



**General Sir John Kotelawala Defence University**

Faculty of Computing

Department of IT

**Service Oriented Web Programming (IT 6023)**

**INSTRUCTOR ALLOCATION SYSTEM**

LEVEL 6 - 2016

**Student Declaration:**

*I declare  
that:*

- *I understand what is meant by plagiarism*
- *The implications of plagiarism have been explained to me by my institution*
- *This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.*

Student's signature:

Date: 20.10.2016

Total number of pages including this cover page			63
Submission Date	20.10.2016	Due Date	20.10.2016
Student's ID	ICT / 14 / 051	Class Code	ICT_INTAKE XXXI
Student's Full Name	JKC Shayalika		
Lecturer's Name	Mr. TMKK Jinasena		

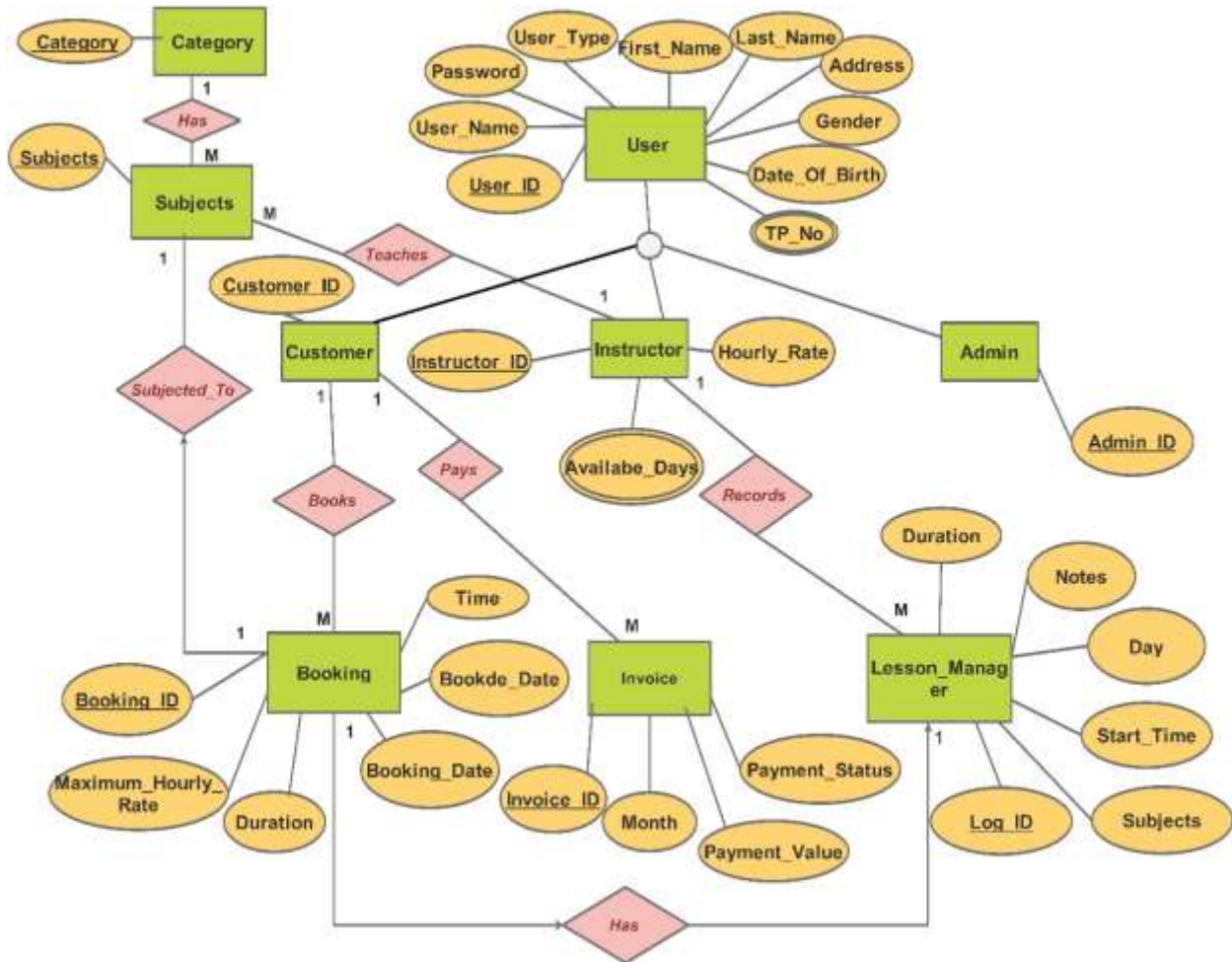
**OFFICIAL USE ONLY**

MARKER'S COMMENTS	
Marker's Name	Marks Awarded (100%)

## Deliverables

### 1) Database

#### ER Diagram



## Relational Mapping

### 1) Mapping regular entity types

User (User\_ID, User\_Name, Password, First\_Name, Last\_Name, Address, Gender, DOB, TP\_No)

- User (User\_ID, User\_Name, Password, First\_Name, Last\_Name, Address, Gender, DOB)
- User\_TP (User\_ID, TP\_No)

Instructor (Instructor\_ID, Available\_Days, Hourly\_Rate)

- Instructor (Instructor\_ID, Hourly\_Rate)
- Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

### 2) Mapping regular entity types

There are no weak entity types.

### 3) Mapping of Binary (one to one relationship)

Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date)

Lesson\_Manager (Log\_ID, Start\_Time, End\_Time, Duration, Notes)

- Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date)
- Lesson\_Manager (Log\_ID, Start\_Time, End\_Time, Duration, Notes, Booking\_ID(fk))

### 4) Mapping of Binary (1-M relationship)

Instructor (Instructor\_ID, Hourly\_Rate)

Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date, Customer\_ID(fk))

- Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date, Instructor\_ID(fk))

Instructor (Instructor\_ID, Hourly\_Rate)

Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

Invoice (Invoice\_ID, Month, Payment\_Value, Payment\_Status)

- Invoice (Invoice\_ID, Month, Payment\_Value, Payment\_Status, Instructor\_ID(fk))

Customer (Customer\_ID)

Invoice (Invoice\_ID, Month, Payment\_Value, Payment\_Status, Instructor\_ID(fk))

- Lesson\_Manager (Log\_ID, Start\_Time, End\_Time, Duration, Notes, Booking\_ID(fk), Instructor\_ID(fk))

- Invoice (Invoice\_ID, Month, Payment\_Value, Payment\_Status, Instructor\_ID(fk), Customer\_ID(fk))

Instructor (Instructor\_ID, Hourly\_Rate)

Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

Lesson\_Manager (Log\_ID, Start\_Time, End\_Time, Duration, Notes, Booking\_ID(fk))

Instructor (Instructor\_ID, Hourly\_Rate)

Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

Subjects (Subjects)

- Subjects (Subjects, Instructor\_ID(fk))

Subjects (Subjects, Instructor\_ID(fk))

Category (Category)

- Subjects (Subjects, Instructor\_ID(fk), Category(fk))

Subjects (Subjects, Instructor\_ID(fk), Category(fk))

Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date, Instructor\_ID(fk))

- Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date, Instructor\_ID(fk), Subjects(fk))

### 5) Mapping of Binary (M-N relationship)

There are no M-N relationships.

### 6) Mapping of Unary relationship

There are no Unary relationships.

### 7) Mapping of Ternary (N-ary)

There are no Ternary relationships.

### 8) Mapping of Super / Subtype relationship

User (User\_ID, User\_Name, Password, First\_Name, Last\_Name, Address, Gender, DOB)

User\_TP (User\_ID, TP\_No)

Customer (Customer\_ID)

Instructor (Instructor\_ID, Hourly\_Rate)

Instructor\_Available\_Days (Instructor\_ID, Available\_Days)

Admin (Admin\_ID)

- User (User\_ID, User\_Name, Password, First\_Name, Last\_Name, Address, Gender, DOB)

- User\_TP (User\_ID, TP\_No)
- Customer (Customer\_ID, User\_ID)
- Instructor (Instructor\_ID, Hourly\_Rate, User\_ID)
- Instructor\_Available\_Days (Instructor\_ID, Available\_Days)
- Admin (Admin\_ID, User\_ID)

### Final Table Set

- User (User\_ID, User\_Name, Password, First\_Name, Last\_Name, Address, Gender, DOB)
- User\_TP (User\_ID, TP\_No)
- Booking (Booking\_ID, Maximum\_hourly\_rate, Duration, Booking\_Date, Time, Booked\_Date, Instructor\_ID(fk), Subjects(fk))
- Invoice (Invoice\_ID, Month, Payment\_Value, Payment\_Status, Instructor\_ID(fk), Customer\_ID(fk))
- Lesson\_Manager (Log\_ID, Start\_Time, End\_Time, Duration, Notes, Booking\_ID(fk), Instructor\_ID(fk))
- Customer (Customer\_ID, User\_ID)
- Instructor (Instructor\_ID, Hourly\_Rate, User\_ID)
- Instructor\_Available\_Days (Instructor\_ID, Available\_Days)
- Admin (Admin\_ID, User\_ID)
- Category (Category)
- Subjects (Subjects, Instructor\_ID(fk), Category(fk))

