubjectId	LecturerId	UnitId
▶	1	2023-03-04	8-10	1	3	1
	2	2022-06-06	8-10	2	1	2
	3	2023-06-09	8-10.30	3	1	4
	5	2023-04-12	9.30-10.45	3	4	2
	6	2023-06-15	9-11	5	4	2
*	NULL	NULL	NULL	NULL	NULL	NULL

2. ExaminationAttendance



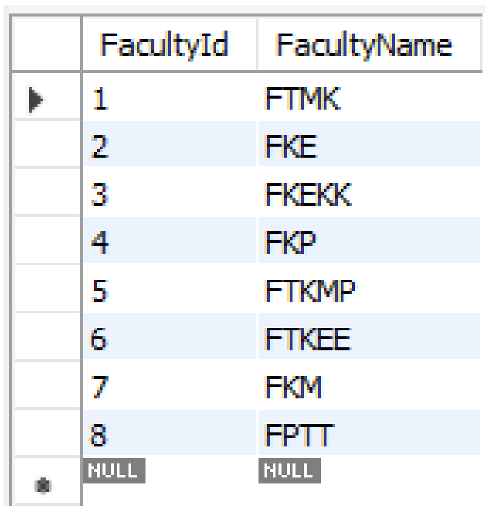
ExamAttendId	ExamAttendStatus	InputType	ExaminationId	StudentId
5	hadir	fingerprint	1	1
6	hadir	QRCode	3	2
7	hadirrr	QRCode	2	21
8	hadirrr	Matric Card	1	3
9	hadirrr	Self Check In	2	6
10	hadirrr	QRCode	3	5
25	hadir	MatricCard	1	4
26	hadir	MatricCard	1	4
27	hadir	Fingerprint	5	1
28	hadir	MatricCard	1	4
29	late	MatricCard	1	3
30	hadir	MatricCard	3	1
31	hadir	QRCode	5	2
NULL	NULL	NULL	NULL	NULL

3. ExaminationUnit



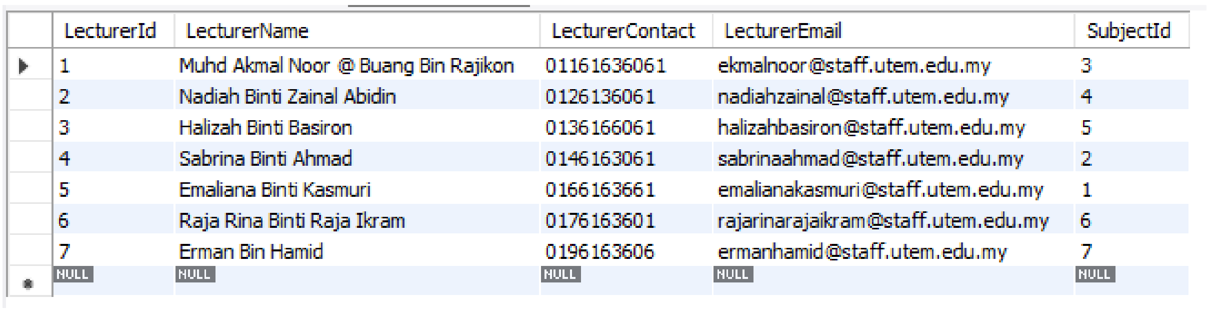
UnitId	UnitName	UnitAvailability
1	DEWAN CHANCELLOR	1
2	PUSAT SUKAN	1
3	DEWAN SEMINAR	1
4	RECAP ROOM	1
NULL	NULL	NULL

4. Faculty



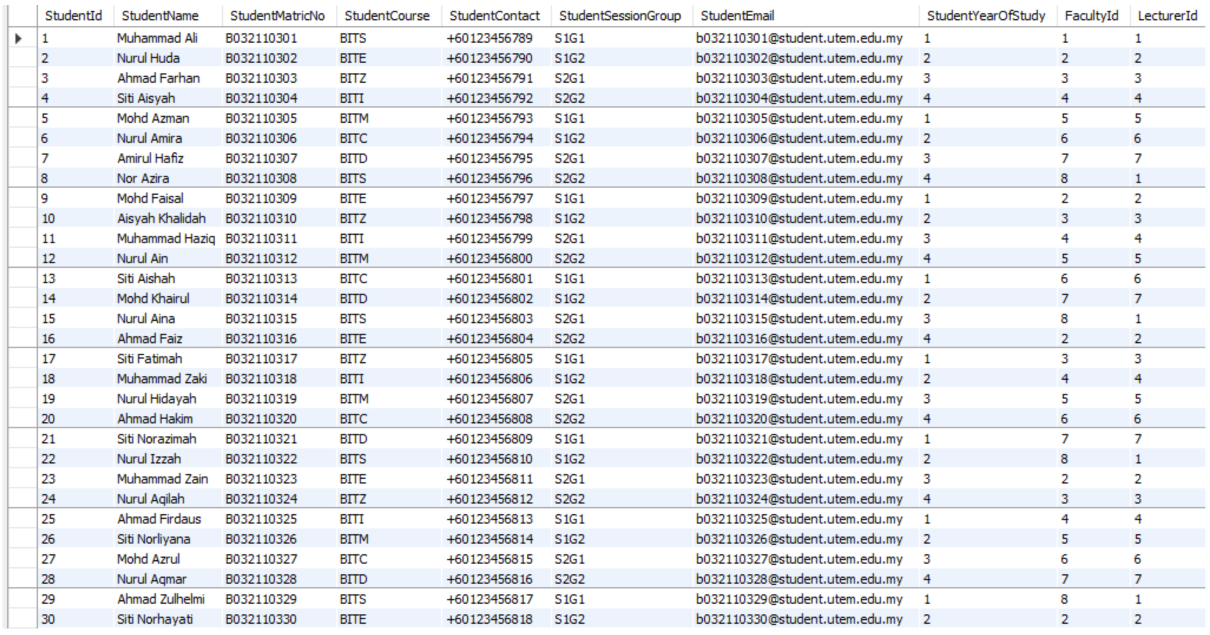
FacultyId	FacultyName
1	FTMK
2	FKE
3	FKEKK
4	FKP
5	FTKMP
6	FTKEE
7	FKM
8	FPTT
NULL	NULL

5. Lecturer



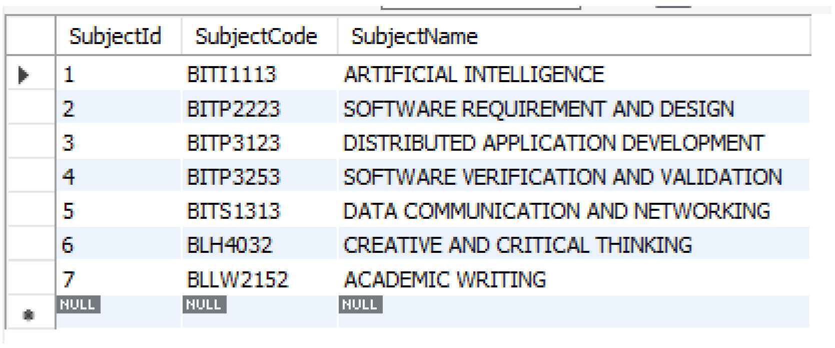
LecturerId	LecturerName	LecturerContact	LecturerEmail	SubjectId
1	Muhd Akmal Noor @ Buang Bin Rajikon	01161636061	ekmalnoor@staff.utem.edu.my	3
2	Nadiah Binti Zainal Abidin	0126136061	nadiahzainal@staff.utem.edu.my	4
3	Halizah Binti Basiron	0136166061	halizahbasiron@staff.utem.edu.my	5
4	Sabrina Binti Ahmad	0146163061	sabrinaahmad@staff.utem.edu.my	2
5	Emaliana Binti Kasmuri	0166163661	emalianakasmuri@staff.utem.edu.my	1
6	Raja Rina Binti Raja Ikram	0176163601	rajarinarajaikram@staff.utem.edu.my	6
7	Erman Bin Hamid	0196163606	ermanhamid@staff.utem.edu.my	7
NULL	NULL	NULL	NULL	NULL

6. Student



StudentId	StudentName	StudentMatricNo	StudentCourse	StudentContact	StudentSessionGroup	StudentEmail	StudentYearOfStudy	FacultyId	LecturerId
1	Muhammad Ali	B032110301	BITS	+60123456789	S1G1	b032110301@student.utem.edu.my	1	1	1
2	Nurul Huda	B032110302	BITE	+60123456790	S1G2	b032110302@student.utem.edu.my	2	2	2
3	Ahmad Farhan	B032110303	BITZ	+60123456791	S2G1	b032110303@student.utem.edu.my	3	3	3
4	Siti Aisyah	B032110304	BITI	+60123456792	S2G2	b032110304@student.utem.edu.my	4	4	4
5	Mohd Azman	B032110305	BITM	+60123456793	S1G1	b032110305@student.utem.edu.my	1	5	5
6	Nurul Amira	B032110306	BITC	+60123456794	S1G2	b032110306@student.utem.edu.my	2	6	6
7	Amirul Hafiz	B032110307	BITD	+60123456795	S2G1	b032110307@student.utem.edu.my	3	7	7
8	Nor Azira	B032110308	BITS	+60123456796	S2G2	b032110308@student.utem.edu.my	4	8	1
9	Mohd Faisal	B032110309	BITE	+60123456797	S1G1	b032110309@student.utem.edu.my	1	2	2
10	Aisyah Khalidah	B032110310	BITZ	+60123456798	S1G2	b032110310@student.utem.edu.my	2	3	3
11	Muhammad Haziq	B032110311	BITI	+60123456799	S2G1	b032110311@student.utem.edu.my	3	4	4
12	Nurul Ain	B032110312	BITM	+60123456800	S2G2	b032110312@student.utem.edu.my	4	5	5
13	Siti Aishah	B032110313	BITC	+60123456801	S1G1	b032110313@student.utem.edu.my	1	6	6
14	Mohd Khairul	B032110314	BITD	+60123456802	S1G2	b032110314@student.utem.edu.my	2	7	7
15	Nurul Aina	B032110315	BITS	+60123456803	S2G1	b032110315@student.utem.edu.my	3	8	1
16	Ahmad Faiz	B032110316	BITE	+60123456804	S2G2	b032110316@student.utem.edu.my	4	2	2
17	Siti Fatimah	B032110317	BITZ	+60123456805	S1G1	b032110317@student.utem.edu.my	1	3	3
18	Muhammad Zaki	B032110318	BITI	+60123456806	S1G2	b032110318@student.utem.edu.my	2	4	4
19	Nurul Hidayah	B032110319	BITM	+60123456807	S2G1	b032110319@student.utem.edu.my	3	5	5
20	Ahmad Hakim	B032110320	BITC	+60123456808	S2G2	b032110320@student.utem.edu.my	4	6	6
21	Siti Norazimah	B032110321	BITD	+60123456809	S1G1	b032110321@student.utem.edu.my	1	7	7
22	Nurul Izzah	B032110322	BITS	+60123456810	S1G2	b032110322@student.utem.edu.my	2	8	1
23	Muhammad Zain	B032110323	BITE	+60123456811	S2G1	b032110323@student.utem.edu.my	3	2	2
24	Nurul Aqilah	B032110324	BITZ	+60123456812	S2G2	b032110324@student.utem.edu.my	4	3	3
25	Ahmad Firdaus	B032110325	BITI	+60123456813	S1G1	b032110325@student.utem.edu.my	1	4	4
26	Siti Norliyana	B032110326	BITM	+60123456814	S1G2	b032110326@student.utem.edu.my	2	5	5
27	Mohd Azrul	B032110327	BITC	+60123456815	S2G1	b032110327@student.utem.edu.my	3	6	6
28	Nurul Aqmar	B032110328	BITD	+60123456816	S2G2	b032110328@student.utem.edu.my	4	7	7
29	Ahmad Zulhelmi	B032110329	BITS	+60123456817	S1G1	b032110329@student.utem.edu.my	1	8	1
30	Siti Norhayati	B032110330	BITE	+60123456818	S1G2	b032110330@student.utem.edu.my	2	2	2

7. Subject

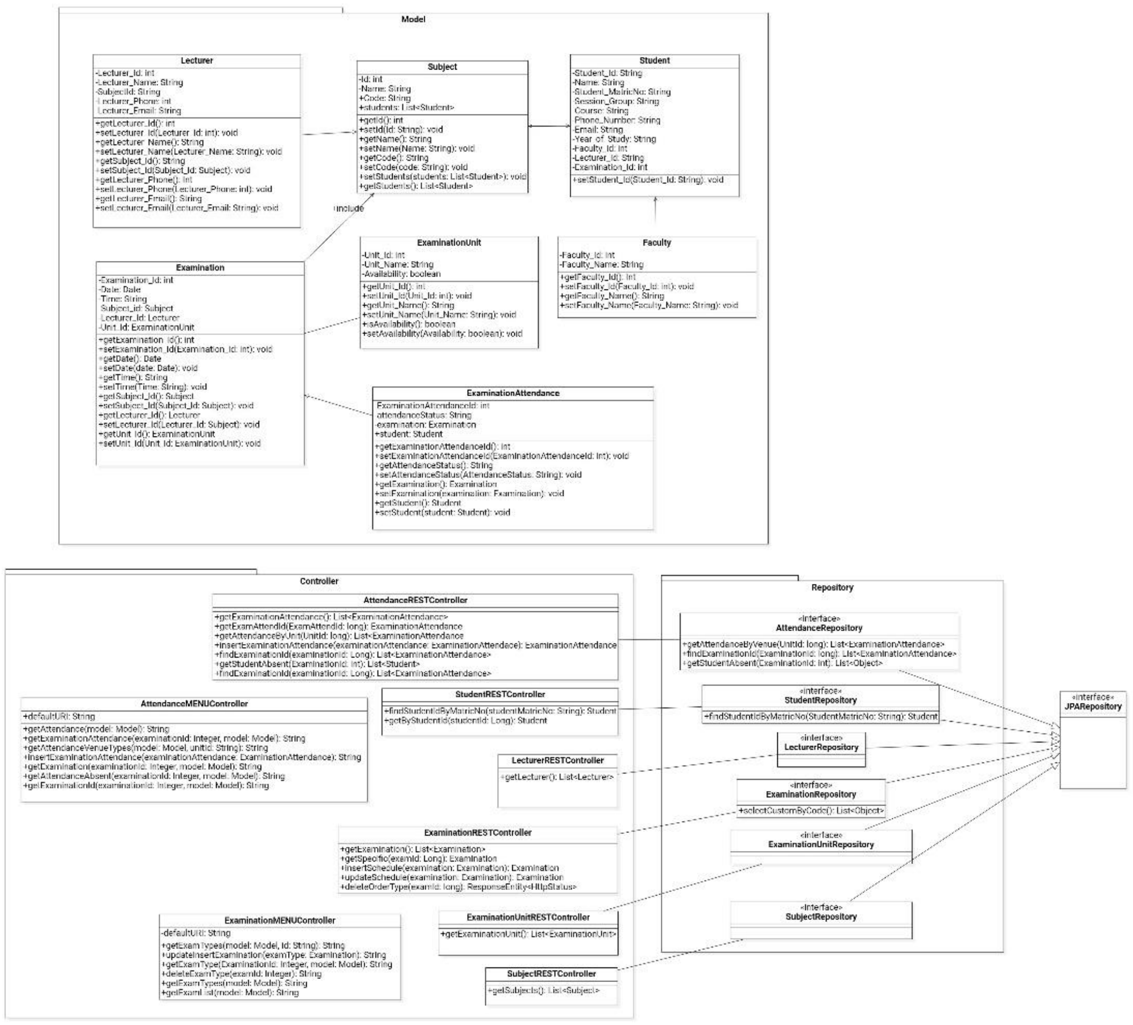


SubjectId	SubjectCode	SubjectName
1	BITI1113	ARTIFICIAL INTELLIGENCE
2	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
3	BITP3123	DISTRIBUTED APPLICATION DEVELOPMENT
4	BITP3253	SOFTWARE VERIFICATION AND VALIDATION
5	BITS1313	DATA COMMUNICATION AND NETWORKING
6	BLH4032	CREATIVE AND CRITICAL THINKING
7	BLLW2152	ACADEMIC WRITING
NULL	NULL	NULL

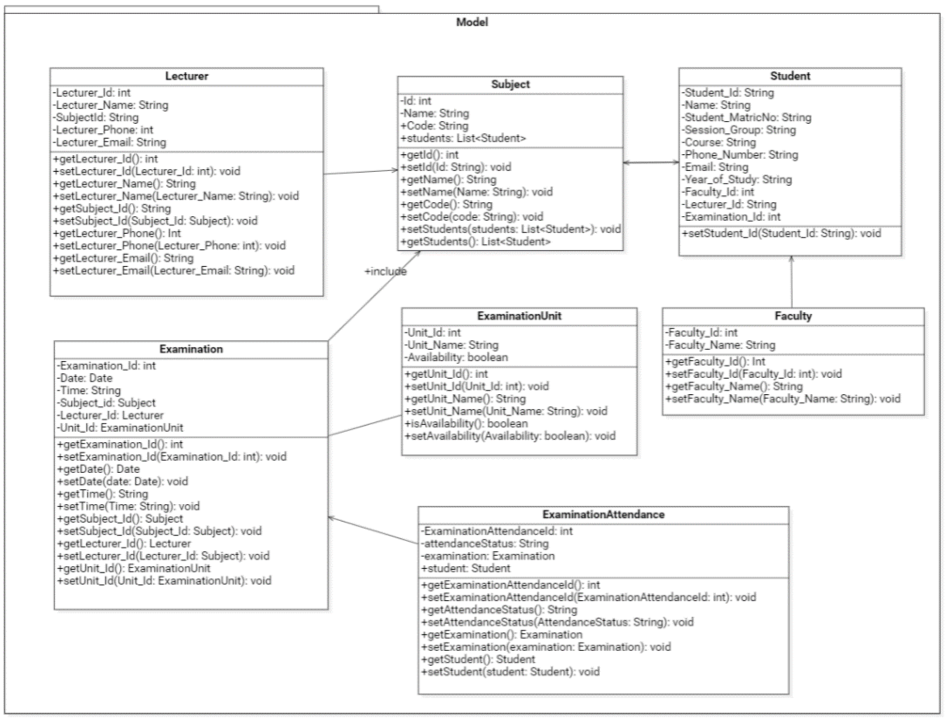
Implementation of Data Layer

Class Diagram

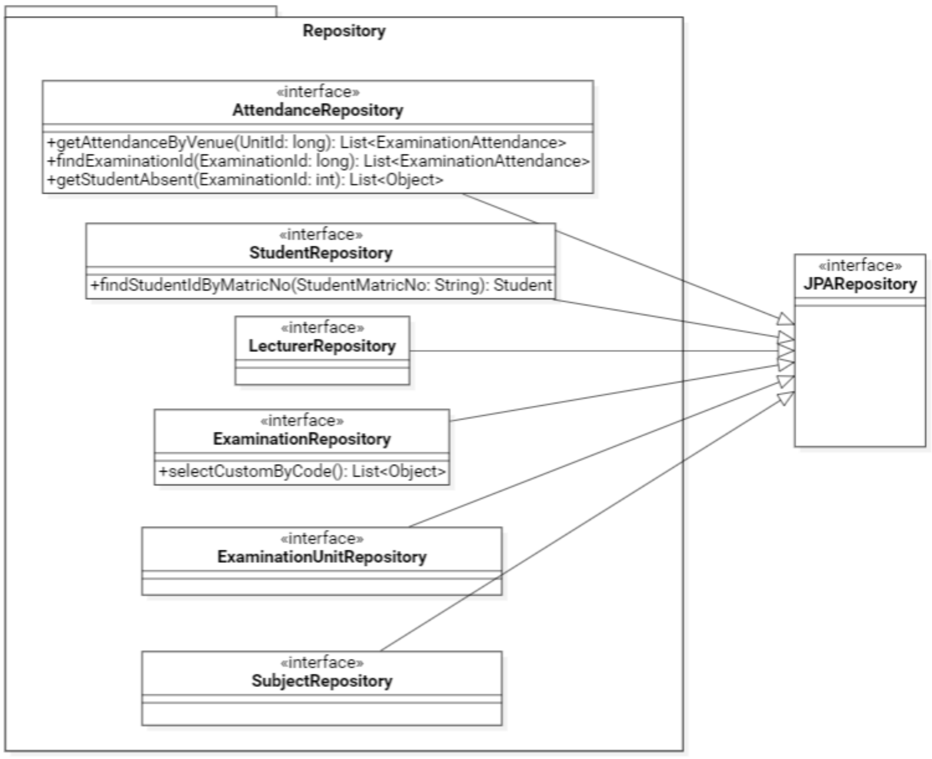
Overview Class Diagram


Model
Lecturer
-Lecturer_Id: int
-Lecturer_Name: String
-SubjectId: String
-Lecturer_Phone: int
-Lecturer_Email: String
+getLecturer_Id(): int
+setLecturer_Id(Lecturer_Id: int): void
+getLecturer_Name(): String
+setLecturer_Name(Lecturer_Name: String): void
+getSubject_Id(): String
+setSubject_Id(Subject_Id: Subject): void
+getLecturer_Phone(): int
+setLecturer_Phone(Lecturer_Phone: int): void
+getLecturer_Email(): String
+setLecturer_Email(Lecturer_Email: String): void
Subject
-Id: int
-Name: String
+Code: String
+students: List<Student>
+getId(): int
+setId(Id: String): void
+getName(): String
+setName(Name: String): void
+getCode(): String
+setCode(code: String): void
+setStudents(students: List<Student>): void
+getStudents(): List<Student>
Student
-Student_Id: String
-Name: String
-Student_MatricNo: String
-Session_Group: String
-Course: String
-Phone_Number: String
-Email: String
-Year_of_Study: String
-Faculty_Id: int
-Lecturer_Id: String
-Examination_Id: int
+setStudent_Id(Student_Id: String): void
«include»
Examination
-Examination_Id: int
-Date: Date
-Time: String
-Subject_Id: Subject
-Lecturer_Id: Lecturer
-Unit_Id: ExaminationUnit
+getExamination_Id(): int
+setExamination_Id(Examination_Id: int): void
+getDate(): Date
+setDate(date: Date): void
+getTime(): String
+setTime(Time: String): void
+getSubject_Id(): Subject
+setSubject_Id(Subject_Id: Subject): void
+getLecturer_Id(): Lecturer
+setLecturer_Id(Lecturer_Id: Subject): void
+getUnit_Id(): ExaminationUnit
+setUnit_Id(Unit_Id: ExaminationUnit): void
ExaminationUnit
-Unit_Id: int
-Unit_Name: String
-Availability: boolean
+getUnit_Id(): int
+setUnit_Id(Unit_Id: int): void
+getUnit_Name(): String
+setUnit_Name(Unit_Name: String): void
+isAvailability(): boolean
+setAvailability(Availability: boolean): void
Faculty
-Faculty_Id: int
-Faculty_Name: String
+getFaculty_Id(): int
+setFaculty_Id(Faculty_Id: int): void
+getFaculty_Name(): String
+setFaculty_Name(Faculty_Name: String): void
ExaminationAttendance
-ExaminationAttendanceId: int
-attendanceStatus: String
-examination: Examination
+student: Student
+getExaminationAttendanceId(): int
+setExaminationAttendanceId(ExaminationAttendanceId: int): void
+getAttendanceStatus(): String
+setAttendanceStatus(AttendanceStatus: String): void
+getExamination(): Examination
+setExamination(examination: Examination): void
+getStudent(): Student
+setStudent(student: Student): void
Controller
AttendanceRESTController
+getExaminationAttendance(): List<ExaminationAttendance>
+getExamAttendId(ExamAttendId: long): ExaminationAttendance
+getAttendanceByUnit(UnitId: long): List<ExaminationAttendance>
+insertExaminationAttendance(examinationAttendance: ExaminationAttendace): ExaminationAttendance
+findExaminationId(examinationId: Long): List<ExaminationAttendance>
+getStudentAbsent(ExaminationId: int): List<Student>
+findExaminationId(examinationId: Long): List<ExaminationAttendance>
AttendanceMENUController
+defaultURI: String
+getAttendance(model: Model): String
+getExaminationAttendance(examinationId: Integer, model: Model): String
+getAttendanceVenueTypes(model: Model, unitId: String): String
+insertExaminationAttendance(examinationAttendance: ExaminationAttendance): String
+getExamination(examinationId: Integer, model: Model): String
+getAttendanceAbsent(examinationId: Integer, model: Model): String
+getExaminationId(examinationId: Integer, model: Model): String
StudentRESTController
+findStudentIdByMatricNo(studentMatricNo: String): Student
+getByStudentId(studentId: Long): Student
LecturerRESTController
+getLecturer(): List<Lecturer>
ExaminationRESTController
+getExamination(): List<Examination>
+getSpecific(examId: Long): Examination
+insertSchedule(examination: Examination): Examination
+updateSchedule(examination: Examination): Examination
+deleteOrderType(examId: long): ResponseEntity<HttpStatus>
ExaminationMENUController
-defaultURI: String
+getExamTypes(model: Model, id: String): String
+updateInsertExamination(examType: Examination): String
+getExamType(ExaminationId: Integer, model: Model): String
+deleteExamType(examId: Integer): String
+getExamTypes(model: Model): String
+getExamList(model: Model): String
ExaminationUnitRESTController
+getExaminationUnit(): List<ExaminationUnit>
SubjectRESTController
+getSubjects(): List<Subject>
Repository
«interface»
AttendanceRepository
+getAttendanceByVenue(UnitId: long): List<ExaminationAttendance>
+findExaminationId(ExaminationId: long): List<ExaminationAttendance>
+getStudentAbsent(ExaminationId: int): List<Object>
«interface»
StudentRepository
+findStudentIdByMatricNo(StudentMatricNo: String): Student
«interface»
LecturerRepository
«interface»
ExaminationRepository
+selectCustomByCode(): List<Object>
«interface»
ExaminationUnitRepository
«interface»
SubjectRepository
«interface»
JPARepository