## MySQL Database Query

```
-- phpMyAdmin SQL Dump
-- version 4.1.14
-- http://www.phpmyadmin.net
--
-- Host: 127.0.0.1
-- Generation Time: Oct 19, 2016 at 05:22 PM
-- Server version: 5.1.52-community
-- PHP Version: 5.5.12

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `sowpassignmentnew`
--

--
-- -----
--
-- Table structure for table `admin`
--

CREATE TABLE IF NOT EXISTS `admin` (
  `Admin_ID` int(11) NOT NULL AUTO_INCREMENT,
  `User_ID` int(11) NOT NULL,
  PRIMARY KEY (`Admin_ID`),
  KEY `User_ID` (`User_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

--
-- Dumping data for table `admin`
--

INSERT INTO `admin` (`Admin_ID`, `User_ID`) VALUES
(1, 4),
(2, 11);

--
-- -----
--
-- Table structure for table `booking`
--

CREATE TABLE IF NOT EXISTS `booking` (
  `Booking_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Subjects` varchar(255) NOT NULL,
```

```

`Booking_Date` datetime NOT NULL,
`Booked_Date` datetime NOT NULL,
`Maximum_Hourly_Rate` decimal(10,0) NOT NULL,
`Start_Time` time NOT NULL,
`Duration` decimal(10,0) NOT NULL,
`CustomerID` int(11) NOT NULL,
`Day` varchar(255) NOT NULL,
PRIMARY KEY (`Booking_ID`),
KEY `CustomerID` (`CustomerID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=8 ;

--
-- Dumping data for table `booking`
--

INSERT INTO `booking` (`Booking_ID`, `Subjects`, `Booking_Date`,
`Booked_Date`, `Maximum_Hourly_Rate`, `Start_Time`, `Duration`, `CustomerID`,
`Day`) VALUES
(1, 'Chemistry', '2016-10-07 10:00:00', '2016-09-27 08:17:21', '788',
'08:00:00', '3', 1, 'Tuesday'),
(2, 'Chemistry', '2016-10-08 09:30:00', '2016-10-18 00:00:00', '150',
'11:00:00', '2', 2, 'Tuesday'),
(3, 'Chemistry', '2016-10-09 09:47:00', '2016-10-22 00:00:00', '200',
'18:00:00', '1', 1, 'Saturday'),
(7, 'Western Music', '2016-10-17 07:19:00', '2016-10-27 00:00:00', '300',
'12:00:00', '4', 1, 'Thursday');

-- -----

--
-- Table structure for table `category`
--

CREATE TABLE IF NOT EXISTS `category` (
  `Category` varchar(255) NOT NULL,
  PRIMARY KEY (`Category`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `category`
--

INSERT INTO `category` (`Category`) VALUES
('A/L'),
('Aesthetics'),
('Arts'),
('Commerce'),
('Management'),
('O/L'),
('Other'),
('Primary'),
('Technical Subjects');

-- -----

```

```

-- Table structure for table `customer`
--

CREATE TABLE IF NOT EXISTS `customer` (
  `Customer_ID` int(11) NOT NULL AUTO_INCREMENT,
  `User_ID` int(11) NOT NULL,
  PRIMARY KEY (`Customer_ID`),
  KEY `User_ID` (`User_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

--
-- Dumping data for table `customer`
--

INSERT INTO `customer` (`Customer_ID`, `User_ID`) VALUES
(1, 2),
(2, 3),
(3, 9),
(4, 10);

-----

--
-- Table structure for table `instructor`
--

CREATE TABLE IF NOT EXISTS `instructor` (
  `Instructor_ID` int(11) NOT NULL AUTO_INCREMENT,
  `User_ID` int(255) NOT NULL,
  `Hourly_Rate` decimal(10,0) NOT NULL,
  PRIMARY KEY (`Instructor_ID`),
  KEY `User_ID` (`User_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;

--
-- Dumping data for table `instructor`
--

INSERT INTO `instructor` (`Instructor_ID`, `User_ID`, `Hourly_Rate`) VALUES
(1, 1, '2000'),
(2, 5, '2000'),
(3, 6, '300'),
(4, 7, '200'),
(5, 8, '200');

-----

--
-- Table structure for table `instructor_available_days`
--

CREATE TABLE IF NOT EXISTS `instructor_available_days` (
  `Instructor_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Available_Days` varchar(255) NOT NULL,
  PRIMARY KEY (`Instructor_ID`)

```



```

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;

--
-- Dumping data for table `instructor_available_days`
--

INSERT INTO `instructor_available_days` (`Instructor_ID`, `Available_Days`)
VALUES
(1, 'Tuesday,Wednesday'),
(2, 'Tuesday'),
(3, 'Thursday'),
(4, 'Wednesday'),
(5, 'Wednesday');

-----

--
-- Table structure for table `invoice`
--

CREATE TABLE IF NOT EXISTS `invoice` (
  `Invoice_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Customer_ID` int(11) NOT NULL,
  `Month` varchar(255) NOT NULL,
  `Payment_Value` decimal(10,0) NOT NULL,
  `Payment_Status` varchar(100) NOT NULL,
  PRIMARY KEY (`Invoice_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

--
-- Dumping data for table `invoice`
--

INSERT INTO `invoice` (`Invoice_ID`, `Customer_ID`, `Month`, `Payment_Value`,
`Payment_Status`) VALUES
(1, 1, 'August', '1000', 'Paid'),
(2, 2, 'August', '2000', 'Paid'),
(3, 1, 'September', '1250', 'Not Paid');

-----

--
-- Table structure for table `lessonmanager`
--

CREATE TABLE IF NOT EXISTS `lessonmanager` (
  `Log_ID` int(11) NOT NULL,
  `Start_Time` varchar(200) NOT NULL,
  `Duration` varchar(200) NOT NULL,
  `Notes` varchar(255) NOT NULL,
  `Instructor_ID` varchar(255) NOT NULL,
  PRIMARY KEY (`Log_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

```

```

-- Dumping data for table `lessonmanager`
--

INSERT INTO `lessonmanager` (`Log_ID`, `Start_Time`, `Duration`, `Notes`,
`Instructor_ID`) VALUES
(1, '4', '1', 'Conducted Successfully', '1'),
(2, '3', '2', 'Conducted Successfully', '2'),
(3, '7', '11', 'Conducted', '3'),
(4, '3', '3', 'Success', '1'),
(8, '5', '2', 'Not good', '5'),
(9, '3', '1', 'Successfull', '1'),
(10, '7', '2', 'Successfull lecture', '1'),
(11, '6', '1', 'null', '3');

-- -----

--
-- Table structure for table `lessonmanagerapp`
--

CREATE TABLE IF NOT EXISTS `lessonmanagerapp` (
  `Log_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Booking_ID` int(11) NOT NULL DEFAULT '0',
  `Start_Time` time NOT NULL DEFAULT '00:00:00',
  `Duration` decimal(10,0) NOT NULL DEFAULT '0',
  `Day` varchar(100) NOT NULL DEFAULT '0',
  `Subject` varchar(100) NOT NULL DEFAULT '0',
  `Instructor_ID` int(11) NOT NULL DEFAULT '0',
  `Note` varchar(255) NOT NULL DEFAULT '0',
  PRIMARY KEY (`Log_ID`),
  KEY `Booking_ID` (`Booking_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=8 ;

--
-- Dumping data for table `lessonmanagerapp`
--

INSERT INTO `lessonmanagerapp` (`Log_ID`, `Booking_ID`, `Start_Time`,
`Duration`, `Day`, `Subject`, `Instructor_ID`, `Note`) VALUES
(1, 0, '00:00:00', '0', '0', '0', 2, '0'),
(2, 0, '00:00:00', '0', '0', '0', 3, '0'),
(4, 1, '08:00:00', '3', 'Tuesday', 'Chemistry', 0, '0'),
(5, 2, '11:00:00', '2', 'Tuesday', 'Chemistry', 0, '0'),
(6, 3, '18:00:00', '1', 'Saturday', 'Chemistry', 0, '0'),
(7, 7, '12:00:00', '4', 'Thursday', 'Western Music', 0, '0');

-- -----

--
-- Table structure for table `schedule`
--

CREATE TABLE IF NOT EXISTS `schedule` (
  `Schedule_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Booking_ID` int(11) NOT NULL,

```

```

`Instructor_ID` int(11) NOT NULL,
`Subject` varchar(255) NOT NULL,
`Booked_Date` datetime NOT NULL,
`Start_Time` time NOT NULL,
`Duration` decimal(10,0) NOT NULL,
`Customer_ID` int(11) NOT NULL,
`Day` varchar(255) NOT NULL,
PRIMARY KEY (`Schedule_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-- -----

--
-- Table structure for table `subjectcategory`
--

CREATE TABLE IF NOT EXISTS `subjectcategory` (
  `Subjects` varchar(100) NOT NULL,
  `Category` varchar(100) NOT NULL,
  PRIMARY KEY (`Subjects`,`Category`),
  KEY `Subjects` (`Subjects`),
  KEY `Category` (`Category`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `subjectcategory`
--

INSERT INTO `subjectcategory` (`Subjects`, `Category`) VALUES
('BC', 'Arts'),
('Biology', 'A/L'),
('Business Studies', 'Management'),
('Chemistry', 'A/L'),
('Combined Maths', 'A/L'),
('Eastern Music', 'Aesthetics'),
('Scholarship', 'Primary'),
('Sinhala', 'Arts'),
('Western Music', 'Aesthetics');

-- -----

--
-- Table structure for table `subjects`
--

CREATE TABLE IF NOT EXISTS `subjects` (
  `Subjects` varchar(255) NOT NULL,
  `Instructor_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Category` varchar(255) NOT NULL,
  PRIMARY KEY (`Subjects`,`Instructor_ID`),
  KEY `Instructor_ID` (`Instructor_ID`,`Category`),
  KEY `Category` (`Category`),
  KEY `Instructor_ID_2` (`Instructor_ID`),
  KEY `Subjects` (`Subjects`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;

```

```
--
-- Dumping data for table `subjects`
--

INSERT INTO `subjects` (`Subjects`, `Instructor_ID`, `Category`) VALUES
('Biology', 5, 'A/L'),
('Chemistry', 2, 'A/L'),
('Eastern Music,Western Music', 1, 'Aesthetics'),
('Western Music', 3, 'Aesthetics'),
('BC', 4, 'Arts');

-----

--
-- Table structure for table `user`
--
CREATE TABLE IF NOT EXISTS `user` (
  `User_ID` int(11) NOT NULL AUTO_INCREMENT,
  `User_Name` varchar(100) NOT NULL,
  `Password` varchar(100) NOT NULL,
  `User_Type` varchar(100) NOT NULL,
  `First_Name` varchar(100) NOT NULL,
  `Last_Name` varchar(100) NOT NULL,
  `Address` varchar(100) NOT NULL,
  `Gender` varchar(100) NOT NULL,
  `Date_Of_Birth` date NOT NULL,
  `TP_No` varchar(100) NOT NULL,
  PRIMARY KEY (`User_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=12 ;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`User_ID`, `User_Name`, `Password`, `User_Type`,
`First_Name`, `Last_Name`, `Address`, `Gender`, `Date_Of_Birth`, `TP_No`)
VALUES
(1, 'chathurangijks@gmail.com', 'sc', 'Instructor', 'Chathurangi', 'Shyalika',
'No,57, Madelgamuwa, Gampaha', 'Female', '1998-02-02', '0772150269'),
(2, 'chamanijks2@gmail.com', 'cd', 'Customer', 'Chamani', 'Shiranthika',
'No56, Silva Rd, Moratuwa', 'Female', '1008-02-02', '0111234561'),
(3, 'jayakodydw@gmail.com', 's', 'Customer', 'Jaya', 'Sampath',
'No56,289Road', 'Male', '9988-02-09', '0774567891'),
(4, 'dwjayakody@gmail.com', 'd', 'Administrator', 'Dayawansa', 'Jayakody',
'No56, Silva Rd, Moratuwa', 'Male', '1980-12-02', '0779886575'),
(5, 'd@gmail.com', 'da', 'Instructor', 'Damith', 'Sumathipala', 'No 45,
Nedagamuwa, Gampaha', 'Male', '1978-02-11', '0774567891'),
(6, 'pradeepa@yahoo.com', 'mw', 'Instructor', 'Pradeepa', 'Satharasinghe',
'No.45, Asgiriya, Gampaha', 'Female', '2015-12-10', '0332230981'),
(7, 'sagara@gmail.com', 'pw', 'Instructor', 'Sagara', 'Palansuriya', 'No.5,
Kandy Rd', 'Female', '2016-10-18', '0772150260'),
(8, 'dil@gmail.com', 'k', 'Instructor', 'Dilruwan', 'Senadeera', 'No.56,
Palama Rd, Asgiriya', 'Female', '2016-10-11', '0772150234'),
```

```
(9, 'gimi@yahoo.com', 'gm', 'Customer', 'Gimhani', 'Hangawaththa', 'No.23
Samagi Mw, Piliyandala', 'Female', '2016-10-10', '0725037410'),
(10, 'sri@gmail.com', 's', 'Customer', 'Srimali', 'Manchanayaka', 'No 89,
Ihala Imbulgoda', 'Female', '2016-10-05', '0725048961'),
(11, 'kamal@hotmail.com', 'km', 'Administrator', 'Kamal', 'Jayakody', 'No.
23/A, Panadura', 'Male', '2016-10-10', '0779883572');
```

```
-- -----
```

```
--
-- Table structure for table `user_tp`
--
```

```
CREATE TABLE IF NOT EXISTS `user_tp` (
  `User_ID` int(11) NOT NULL AUTO_INCREMENT,
  `TP_No` varchar(100) NOT NULL,
  PRIMARY KEY (`User_ID`,`TP_No`),
  KEY `TP_No` (`TP_No`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```
--
-- Constraints for dumped tables
--
```

```
--
-- Constraints for table `admin`
--
```

```
ALTER TABLE `admin`
  ADD CONSTRAINT `admin_ibfk_1` FOREIGN KEY (`User_ID`) REFERENCES `user`
  (`User_ID`);
```

```
--
-- Constraints for table `customer`
--
```

```
ALTER TABLE `customer`
  ADD CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`User_ID`) REFERENCES `user`
  (`User_ID`);
```

```
--
-- Constraints for table `instructor`
--
```

```
ALTER TABLE `instructor`
  ADD CONSTRAINT `instructor_ibfk_1` FOREIGN KEY (`User_ID`) REFERENCES `user`
  (`User_ID`);
```

```
--
-- Constraints for table `instructor_available_days`
--
```

```
ALTER TABLE `instructor_available_days`
  ADD CONSTRAINT `instructor_available_days_ibfk_1` FOREIGN KEY
  (`Instructor_ID`) REFERENCES `instructor` (`Instructor_ID`);
```

```
--
-- Constraints for table `subjectcategory`
--
```

```

ALTER TABLE `subjectcategory`
  ADD CONSTRAINT `subjectcategory_ibfk_1` FOREIGN KEY (`Category`) REFERENCES
`category` (`Category`);

--
-- Constraints for table `subjects`
--
ALTER TABLE `subjects`
  ADD CONSTRAINT `subjects_ibfk_1` FOREIGN KEY (`Instructor_ID`) REFERENCES
`instructor` (`Instructor_ID`),
  ADD CONSTRAINT `subjects_ibfk_2` FOREIGN KEY (`Category`) REFERENCES
`category` (`Category`);

--
-- Constraints for table `user_tp`
--
ALTER TABLE `user_tp`
  ADD CONSTRAINT `user_tp_ibfk_1` FOREIGN KEY (`User_ID`) REFERENCES `user`
(`User_ID`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

## 2) Web Application

### Customer Interfaces

#### Customer Login



The image shows the 'Customer Login' interface for 'E-Explore Online E-Learning Academy'. The background features a stylized illustration of trees with lightbulbs hanging from them. The login form is a white box with the title 'Customer Login' and a 'Register' button in the top right corner. It contains two input fields: 'UserName' with a placeholder 'username' and 'Password' with a placeholder 'password'. A blue 'LOGIN' button is positioned below the password field.

#### Registering a new Customer



The image shows the 'Customer Details' registration form for 'E-Explore Online E-Learning Academy'. The header includes a graduation cap icon, the title 'Customer Details', and the subtitle 'Register A Customer' next to a stack of books and a rolled diploma. The form is a white box with a red asterisk and the text '\* required fields' at the top left. It contains five input fields, each with a red asterisk on the right: 'User Type' (a dropdown menu with the placeholder '---Select User Type---'), 'First Name' (placeholder 'First Name'), 'Last Name' (placeholder 'Last Name'), 'Address' (placeholder 'Address'), and 'Contact Number' (placeholder 'Contact Number').

Email (This is your User Name)  
Enter a Password  
Confirm Password  
Gender  
Date Of Birth  
Category  
Select Subject /s

☒ Male ☐ Female  
  

---

Select the Category---

▼

Select a category to view Subject info here...

Available Days

Monday  
Tuesday  
Wednesday  
Thursday

Hourly Rate

Date

Add User

## Booking Lessons



Book A Lesson

Booking Section



Log out

\* required fields

Customer ID

Subject

Maximum Hourly Rate

Start Time

---

Select Your ID---

▼

---

Select the Subject---

BC  
Biology  
Business Studies

▼

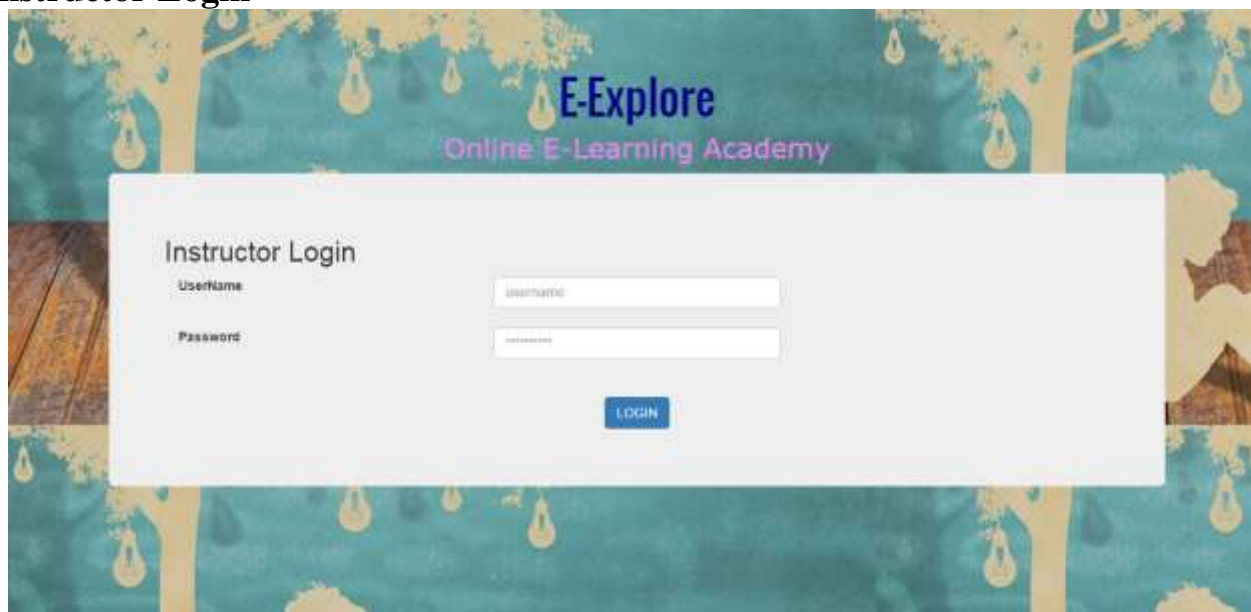




A form for booking a lesson. It contains three input fields: 'Booking Date' with the value '10/00/2016', 'Booked Date' with the placeholder 'mm/dd/yyyy', and 'Duration' with the placeholder 'Duration'. A blue 'Book Lesson' button is located at the bottom right of the form area.

## Instructor Interfaces

### Instructor Login



The 'Instructor Login' form is centered on a background featuring stylized trees and light bulbs. The form has a title 'Instructor Login' and two input fields: 'Username' and 'Password'. A blue 'LOGIN' button is positioned below the password field.

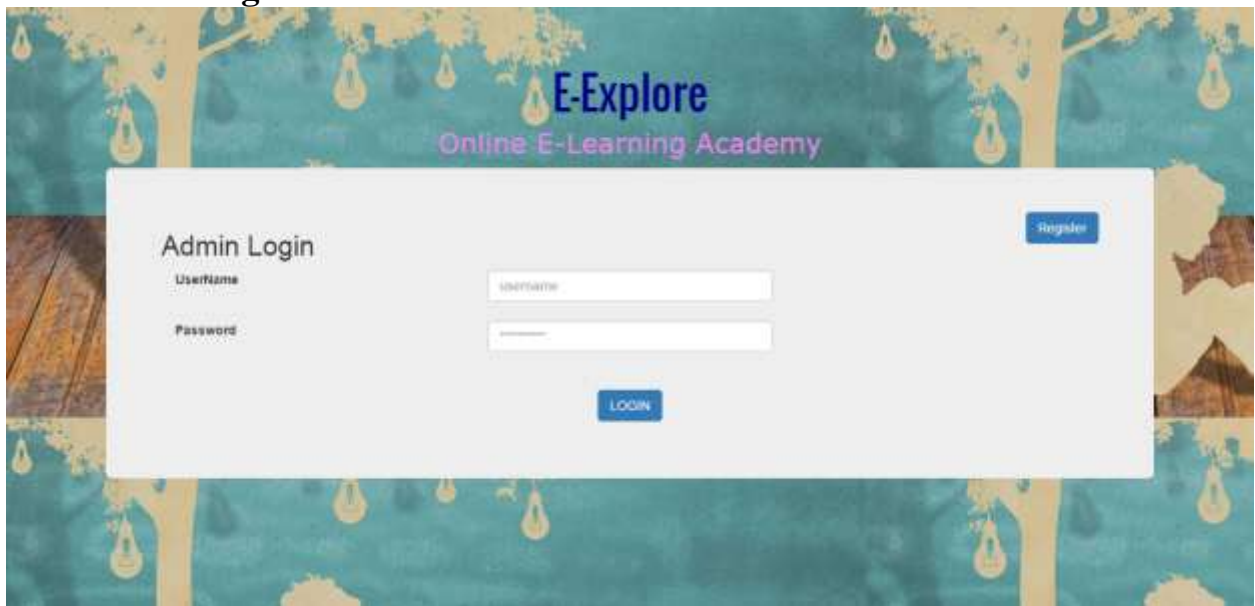
### Viewing Lesson Manager Details

Scheduled Lessons as at 2016/10/18 04:46:53 pm Log out

Log ID	Booking ID	Start Time	Duration	Day	Subject	Instructor ID	Note
1	0	00:00:00	0	0	0	2	0
2	0	00:00:00	0	0	0	3	0
4	1	08:00:00	3	Tuesday	Chemistry	0	0
5	2	11:00:00	2	Tuesday	Chemistry	0	0
6	3	18:00:00	1	Saturday	Chemistry	0	0
7	7	12:00:00	4	Thursday	Western Music	0	0