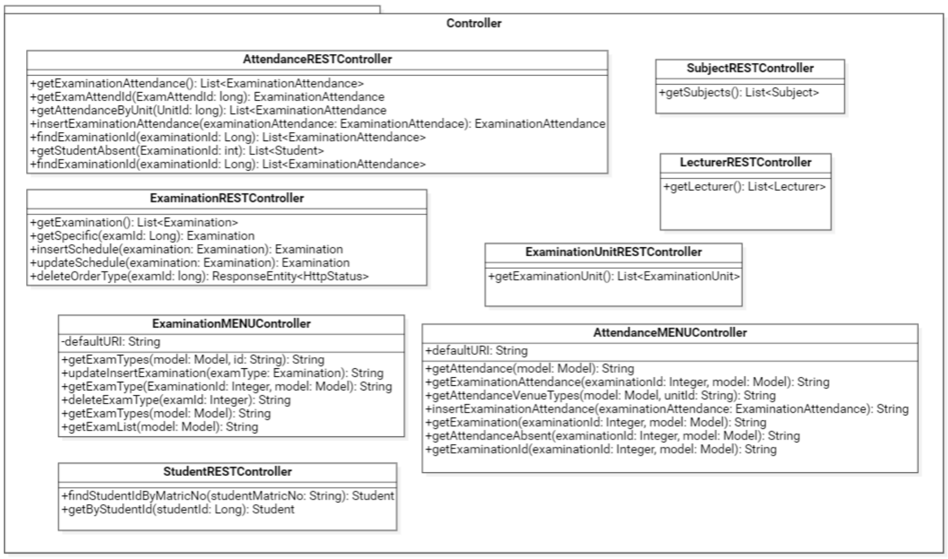
Package: Model


Model
Lecturer
-Lecturer_Id: int
-Lecturer_Name: String
-SubjectId: String
-Lecturer_Phone: int
-Lecturer_Email: String
+getLecturer_Id(): int
+setLecturer_Id(Lecturer_Id: int): void
+getLecturer_Name(): String
+setLecturer_Name(Lecturer_Name: String): void
+getSubject_Id(): String
+setSubject_Id(Subject_Id: Subject): void
+getLecturer_Phone(): Int
+setLecturer_Phone(Lecturer_Phone: int): void
+getLecturer_Email(): String
+setLecturer_Email(Lecturer_Email: String): void
Subject
-Id: int
-Name: String
+Code: String
+students: List<Student>
+getId(): int
+setId(Id: String): void
+getName(): String
+setName(Name: String): void
+getCode(): String
+setCode(code: String): void
+setStudents(students: List<Student>): void
+getStudents(): List<Student>
Student
-Student_Id: String
-Name: String
-Student_MatricNo: String
-Session_Group: String
-Course: String
-Phone_Number: String
-Email: String
-Year_of_Study: String
-Faculty_Id: int
-Lecturer_Id: String
-Examination_Id: int
+setStudent_Id(Student_Id: String): void
+include
Examination
-Examination_Id: int
-Date: Date
-Time: String
-Subject_id: Subject
-Lecturer_Id: Lecturer
-Unit_Id: ExaminationUnit
+getExamination_Id(): int
+setExamination_Id(Examination_Id: int): void
+getDate(): Date
+setDate(date: Date): void
+getTime(): String
+setTime(Time: String): void
+getSubject_Id(): Subject
+setSubject_Id(Subject_Id: Subject): void
+getLecturer_Id(): Lecturer
+setLecturer_Id(Lecturer_Id: Subject): void
+getUnit_Id(): ExaminationUnit
+setUnit_Id(Unit_Id: ExaminationUnit): void
ExaminationUnit
-Unit_Id: int
-Unit_Name: String
-Availability: boolean
+getUnit_Id(): int
+setUnit_Id(Unit_Id: int): void
+getUnit_Name(): String
+setUnit_Name(Unit_Name: String): void
+isAvailability(): boolean
+setAvailability(Availability: boolean): void
Faculty
-Faculty_Id: int
-Faculty_Name: String
+getFaculty_Id(): Int
+setFaculty_Id(Faculty_Id: int): void
+getFaculty_Name(): String
+setFaculty_Name(Faculty_Name: String): void
ExaminationAttendance
-ExaminationAttendanceId: int
-attendanceStatus: String
-examination: Examination
+student: Student
+getExaminationAttendanceId(): int
+setExaminationAttendanceId(ExaminationAttendanceId: int): void
+getAttendanceStatus(): String
+setAttendanceStatus(AttendanceStatus: String): void
+getExamination(): Examination
+setExamination(examination: Examination): void
+getStudent(): Student
+setStudent(student: Student): void

Package: Repository


Repository
«interface»
AttendanceRepository
+getAttendanceByVenue(UnitId: long): List<ExaminationAttendance>
+findExaminationId(ExaminationId: long): List<ExaminationAttendance>
+getStudentAbsent(ExaminationId: int): List<Object>
«interface»
StudentRepository
+findStudentIdByMatricNo(StudentMatricNo: String): Student
«interface»
LecturerRepository
«interface»
ExaminationRepository
+selectCustomByCode(): List<Object>
«interface»
ExaminationUnitRepository
«interface»
SubjectRepository
«interface»
JPARepository

Package: Controller


Controller
AttendanceRESTController
+getExaminationAttendance(): List<ExaminationAttendance>
+getExamAttendId(ExamAttendId: long): ExaminationAttendance
+getAttendanceByUnit(UnitId: long): List<ExaminationAttendance
+insertExaminationAttendance(examinationAttendance: ExaminationAttendace): ExaminationAttendance
+findExaminationId(examinationId: Long): List<ExaminationAttendance>
+getStudentAbsent(ExaminationId: int): List<Student>
+findExaminationId(examinationId: Long): List<ExaminationAttendance>
SubjectRESTController
+getSubjects(): List<Subject>
LecturerRESTController
+getLecturer(): List<Lecturer>
ExaminationRESTController
+getExamination(): List<Examination>
+getSpecific(examId: Long): Examination
+insertSchedule(examination: Examination): Examination
+updateSchedule(examination: Examination): Examination
+deleteOrderType(examId: long): ResponseEntity<HttpStatus>
ExaminationUnitRESTController
+getExaminationUnit(): List<ExaminationUnit>
ExaminationMENUController
-defaultURI: String
+getExamTypes(model: Model, id: String): String
+updateInsertExamination(examType: Examination): String
+getExamType(ExaminationId: Integer, model: Model): String
+deleteExamType(examId: Integer): String
+getExamTypes(model: Model): String
+getExamList(model: Model): String
AttendanceMENUController
+defaultURI: String
+getAttendance(model: Model): String
+getExaminationAttendance(examinationId: Integer, model: Model): String
+getAttendanceVenueTypes(model: Model, unitId: String): String
+insertExaminationAttendance(examinationAttendance: ExaminationAttendance): String
+getExamination(examinationId: Integer, model: Model): String
+getAttendanceAbsent(examinationId: Integer, model: Model): String
+getExaminationId(examinationId: Integer, model: Model): String
StudentRESTController
+findStudentIdByMatricNo(studentMatricNo: String): Student
+getByStudentId(studentId: Long): Student

Entity Mapping to the Table

Class	Attributes	Repository	Annotation	Relationship
Lecturer	Id, name, contact, email, subject id.	LecturerRepository	@Entity @Table @Column	
Subject	Id, code, name	SubjectRepository	@Entity @Table @Column	
Examination Unit	Id, name, availability	ExaminationUnitRepository	@Entity @Table @Column	

Faculty	Id, name	FacultyRepository	@Entity @Table @Column	A faculty may have many students, and a student can only be in a faculty.
Student	Id, name, matric no, course, contact, session group, email, year of study, faculty id, lecturer id.	StudentRepository	@Entity @Table @Column	A student can only be in a faculty, and a faculty may have many students. A student can only have one academic advisor, and an academic advisor may advise many students.

Examination	Id, date, time, subject id, lecturer id, unit id	ExaminationRepository	@Entity @Table @Column	An examination unit may have one examination, and an examination can be located at an examination unit. A subject may include in an examination, and an examination can include only one subject. A lecturer may invigilate an examination, and an examination can be invigilated by a lecturer.
Examination Attendance	Id, status, input type, examination id, student id	ExaminationAttendanceRepository	@Entity @Table @Column	An attendance may only have an examination, and an examination can be in many attendances. An attendance can only have a student, and a student can only

				include in an attendance.

Implementation of Web Services

The web service of this Examination Attendance System is basically divided into several classes including web methods where JSON data will be returned.

<u>First Class: AttendanceRESTController</u>

This class is mainly used to handle the student attendance based on Insert, Read and Update operation. In order to do so, an @autowired object from AttendanceRepository named attendanceRepository will be created.

Default API: http://localhost:8080/examinationattendancesystem/api/attend

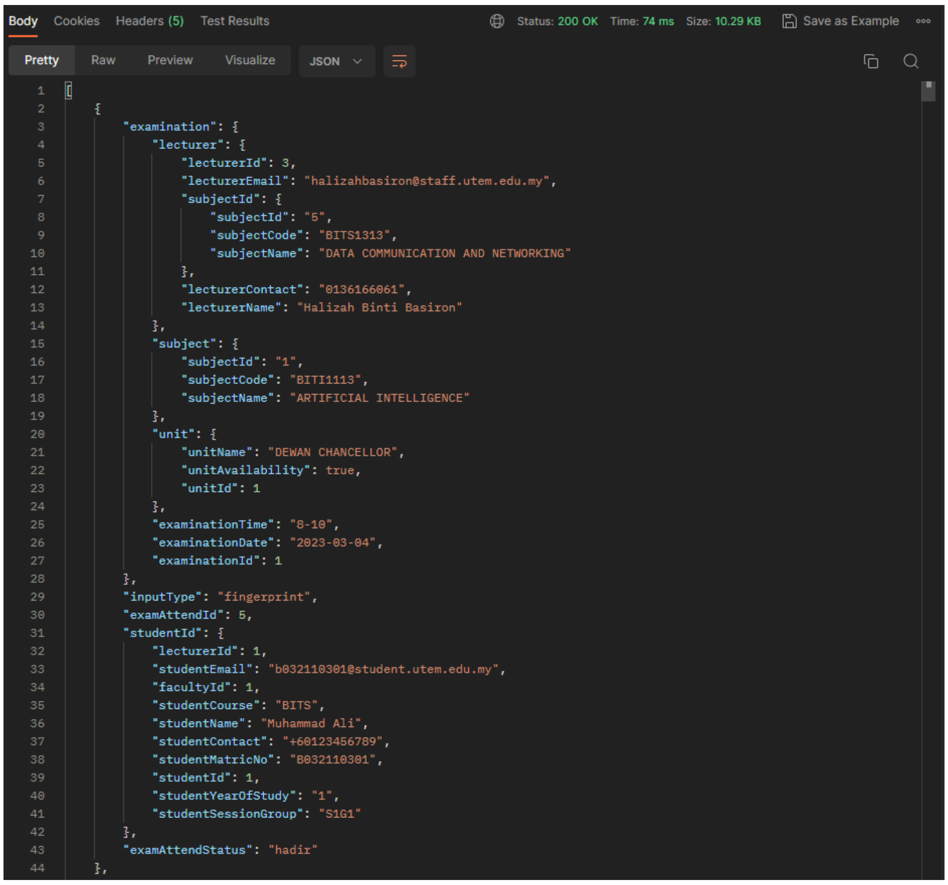
The following shows the web methods implemented:

1. getExaminationAttendance()

   This web method uses "**GET**" mapping method to retrieve the list of all ExaminationAttendance entities.

   Web API: http://localhost:8080/examinationattendancesystem/api/attend

   Example of JSON data retrieved from Postman:



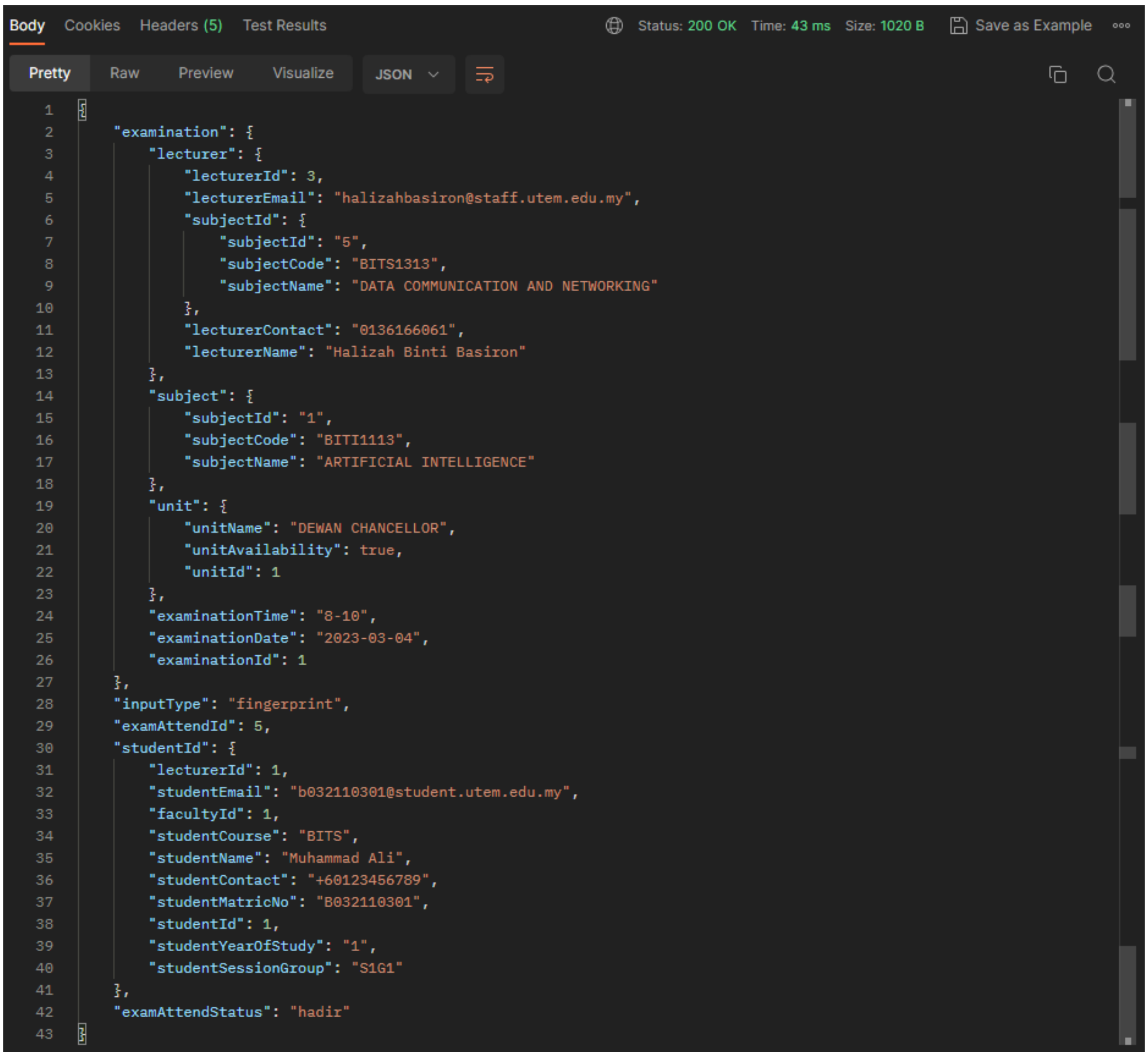
```
[
    {
        "examination": {
            "lecturer": {
                "lecturerId": 3,
                "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
                "subjectId": {
                    "subjectId": "5",
                    "subjectCode": "BITS1313",
                    "subjectName": "DATA COMMUNICATION AND NETWORKING"
                },
                "lecturerContact": "0136166061",
                "lecturerName": "Halizah Binti Basiron"
            },
            "subject": {
                "subjectId": "1",
                "subjectCode": "BITI1113",
                "subjectName": "ARTIFICIAL INTELLIGENCE"
            },
            "unit": {
                "unitName": "DEWAN CHANCELLOR",
                "unitAvailability": true,
                "unitId": 1
            },
            "examinationTime": "8-10",
            "examinationDate": "2023-03-04",
            "examinationId": 1
        },
        "inputType": "fingerprint",
        "examAttendId": 5,
        "studentId": {
            "lecturerId": 1,
            "studentEmail": "b032110301@student.utem.edu.my",
            "facultyId": 1,
            "studentCourse": "BITS",
            "studentName": "Muhammad Ali",
            "studentContact": "+60123456789",
            "studentMatricNo": "B032110301",
            "studentId": 1,
            "studentYearOfStudy": "1",
            "studentSessionGroup": "S1G1"
        },
        "examAttendStatus": "hadir"
    },
```

2. getExamAttendId(@PathVariable long ExamAttendId)

   This method uses "**GET**" mapping method to retrieve the ExaminationAttendance entity with the specified examinationAttendanceId.

   Web API: http://localhost:8080/examinationattendancesystem/api/attend/{ExamAttendId}

   Example of JSON data retrieved from Postman based on ExamAttendId = 5:



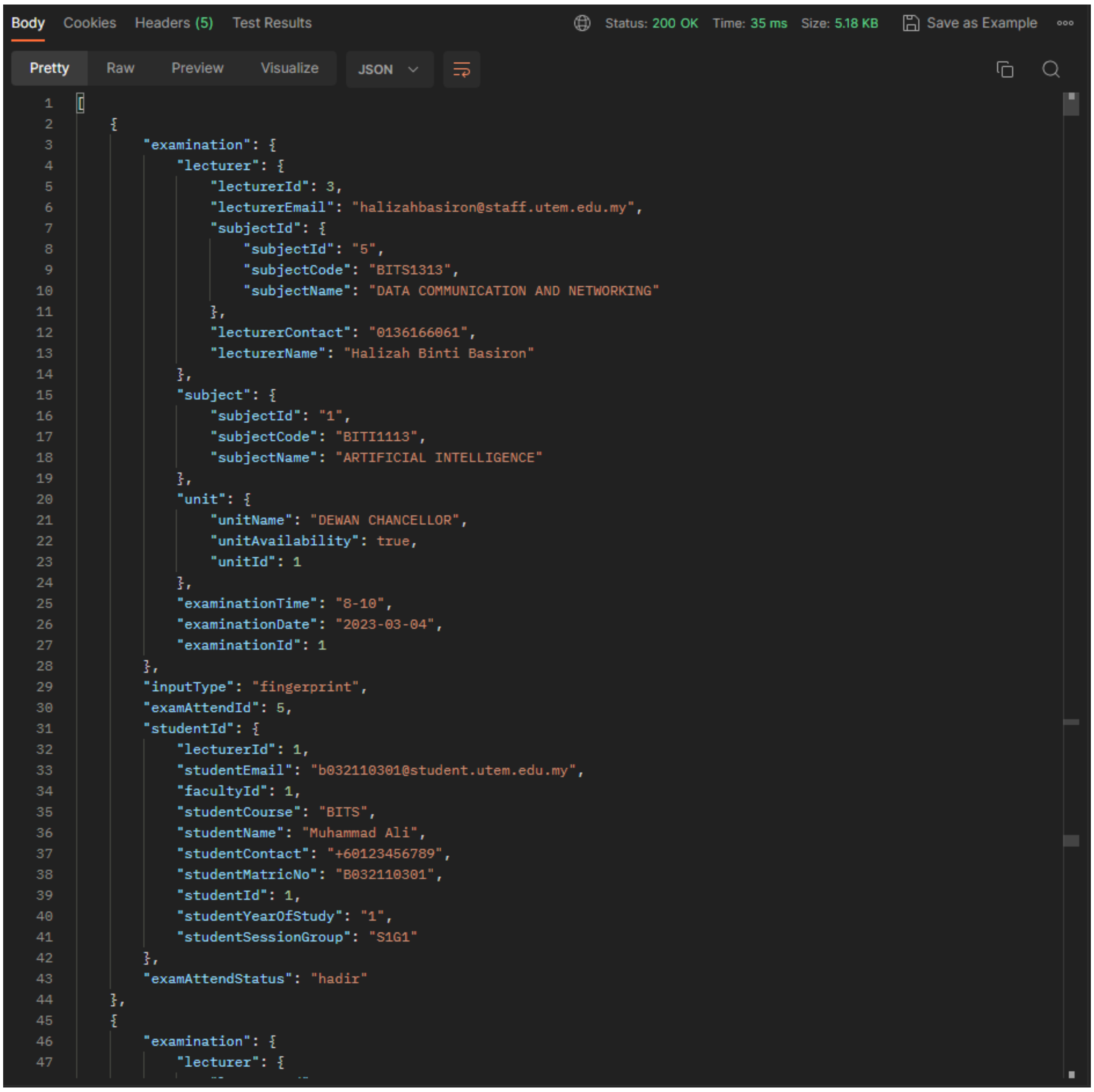
```
{
    "examination": {
        "lecturer": {
            "lecturerId": 3,
            "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
            "subjectId": {
                "subjectId": "5",
                "subjectCode": "BITS1313",
                "subjectName": "DATA COMMUNICATION AND NETWORKING"
            },
            "lecturerContact": "0136166061",
            "lecturerName": "Halizah Binti Basiron"
        },
        "subject": {
            "subjectId": "1",
            "subjectCode": "BITI1113",
            "subjectName": "ARTIFICIAL INTELLIGENCE"
        },
        "unit": {
            "unitName": "DEWAN CHANCELLOR",
            "unitAvailability": true,
            "unitId": 1
        },
        "examinationTime": "8-10",
        "examinationDate": "2023-03-04",
        "examinationId": 1
    },
    "inputType": "fingerprint",
    "examAttendId": 5,
    "studentId": {
        "lecturerId": 1,
        "studentEmail": "b032110301@student.utem.edu.my",
        "facultyId": 1,
        "studentCourse": "BITS",
        "studentName": "Muhammad Ali",
        "studentContact": "+60123456789",
        "studentMatricNo": "B032110301",
        "studentId": 1,
        "studentYearOfStudy": "1",
        "studentSessionGroup": "S1G1"
    },
    "examAttendStatus": "hadir"
}
```

3. getAttendanceByUnit (@PathVariable long UnitId)

   This method will use "**GET**" mapping method to retrieve retrieves the list of ExaminationAttendance entities associated with a specific unit (venue) based on UnitId.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/attend/Venue/{UnitId}

   Example of JSON data retrieved from Postman using UnitId = 1:



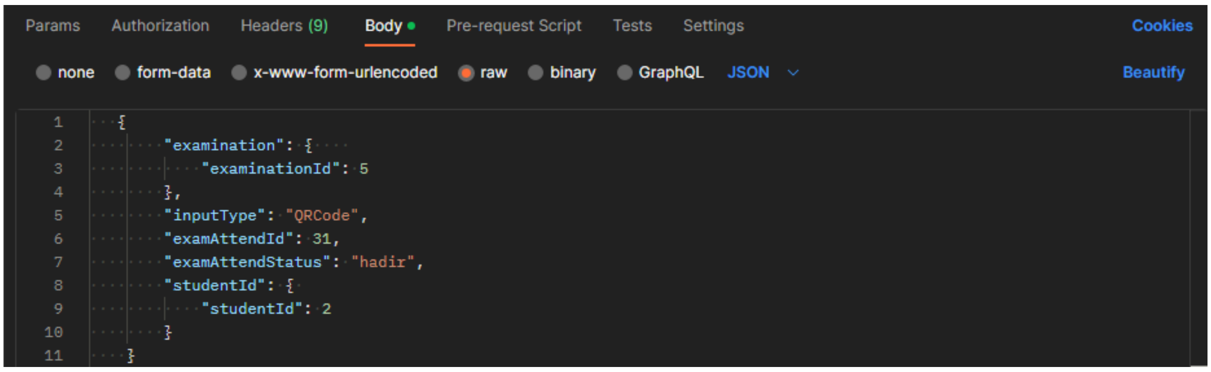
```
[
    {
        "examination": {
            "lecturer": {
                "lecturerId": 3,
                "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
                "subjectId": {
                    "subjectId": "5",
                    "subjectCode": "BITS1313",
                    "subjectName": "DATA COMMUNICATION AND NETWORKING"
                },
                "lecturerContact": "0136166061",
                "lecturerName": "Halizah Binti Basiron"
            },
            "subject": {
                "subjectId": "1",
                "subjectCode": "BITI1113",
                "subjectName": "ARTIFICIAL INTELLIGENCE"
            },
            "unit": {
                "unitName": "DEWAN CHANCELLOR",
                "unitAvailability": true,
                "unitId": 1
            },
            "examinationTime": "8-10",
            "examinationDate": "2023-03-04",
            "examinationId": 1
        },
        "inputType": "fingerprint",
        "examAttendId": 5,
        "studentId": {
            "lecturerId": 1,
            "studentEmail": "b032110301@student.utem.edu.my",
            "facultyId": 1,
            "studentCourse": "BITS",
            "studentName": "Muhammad Ali",
            "studentContact": "+60123456789",
            "studentMatricNo": "B032110301",
            "studentId": 1,
            "studentYearOfStudy": "1",
            "studentSessionGroup": "S1G1"
        },
        "examAttendStatus": "hadir"
    },
    {
        "examination": {
            "lecturer": {
```

4. insertExaminationAttendance(@RequestBody ExaminationAttendance examinationAttendance)

   This method use "**POST**" mapping method to insert a new ExaminationAttendance entity into the database.