## Admin Interfaces

### Administrator Login



The image shows the 'Admin Login' interface for 'E-Explore Online E-Learning Academy'. The background features a stylized illustration of trees with lightbulbs hanging from them. The login form is a white box with the following elements:

- Admin Login** (Section Header)
- Register** (Blue button in the top right corner)
- Username** label and a corresponding text input field.
- Password** label and a corresponding text input field.
- LOGIN** (Blue button at the bottom center)

### Registering of Another administrators



The image shows the 'Admin Details' interface for 'Register An Administrator'. The background is light blue with a graduation cap icon on the left and a stack of books with a graduation cap on the right. The registration form is a white box with the following elements:

- Admin Details** (Section Header)
- Register An Administrator** (Section Header)
- \* required fields** (Red text label)
- User Type** label and a dropdown menu with the option '—Select User Type—'.
- First Name** label and a corresponding text input field.
- Last Name** label and a corresponding text input field.
- Address** label and a corresponding text input field.
- Contact Number** label and a corresponding text input field.

Email (This is your User Name):

Enter a Password:

Confirm Password:

Gender: ☒ Male ☐ Female

Date Of Birth:

Category:

Select Subject /s:

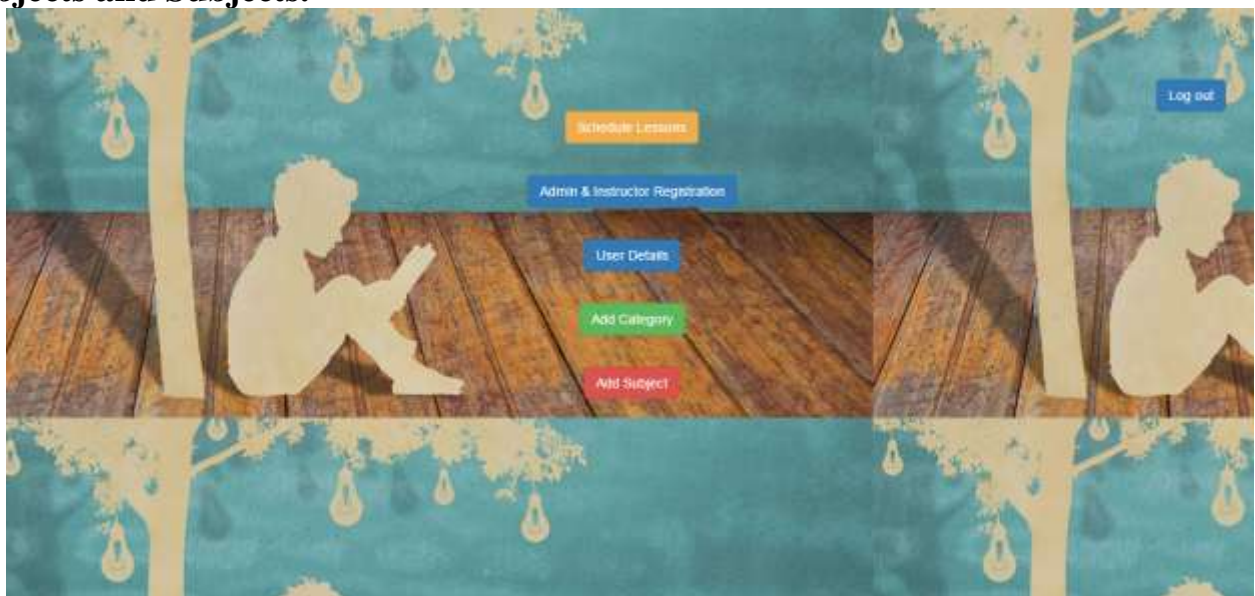
Available Days:

Hourly Rate:

Date:

[Add User](#)

**Admin Panel- Admin can schedule lessons, Register Instructors and another administrators for the system, View, Update User details, Add Categories of subjects and Subjects.**



## Scheduling Lessons by the Admin



The interface features a light blue header with a graduation cap icon on the left, the title "Schedule Lessons" in large blue font, the subtitle "Assign the schedule" in pink, and a "Log out" button on the right. Below the header is a white form area. On the left of the form, it says "\* required fields" in red, followed by "Schedule for the week starting from :". To the right of this text is a date input field containing "10/09/2016". On the right side of the form, there is a green "View All Schedules" button and a blue "Generate Schedule" button.

**Schedule Lessons**  
Assign the schedule

\* required fields  
Schedule for the week starting from : 10/09/2016

View All Schedules  
Generate Schedule

## Success message after scheduling lessons for a given week



This interface is identical to the one above, but it includes a success message at the top of the white form area: "New Schedule created successfully". The rest of the form, including the date input field with "10/09/2016" and the "View All Schedules" and "Generate Schedule" buttons, remains the same.

**Schedule Lessons**  
Assign the schedule

New Schedule created successfully

\* required fields  
Schedule for the week starting from : 10/09/2016

View All Schedules  
Generate Schedule

## View Scheduled Lessons

Scheduled Lessons as at 2016/10/19 05:42:00 pm							
Log ID	Booking ID	Start Time	Duration	Day	Subject	Instructor ID	Note
1	0	00:00:00	0	0	0	2	0
2	0	00:00:00	0	0	0	3	0
4	1	09:00:00	3	Tuesday	Chemistry	0	0
5	2	11:00:00	2	Tuesday	Chemistry	0	0
6	3	18:00:00	1	Saturday	Chemistry	0	0
7	7	12:00:00	4	Thursday	Western Music	0	0

## View, Update User details

User Details as at 2016/10/19 05:42:35 pm							
User Name	Password	FName	LName	Address	Gender	DOB	TNo
chathurangika@gmail.com	sc	sc	Shiyalka	No.57, Madelgamuwa, Gampaha	Female	1996-02-02	0772150269
chamanika2@gmail.com	cd	cd	Shranthika	No56, Silva Rd, Moratuwa	Female	1006-02-02	0111234561
jayakodya@gmail.com	s	s	Sampath	No56 255Road	Male	9955-02-09	0774567891
dajayakody@gmail.com	d	d	Jayakody	No56, Silva Rd, Moratuwa	Male	1980-12-02	0779686575
o@gmail.com	da	da	Sumathipala	No 40, Nedagamulla, Gampaha	Male	1978-02-11	0774567891
pradeepa@yahoo.com	mw	mw	Sathrasinghe	No 45, Agriya, Gampaha	Female	2015-12-10	0332230981
sagara@gmail.com	pw	pw	Paransuriya	No 5, Kandy Rd	Female	2016-10-18	0772150260
di@gmail.com	k	k	Senadeera	No 56, Palama Rd, Agriya	Female	2016-10-11	0772150234
gms@yahoo.com	gm	gm	Hangawaththa	No 23 Samagi Ma, Pitiyandala	Female	2016-10-10	0725037410
ln@gmail.com	s	s	Manchanayaka	No 89, thala ambukoda	Female	2016-10-05	0725048961
kamal@hotmail.com	km	km	Jayakody	No. 23/A, Panadura	Male	2016-10-10	0775683572

## Add Categories



### Category Details

Add A Category



\* required fields

Category Name \*

[View Category List](#)

[Add Category](#)

## View Categories



### Category Details

Add A Category



Category Names
A/L
Aesthetics
Arts
Commerce
Management
Q/L
Other
Technical Subjects

[View Category List](#)

## Add Subjects



### Subjects Details

Add A Subject



View Subjects List

\* required fields

Category \*

-----Select the Category-----

Subject Name \*

Subject Name

Add Subject

## View Subjects



### Subjects Details

Add A Subject



Category Name	Subject Name
Arts	BC
Av/L	Biology
Management	Business Studies
Av/L	Chemistry
Av/L	Combined Maths
Aesthetics	Eastern Music
Arts	Sinhala
Aesthetics	Western Music

View Subjects List



## View & Edit subjects and categories in button click

Subjects Online as at 2016/10/19 05:57:29 pm [Back To Admin Panel](#) [Log out](#)

Category	Subjects	
Arts	BC	<a href="#">Edit</a>
A/L	Biology	<a href="#">Edit</a>
Arts	Buddhist Culture	<a href="#">Edit</a>
Management	Business Studies	<a href="#">Edit</a>
A/L	Chemistry	<a href="#">Edit</a>
A/L	Combined Maths	<a href="#">Edit</a>
Aesthetics	Eastern Music	<a href="#">Edit</a>
Primary	Scotiabp	<a href="#">Edit</a>
Arts	Simata	<a href="#">Edit</a>
Aesthetics	Western Music	<a href="#">Edit</a>

## Data Validation

\* required fields

User Type	<input type="text" value="—Select User Type—"/>	* User Type is required
First Name	<input type="text" value="First Name"/>	* First Name is required
Last Name	<input type="text" value="Last Name"/>	* Last Name is required
Address	<input type="text" value="Address"/>	* Address is required
Contact Number	<input type="text" value="Contact Number"/>	* Contact Number is required
Email (This is your User Name)	<input type="text" value="user@gmail.com"/>	* Email is required
Enter a Password	<input type="text" value="Password"/>	* Password is required
Confirm Password	<input type="text" value="Confirm Password"/>	* Password is required
Gender	<input checked="" type="radio"/> Male <input type="radio"/> Female	* Gender is required
Date Of Birth	<input type="text" value="mm/dd/yyyy"/>	* Date of birth is required
Category	<input type="text" value="—Select the Category—"/>	* Category is are required. ignore Error if You Are An Administrator



Select Subject is	Select a category to view Subject info here...	* Subject/s are required. Ignore Error if You Are An Administrator!
Available Days	None selected ▾	* Available day /s are required. Ignore Error if You Are An Administrator!
Hourly Rate	Rate	* Rate is required. Ignore Error if You Are An Administrator!
Date	10/00/2016	* Date is required

## User Authentication

Customer Login

Register

UserName	dejayakody@gmail.com
Password	

LOGIN

Username or Password is invalid!

### 3) SOAP Web Service

#### Methods in WebService1.asmx

```
[WebMethod]
public string GetLesson(int Instructor_ID)
{
    String sql = "Select * from lessonmanagerapp";
    DataSet dataset = conObj.execQuery(sql);
    StringWriter sw = new StringWriter();
    dataset.WriteXml(sw, XmlWriteMode.IgnoreSchema);
    string xmlResult = sw.ToString();
    return xmlResult;
}

[WebMethod]
public string GetLessonMgrDetails(int Booking_ID)
{
    String sql = "Select * from lessonmanagerapp where Booking_ID =' " +
Booking_ID + " ' ";
    DataSet dataset = conObj.execQuery(sql);
    StringWriter sw = new StringWriter();
    dataset.WriteXml(sw, XmlWriteMode.IgnoreSchema);
    string xmlResult = sw.ToString();
    return xmlResult;
}

[WebMethod]
public string UploadNotes(string Start_Time, decimal duration, string note)
{
    string conetionString = null;
    MySqlConnection connection;
    MySqlCommand command;
    MySqlDataAdapter adpter = new MySqlDataAdapter();
    DataSet ds = new DataSet();
    XmlReader xmlFile;
    string sql = null;
    string filePath = "";

    string StartTime = "";
    string Duration = "";
    string Note = "";
    string InstructorID = "";

    conetionString = "Database=sowpassignmentnew; Data Source=localhost; User
Id=root; Password=123";

    connection = new MySqlConnection(conetionString);
    connection.Open();

    xmlFile = XmlReader.Create("D:\\LessonManagerApp.xml", new
XmlReaderSettings());
    ds.ReadXml(xmlFile);
}
```

```

int i = 0;

XmlDocument xml = new XmlDocument();
filePath = @"D:\\LessonManagerApp.xml";
xml.Load(filePath);

int count = xml.SelectNodes("Users/User").Count;

StartTime = xml.SelectSingleNode("Users/User[last()]/StartTime").InnerText;
Duration = xml.SelectSingleNode("Users/User[last()]/Duration").InnerText;
Note = xml.SelectSingleNode("Users/User[last()]/Notes").InnerText;
InstructorID =
xml.SelectSingleNode("Users/User[last()]/Instructor_ID").InnerText;

sql = "insert into lessonmanager values('" + (count) + "','" + StartTime +
"', '" + Duration + "','" + Note + "','" + InstructorID + "')";

command = new MySqlCommand(sql, connection);
adpter.InsertCommand = command;
adpter.InsertCommand.ExecuteNonQuery();
connection.Close();