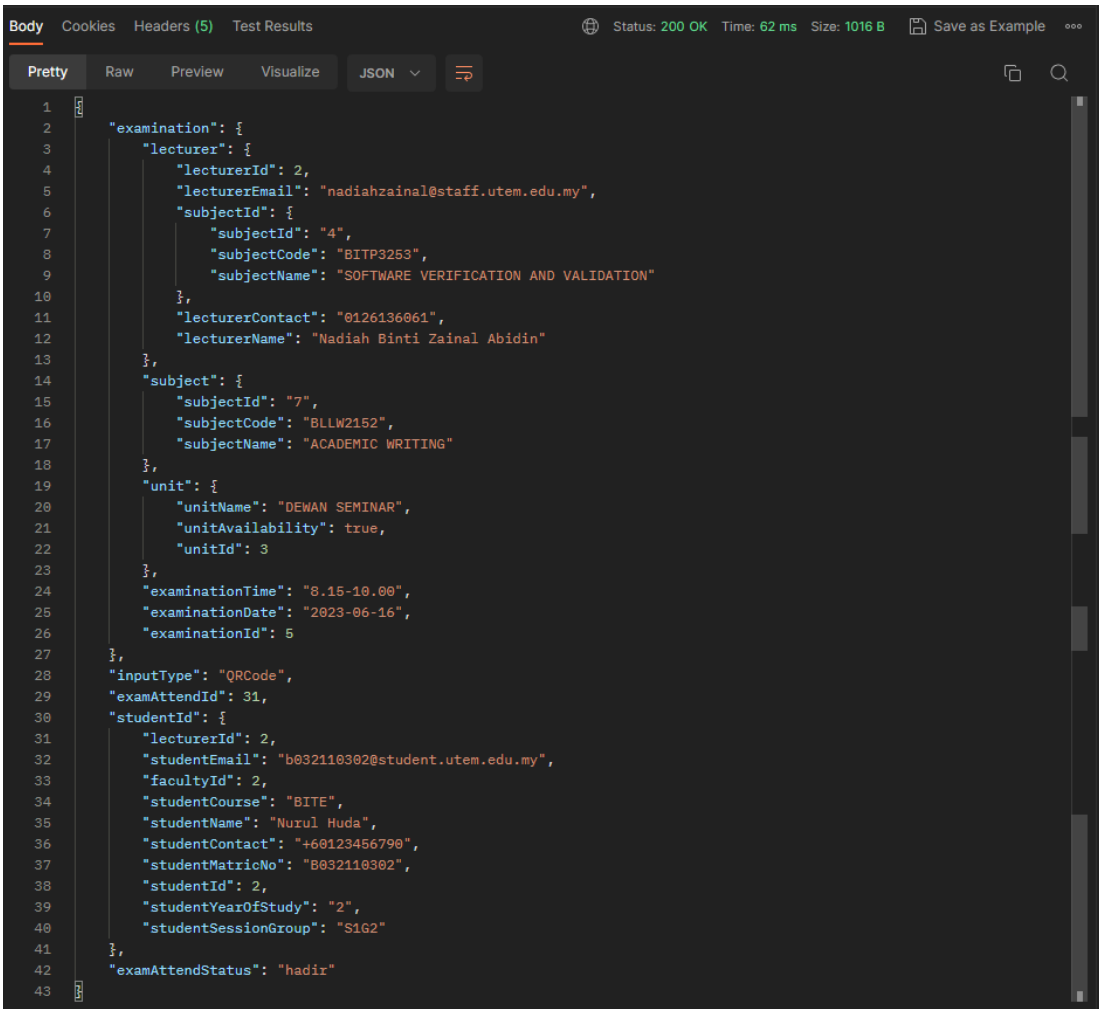
   Web API: http://localhost:8080/examinationattendancesystem/api/attend

   Example of new data to be inserted at Postman in JSON format:



```
{
    "examination": {
        "examinationId": 5
    },
    "inputType": "QRCode",
    "examAttendId": 31,
    "examAttendStatus": "hadir",
    "studentId": {
        "studentId": 2
    }
}
```

   Example of JSON data returned from Postman after insert data successfully:



```
{
    "examination": {
        "lecturer": {
            "lecturerId": 2,
            "lecturerEmail": "nadiahzainal@staff.utem.edu.my",
            "subjectId": {
                "subjectId": "4",
                "subjectCode": "BITP3253",
                "subjectName": "SOFTWARE VERIFICATION AND VALIDATION"
            },
            "lecturerContact": "0126136061",
            "lecturerName": "Nadiah Binti Zainal Abidin"
        },
        "subject": {
            "subjectId": "7",
            "subjectCode": "BLLW2152",
            "subjectName": "ACADEMIC WRITING"
        },
        "unit": {
            "unitName": "DEWAN SEMINAR",
            "unitAvailability": true,
            "unitId": 3
        },
        "examinationTime": "8.15-10.00",
        "examinationDate": "2023-06-16",
        "examinationId": 5
    },
    "inputType": "QRCode",
    "examAttendId": 31,
    "studentId": {
        "lecturerId": 2,
        "studentEmail": "b032110302@student.utem.edu.my",
        "facultyId": 2,
        "studentCourse": "BITE",
        "studentName": "Nurul Huda",
        "studentContact": "+60123456790",
        "studentMatricNo": "B032110302",
        "studentId": 2,
        "studentYearOfStudy": "2",
        "studentSessionGroup": "S1G2"
    },
    "examAttendStatus": "hadir"
}
```

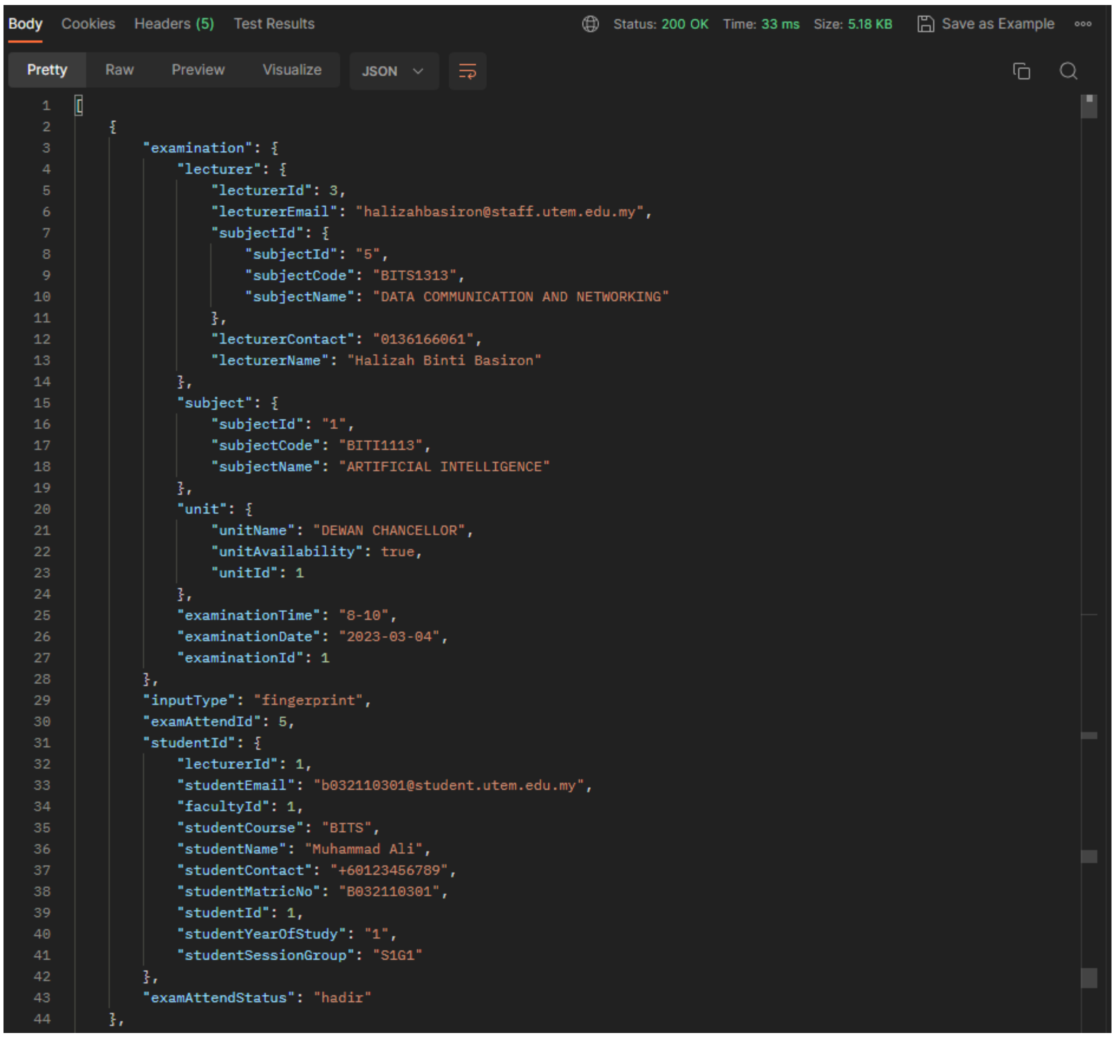
5. findExaminationId(@PathVariable Long examinationId)

   This method will use "**GET**" mapping method to retrieve the list of ExaminationAttendance entities for a specific examination based on examinationId.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/attend/report/{examinationId}

   Custom Query used in attendanceRepository:
   SELECT * from examinationAttendance WHERE ExaminationId = :ExaminationId

   Example of JSON data retrieved from Postman using examinationId = 1:



```json
[
    {
        "examination": {
            "lecturer": {
                "lecturerId": 3,
                "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
                "subjectId": {
                    "subjectId": "5",
                    "subjectCode": "BITS1313",
                    "subjectName": "DATA COMMUNICATION AND NETWORKING"
                },
                "lecturerContact": "0136166061",
                "lecturerName": "Halizah Binti Basiron"
            },
            "subject": {
                "subjectId": "1",
                "subjectCode": "BITI1113",
                "subjectName": "ARTIFICIAL INTELLIGENCE"
            },
            "unit": {
                "unitName": "DEWAN CHANCELLOR",
                "unitAvailability": true,
                "unitId": 1
            },
            "examinationTime": "8-10",
            "examinationDate": "2023-03-04",
            "examinationId": 1
        },
        "inputType": "fingerprint",
        "examAttendId": 5,
        "studentId": {
            "lecturerId": 1,
            "studentEmail": "b032110301@student.utem.edu.my",
            "facultyId": 1,
            "studentCourse": "BITS",
            "studentName": "Muhammad Ali",
            "studentContact": "+60123456789",
            "studentMatricNo": "B032110301",
            "studentId": 1,
            "studentYearOfStudy": "1",
            "studentSessionGroup": "S1G1"
        },
        "examAttendStatus": "hadir"
    },
```

6. getStudentsWithNullAttendStatusAndExaminationId(@PathVariable int ExaminationId)

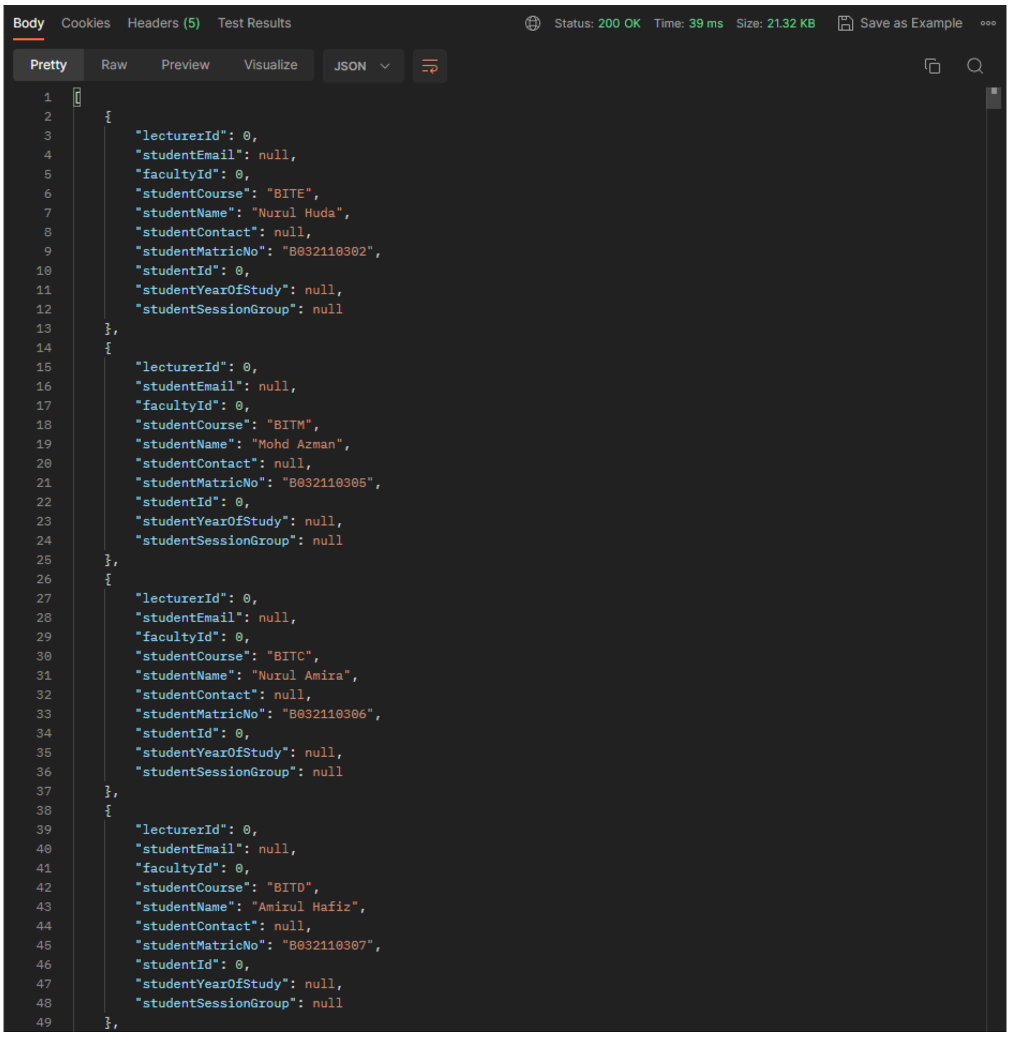
   This method will use "**GET**" mapping method to retrieve the list of students who have a null attendance status or absent for a specific examination.

   Web API: http://localhost:8080/examinationattendancesystem/api/attend/students/absent/{ExaminationId}

   Custom Query Used in attendanceRepository:

   SELECT * FROM Student s LEFT JOIN ExaminationAttendance e ON s.StudentId = e.StudentId AND e.ExaminationId = :ExaminationId WHERE e.ExamAttendStatus IS NULL OR e.ExamAttendStatus = ' '

   Example of JSON data retrieved from Postman using ExaminationId = 1:



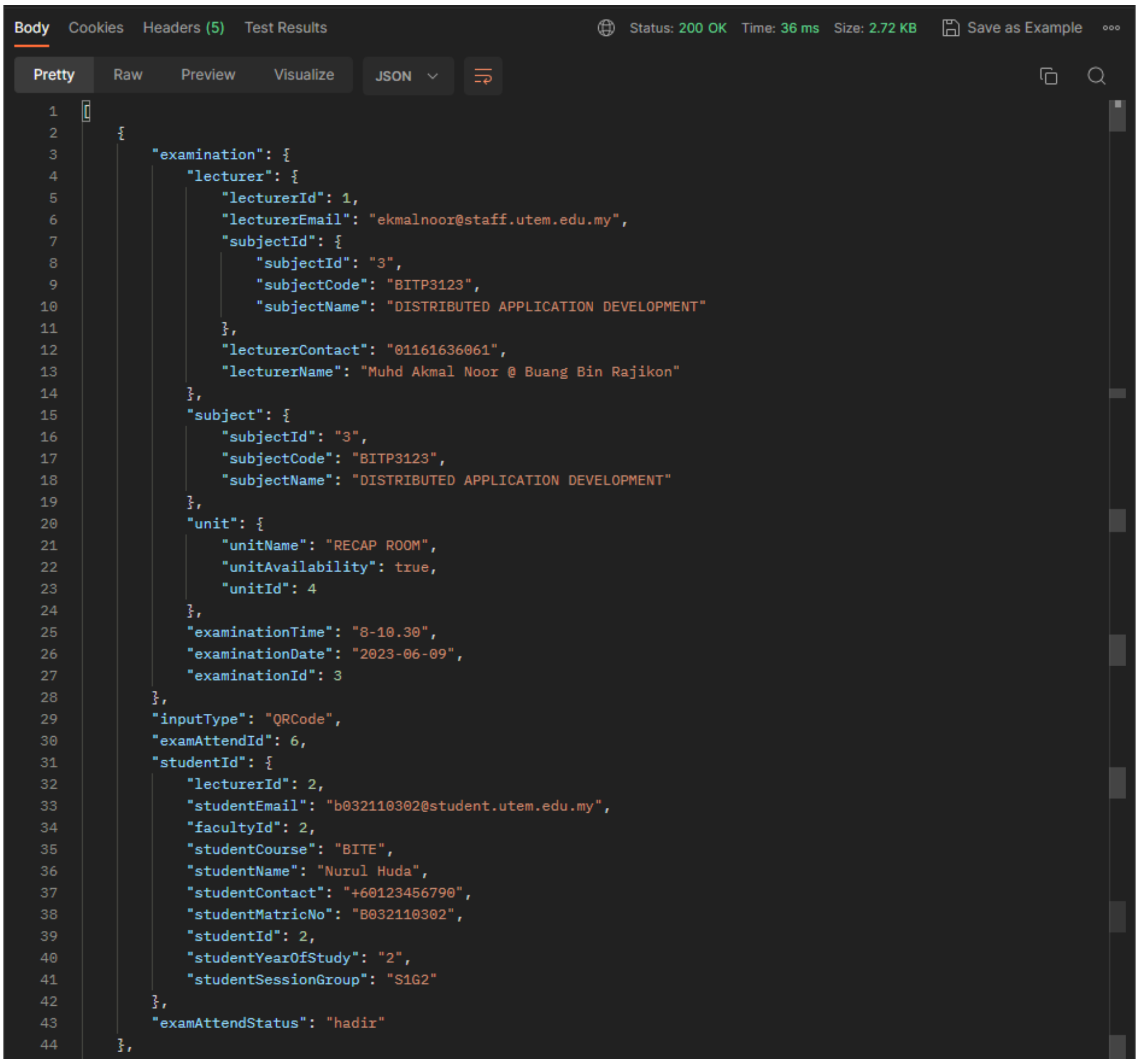
```
[
    {
        "lecturerId": 0,
        "studentEmail": null,
        "facultyId": 0,
        "studentCourse": "BITE",
        "studentName": "Nurul Huda",
        "studentContact": null,
        "studentMatricNo": "B032110302",
        "studentId": 0,
        "studentYearOfStudy": null,
        "studentSessionGroup": null
    },
    {
        "lecturerId": 0,
        "studentEmail": null,
        "facultyId": 0,
        "studentCourse": "BITM",
        "studentName": "Mohd Azman",
        "studentContact": null,
        "studentMatricNo": "B032110305",
        "studentId": 0,
        "studentYearOfStudy": null,
        "studentSessionGroup": null
    },
    {
        "lecturerId": 0,
        "studentEmail": null,
        "facultyId": 0,
        "studentCourse": "BITC",
        "studentName": "Nurul Amira",
        "studentContact": null,
        "studentMatricNo": "B032110306",
        "studentId": 0,
        "studentYearOfStudy": null,
        "studentSessionGroup": null
    },
    {
        "lecturerId": 0,
        "studentEmail": null,
        "facultyId": 0,
        "studentCourse": "BITD",
        "studentName": "Amirul Hafiz",
        "studentContact": null,
        "studentMatricNo": "B032110307",
        "studentId": 0,
        "studentYearOfStudy": null,
        "studentSessionGroup": null
    },
```

7. findExaminationI(@PathVariable Long examinationId)

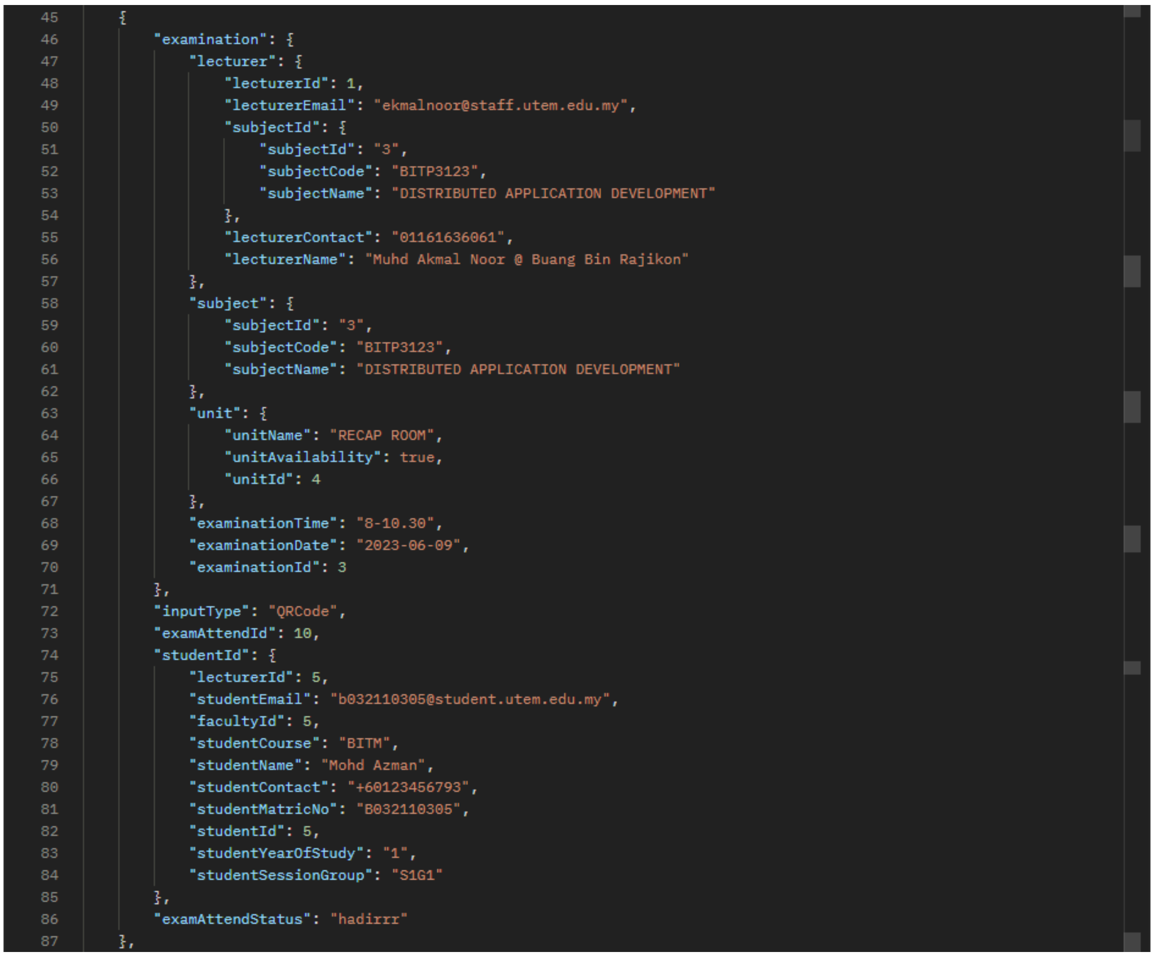
   This method uses "**GET**" mapping method to retrieve the list of ExaminationAttendance entities for a specific examination.

   Web API: http://localhost:8080/examinationattendancesystem/api/attend/examination/{examinationId}

   Example of JSON data retrieved from Postman given examinationId = 3:



```
[
    {
        "examination": {
            "lecturer": {
                "lecturerId": 1,
                "lecturerEmail": "ekmalnoor@staff.utem.edu.my",
                "subjectId": {
                    "subjectId": "3",
                    "subjectCode": "BITP3123",
                    "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
                },
                "lecturerContact": "01161636061",
                "lecturerName": "Muhd Akmal Noor @ Buang Bin Rajikon"
            },
            "subject": {
                "subjectId": "3",
                "subjectCode": "BITP3123",
                "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
            },
            "unit": {
                "unitName": "RECAP ROOM",
                "unitAvailability": true,
                "unitId": 4
            },
            "examinationTime": "8-10.30",
            "examinationDate": "2023-06-09",
            "examinationId": 3
        },
        "inputType": "QRCode",
        "examAttendId": 6,
        "studentId": {
            "lecturerId": 2,
            "studentEmail": "b032110302@student.utem.edu.my",
            "facultyId": 2,
            "studentCourse": "BITE",
            "studentName": "Nurul Huda",
            "studentContact": "+60123456790",
            "studentMatricNo": "B032110302",
            "studentId": 2,
            "studentYearOfStudy": "2",
            "studentSessionGroup": "S1G2"
        },
        "examAttendStatus": "hadir"
    },
```



```
    {
        "examination": {
            "lecturer": {
                "lecturerId": 1,
                "lecturerEmail": "ekmalnoor@staff.utem.edu.my",
                "subjectId": {
                    "subjectId": "3",
                    "subjectCode": "BITP3123",
                    "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
                },
                "lecturerContact": "01161636061",
                "lecturerName": "Muhd Akmal Noor @ Buang Bin Rajikon"
            },
            "subject": {
                "subjectId": "3",
                "subjectCode": "BITP3123",
                "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
            },
            "unit": {
                "unitName": "RECAP ROOM",
                "unitAvailability": true,
                "unitId": 4
            },
            "examinationTime": "8-10.30",
            "examinationDate": "2023-06-09",
            "examinationId": 3
        },
        "inputType": "QRCode",
        "examAttendId": 10,
        "studentId": {
            "lecturerId": 5,
            "studentEmail": "b032110305@student.utem.edu.my",
            "facultyId": 5,
            "studentCourse": "BITM",
            "studentName": "Mohd Azman",
            "studentContact": "+60123456793",
            "studentMatricNo": "B032110305",
            "studentId": 5,
            "studentYearOfStudy": "1",
            "studentSessionGroup": "S1G1"
        },
        "examAttendStatus": "hadirrr"
    },
```

<u>**Second Class: ExaminationRESTController**</u>

This class is mainly used to handle the examination schedule based on Insert, Read, Update and Delete operation. In order to do so, an @autowired object from ExaminationRepository named exams will be created.