StringWriter sw = new StringWriter();
string xmlResult = sw.ToString();
return xmlResult;
}

```

## Result of GetLesson method



## Result of GetLessonMgrDetails- Returns the booking details according to the booking id.



#### 4) Lesson Manager Application for Instructors

The screenshot shows a window titled "Form1" with a large gray rectangular area at the top. Below this area are two buttons: "View All" and "Select from Instructor". Further down, there are three input fields: "Start Time", "Duration", and "Notes". To the right of these fields is a dropdown menu labeled "Instructor ID". An "Upload" button is located at the bottom right of the form.

**“View All” Button- returns all the lessons booked so far**

The screenshot shows the same window "Form1" but with a table of data displayed in the large gray area. The table has columns: Log\_ID, Booking\_ID, Start\_Time, Duration, Subject, Instructor\_ID, and Note. The "View All" button is highlighted with a blue border. The "Instructor ID" dropdown menu is set to "0".

Log_ID	Booking_ID	Start_Time	Duration	Subject	Instructor_ID	Note
1	0	PT0S	0	0	2	0
2	0	PT0S	0	0	3	0
4	1	PT8H	3	Chemistry	0	0
5	2	PT11H	2	Chemistry	0	0
6	3	PT18H	1	Chemistry	0	0
7	7	PT12H	4	Western Music	0	0
*						

**“Select from Instructor” button-returns lessons according to particular Instructor**

The screenshot shows a Windows application window titled "Form1". Inside the window, there is a table with the following columns: Log\_ID, Booking\_ID, Start\_Time, Duration, Subject, Instructor\_ID, and Note. The first row of data is highlighted in blue and contains the values: 5, 2, PT11H, 2, Chemistry, 0, and 0. Below the table, there is a large grey rectangular area. At the bottom of the window, there is a form with two buttons: "View All" and "Select from Instructor". The "Select from Instructor" button is highlighted with a blue border. Below these buttons, there are input fields for "Start Time", "Duration", and "Notes". To the right of these fields is a dropdown menu for "Instructor ID". An "Upload" button is located at the bottom right of the form.

Log_ID	Booking_ID	Start_Time	Duration	Subject	Instructor_ID	Note
5	2	PT11H	2	Chemistry	0	0
*						

View All    Select from Instructor

Start Time    Instructor ID

Duration

Notes    Upload

**Saving details of a particular lesson**

The screenshot shows a form for saving lesson details. It has three rows of input fields: "Start Time" with the value "7", "Duration" with the value "2", and "Notes" with the value "Successfull lecture". To the right of the "Start Time" field is a dropdown menu for "Instructor ID" with the value "1". An "Upload" button is located at the bottom right of the form.

Start Time    Instructor ID

Duration

Notes    Upload

## Message when Data is Uploaded Successfully to the lesson manager

The screenshot shows a Windows-style window titled "Form1". Inside, there is a table with the following data:

	Log_ID	Booking_ID	Start_Time	Duration	Subject	Instructor_ID	Note
▶	5	2	PT11H	2	Chemistry	0	0
*							

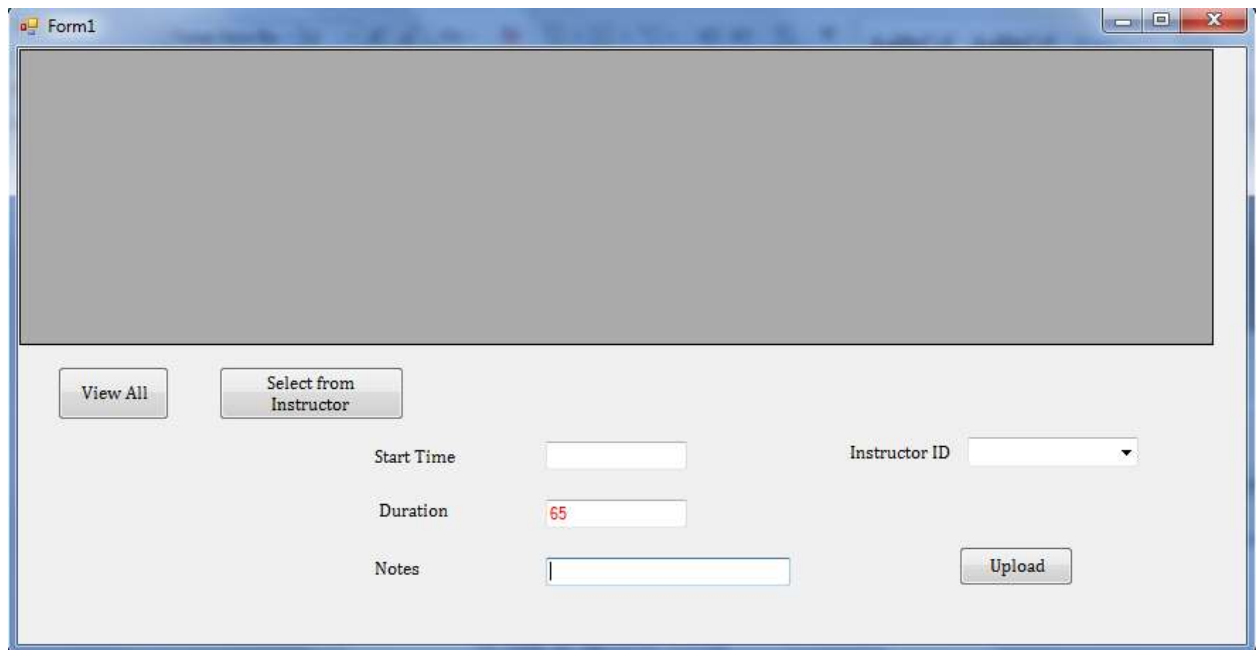
Below the table, there are two buttons: "View All" and "Select from Instructor". To the right, there are input fields for "Start Time" (value: 7), "Duration" (value: 2), and "Notes" (value: Successfull lecture). There is also a dropdown for "Instructor ID" (value: 1) and an "Upload" button. A small dialog box is overlaid on the form, titled "Uploaded Successfully", with an "OK" button.

## 5) Extended Textbox Component

Adding " Duration < 61 "

The screenshot shows the same "Form1" window. The "Duration" input field now has the value "60" and a yellow warning icon to its left, indicating a constraint. The "Notes" field is empty. The "Upload" button is still present.

Adding “ Duration > 61 “



### Code of CustomControl1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ClassLibrary1
{
    public partial class CustomControl1 : TextBox
    {
        private int limit;

        public int Limit
        {
            get { return limit; }
            set { limit = value; }
        }

        public CustomControl1()
        {
            InitializeComponent();
            this.TextChanged += new EventHandler(textBox1_TextChanged);
        }

        protected override void OnPaint(PaintEventArgs pe)
        {
            base.OnPaint(pe);
        }
    }
}
```

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    //MessageBox.Show("Hi");
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    int val = Convert.ToInt32(this.Text);
    if (val > limit)
        this.ForeColor = Color.Red;
    else
        this.ForeColor = Color.Black;
}
}
```



## 6) RESTful Web Service

### C# Code of Service1.svc.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfService2
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class
    // name "Service1" in code, svc and config file together.
    // NOTE: In order to launch WCF Test Client for testing this service, please select
    // Service1.svc or Service1.svc.cs at the Solution Explorer and start debugging.
    public class Service1 : IService1
    {
        dbconn conObj = new dbconn();

        public string GetData(int value)
        {
            return string.Format("You entered: {0}", value);
        }

        public List<Booking> allBooking()
        {
            string query = "SELECT * FROM booking";
            DataSet ds = conObj.execQuery(query);
            DataTable dt = new DataTable();
            dt = ds.Tables[0];
            List<Booking> result = new List<Booking>();

            foreach (DataRow dr in dt.Rows)
            {
                Booking bk = new Booking
                {
                    Booking_ID = dr["Booking_ID"].ToString(),
                    Subjects = dr["Subjects"].ToString(),
                    Booking_Date = dr["Booking_Date"].ToString(),
                    Booked_Date = dr["Booked_Date"].ToString(),
                    Maximum_Hourly_Rate = dr["Maximum_Hourly_Rate"].ToString(),
                    Start_Time = dr["Start_Time"].ToString(),
                    Duration = dr["Duration"].ToString(),
                    CustomerID = dr["CustomerID"].ToString(),
                    Day = dr["Day"].ToString(),
                };
                result.Add(bk);
            }
            return result;
        }
    }
}
```

```

public List<Booking> allBookingCustomer(string customerID)
{
    string query = "SELECT * FROM booking where CustomerID = '" + customerID +
    """;

    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<Booking> result = new List<Booking>();

    foreach (DataRow dr in dt.Rows)
    {
        Booking bk = new Booking
        {
            Booking_ID = dr["Booking_ID"].ToString(),
            Booking_Date = dr["Booking_Date"].ToString(),
            Maximum_Hourly_Rate = dr["Maximum_Hourly_Rate"].ToString(),
            Duration = dr["Duration"].ToString(),
            Subjects = dr["Subjects"].ToString(),
            CustomerID = dr["CustomerID"].ToString(),
            Start_Time = dr["Start_Time"].ToString(),
            Day = dr["Day"].ToString()
        };
        result.Add(bk);
    }
    return result;
}

public List<user> allCus()
{
    string query = "SELECT * FROM user";
    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<user> result = new List<user>();

    foreach (DataRow dr in dt.Rows)
    {
        user us = new user
        {
            UserID = dr["User_ID"].ToString(),
            UserName = dr["User_Name"].ToString(),
            UserType = dr["User_Type"].ToString(),
            First_name = dr["First_Name"].ToString(),
            Last_name = dr["Last_Name"].ToString(),
            Address = dr["Address"].ToString(),
            Gender = dr["Gender"].ToString(),
            Dob=dr["Date_Of_Birth"].ToString(),
            Tp = dr["TP_No"].ToString(),
        };
        result.Add(us);
    }
    return result;
}

public List<user> allCusType(string userType)
{
    string query = "SELECT * FROM user where User_Type = '" + userType + "'";
    DataSet ds = conObj.execQuery(query);

```

```

DataTable dt = new DataTable();
dt = ds.Tables[0];
List<user> result = new List<user>();

foreach (DataRow dr in dt.Rows)
{
    user us = new user
    {
        UserID = dr["User_ID"].ToString(),
        UserName = dr["User_Name"].ToString(),
        UserType = dr["User_Type"].ToString(),
        First_name = dr["First_Name"].ToString(),
        Last_name = dr["Last_Name"].ToString(),
        Address = dr["Address"].ToString(),
        Gender = dr["Gender"].ToString(),
        Dob = dr["Date_Of_Birth"].ToString(),
        Tp = dr["TP_No"].ToString(),
    };
    result.Add(us);
}
return result;
}

public List<user> allCusTypeEmail(string userType, string email)
{
    string query = "SELECT * FROM user where User_Type = '" + userType + "'and
User_Name= '" + email + "'";
    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<user> result = new List<user>();

    foreach (DataRow dr in dt.Rows)
    {
        user us = new user
        {
            UserID = dr["User_ID"].ToString(),
            UserName = dr["User_Name"].ToString(),
            UserType = dr["User_Type"].ToString(),
        };
        result.Add(us);
    }
    return result;
}

public List<Invoice> allInvoice()
{
    string query = "SELECT * FROM invoice";
    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<Invoice> result = new List<Invoice>();

    foreach (DataRow dr in dt.Rows)
    {
        Invoice inv = new Invoice
        {
            Invoice_ID = dr["Invoice_ID"].ToString(),

```

```

        Customer_ID = dr["Customer_ID"].ToString(),
        Month = dr["Month"].ToString(),
        Payment_Value = dr["Payment_Value"].ToString(),
        Payment_Status = dr["Payment_Status"].ToString(),
    };
    result.Add(inv);
}
return result;
}

public List<Invoice> allInvoiceCus(string customerID)
{
    string query = "SELECT * FROM invoice where Customer_ID = '" + customerID +
    """;

    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<Invoice> result = new List<Invoice>();

    foreach (DataRow dr in dt.Rows)
    {
        Invoice inv = new Invoice
        {
            Invoice_ID = dr["Invoice_ID"].ToString(),
            Customer_ID = dr["Customer_ID"].ToString(),
            Month = dr["Month"].ToString(),
            Payment_Value = dr["Payment_Value"].ToString(),
            Payment_Status = dr["Payment_Status"].ToString(),
        };
        result.Add(inv);
    }
    return result;
}

public List<Invoice> allInvoiceCusMonth(string customerID, string month)
{
    string query = "SELECT * FROM invoice where Customer_ID = '" + customerID +
    "'and Month= '" + month + """;
    DataSet ds = conObj.execQuery(query);
    DataTable dt = new DataTable();
    dt = ds.Tables[0];
    List<Invoice> result = new List<Invoice>();

    foreach (DataRow dr in dt.Rows)
    {
        Invoice inv = new Invoice
        {
            Invoice_ID = dr["Invoice_ID"].ToString(),
            Customer_ID = dr["Customer_ID"].ToString(),
            Month = dr["Month"].ToString(),
            Payment_Value = dr["Payment_Value"].ToString(),
            Payment_Status = dr["Payment_Status"].ToString(),
        };
        result.Add(inv);
    }
    return result;
}
}
}

```

## **C# Code of IService1.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfService2
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
    // interface name "IService1" in both code and config file together.
    [ServiceContract]
    public interface IService1
    {

        [OperationContract]
        string GetData(int value);

        [OperationContract]
        [WebGet(UriTemplate = "/booking")]
        List<Booking> allBooking();

        [OperationContract]
        [WebGet(UriTemplate = "/booking/{customerID}")]
        List<Booking> allBookingCustomer(string customerID);

        [OperationContract]
        [WebGet(UriTemplate = "/user")]
        List<user> allCus();

        [OperationContract]
        [WebGet(UriTemplate = "/user/{userType}")]
        List<user> allCusType(string userType);

        [OperationContract]
        [WebGet(UriTemplate = "/user/{userType}/{email}")]
        List<user> allCusTypeEmail(string userType, string email);

        [OperationContract]
        [WebGet(UriTemplate = "/invoice")]
        List<Invoice> allInvoice();

        [OperationContract]
        [WebGet(UriTemplate = "/invoice/{customerID}")]
        List<Invoice> allInvoiceCus(string customerID);

        [OperationContract]
        [WebGet(UriTemplate = "/invoice/{customerID}/{month}")]
        List<Invoice> allInvoiceCusMonth(string customerID, string month);
        // TODO: Add your service operations here
    }

    // Use a data contract as illustrated in the sample below to add composite types to
    // service operations.
    [DataContract]
```

```

public class CompositeType
{
    bool boolValue = true;
    string stringValue = "Hello ";

    [DataMember]
    public bool BoolValue
    {
        get { return boolValue; }
        set { boolValue = value; }
    }

    [DataMember]
    public string StringValue
    {
        get { return stringValue; }
        set { stringValue = value; }
    }
}

[DataContract]
public class customer
{
    string customerID = "C001";
    string username = "DefaultName";
    string email = "DefaultEmail";

    [DataMember]
    public string CustomerID
    {
        get { return customerID; }
        set { customerID = value; }
    }

    [DataMember]
    public string UserName
    {
        get { return username; }
        set { username = value; }
    }

    [DataMember]
    public string Email
    {
        get { return email; }
        set { email = value; }
    }
}

[DataContract]
public class user
{
    string userID = "001";
    string userName = "DefaultName";
    string userType = "DefaultType";
    string first_name = "DefaultFname";
    string last_name = "DefaultLname";
}

```

```

string address = "DefaultAdd";
string gender = "DefaultGender";
string dob = "Defaultdob";
string tp = "DefaultTP";

```

```

[DataMember]
public string UserID
{
    get { return userID; }
    set { userID = value; }
}

```

```

[DataMember]
public string UserName
{
    get { return userName; }
    set { userName = value; }
}

```

```

[DataMember]
public string UserType
{
    get { return userType; }
    set { userType = value; }
}

```

```

[DataMember]
public string First_name
{
    get { return first_name; }
    set { first_name = value; }
}

```

```

[DataMember]
public string Last_name
{
    get { return last_name; }
    set { last_name = value; }
}

```

```

[DataMember]
public string Address
{
    get { return address; }
    set { address = value; }
}

```

```

[DataMember]
public string Gender
{
    get { return gender; }
    set { gender = value; }
}

```

```

[DataMember]
public string Tp
{
    get { return tp; }
}

```

```

        set { tp = value; }
    }

    [DataMember]
    public string Dob
    {
        get { return dob; }
        set { dob = value; }
    }
}

[DataContract]
public class instructor
{
    string instructorID = "I001";
    string hourly_rate = "DefaultRate";

    [DataMember]
    public string InstructorID
    {
        get { return instructorID; }
        set { instructorID = value; }
    }

    [DataMember]
    public string Hourly_rate
    {
        get { return hourly_rate; }
        set { hourly_rate = value; }
    }
}

[DataContract]
public class Booking
{
    string booking_ID = "C001";
    string subjects = "Sinhala";
    string year = "2016";
    string month = "sep";
    string date = "22";
    string booking_Date = "DefaultDate";
    string maximum_Hourly_Rate = "DefaultMonth";
    string duration = "DefaultMonth";
    string customerID = "C001";
    string day = "";
    string start_Time = "";
    string instructor_ID="";

    public string Day
    {
        get { return day; }
        set { day = value; }
    }
}

```



```

public string Start_Time
{
    get { return start_Time; }
    set { start_Time = value; }
}

[DataMember]
public string Booking_ID
{
    get { return booking_ID; }
    set { booking_ID = value; }
}

[DataMember]
public string Instructor_ID
{
    get { return instructor_ID; }
    set { instructor_ID = value; }
}

[DataMember]
public string CustomerID
{
    get { return customerID; }
    set { customerID = value; }
}

[DataMember]
public string Booking_Date
{
    get { return booking_Date; }
    set { booking_Date = value; }
}

[DataMember]
public string Duration
{
    get { return duration; }
    set { duration = value; }
}

[DataMember]
public string Booked_Date
{
    get { return booking_Date; }
    set { booking_Date = value; }
}

[DataMember]
public string Subjects
{
    get { return subjects; }
    set { subjects = value; }
}

[DataMember]
public string Maximum_Hourly_Rate

```

```

        {
            get { return maximum_Hourly_Rate; }
            set { maximum_Hourly_Rate = value; }
        }
    }