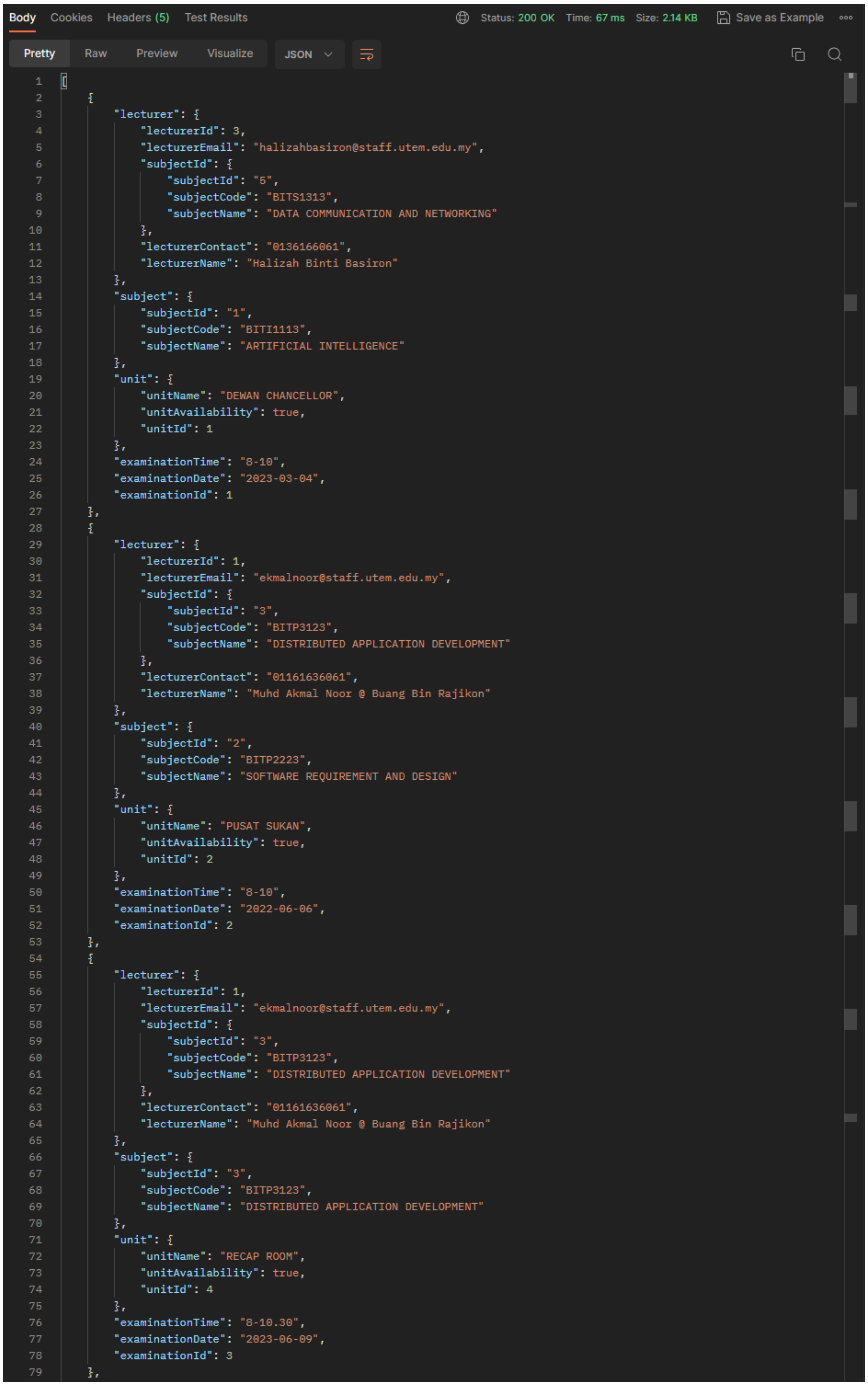
Default API: http://localhost:8080/examinationattendancesystem/api/attend

The following shows the web methods implemented:

1. getExamination()

   This web method will use "**GET**" mapping method to retrieve all examination schedule information based on all venues.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/examination
   Example of JSON data retrieved from Postman:



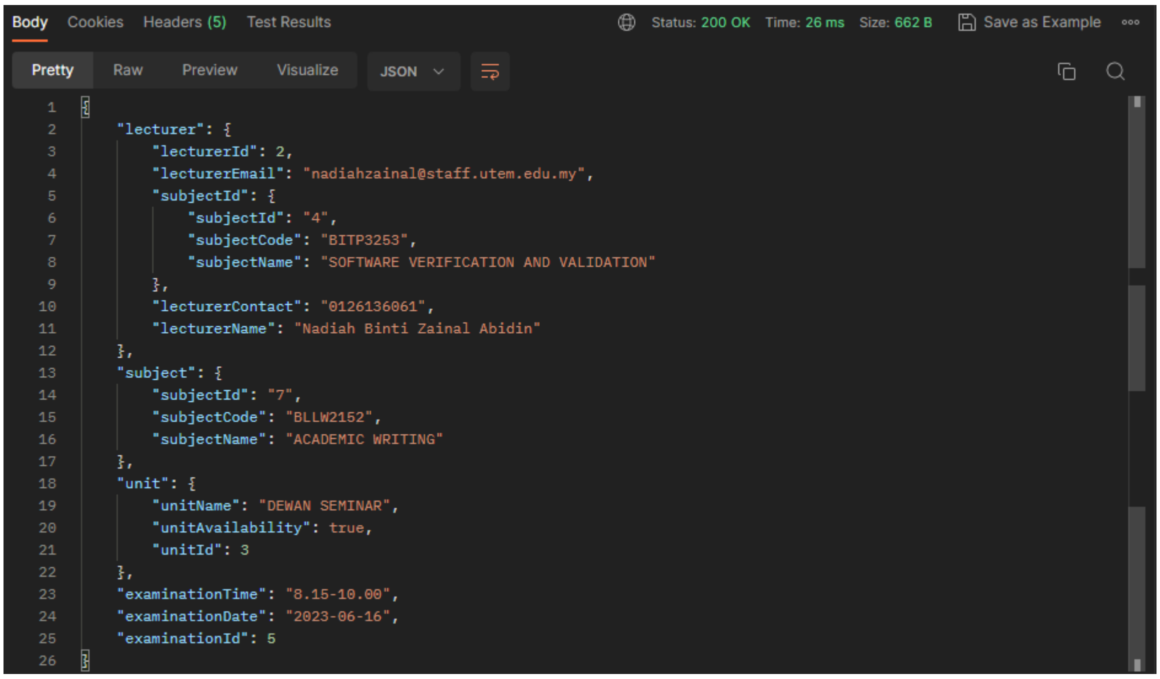
```
[
    {
        "lecturer": {
            "lecturerId": 3,
            "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
            "subjectId": {
                "subjectId": "5",
                "subjectCode": "BITS1313",
                "subjectName": "DATA COMMUNICATION AND NETWORKING"
            },
            "lecturerContact": "0136166061",
            "lecturerName": "Halizah Binti Basiron"
        },
        "subject": {
            "subjectId": "1",
            "subjectCode": "BITI1113",
            "subjectName": "ARTIFICIAL INTELLIGENCE"
        },
        "unit": {
            "unitName": "DEWAN CHANCELLOR",
            "unitAvailability": true,
            "unitId": 1
        },
        "examinationTime": "8-10",
        "examinationDate": "2023-03-04",
        "examinationId": 1
    },
    {
        "lecturer": {
            "lecturerId": 1,
            "lecturerEmail": "ekmalnoor@staff.utem.edu.my",
            "subjectId": {
                "subjectId": "3",
                "subjectCode": "BITP3123",
                "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
            },
            "lecturerContact": "01161636061",
            "lecturerName": "Muhd Akmal Noor @ Buang Bin Rajikon"
        },
        "subject": {
            "subjectId": "2",
            "subjectCode": "BITP2223",
            "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
        },
        "unit": {
            "unitName": "PUSAT SUKAN",
            "unitAvailability": true,
            "unitId": 2
        },
        "examinationTime": "8-10",
        "examinationDate": "2022-06-06",
        "examinationId": 2
    },
    {
        "lecturer": {
            "lecturerId": 1,
            "lecturerEmail": "ekmalnoor@staff.utem.edu.my",
            "subjectId": {
                "subjectId": "3",
                "subjectCode": "BITP3123",
                "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
            },
            "lecturerContact": "01161636061",
            "lecturerName": "Muhd Akmal Noor @ Buang Bin Rajikon"
        },
        "subject": {
            "subjectId": "3",
            "subjectCode": "BITP3123",
            "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
        },
        "unit": {
            "unitName": "RECAP ROOM",
            "unitAvailability": true,
            "unitId": 4
        },
        "examinationTime": "8-10.30",
        "examinationDate": "2023-06-09",
        "examinationId": 3
    },
```

2. getSpecific(@PathVariable Long examId)

   This web method will use "**GET**" mapping method to retrieve all examination schedule by specific Examination Id.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/examination/{examId}

   Example of JSON data retrieved from Postman given examId = 5:



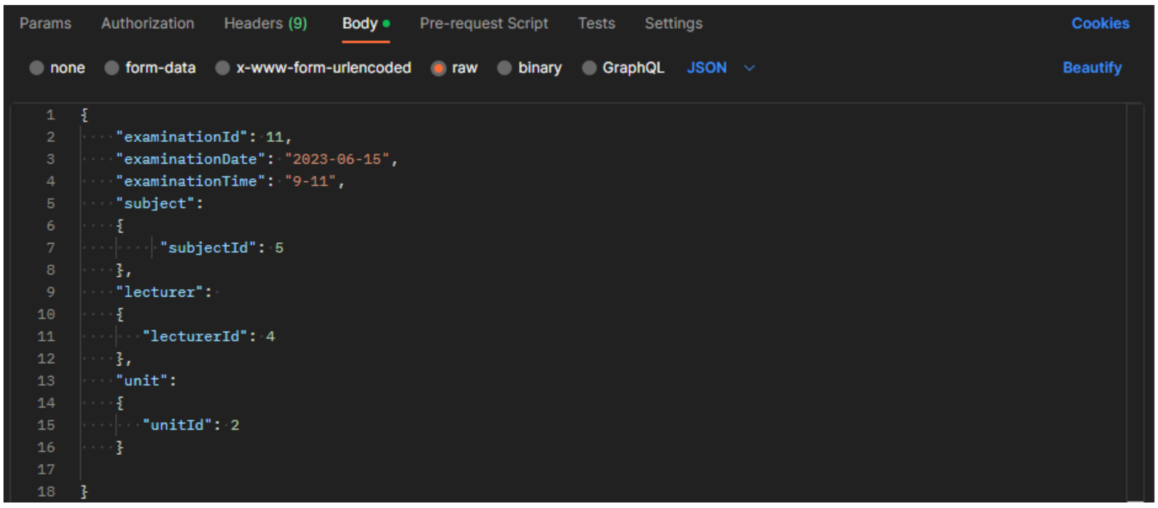
```
{
    "lecturer": {
        "lecturerId": 2,
        "lecturerEmail": "nadiahzainal@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "4",
            "subjectCode": "BITP3253",
            "subjectName": "SOFTWARE VERIFICATION AND VALIDATION"
        },
        "lecturerContact": "0126136061",
        "lecturerName": "Nadiah Binti Zainal Abidin"
    },
    "subject": {
        "subjectId": "7",
        "subjectCode": "BLLW2152",
        "subjectName": "ACADEMIC WRITING"
    },
    "unit": {
        "unitName": "DEWAN SEMINAR",
        "unitAvailability": true,
        "unitId": 3
    },
    "examinationTime": "8.15-10.00",
    "examinationDate": "2023-06-16",
    "examinationId": 5
}
```

3. insertSchedule(@RequestBody Examination examination)

   This web method will use "**POST**" mapping method to insert a new Examination schedule.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/examination

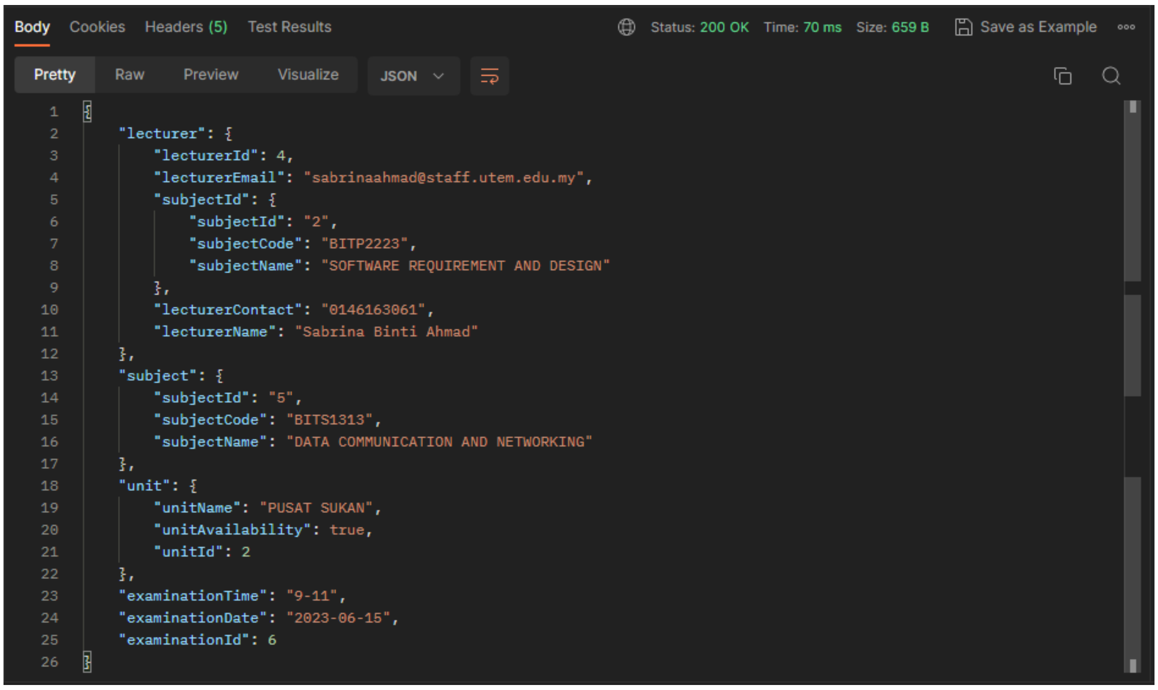
Example of new data to be inserted at Postman in JSON format:



```
{
    "examinationId": 11,
    "examinationDate": "2023-06-15",
    "examinationTime": "9-11",
    "subject":
    {
        "subjectId": 5
    },
    "lecturer":
    {
        "lecturerId": 4
    },
    "unit":
    {
        "unitId": 2
    }

}
```

Example of JSON data retrieved from Postman:



```
{
    "lecturer": {
        "lecturerId": 4,
        "lecturerEmail": "sabrinaahmad@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "2",
            "subjectCode": "BITP2223",
            "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
        },
        "lecturerContact": "0146163061",
        "lecturerName": "Sabrina Binti Ahmad"
    },
    "subject": {
        "subjectId": "5",
        "subjectCode": "BITS1313",
        "subjectName": "DATA COMMUNICATION AND NETWORKING"
    },
    "unit": {
        "unitName": "PUSAT SUKAN",
        "unitAvailability": true,
        "unitId": 2
    },
    "examinationTime": "9-11",
    "examinationDate": "2023-06-15",
    "examinationId": 6
}
```

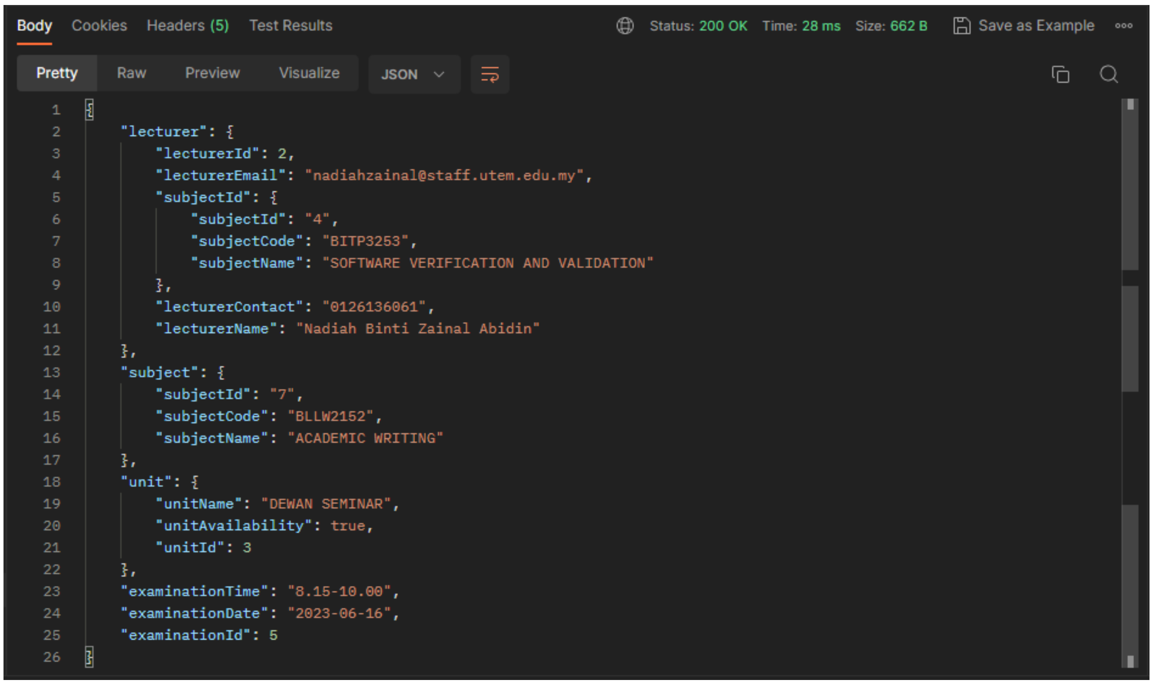
4. UpdateSchedule(@RequestBody Examination examination)

   This web method will use "**PUT**" mapping method to update a current Examination schedule.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/examination

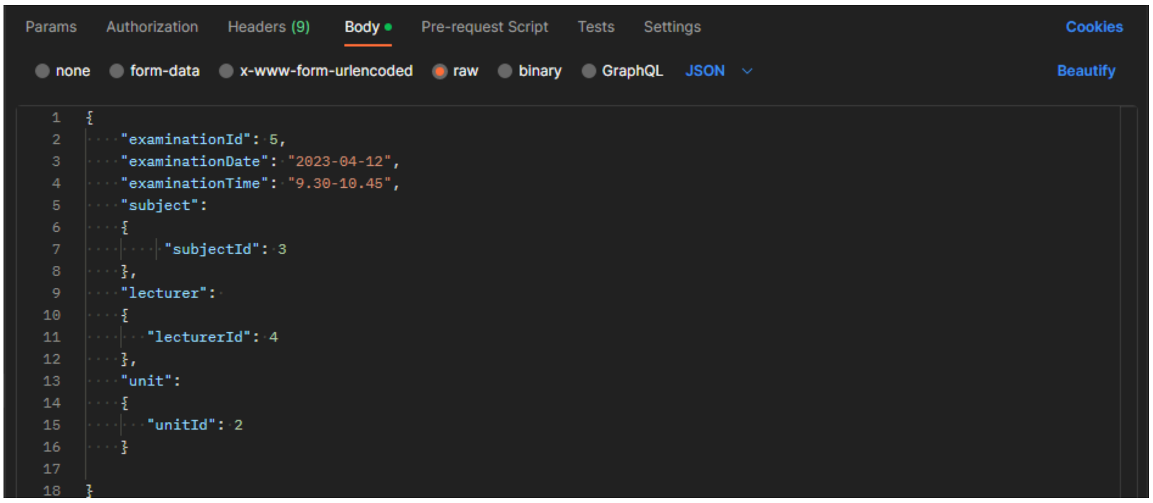
Example of data of Examination Schedule to be updated in JSON format at Postman:

Before Update:



```
{
    "lecturer": {
        "lecturerId": 2,
        "lecturerEmail": "nadiahzainal@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "4",
            "subjectCode": "BITP3253",
            "subjectName": "SOFTWARE VERIFICATION AND VALIDATION"
        },
        "lecturerContact": "0126136061",
        "lecturerName": "Nadiah Binti Zainal Abidin"
    },
    "subject": {
        "subjectId": "7",
        "subjectCode": "BLLW2152",
        "subjectName": "ACADEMIC WRITING"
    },
    "unit": {
        "unitName": "DEWAN SEMINAR",
        "unitAvailability": true,
        "unitId": 3
    },
    "examinationTime": "8.15-10.00",
    "examinationDate": "2023-06-16",
    "examinationId": 5
}
```

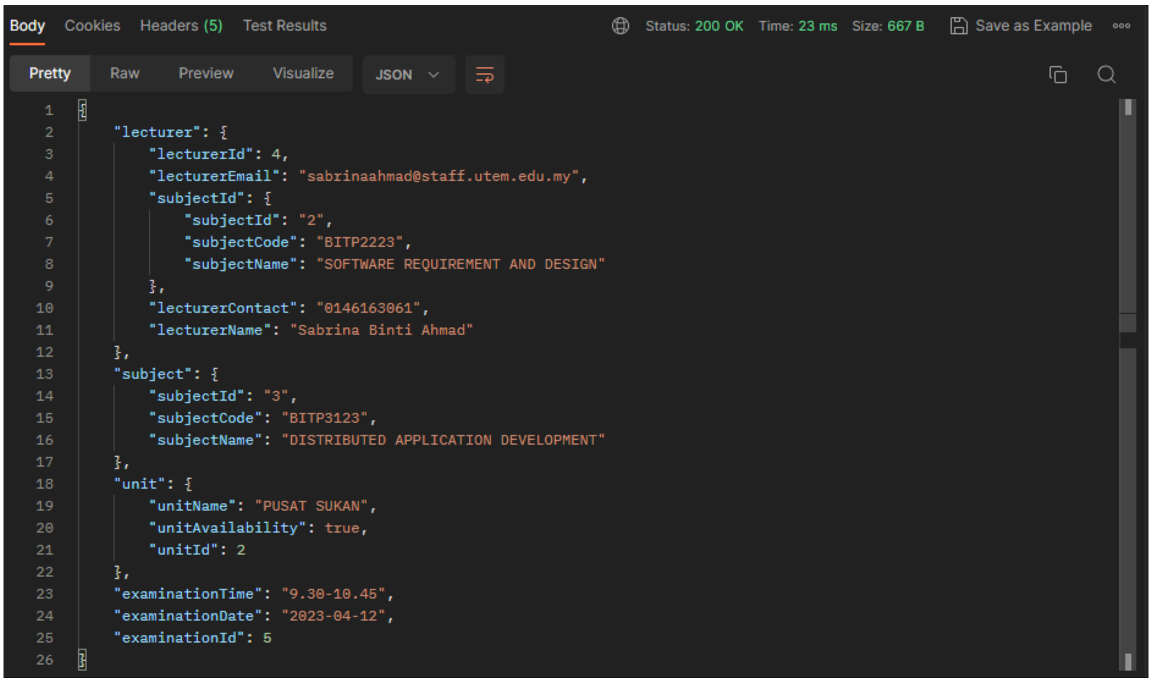
Update: examinationTime: 9.30-10.45 and subjectId = 3 based on examinationId = 5



```
{
    "examinationId": 5,
    "examinationDate": "2023-04-12",
    "examinationTime": "9.30-10.45",
    "subject":
    {
        "subjectId": 3
    },
    "lecturer":
    {
       "lecturerId": 4
    },
    "unit":
    {
       "unitId": 2
    }

}
```

After Example of JSON data retrieved from Postman:



```
{
    "lecturer": {
        "lecturerId": 4,
        "lecturerEmail": "sabrinaahmad@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "2",
            "subjectCode": "BITP2223",
            "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
        },
        "lecturerContact": "0146163061",
        "lecturerName": "Sabrina Binti Ahmad"
    },
    "subject": {
        "subjectId": "3",
        "subjectCode": "BITP3123",
        "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
    },
    "unit": {
        "unitName": "PUSAT SUKAN",
        "unitAvailability": true,
        "unitId": 2
    },
    "examinationTime": "9.30-10.45",
    "examinationDate": "2023-04-12",
    "examinationId": 5
}
```

5. deleteSchedule(@PathVariable long examId)

   This web method will use "**DELETE**" mapping method to delete a current Examination schedule.

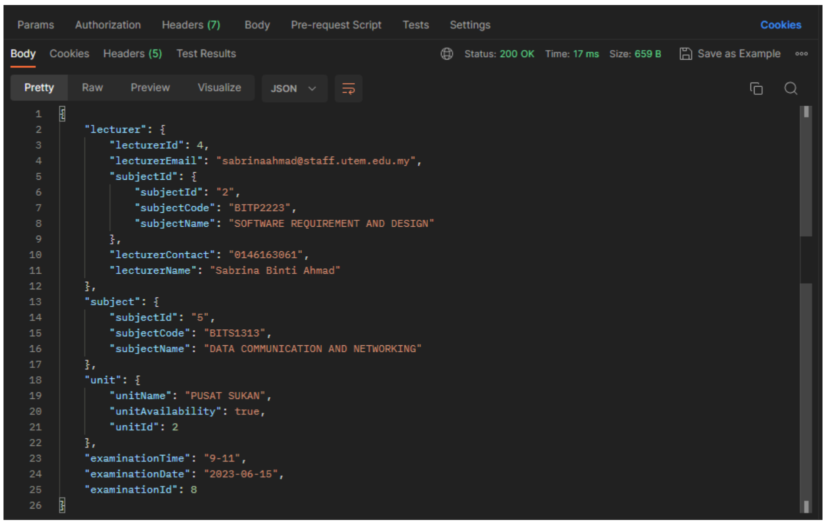
   Web API:

   http://localhost:8080/examinationattendancesystem/api/examination/{examId}

   Example of JSON data retrieved from Postman:

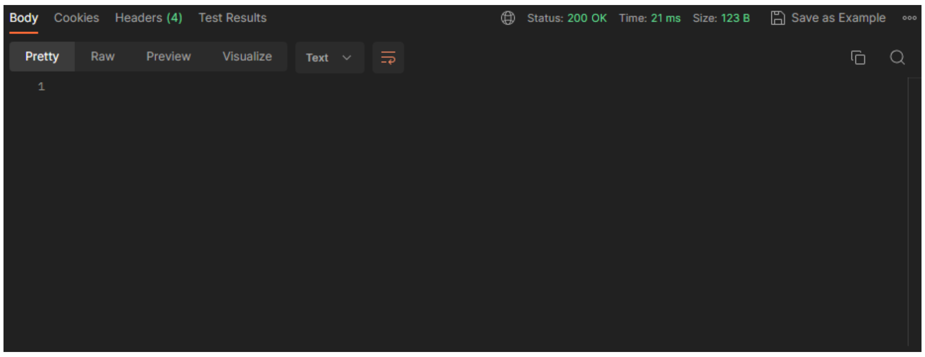
   Delete schedule where examId = 8

   Before delete,



```
{
    "lecturer": {
        "lecturerId": 4,
        "lecturerEmail": "sabrinaahmad@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "2",
            "subjectCode": "BITP2223",
            "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
        },
        "lecturerContact": "0146163061",
        "lecturerName": "Sabrina Binti Ahmad"
    },
    "subject": {
        "subjectId": "5",
        "subjectCode": "BITS1313",
        "subjectName": "DATA COMMUNICATION AND NETWORKING"
    },
    "unit": {
        "unitName": "PUSAT SUKAN",
        "unitAvailability": true,
        "unitId": 2
    },
    "examinationTime": "9-11",
    "examinationDate": "2023-06-15",
    "examinationId": 8
}
```

After Delete,


Body Cookies Headers (4) Test Results Status: 200 OK Time: 21 ms Size: 123 B Save as Example

Pretty Raw Preview Visualize Text

Third Class: ExaminationUnitRESTController

This class is mainly used to handle the examination unit to be invoked for displaying all current examination unit based on Read operation. To do so, an autowired object from ExaminationUnitRepository named examUnit will be created.

Default API: http://localhost:8080/examinationattendancesystem/api/venue

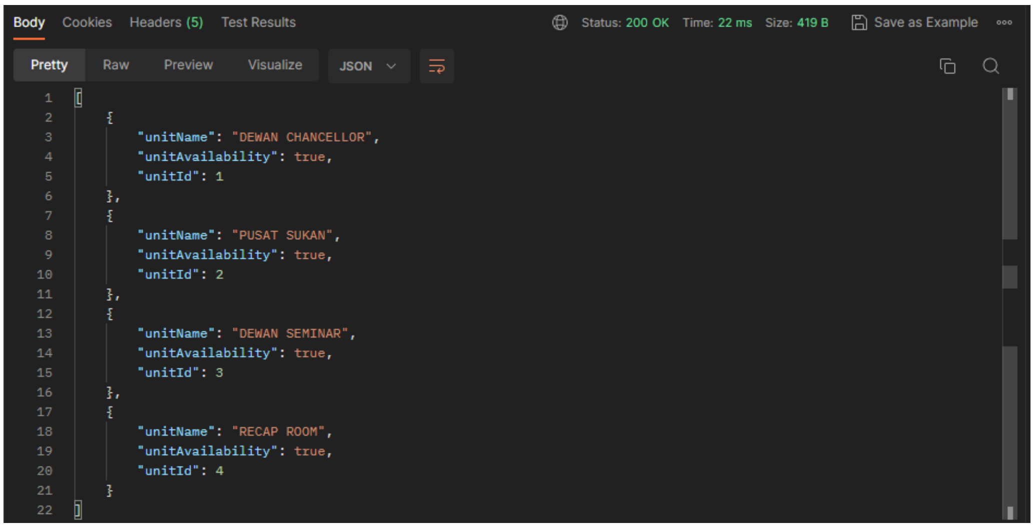
The following shows the web methods implemented:

1. getExaminationUnit()

   This method uses "**GET**" mapping method to retrieve all examination units' information.

   Web API: http://localhost:8080/examinationattendancesystem/api/venue

   Example of JSON data retrieved from Postman:


Body Cookies Headers (5) Test Results Status: 200 OK Time: 22 ms Size: 419 B Save as Example

Pretty Raw Preview Visualize JSON

```
[
    {
        "unitName": "DEWAN CHANCELLOR",
        "unitAvailability": true,
        "unitId": 1
    },
    {
        "unitName": "PUSAT SUKAN",
        "unitAvailability": true,
        "unitId": 2
    },
    {
        "unitName": "DEWAN SEMINAR",
        "unitAvailability": true,
        "unitId": 3
    },
    {
        "unitName": "RECAP ROOM",
        "unitAvailability": true,
        "unitId": 4
    }
]
```

Fourth Class: LecturerRESTController

This class is mainly used to handle the examination unit to be invoked for displaying all current lecturers based on Read operation. To do so, an @autowired object from LecturerRepository named lecturerRepos will be created.

Default API:
http://localhost:8080/examinationattendancesystem/api/lectures

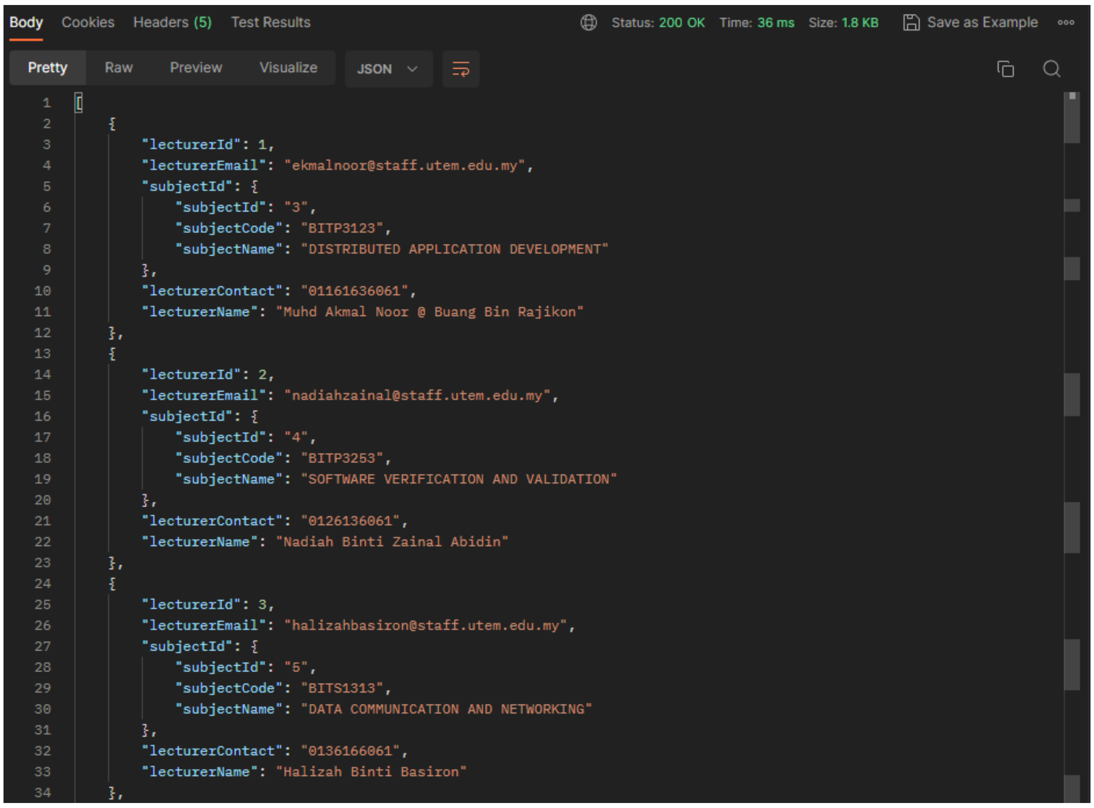
The following shows the web methods implemented:

1. getLecturer()

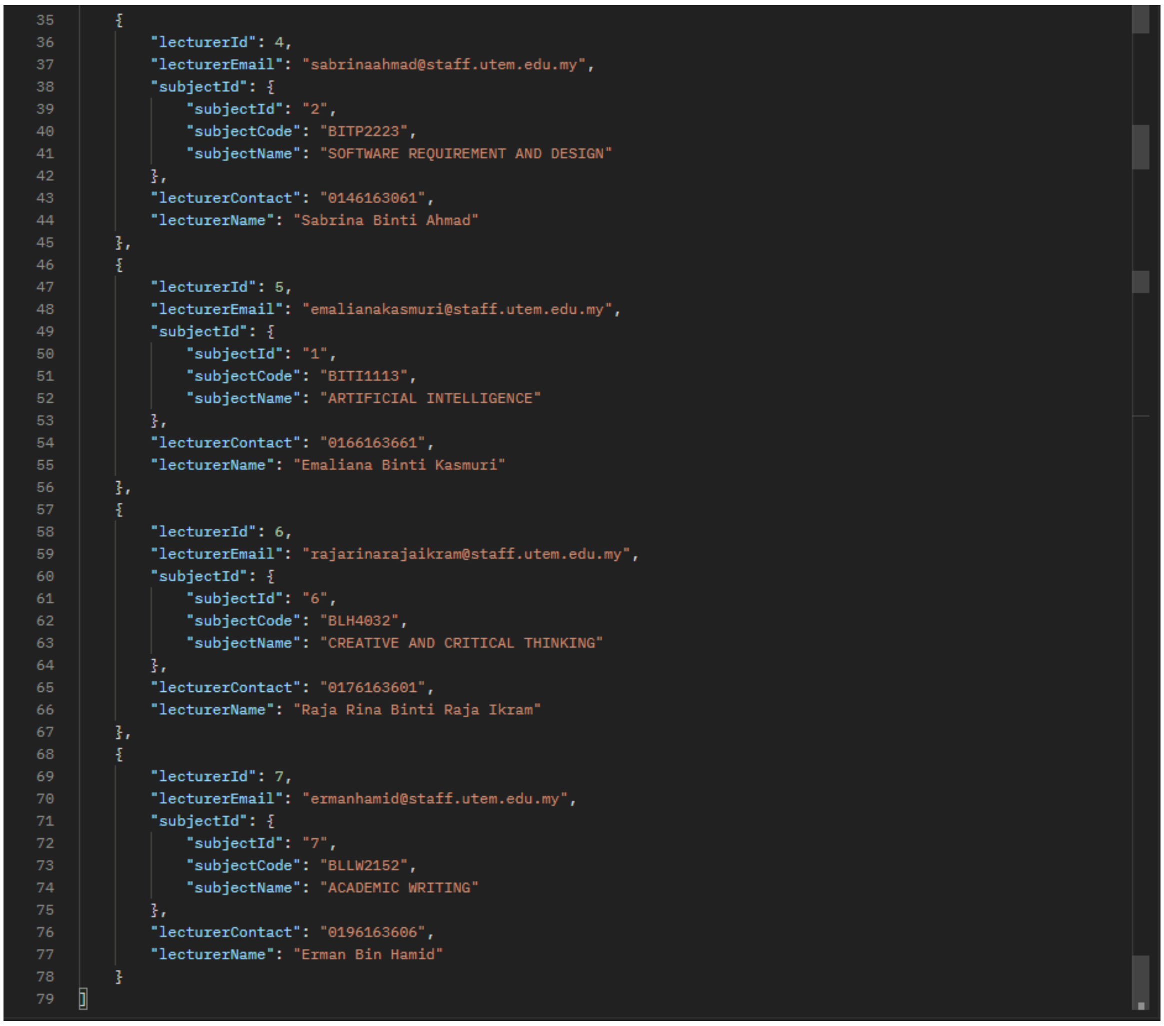
   This method uses "**GET**" mapping method to retrieve all lecturers' information.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/lectures

   Example of JSON data retrieved from Postman:



```
[
    {
        "lecturerId": 1,
        "lecturerEmail": "ekmalnoor@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "3",
            "subjectCode": "BITP3123",
            "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
        },
        "lecturerContact": "01161636061",
        "lecturerName": "Muhd Akmal Noor @ Buang Bin Rajikon"
    },
    {
        "lecturerId": 2,
        "lecturerEmail": "nadiahzainal@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "4",
            "subjectCode": "BITP3253",
            "subjectName": "SOFTWARE VERIFICATION AND VALIDATION"
        },
        "lecturerContact": "0126136061",
        "lecturerName": "Nadiah Binti Zainal Abidin"
    },
    {
        "lecturerId": 3,
        "lecturerEmail": "halizahbasiron@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "5",
            "subjectCode": "BITS1313",
            "subjectName": "DATA COMMUNICATION AND NETWORKING"
        },
        "lecturerContact": "0136166061",
        "lecturerName": "Halizah Binti Basiron"
    },
```



```json
    {
        "lecturerId": 4,
        "lecturerEmail": "sabrinaahmad@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "2",
            "subjectCode": "BITP2223",
            "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
        },
        "lecturerContact": "0146163061",
        "lecturerName": "Sabrina Binti Ahmad"
    },
    {
        "lecturerId": 5,
        "lecturerEmail": "emalianakasmuri@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "1",
            "subjectCode": "BITI1113",
            "subjectName": "ARTIFICIAL INTELLIGENCE"
        },
        "lecturerContact": "0166163661",
        "lecturerName": "Emaliana Binti Kasmuri"
    },
    {
        "lecturerId": 6,
        "lecturerEmail": "rajarinarajaikram@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "6",
            "subjectCode": "BLH4032",
            "subjectName": "CREATIVE AND CRITICAL THINKING"
        },
        "lecturerContact": "0176163601",
        "lecturerName": "Raja Rina Binti Raja Ikram"
    },
    {
        "lecturerId": 7,
        "lecturerEmail": "ermanhamid@staff.utem.edu.my",
        "subjectId": {
            "subjectId": "7",
            "subjectCode": "BLLW2152",
            "subjectName": "ACADEMIC WRITING"
        },
        "lecturerContact": "0196163606",
        "lecturerName": "Erman Bin Hamid"
    }
]
```

Fifth Class: StudentRESTController

This class is mainly used to handle the student for recording examination attendance to be invoked for displaying all current lecturers based on Read operation. To do so, an @autowired object from StudentRepository named students will be created.

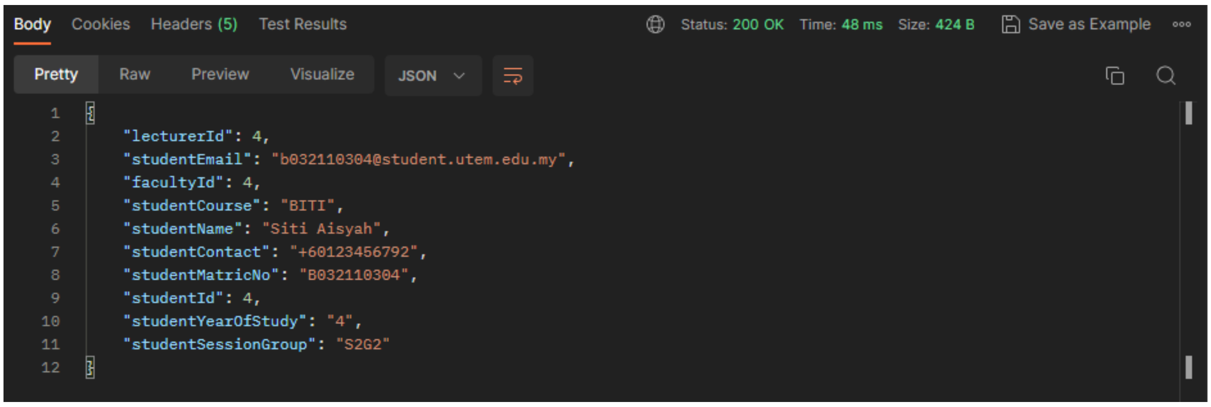
Default API: http://localhost:8080/examinationattendancesystem/api/student

The following shows the web methods implemented:

1. findStudentIdByMatricNo(@PathVariable String studentMatricNo)
   This method uses "**GET**" mapping method to retrieve student based on student's matric number.

Web API:
http://localhost:8080/examinationattendancesystem/api/student/matric/{studentMatricNo}

Example of JSON data retrieved from Postman given studentMatricNo = "B032110304":

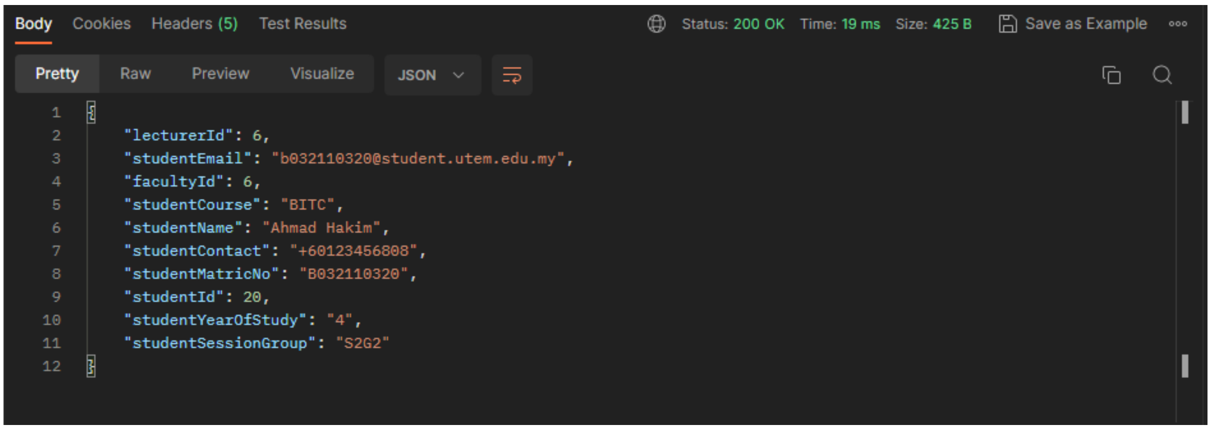


```
{
    "lecturerId": 4,
    "studentEmail": "b032110304@student.utem.edu.my",
    "facultyId": 4,
    "studentCourse": "BITI",
    "studentName": "Siti Aisyah",
    "studentContact": "+60123456792",
    "studentMatricNo": "B032110304",
    "studentId": 4,
    "studentYearOfStudy": "4",
    "studentSessionGroup": "S2G2"
}
```

2. getByStudentId(@PathVariable Long studentId)
   This method uses "**GET**" mapping method to retrieve student based on student id.

   Web API:
   http://localhost:8080/examinationattendancesystem/api/student/matric/{studentId}

   Example of JSON data retrieved from Postman given studentId = 20:



```
{
    "lecturerId": 6,
    "studentEmail": "b032110320@student.utem.edu.my",
    "facultyId": 6,
    "studentCourse": "BITC",
    "studentName": "Ahmad Hakim",
    "studentContact": "+60123456808",
    "studentMatricNo": "B032110320",
    "studentId": 20,
    "studentYearOfStudy": "4",
    "studentSessionGroup": "S2G2"
}
```

<u>Sixth Class: SubjectRESTController</u>

This class is mainly used to handle the student for recording examination attendance to be invoked for displaying all current lecturers based on Read operation. To do so, an @autowired object from StudentRepository named students will be created.

Default API:
http://localhost:8080/examinationattendancesystem/api/subjects

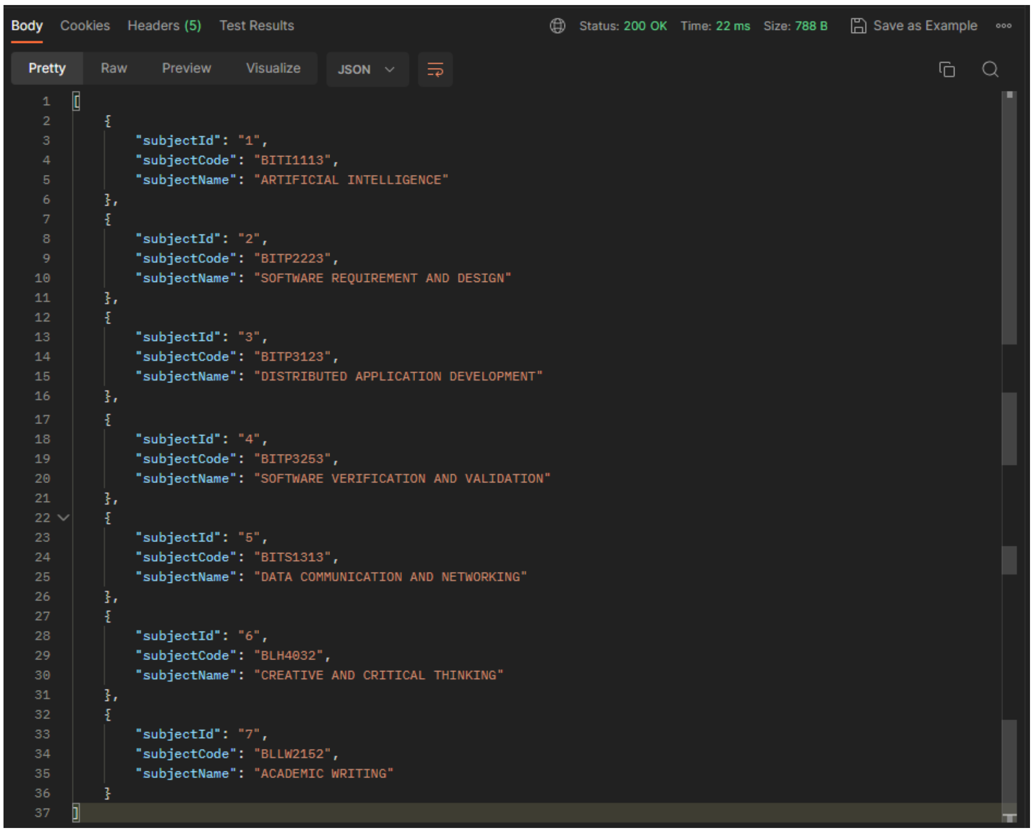
The following shows the web methods implemented:

1. getSubject()

   This method uses "**GET**" mapping method to retrieve all subjects' information.

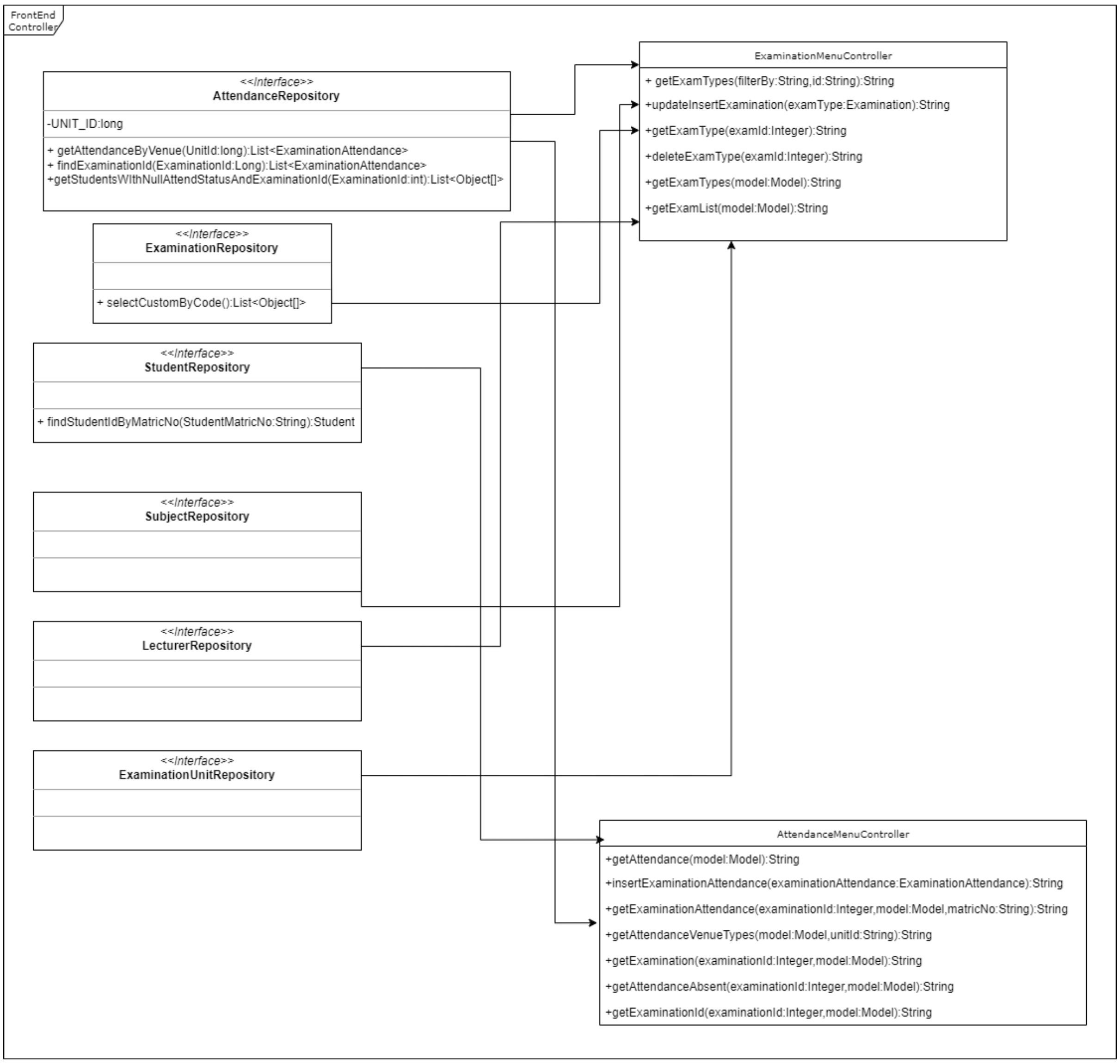
   Web API:
   http://localhost:8080/examinationattendancesystem/api/subjects

   Example of JSON data retrieved from Postman:



```
[
    {
        "subjectId": "1",
        "subjectCode": "BITI1113",
        "subjectName": "ARTIFICIAL INTELLIGENCE"
    },
    {
        "subjectId": "2",
        "subjectCode": "BITP2223",
        "subjectName": "SOFTWARE REQUIREMENT AND DESIGN"
    },
    {
        "subjectId": "3",
        "subjectCode": "BITP3123",
        "subjectName": "DISTRIBUTED APPLICATION DEVELOPMENT"
    },
    {
        "subjectId": "4",
        "subjectCode": "BITP3253",
        "subjectName": "SOFTWARE VERIFICATION AND VALIDATION"
    },
    {
        "subjectId": "5",
        "subjectCode": "BITS1313",
        "subjectName": "DATA COMMUNICATION AND NETWORKING"
    },
    {
        "subjectId": "6",
        "subjectCode": "BLH4032",
        "subjectName": "CREATIVE AND CRITICAL THINKING"
    },
    {
        "subjectId": "7",
        "subjectCode": "BLLW2152",
        "subjectName": "ACADEMIC WRITING"
    }
]
```

Implementation of Front-End Controllers


FrontEnd Controller
<<Interface>>
AttendanceRepository
-UNIT_ID:long
+ getAttendanceByVenue(UnitId:long):List<ExaminationAttendance>
+ findExaminationId(ExaminationId:Long):List<ExaminationAttendance>
+getStudentsWithNullAttendStatusAndExaminationId(ExaminationId:int):List<Object[]>
<<Interface>>
ExaminationRepository
+ selectCustomByCode():List<Object[]>
<<Interface>>
StudentRepository
+ findStudentIdByMatricNo(StudentMatricNo:String):Student
<<Interface>>
SubjectRepository
<<Interface>>
LecturerRepository
<<Interface>>
ExaminationUnitRepository
ExaminationMenuController
+ getExamTypes(filterBy:String,id:String):String
+updateInsertExamination(examType:Examination):String
+getExamType(examId:Integer):String
+deleteExamType(examId:Integer):String
+getExamTypes(model:Model):String
+getExamList(model:Model):String
AttendanceMenuController
+getAttendance(model:Model):String
+insertExaminationAttendance(examinationAttendance:ExaminationAttendance):String
+getExaminationAttendance(examinationId:Integer,model:Model,matricNo:String):String
+getAttendanceVenueTypes(model:Model,unitId:String):String
+getExamination(examinationId:Integer,model:Model):String
+getAttendanceAbsent(examinationId:Integer,model:Model):String
+getExaminationId(examinationId:Integer,model:Model):String

Implementation of Front-end Controllers

<u>**AttendanceMenuController**</u>

The AttendanceMenuController is responsible for handling attendance-related operations in the system. It interacts with various classes and performs actions such as retrieving attendance lists, updating or adding attendance, generating reports, and more.

The following methods are implemented in the AttendanceMenuController:

- getAttendance(Model model)

This method retrieves the attendance list from the web service and attaches it to the model as an attribute. It returns the "attendancelist" HTML file to the browser.

- insertExaminationAttendance(@ModelAttribute ExaminationAttendance examinationAttendance)

This method updates or adds an attendance record by sending a request to the web service. It then redirects the request to display a list of attendance.

- getExaminationAttendance (@PathVariable Integer examinationId, Model model,@RequestParam(name = "matricNo",required=false) String matricNo)

This method gets the examination attendance based on the examination ID. It retrieves the corresponding student information from the web service if a matriculation number is provided. The values are attached to the model as attributes, and the "examinationattendance" HTML file is returned to the browser.

- getAttendanceVenueTypes(Model model,@RequestParam(name = "unitid",required = false) String unitId)

  This method retrieves attendance information based on the venue. It filters the table based on the unit ID and retrieves the attendance list from the web service. The list of attendance and examination units are attached to the model as attributes, and the "AttendanceVenue" HTML file is returned to the browser.

- getExamination (@PathVariable Integer examinationId, Model model)

This method displays the attendance report based on the examination ID. It retrieves the attendance data for the specified examination from the web service and attaches it to the model as an attribute. The "Report" HTML file is returned to the browser.

- getAttendanceAbsent (@PathVariable Integer examinationId, Model model)

This method displays the attendance information for students who are absent from the examination. It retrieves the list of absent students from the web service based on the examination ID and attaches it to the model as an attribute. The "attendanceabsent" HTML file is returned to the browser.

- getExaminationId (@PathVariable Integer examinationId, Model model)

This method retrieves the attendance details for a specific examination, prepares the data, and passes it to the "AttendanceMenu" view for rendering.

ExaminationMenuController

The ExaminationMenuController is responsible for handling examination schedule-related operations in the system. It interacts with other classes and performs actions such as retrieving examination schedules, adding or updating schedules, and more.

The following methods are implemented in the ExaminationMenuController:

- getExamTypes(Model model,@RequestParam(name = "filter",required=false) String filterBy,@RequestParam(name = "id",required = false) String id)

This method is mapped to the `/schedule` endpoint with the HTTP GET method. It retrieves a list of examination schedules based on the provided filters (if any) and adds the list to the model attribute "Examinations". The method returns the "schedule" view.

- updateInsertExamination(@ModelAttribute Examination examType)

This method is mapped to the `/schedule/save` endpoint with the HTTP POST method. It handles the updating or insertion of an examination schedule. If the examination already exists (based on the `ExaminationId`), it sends a PUT request to update the schedule. Otherwise, it sends a POST request to insert a new schedule. After processing the request, it redirects to the `/schedule` endpoint.

- getExamType (@PathVariable Integer ExaminationId, Model model)

This method is mapped to the `/schedule/{ExaminationId}` endpoint with the HTTP GET method. It retrieves an examination schedule based on the provided `ExaminationId` and adds it to the model attribute "examType". It also retrieves a list of subjects, examination units, and lecturers to populate dropdown menus in the view. The method returns the "scheduleinfo" view.

- deleteExamType(@PathVariable Integer examId)

This method is mapped to the `/schedule/delete/{examId}` endpoint with the HTTP POST method. It deletes an examination schedule based on the provided `examId`. It sends a DELETE request to the web service and redirects to the `/schedule` endpoint.

- getExamTypes(Model model)

This method (with a different model) is also used for generating reports. It is mapped to the `/report` endpoint with the HTTP GET method. It retrieves a list of examination schedules and adds them to the model attribute "ExaminationsReport". The method returns the "AttendanceReport" view.

- getExamList(Model model)

This method is mapped to the `/ExaminationList` endpoint with the HTTP GET method. It retrieves a list of examination schedules and adds them to the model attribute "ExaminationsReport". It also sets the page title to "Report For

Absent Student". The method returns the "AttendanceExaminationSchedule" view.

Implementation of Front-End

<u>**AttendanceExaminationSchedule.html**</u>

Header: The <head> section contains the document title and includes the necessary CSS and JavaScript files. It also defines a JavaScript function rowClick(event) that is used when a row in the table is clicked.

Body: The <body> section contains the main content of the web page.

Header: The line <div th:replace="fragments/header :: header"></div>` indicates that the header section is replaced by another Thymeleaf fragment called `header.html`.

Title: The <h2> heading displays the title "Absent Attendance Report" in a centered format.

Message: If the message variable is not null, an alert div is shown with a success message. The alert can be dismissed by clicking the close button.

Table: If there are items in the ExaminationsReport list, a table is displayed with the attendance report. Each row in the table represents an examination entry. The columns include the date, time, subject code, subject name, and venue. The table rows are generated dynamically using Thymeleaf's `th:each` attribute, and each row is assigned a click event that triggers the `rowClick(event)` JavaScript function.

Total Records: The total number of records in the `ExaminationsReport` list is displayed.

No Record Found: If the `ExaminationsReport` list is empty, a message is shown indicating that no records were found.

Modal: There is a modal dialog with a confirmation message for deleting entries. The modal is hidden by default and can be triggered by clicking on a "Delete" button.

Footer: The line `<div th:replace="fragments/footer :: footer"></div>` indicates that the footer section is replaced by another Thymeleaf fragment called `footer.html`.

JavaScript: The provided JavaScript code contains two event handlers

- **Delete Confirmation**: When a "Delete" button with the class "btn-delete" is clicked, a modal dialog is displayed with a confirmation message for deleting the corresponding entry. The confirmation message includes the examTypeCode, which is extracted from the clicked button's attributes.
- **Clear Button**: When the "Clear" button with the id "btnClear" is clicked, the value of the input field with the id "keyword" is cleared, and the web page is redirected to a specific URL (`[[@{/ordertype/list}]]`).

Data Processed: The front-end code expects data to be provided from the back end in the form of a list of `ExaminationsReport` objects. Each object in the list represents an examination entry with properties such as `ExaminationId`, `ExaminationDate`, `ExaminationTime`, `subject`, and `unit`. The front end dynamically generates a table based on this data and provides interactive functionalities such as row click events and delete confirmation dialogs. Based on the provided front-end code, the data processed by the front end includes:

Examinations Report Data:

- The front end expects a list of examination reports (`ExaminationsReport`) to be passed from the server-side code.
- Each examination report in the list should have attributes like ExaminationId, ExaminationDate, ExaminationTime, subject (with

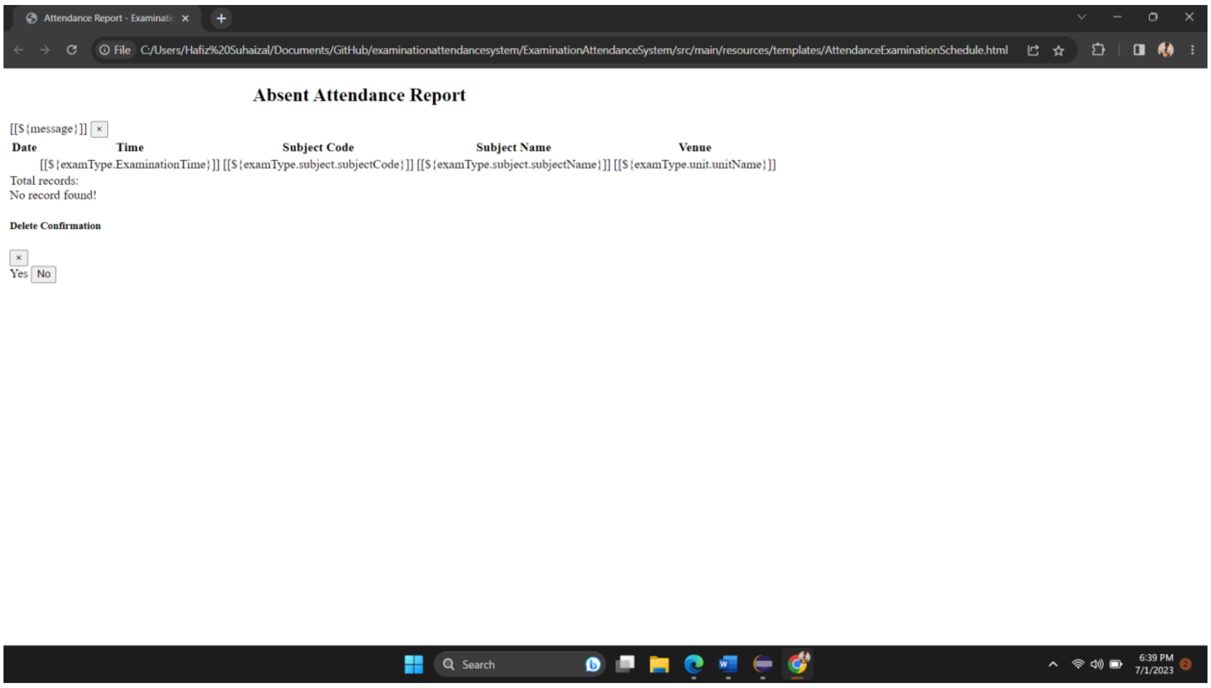
attributes like subjectCode and subjectName), and unit (with attributes like unitName).

- The front end processes this data to populate a table that displays the examination report information, including the date, time, subject code, subject name, and venue.

The front-end code retrieves the necessary data from the `ExaminationsReport` list and uses it to populate the table on the webpage.

Screenshot

- Without CSS


Attendance Report - Examinatic
File C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/AttendanceExaminationSchedule.html
Search
6:39 PM
7/1/2023

Absent Attendance Report

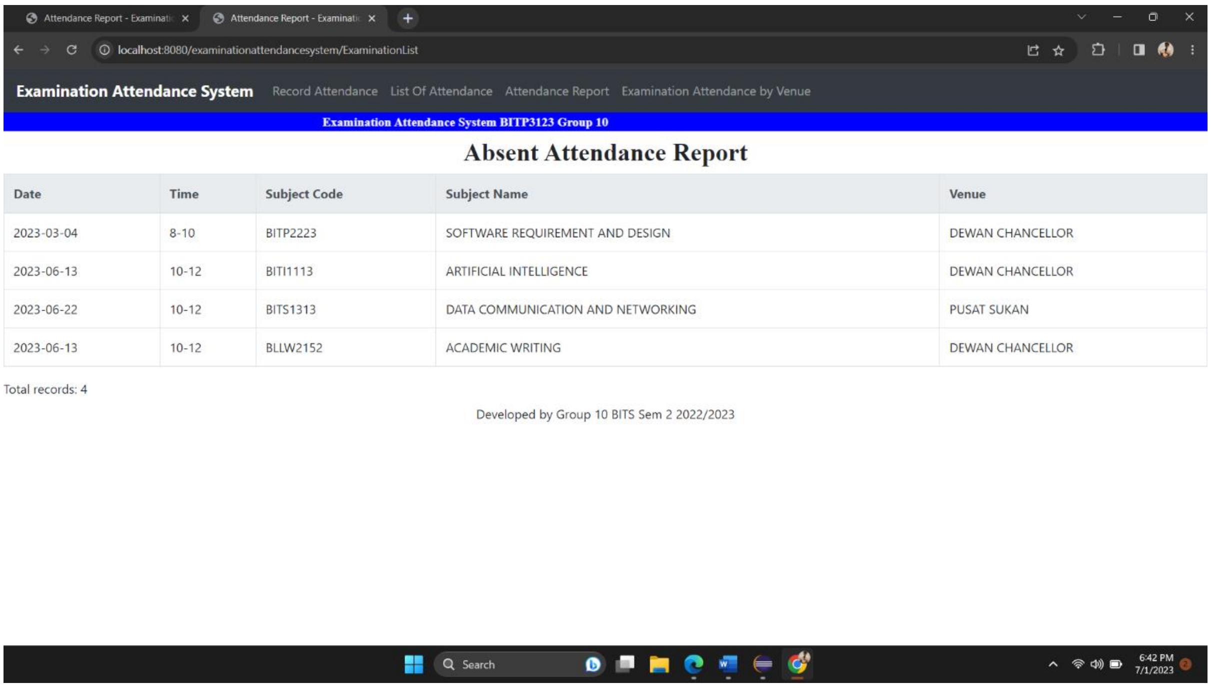
[[${message}]] ×

```
Date        Time        Subject Code        Subject Name        Venue
    [[${examType.ExaminationTime}]] [[${examType.subject.subjectCode}]] [[${examType.subject.subjectName}]] [[${examType.unit.unitName}]]
Total records:
No record found!
```

Delete Confirmation

×
Yes No

- With CSS


localhost:8080/examinationattendancesystem/ExaminationList

Examination Attendance System | Record Attendance | List Of Attendance | Attendance Report | Examination Attendance by Venue

Examination Attendance System BITP3123 Group 10

Absent Attendance Report

Date	Time	Subject Code	Subject Name	Venue
2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	DEWAN CHANCELLOR
2023-06-13	10-12	BITI1113	ARTIFICIAL INTELLIGENCE	DEWAN CHANCELLOR
2023-06-22	10-12	BITS1313	DATA COMMUNICATION AND NETWORKING	PUSAT SUKAN
2023-06-13	10-12	BLLW2152	ACADEMIC WRITING	DEWAN CHANCELLOR

Total records: 4

Developed by Group 10 BITS Sem 2 2022/2023

attendancelist.html

`<head>` section: Includes necessary meta tags, title, and external CSS and JavaScript files. It imports Bootstrap CSS, a custom CSS file, and the Font Awesome library. It also imports jQuery and Bootstrap JavaScript files.

Header and Footer: These sections are included using Thymeleaf's fragment feature. They are likely defined in separate HTML files and are included in this template using the `th:replace` attribute.

Main Content:

- `<h2>` heading with the ID "ATTEND" and the text "List of Attendance".
- `<div>` to display success messages if the "message" attribute is not null.
- A table to display the attendance records. The table headers define the columns, and the table rows are generated using Thymeleaf's iteration (`th:each`) feature to loop through the `examinationAttendance` list. Each row displays the respective values from the `examinationAttendance` object.
- A label to display the total number of records (`examinationAttendance.size()`).

- A "Print" button that triggers the `printDivContent()` JavaScript function.
- A JavaScript function `printDivContent()` that retrieves the content of the "ATTEND" and "TABLES" elements, opens a new window, writes the content to the new window, and prints it.

`<div>` with no records: This section is displayed when there are no records in the `examinationAttendance` list.

Modal: This section defines a modal dialog that can be used for delete confirmation. It includes a title, a body section to display the confirmation text, and footer buttons for "Yes" and "No" options.

Footer: Similar to the header, this section is included using Thymeleaf's fragment feature.

JavaScript: The script at the bottom of the file handles click events for the "Delete" buttons (`.btn-delete`) and the "Clear" button (`#btnClear`). It opens the delete confirmation modal and sets the confirmation text based on the clicked button's attributes. The "Clear" button clears the search keyword and redirects the user to the attendance list page.

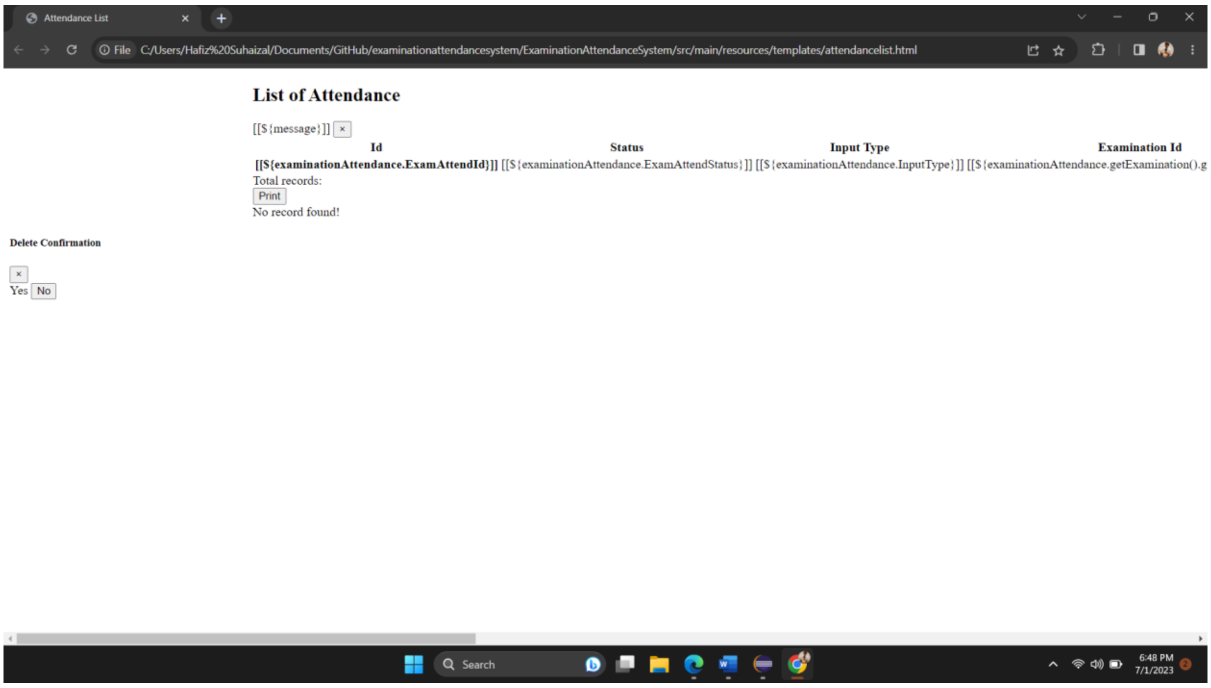
Data processed: by the front-end includes the following attributes for each attendance record:

- **ExamAttendId**: The ID of the attendance record.
- **ExamAttendStatus**: The status of the attendance (e.g., present, absent).
- **InputType**: The type of input for the attendance (e.g., manual, automated).
- **ExaminationId**: The ID of the examination associated with the attendance record.
- **StudentId**: The ID of the student associated with the attendance record.
- **StudentMatricNo**: The matriculation number of the student.
- **StudentName**: The name of the student.
- **SubjectCode**: The code of the subject related to the attendance.

- **SubjectName**: The name of the subject related to the attendance.

Screenshot

- Without CSS



C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/attendancelist.html

List of Attendance

[[${message}]]

Id	Status	Input Type	Examination Id
[[${examinationAttendance.ExamAttendId}]]	[[${examinationAttendance.ExamAttendStatus}]]	[[${examinationAttendance.InputType}]]	[[${examinationAttendance.getExamination().g

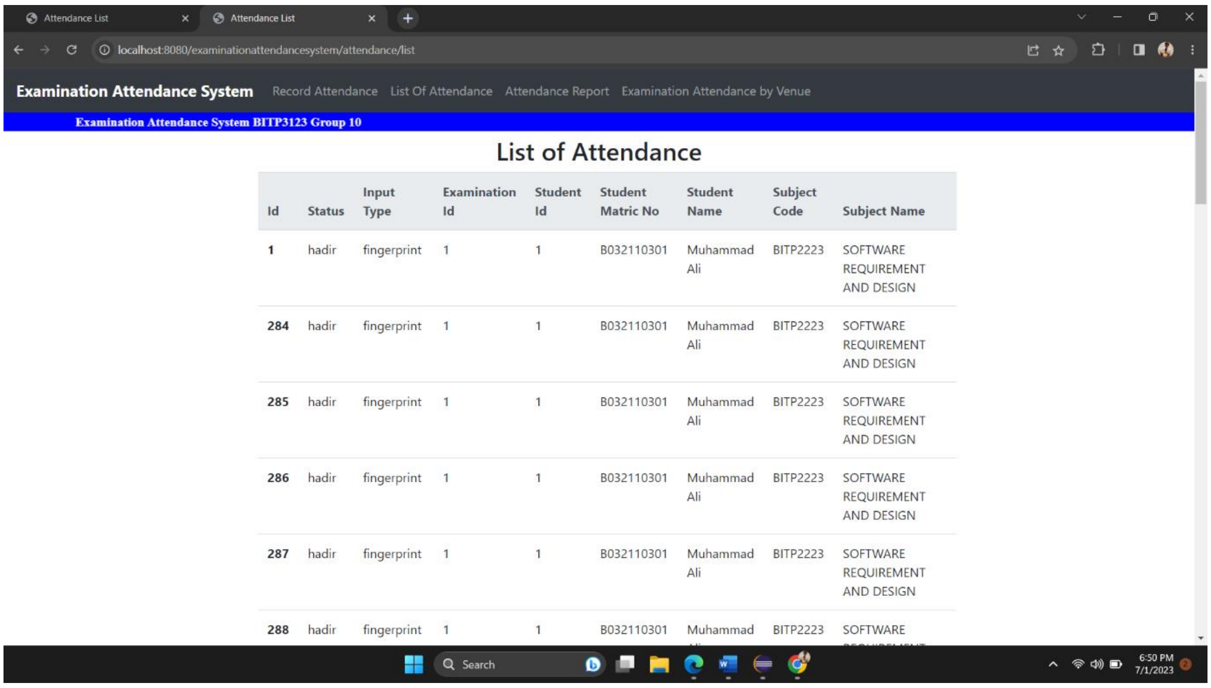
Total records:

No record found!

Delete Confirmation

Yes

- With CSS



localhost:8080/examinationattendancesystem/attendance/list

Examination Attendance System BITP3123 Group 10

List of Attendance

Id	Status	Input Type	Examination Id	Student Id	Student Matric No	Student Name	Subject Code	Subject Name
1	hadir	fingerprint	1	1	B032110301	Muhammad Ali	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
284	hadir	fingerprint	1	1	B032110301	Muhammad Ali	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
285	hadir	fingerprint	1	1	B032110301	Muhammad Ali	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
286	hadir	fingerprint	1	1	B032110301	Muhammad Ali	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
287	hadir	fingerprint	1	1	B032110301	Muhammad Ali	BITP2223	SOFTWARE REQUIREMENT AND DESIGN
288	hadir	fingerprint	1	1	B032110301	Muhammad	BITP2223	SOFTWARE

<u>AttendanceMenu.html</u>

`<head>` section: includes necessary meta tags, stylesheets, and scripts for proper rendering and styling of the content.

Page title: is set as "Attendance Report - Examination Attendance System."

`<body>` section: starts with the inclusion of the header using Thymeleaf's `th:replace` attribute, referencing a fragment named "header."

attendance data: is presented in a table structure (`<table>`) with class styles for border, hover effect, and positioning.

Table: has a header row (`<thead>`) containing columns for "Subject Code," "Subject Name," "Student Matric Number," "Student Name," and "Student Course."

attendance data: is dynamically displayed using Thymeleaf's iteration attribute `th:each`. The attendance records are retrieved from the `studentAttendance` collection.

values for each column: are extracted from the corresponding related classes and objects, such as `getExamination().getSubject().getSubjectCode()` and `getStudentId().getStudentMatricNo()`.

If there are attendance records (`studentAttendance.size() > 0`), the table is displayed with the attendance data.

If there are no attendance records (`studentAttendance.size() <= 0`), a message "No record found!" is shown.

Footer: is included using Thymeleaf's `th:replace` attribute, referencing a fragment named "footer."

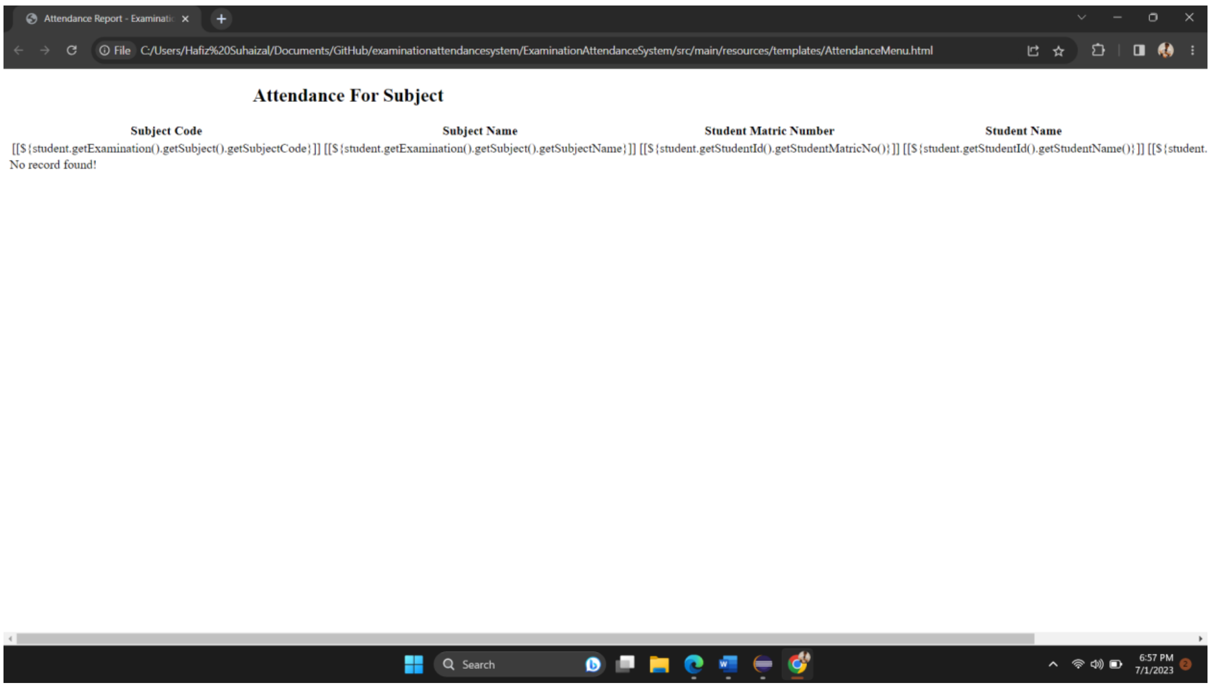
Data processed: by the front-end in this template is the attendance information for each student related to a specific subject. The template expects to receive the `studentAttendance` collection from the backend, which should contain attendance records.

For each student in the `studentAttendance` collection, the template accesses the relevant properties to populate the table columns. The following data is processed and displayed for each student:

- **Subject Code**: `student.getExamination().getSubject().getSubjectCode()`
- **Subject Name**: `student.getExamination().getSubject().getSubjectName()`
- **Student Matric Number**: `student.getStudentId().getStudentMatricNo()`
- **Student Name**: `student.getStudentId().getStudentName()`
- **Student Course**: `student.getStudentId().getStudentCourse()`

Screenshot

- Without CSS

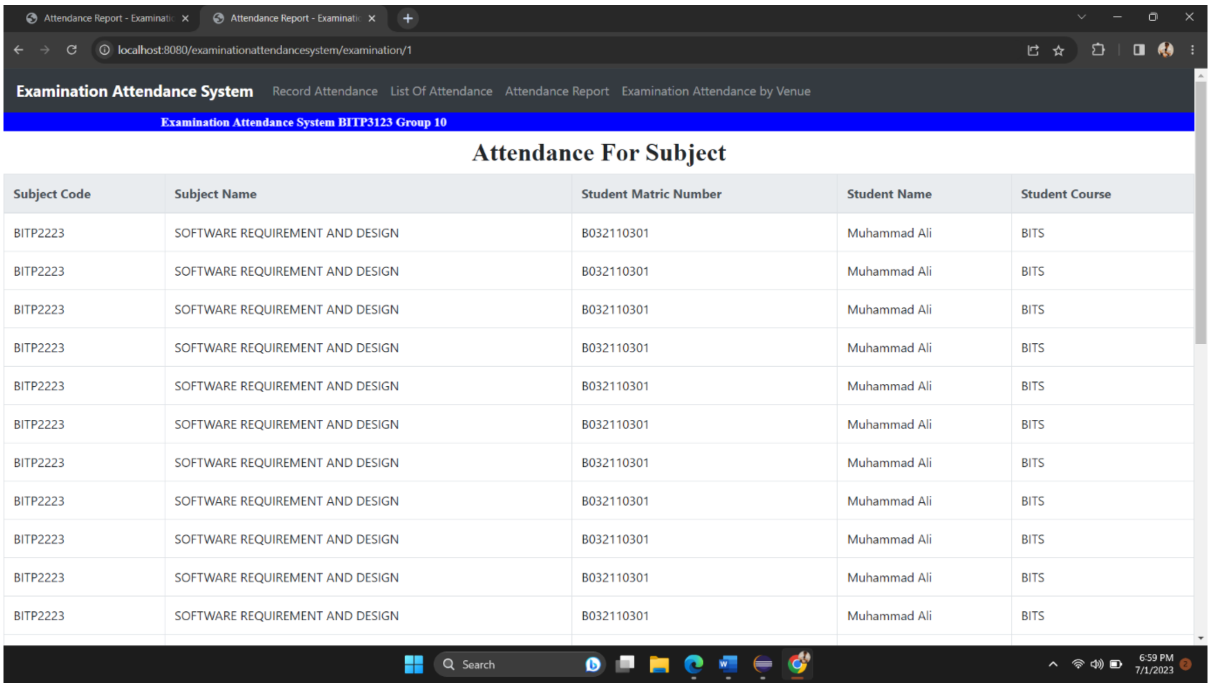

Attendance Report - Examinatic
File C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/AttendanceMenu.html
Search
6:57 PM
7/1/2023

Attendance For Subject

Subject Code	Subject Name	Student Matric Number	Student Name	
[[${student.getExamination().getSubject().getSubjectCode()}]]	[[${student.getExamination().getSubject().getSubjectName()}]]	[[${student.getStudentId().getStudentMatricNo()}]]	[[${student.getStudentId().getStudentName()}]]	[[${student.

No record found!

- With CSS


Attendance Report - Examinati
Attendance Report - Examinati
localhost:8080/examinationattendancesystem/examination/1
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue
Search
6:59 PM
7/1/2023

Examination Attendance System BITP3123 Group 10

Attendance For Subject

Subject Code	Subject Name	Student Matric Number	Student Name	Student Course
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS
BITP2223	SOFTWARE REQUIREMENT AND DESIGN	B032110301	Muhammad Ali	BITS

AttendanceReport.html

`<head>` section: includes meta tags for character encoding, viewport settings, and a title for the page.

`<body>` section: starts with a header fragment included using `<div th:replace="fragments/header :: header"></div>`.

It includes a container `<div>` with an id of "SCHEDULE" that displays the heading "Attendance Report".

If a message is present (e.g., a success message), it is displayed using an alert box.

The template includes a table with attendance information, where each row represents an examination. The attendance data is expected to be provided in the `ExaminationsReport` collection from the back-end.

The table displays columns for Date, Time, Subject Code, Subject Name, and Venue. The values for each examination are dynamically inserted into the respective table cells using Thymeleaf expressions (`[[...]]`).

Each row in the table is clickable and triggers the JavaScript function `rowClick(event)`. It retrieves the examination ID from the clicked row and redirects the user to a detailed report page.

The template includes an input button labeled "Print" that triggers the `printDivContent()` JavaScript function. This function opens a new window, copies the content of the "SCHEDULE" and "TABLES" divs, and prints the window's content.

If no attendance records are available (i.e., `ExaminationsReport` is empty), a message stating "No record found!" is displayed.

The template includes a modal dialog for delete confirmation, although its functionality is not fully implemented.

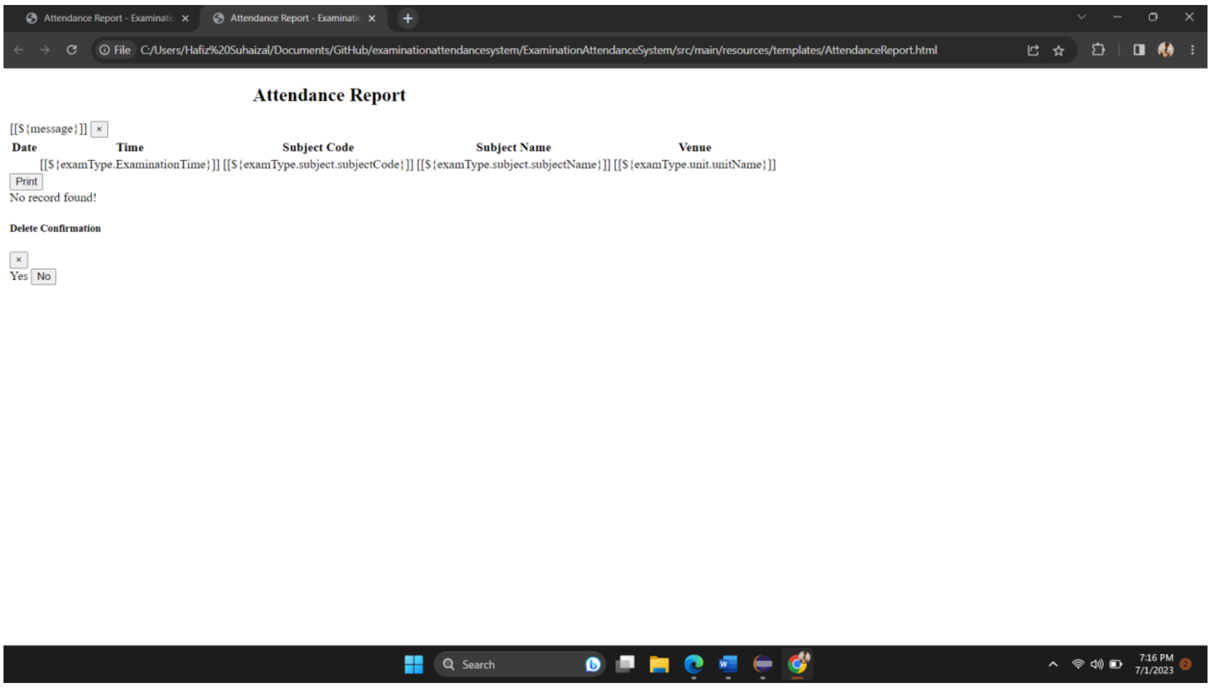
The footer fragment is included using `<div th:replace="fragments/footer :: footer"></div>`.

Data processed:

- **ExaminationsReport** : This data is used to populate the attendance report table. It is expected to be a collection of objects representing examinations. Each object should have properties such as `ExaminationId`, `ExaminationDate`, `ExaminationTime`, `subject`, and `unit`. The template uses Thymeleaf expressions (`[[...]]`) to retrieve and display the values of these properties in the table cells.

Screenshot

- Without CSS



C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/AttendanceReport.html

Attendance Report

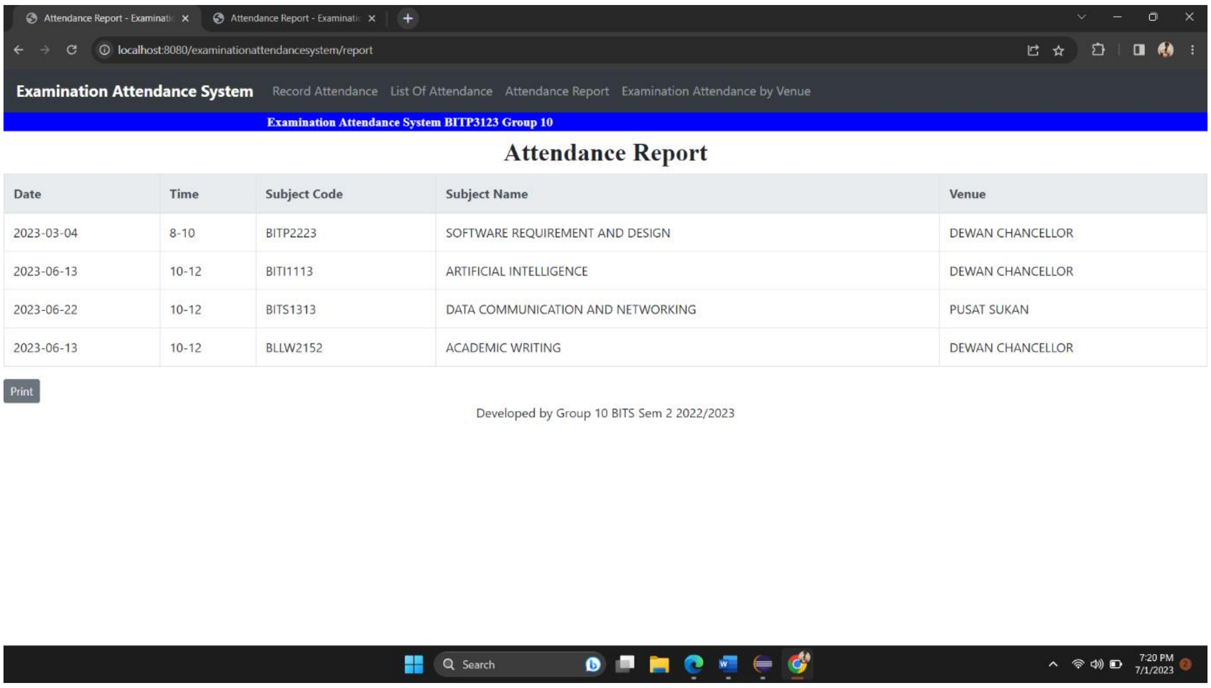
[[${message}]]

```
Date          Time                    Subject Code                    Subject Name                    Venue
       [[${examType.ExaminationTime}]] [[${examType.subject.subjectCode}]] [[${examType.subject.subjectName}]] [[${examType.unit.unitName}]]
Print
No record found!

Delete Confirmation

Yes No
```

- With CSS



localhost:8080/examinationattendancesystem/report

Examination Attendance System | Record Attendance | List Of Attendance | Attendance Report | Examination Attendance by Venue

Examination Attendance System BITP3123 Group 10

Attendance Report

Date	Time	Subject Code	Subject Name	Venue
2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	DEWAN CHANCELLOR
2023-06-13	10-12	BITI1113	ARTIFICIAL INTELLIGENCE	DEWAN CHANCELLOR
2023-06-22	10-12	BITS1313	DATA COMMUNICATION AND NETWORKING	PUSAT SUKAN
2023-06-13	10-12	BLLW2152	ACADEMIC WRITING	DEWAN CHANCELLOR

Print

Developed by Group 10 BITS Sem 2 2022/2023

AttendanceVenue.html

examUnit : This data is used to populate the `<select>` dropdown for filtering the attendance by venue. It is expected to be a collection of objects representing examination units. Each object should have properties such as `UnitId` and `UnitName`. The template uses Thymeleaf expressions (`th:each`) to iterate over this collection and generate the `<option>` elements of the dropdown.

attendVenue : This data is used to populate the attendance table based on the selected venue. It is expected to be a collection of objects representing attendance records. Each object should have properties such as `ExamAttendId`, `StudentId`, `Examination`, `ExamAttendStatus`, and `InputType`. The template uses Thymeleaf expressions (`[[...]]` and `th:text`) to retrieve and display the values of these properties in the table cells.

The front-end template allows the user to filter attendance records by venue using the dropdown `<select>` element. When the user selects a venue, the `filterAttendance(event)` JavaScript function is triggered, which updates the URL with the selected venue and reloads the page to fetch the filtered data.

The template also includes a "Print" button that invokes the `printDivContent()` JavaScript function. This function opens a new window, copies the content of the "AttendanceVenue" and "TABLES" sections, and prints them.

Overall, the front-end template provides a user interface to view and filter attendance records by venue and supports printing functionality.

Data processed:

- **examUnit**: This data is used to populate the `<select>` dropdown for filtering attendance by venue. The template uses a Thymeleaf expression (`th:each`) to iterate over the `examUnit` collection and generate the `<option>` elements of the dropdown. Each `Unit` object in the collection is expected to have properties `UnitId` and `UnitName`.
- **attendVenue**: This data is used to populate the attendance table. The template uses a Thymeleaf expression (`th:each`) to iterate over the `attendVenue` collection and generate the rows of the table. Each `attendVenue` object in the collection is expected to have properties such as `ExamAttendId`, `StudentId`, `Examination`,

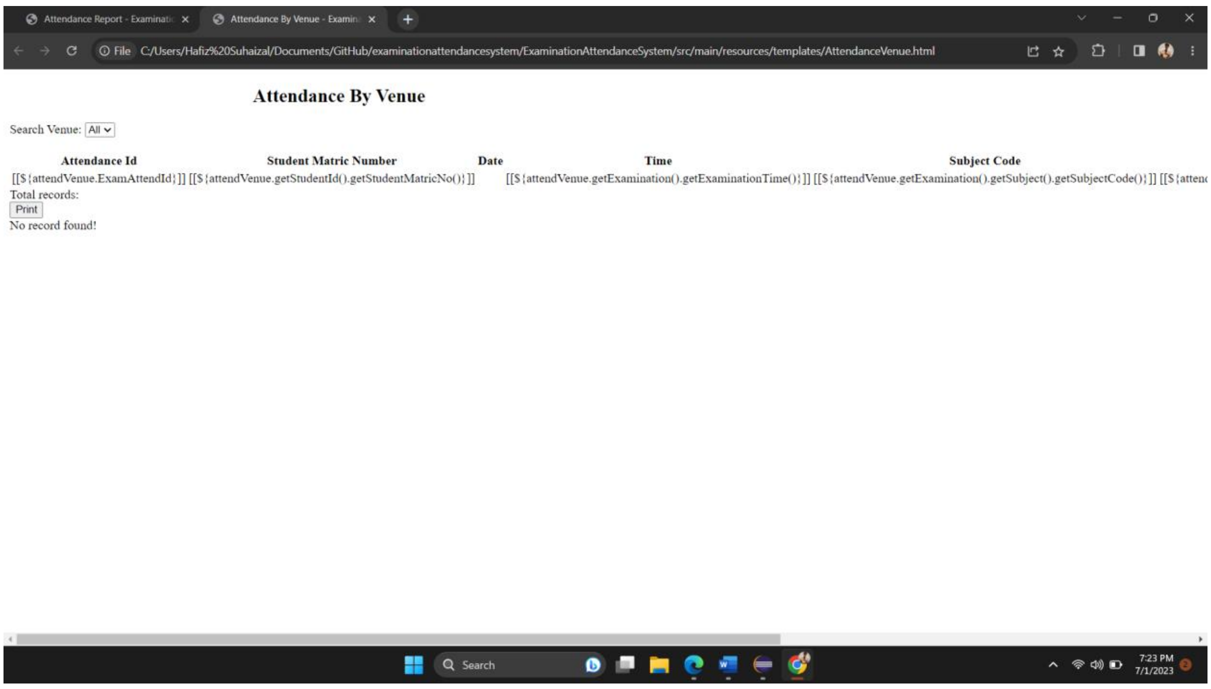
`ExamAttendStatus`, and `InputType`. The values of these properties are displayed in the respective columns of the table.

The template also includes JavaScript functions for filtering the attendance records based on the selected venue and for printing the content of the "AttendanceVenue" and "TABLES" sections.

In summary, the front-end template processes `examUnit` and `attendVenue` data to populate the dropdown and attendance table, respectively.

Screenshot

- Without CSS

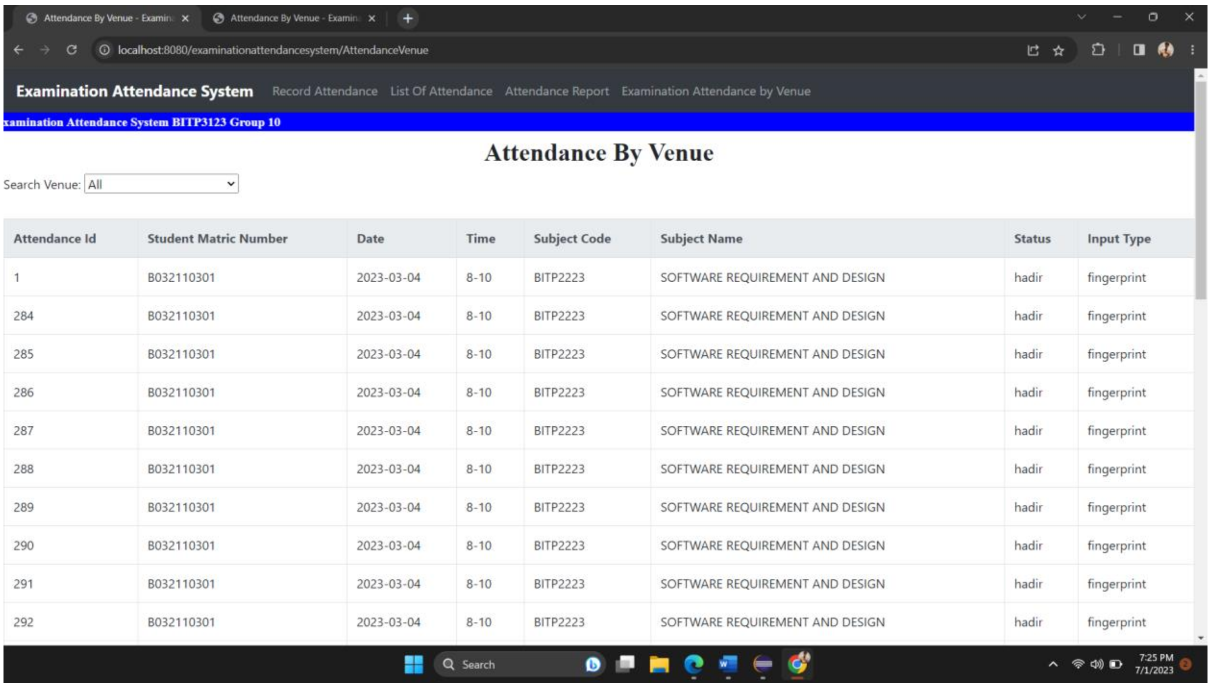


Attendance By Venue

Search Venue: All

```
Attendance Id                       Student Matric Number                          Date              Time                                                  Subject Code
[[${attendVenue.ExamAttendId}]] [[${attendVenue.getStudentId().getStudentMatricNo()}]]   [[${attendVenue.getExamination().getExaminationTime()}]] [[${attendVenue.getExamination().getSubject().getSubjectCode()}]] [[${attenc
Total records:
Print
No record found!
```

- With CSS


Attendance By Venue - Examin
Attendance By Venue - Examin
localhost:8080/examinationattendancesystem/AttendanceVenue
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue
xamination Attendance System BITP3123 Group 10
Search
7:25 PM
7/1/2023

Attendance By Venue

Search Venue: All

Attendance Id	Student Matric Number	Date	Time	Subject Code	Subject Name	Status	Input Type
1	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
284	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
285	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
286	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
287	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
288	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
289	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
290	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
291	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint
292	B032110301	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	hadir	fingerprint

examinationattendance.html

The template includes a logo of UTeM (Universiti Teknikal Malaysia Melaka) using an image tag.

The form allows the user to record attendance by providing the following information:

- **Matric Card**: The user enters the matric card number of the student. There is a "Check" button to verify the student's information.
- **Reader Input**: The user selects the type of reader input (MatricCard, Fingerprint, QRCode, or SelfCheck-In) from a dropdown menu.
- **Attendance**: The user selects the attendance status (Hadir) using a radio button.

The form is submitted to the server using the POST method to the `/examinationattendance/save` endpoint.

The template includes JavaScript functions:

- **checkStudent()**: This function is called when the "Check" button is clicked. It appends the matric card value to the current URL and redirects the user to the updated URL.
- **validateStudentId(event)**: This function is called when the form is submitted. It checks if the hidden input field `studentIdVal` has a value of '0' and displays an error message if so.
- **$(document).ready()**: This function attaches a click event handler to the "Cancel" button. When clicked, it redirects the user to the `/examinationAttendance` URL.

In summary, the front-end template provides a form for recording attendance by capturing student information, reader input type, and attendance status. The template incorporates Thymeleaf expressions for data binding and includes JavaScript functions for form validation and navigation.

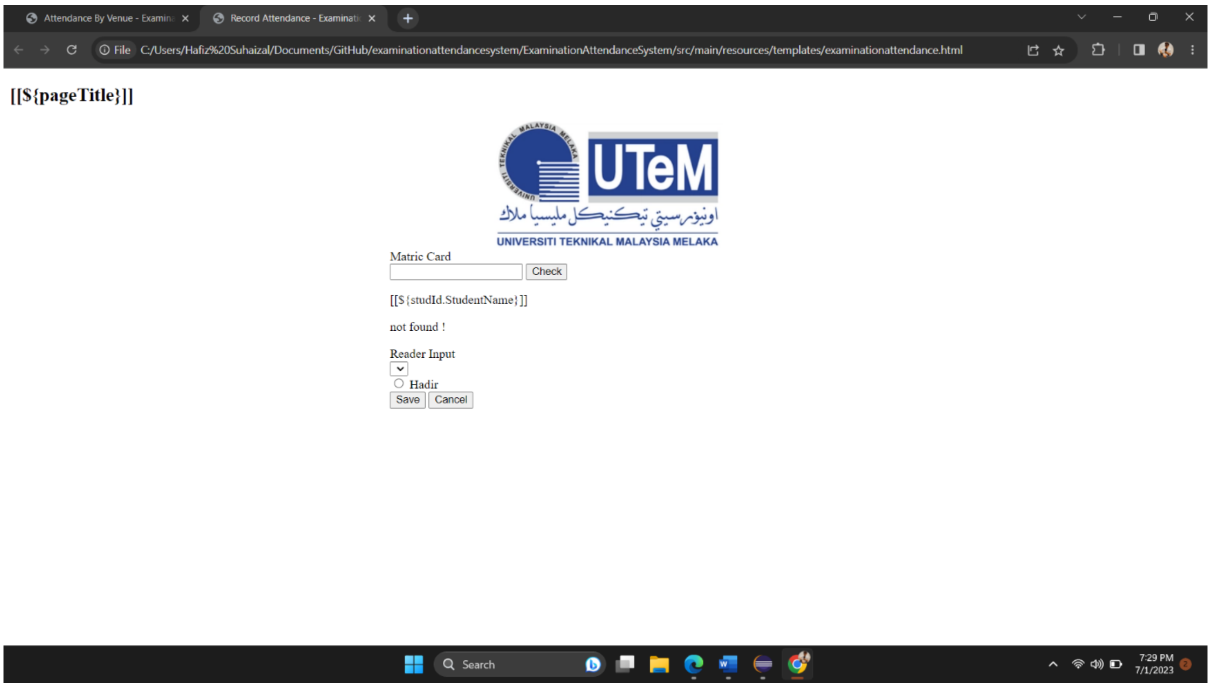
Data processed:

- **Matric Card**: The user enters the matric card number of the student. This data is captured when the form is submitted.
- **Reader Input**: The user selects the type of reader input (MatricCard, Fingerprint, QRCode, or SelfCheck-In) from a dropdown menu. The selected value is captured when the form is submitted.
- **Attendance**: The user selects the attendance status (Hadir) using a radio button. The selected value is captured when the form is submitted.
- **Other data**: The template also includes hidden input fields for `ExamAttendId`, `StudentId.StudentId`, and `Examination.ExaminationId`. These values are captured when the form is submitted.

Once the form is submitted, the data is sent to the server-side endpoint specified in the form's `th:action` attribute (`/examinationattendance/save`) using the HTTP POST method. The server-side code handling this endpoint will further process the data and perform any necessary operations, such as saving the attendance record in a database or performing additional validations.

Screenshot

- Without CSS



[[${pageTitle}]]

Matric Card

[[${studId.StudentName}]]

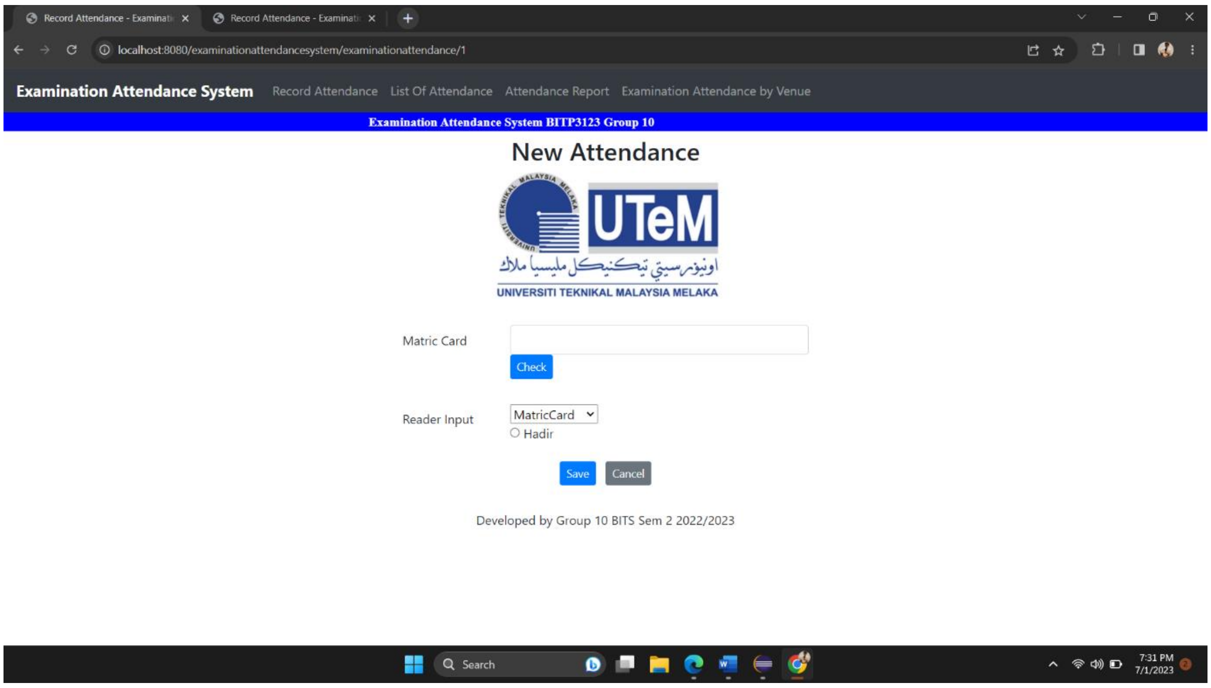
not found !

Reader Input

Hadir

Attendance By Venue - Examin
Record Attendance - Examinati
File C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/examinationattendance.html
Check
Save
Cancel
Search
7:29 PM
7/1/2023

- With CSS



Examination Attendance System BITP3123 Group 10

New Attendance

Matric Card

Reader Input: MatricCard

Hadir

Developed by Group 10 BITS Sem 2 2022/2023

Record Attendance - Examinati
Record Attendance - Examinati
localhost:8080/examinationattendancesystem/examinationattendance/1
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue
Check
Save
Cancel
Search
7:31 PM
7/1/2023

Report.html

head section: contains meta tags for character encoding and viewport settings, as well as the page title.

Header: included using the Thymeleaf `th:replace` attribute, referencing a header fragment defined in another file.

Attendance Report Title: The `<div id="Attendance">` container is used to display the title of the Attendance Report.

Subject Information: A table is used to display the subject code and subject name.The data is obtained from the `studentAttendance` variable, which seems to be a list of student attendance records.The subject code and subject name are dynamically populated using Thymeleaf expressions.

Student Attendance: The total number of students who attended is displayed.If there are students in attendance (`studentAttendance.size() > 0`), a table is generated to display the student details.The student details are obtained from the `studentAttendance` list using Thymeleaf expressions.

Student Absent: The total number of absent students is displayed. If there are absent students (`studentAbsent.size() > 0`), a table is generated to display their details.The absent student details are obtained from the `studentAbsent` list using Thymeleaf expressions.

No Record Found:If there are no absent students (`studentAbsent.size() <= 0`), a message is displayed indicating no records were found.

Print Button: The "Print" button allows the user to print the content of the page.Clicking the button triggers the `printDivContent()` JavaScript function.The function retrieves the HTML content of the "Attendance" and "TABLES" elements, opens a new window, and writes the content into it.The new window is then printed.

Footer: The footer is included using the Thymeleaf `th:replace` attribute, referencing a footer fragment defined in another file.

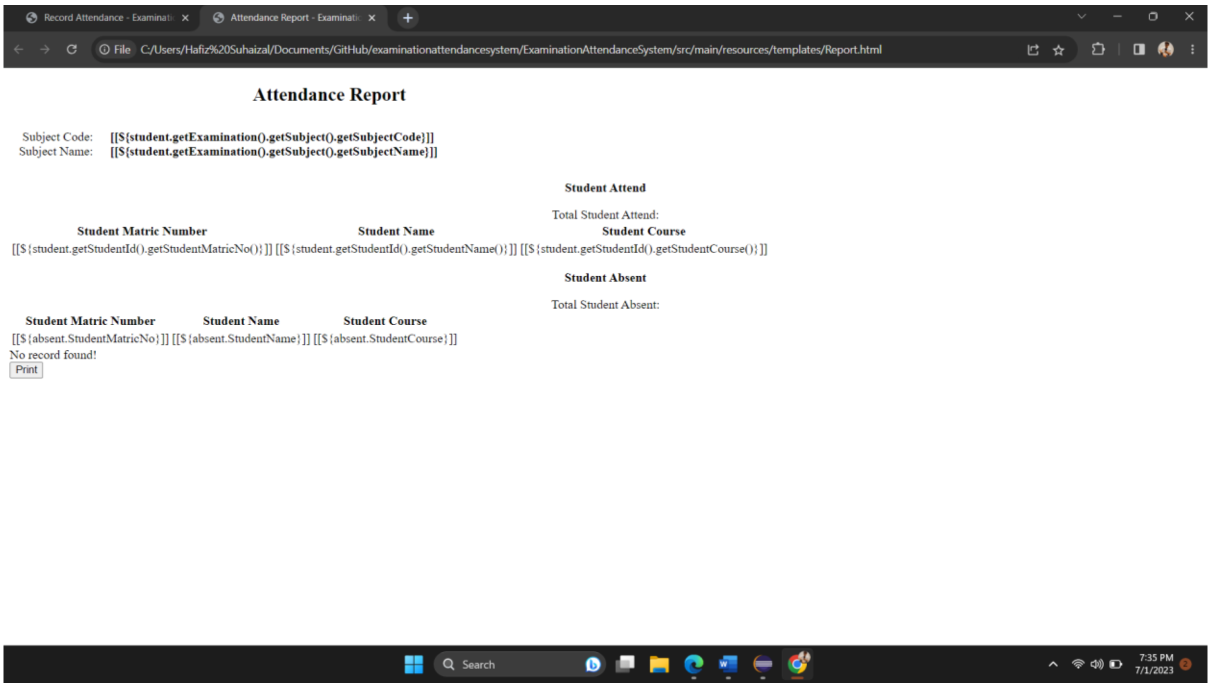
Overall, this code is designed to generate an Attendance Report page with student attendance and absence details. The front-end relies on Thymeleaf expressions to dynamically populate the data from backend variables (`studentAttendance` and `studentAbsent`).

Data Processed:

- **studentAttendance**: This is a list of students who attended the examination. The front end iterates over this list to display the student's matric number, name, and course in a table.
- **studentAbsent**: This is a list of students who were absent from the examination. The front end iterates over this list to display the absent student's matric number, name, and course in a separate table.

Screenshot

- Without CSS



Attendance Report

Subject Code: **[[${student.getExamination().getSubject().getSubjectCode}]]**
Subject Name: **[[${student.getExamination().getSubject().getSubjectName}]]**

Student Attend

Total Student Attend:

Student Matric Number	Student Name	Student Course
[[${student.getStudentId().getStudentMatricNo()}]]	[[${student.getStudentId().getStudentName()}]]	[[${student.getStudentId().getStudentCourse()}]]

Student Absent

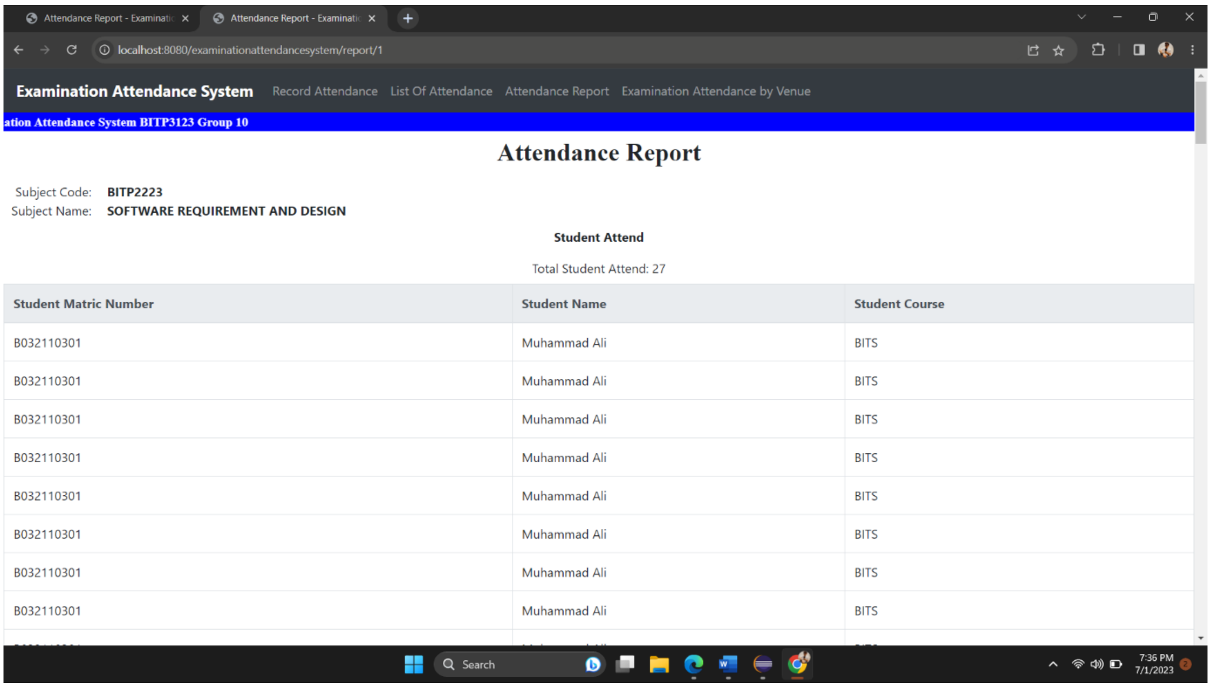
Total Student Absent:

Student Matric Number	Student Name	Student Course
[[${absent.StudentMatricNo}]]	[[${absent.StudentName}]]	[[${absent.StudentCourse}]]

No record found!

Record Attendance - Examinatic
Attendance Report - Examinatic
File C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/Report.html
Print
Search
7:35 PM
7/1/2023

- With CSS


Attendance Report - Examinati
localhost:8080/examinationattendancesystem/report/1
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue
ation Attendance System BITP3123 Group 10

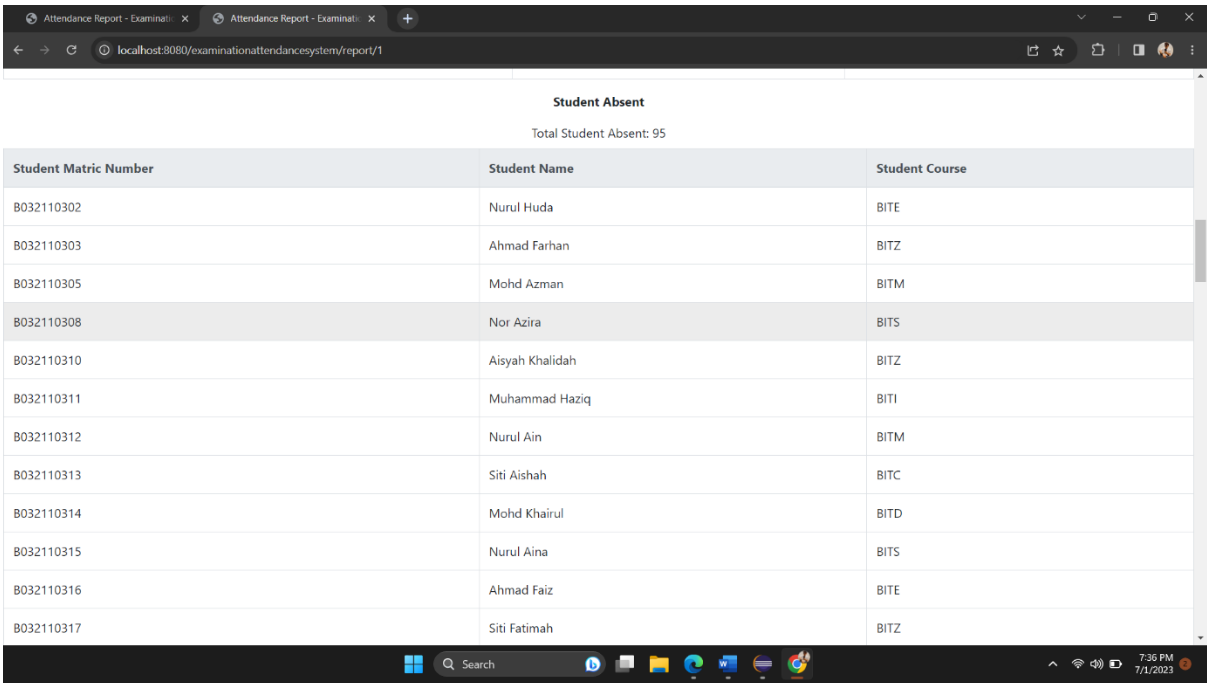
Attendance Report

Subject Code: BITP2223
Subject Name: SOFTWARE REQUIREMENT AND DESIGN

Student Attend

Total Student Attend: 27

Student Matric Number	Student Name	Student Course
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS
B032110301	Muhammad Ali	BITS


Attendance Report - Examinati
localhost:8080/examinationattendancesystem/report/1

Student Absent

Total Student Absent: 95

Student Matric Number	Student Name	Student Course
B032110302	Nurul Huda	BITE
B032110303	Ahmad Farhan	BITZ
B032110305	Mohd Azman	BITM
B032110308	Nor Azira	BITS
B032110310	Aisyah Khalidah	BITZ
B032110311	Muhammad Haziq	BITI
B032110312	Nurul Ain	BITM
B032110313	Siti Aishah	BITC
B032110314	Mohd Khairul	BITD
B032110315	Nurul Aina	BITS
B032110316	Ahmad Faiz	BITE
B032110317	Siti Fatimah	BITZ

schedule.html

Header and Footer: The header and footer sections are included using Thymeleaf fragments. These sections typically contain navigation links, branding, or other common elements.

Examination Schedule Title: The page displays the title "Examination Schedule" in a heading.

Success Message: If a success message is present (variable `message`), it is displayed as a success alert at the top of the page.

Add Schedule Button: The "Add Schedule" button allows users to add a new examination schedule. Clicking on this button redirects the user to a specific URL (`/schedule/0`).

Examination Schedule Table: The table displays the examination schedules (`Examinations`). Each row represents an examination schedule and contains the following columns:

- **Id**: Unique identifier of the schedule
- **Date**: The date of the examination
- **Time**: The time of the examination
- **Subject Code**: The code of the subject
- **Subject Name**: The name of the subject
- **Venue**: The examination venue
- **Action**: Provides options to edit or delete the schedule. Clicking on the edit icon redirects the user to edit the specific schedule. Clicking on the delete icon shows a confirmation modal and allows the user to delete the schedule.

Print Button: The "Print" button allows users to print the content of the page. Clicking on this button opens a new window with the content and triggers the print functionality.

No Record Found: If no examination schedules (`Examinations`) are available, a message "No record found!" is displayed.

Modal: A confirmation modal dialog is shown when the user clicks on the delete icon. It displays a confirmation message and provides "Yes" and "No" buttons to proceed or cancel the deletion, respectively.

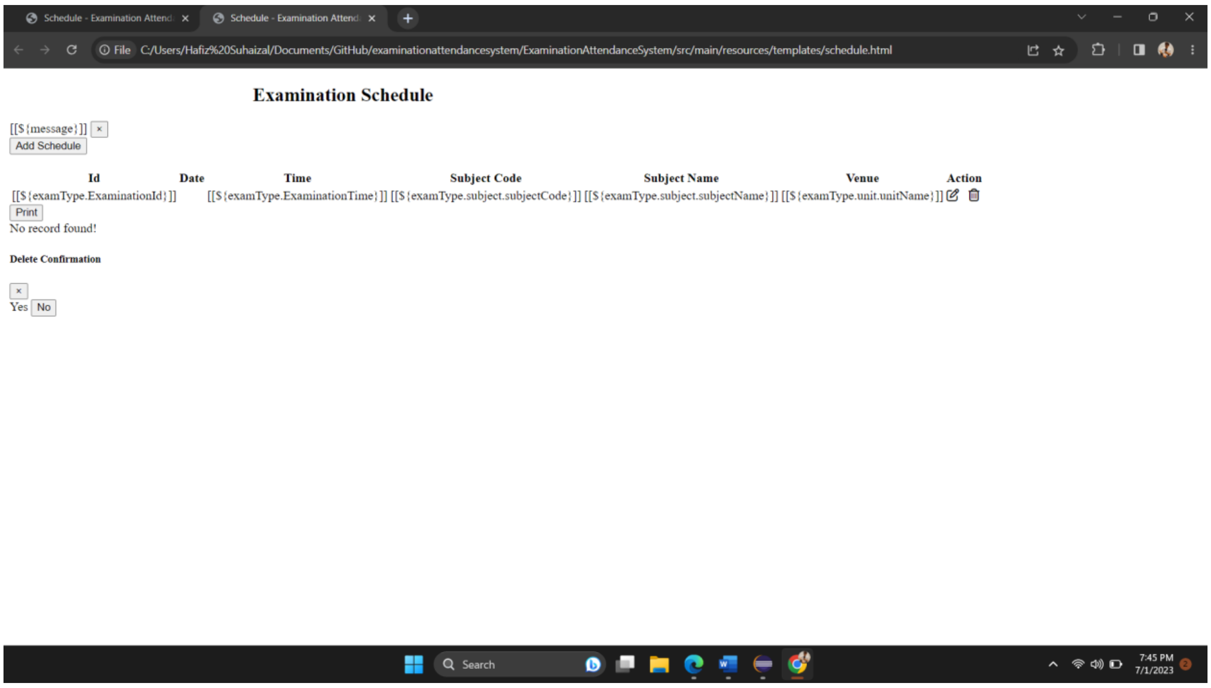
Data Processed:

Examination Schedules:The front end receives a list of examination schedules (`Examinations`) from the server-side code.Each schedule contains the following data:

- **ExaminationId**: Unique identifier of the schedule.
- **ExaminationDate**: The date of the examination.
- **ExaminationTime**: The time of the examination.
- **subject**: Object representing the subject associated with the schedule, containing properties like subjectCode and subjectName.
- **unit**: Object representing the unit associated with the schedule, containing properties like unitName.

Screenshot

- Without CSS


C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/schedule.html

Examination Schedule

[[${message}]] ×
Add Schedule

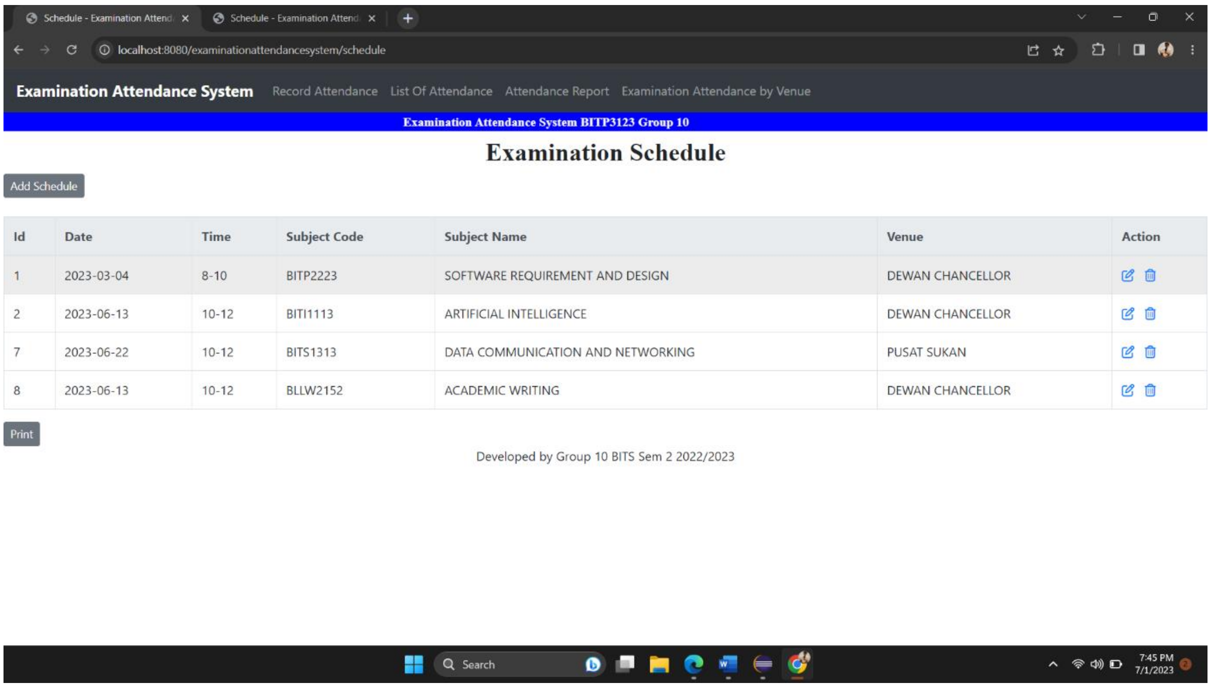
Id	Date	Time	Subject Code	Subject Name	Venue	Action
[[${examType.ExaminationId}]]		[[${examType.ExaminationTime}]]	[[${examType.subject.subjectCode}]]	[[${examType.subject.subjectName}]]	[[${examType.unit.unitName}]]	

Print
No record found!

Delete Confirmation

×
Yes No

- With CSS


Schedule - Examination Attend
Schedule - Examination Attend
localhost:8080/examinationattendancesystem/schedule
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue

Examination Attendance System BITP3123 Group 10

Examination Schedule

Id	Date	Time	Subject Code	Subject Name	Venue	Action
1	2023-03-04	8-10	BITP2223	SOFTWARE REQUIREMENT AND DESIGN	DEWAN CHANCELLOR	
2	2023-06-13	10-12	BITI1113	ARTIFICIAL INTELLIGENCE	DEWAN CHANCELLOR	
7	2023-06-22	10-12	BITS1313	DATA COMMUNICATION AND NETWORKING	PUSAT SUKAN	
8	2023-06-13	10-12	BLLW2152	ACADEMIC WRITING	DEWAN CHANCELLOR	

Developed by Group 10 BITS Sem 2 2022/2023

<u>scheduleinfo.html</u>

The form allows users to enter or select the following information:

- **Date**: The date of the examination.
- **Time**: The time of the examination.
- **Subject**: The subject associated with the schedule. Users can select a subject from a dropdown list.
- **Lecturer**: The lecturer associated with the schedule. Users can select a lecturer from a dropdown list.
- **Unit**: The unit associated with the schedule. Users can select a unit from a dropdown list.

The form also includes buttons to save the schedule and cancel the operation.

Data Processed:

- Subject Data: The form receives a list of subjects (`Subjects`) from the server-side code. Each subject contains the following data:
- SubjectId: Unique identifier of the subject.
- SubjectCode: The code or abbreviation of the subject.

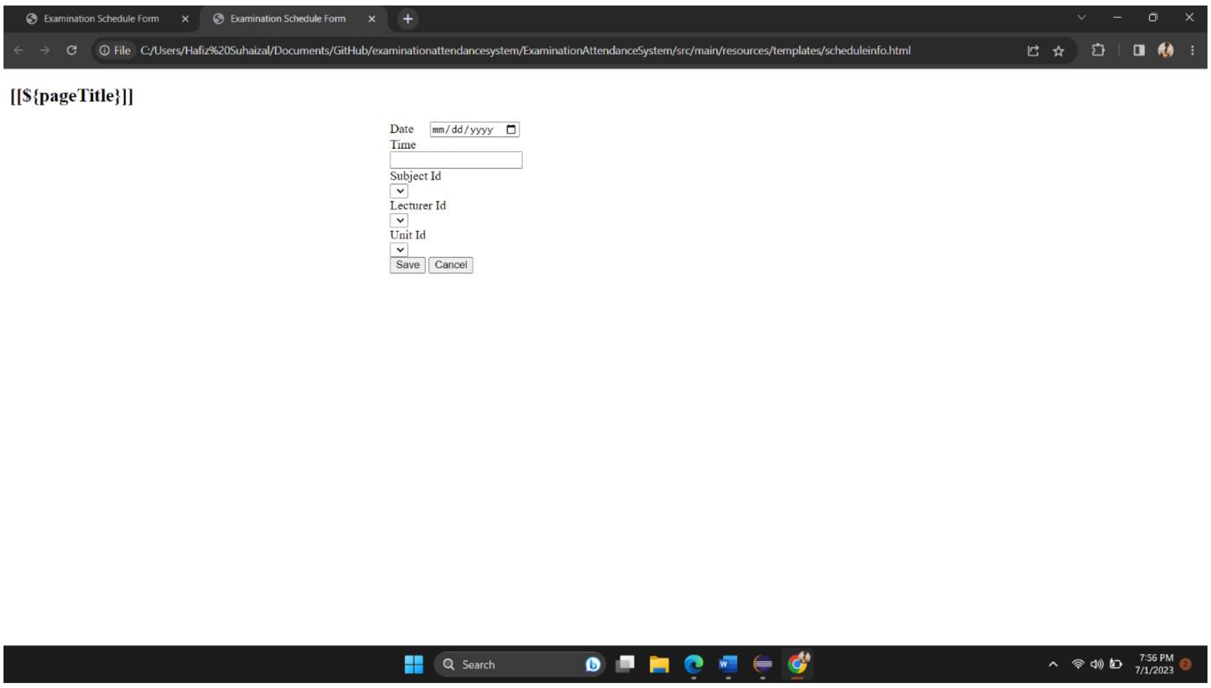
- SubjectName: The name of the subject.
- Lecturer Data: The form receives a list of lecturers (`Lecturers`) from the server-side code. Each lecturer contains the following data:
- LecturerId: Unique identifier of the lecturer.
- LecturerName: The name of the lecturer.
- Unit Data: The form receives a list of units (`examUnit`) from the server-side code. Each unit contains the following data:
- UnitId: Unique identifier of the unit.
- UnitName: The name of the unit.

The front end processes this data to populate the respective dropdown lists in the form, allowing users to select subjects, lecturers, and units associated with the examination schedule.

When the form is submitted, the selected values for date, time, subject, lecturer, and unit are sent back to the server-side code for further processing, such as saving the examination schedule.

Screenshot

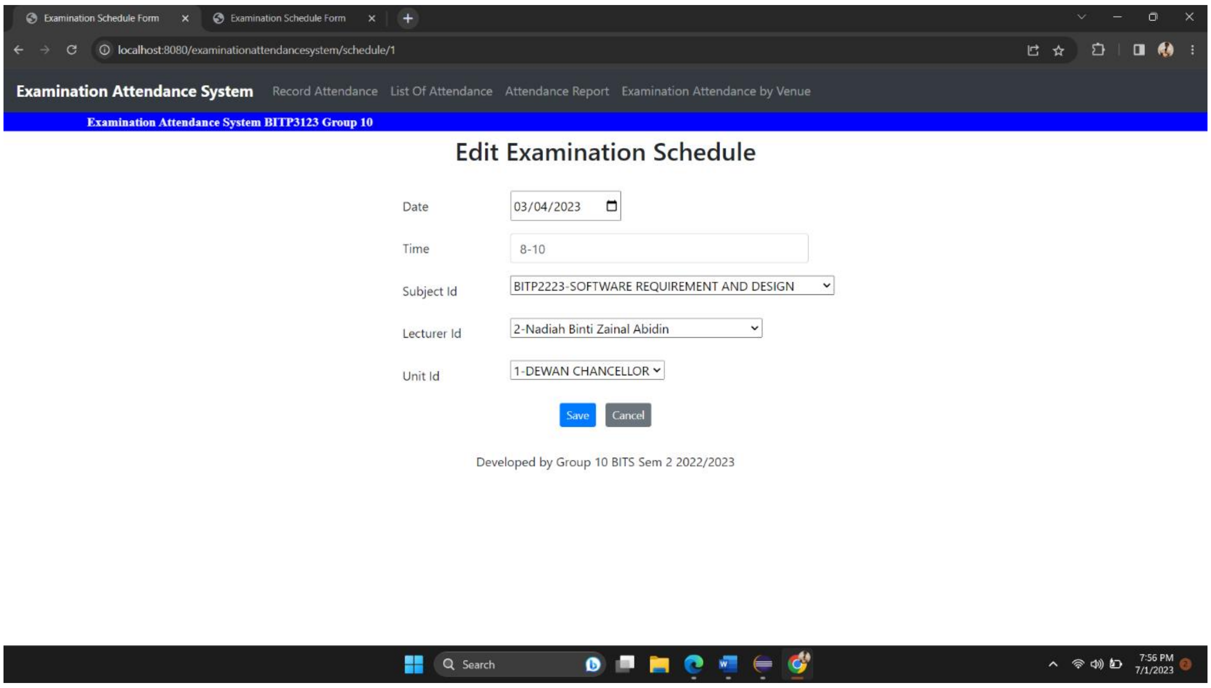
- Without CSS


Examination Schedule Form
Examination Schedule Form
File C:/Users/Hafiz%20Suhaizal/Documents/GitHub/examinationattendancesystem/ExaminationAttendanceSystem/src/main/resources/templates/scheduleinfo.html
Search
7:56 PM
7/1/2023

[[${pageTitle}]]

Date mm/dd/yyyy
Time
Subject Id
Lecturer Id
Unit Id
Save Cancel

- With CSS



Examination Attendance System BITP3123 Group 10

Edit Examination Schedule

Date	03/04/2023
Time	8-10
Subject Id	BITP2223-SOFTWARE REQUIREMENT AND DESIGN
Lecturer Id	2-Nadiah Binti Zainal Abidin
Unit Id	1-DEWAN CHANCELLOR

Developed by Group 10 BITS Sem 2 2022/2023

Examination Schedule Form
localhost:8080/examinationattendancesystem/schedule/1
Examination Attendance System
Record Attendance
List Of Attendance
Attendance Report
Examination Attendance by Venue
Save
Cancel
Search
7:56 PM
7/1/2023