[DataContract]
public class Invoice
{
    string invoice_ID = "";
    public string Invoice_ID
    {
        get { return invoice_ID; }
        set { invoice_ID = value; }
    }

    string customer_ID = "";
    public string Customer_ID
    {
        get { return customer_ID; }
        set { customer_ID = value; }
    }

    string month = "sep";
    public string Month
    {
        get { return month; }
        set { month = value; }
    }

    string payment_Value = "";
    public string Payment_Value
    {
        get { return payment_Value; }
        set { payment_Value = value; }
    }

    string payment_Status = "";
    public string Payment_Status
    {
        get { return payment_Status; }
        set { payment_Status = value; }
    }
}
}

```

### Code of dbconn.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using MySql.Data.MySqlClient;
using MySql.Data;
using System.Data;

namespace WcfService2
{
    public class dbconn
    {
        string connectionString = null;
        public MySqlConnection con;

        public dbconn()
        {
            connectionString = "Database=sowpassassignmentnew; Data Source=localhost; User
Id=root; Password=123";
            try
            {
                con = new MySqlConnection(connectionString);
                con.Open();
                Console.WriteLine("Connection Opened");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Connection Open failed");
            }
        }

        public DataSet execQuery(string query)
        {
            DataSet results = new DataSet();
            try
            {
                MySqlDataAdapter sql = new MySqlDataAdapter(query, connectionString);
                sql.Fill(results);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error : " + ex);
                System.Diagnostics.Debug.Write("Error : " + ex);
            }
            return results;
        }

        public int execNonQuery(string query)
        {
            int condition = -1;
            try
            {
                MySqlCommand cmd1 = new MySqlCommand(query, con);
                condition = cmd1.ExecuteNonQuery();
            }
        }
    }
}
```

```

    }
    catch (Exception ex)
    {
        Console.WriteLine("Error :" + ex);
        System.Diagnostics.Debug.Write("Error :" + ex);
    }
    finally
    {
        con.Close();
    }

    return condition;
}
}
}

```

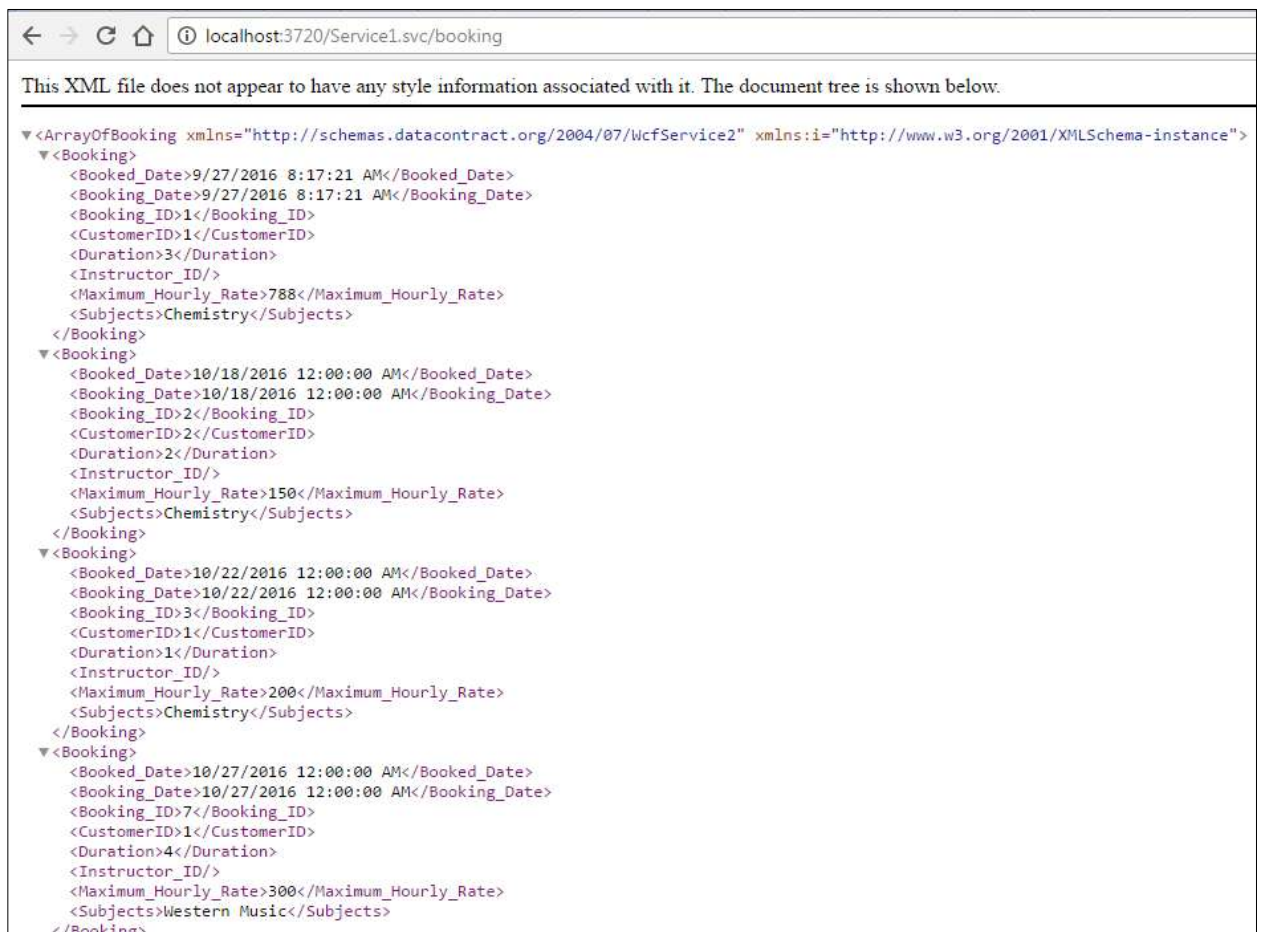
### Code of Service1.svc

```

<%@ ServiceHost Language="C#" Debug="true" Service="WcfService2.Service1"
CodeBehind="Service1.svc.cs"
Factory="System.ServiceModel.Activation.WebServiceHostFactory" %>

```

### Results of Methods



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<ArrayOfBooking xmlns="http://schemas.datacontract.org/2004/07/WcfService2" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <Booking>
    <Booked_Date>9/27/2016 8:17:21 AM</Booked_Date>
    <Booking_Date>9/27/2016 8:17:21 AM</Booking_Date>
    <Booking_ID>1</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>3</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>788</Maximum_Hourly_Rate>
    <Subjects>Chemistry</Subjects>
  </Booking>
  <Booking>
    <Booked_Date>10/18/2016 12:00:00 AM</Booked_Date>
    <Booking_Date>10/18/2016 12:00:00 AM</Booking_Date>
    <Booking_ID>2</Booking_ID>
    <CustomerID>2</CustomerID>
    <Duration>2</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>150</Maximum_Hourly_Rate>
    <Subjects>Chemistry</Subjects>
  </Booking>
  <Booking>
    <Booked_Date>10/22/2016 12:00:00 AM</Booked_Date>
    <Booking_Date>10/22/2016 12:00:00 AM</Booking_Date>
    <Booking_ID>3</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>1</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>200</Maximum_Hourly_Rate>
    <Subjects>Chemistry</Subjects>
  </Booking>
  <Booking>
    <Booked_Date>10/27/2016 12:00:00 AM</Booked_Date>
    <Booking_Date>10/27/2016 12:00:00 AM</Booking_Date>
    <Booking_ID>7</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>4</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>300</Maximum_Hourly_Rate>
    <Subjects>Western Music</Subjects>
  </Booking>
</ArrayOfBooking>

```

← → ↻ 🏠 ⓘ localhost:3720/Service1.svc/booking/1

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼ <ArrayOfBooking xmlns="http://schemas.datacontract.org/2004/07/WcfService2" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  ▼ <Booking>
    <Booked_Date>10/7/2016 10:00:00 AM</Booked_Date>
    <Booking_Date>10/7/2016 10:00:00 AM</Booking_Date>
    <Booking_ID>1</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>3</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>788</Maximum_Hourly_Rate>
    <Subjects>Chemistry</Subjects>
  </Booking>
  ▼ <Booking>
    <Booked_Date>10/9/2016 9:47:00 AM</Booked_Date>
    <Booking_Date>10/9/2016 9:47:00 AM</Booking_Date>
    <Booking_ID>3</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>1</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>200</Maximum_Hourly_Rate>
    <Subjects>Chemistry</Subjects>
  </Booking>
  ▼ <Booking>
    <Booked_Date>10/17/2016 7:19:00 AM</Booked_Date>
    <Booking_Date>10/17/2016 7:19:00 AM</Booking_Date>
    <Booking_ID>7</Booking_ID>
    <CustomerID>1</CustomerID>
    <Duration>4</Duration>
    <Instructor_ID/>
    <Maximum_Hourly_Rate>300</Maximum_Hourly_Rate>
    <Subjects>Western Music</Subjects>
  </Booking>
</ArrayOfBooking>

```

← → ↻ 🏠 ⓘ localhost:3720/Service1.svc/user

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼ <ArrayOfuser xmlns="http://schemas.datacontract.org/2004/07/WcfService2" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  ▼ <user>
    <Address>No,57, Madelgamuwa, Gampaha</Address>
    <Dob>2/2/1998 12:00:00 AM</Dob>
    <First_name>Chathurangi</First_name>
    <Gender>Female</Gender>
    <Last_name>Shyalika</Last_name>
    <Tp>0772150269</Tp>
    <UserID>1</UserID>
    <UserName>chathurangijks@gmail.com</UserName>
    <UserType>Instructor</UserType>
  </user>
  ▼ <user>
    <Address>No56, Silva Rd, Moratuwa</Address>
    <Dob>2/2/1008 12:00:00 AM</Dob>
    <First_name>Chamani</First_name>
    <Gender>Female</Gender>
    <Last_name>Shiranthika</Last_name>
    <Tp>0111234561</Tp>
    <UserID>2</UserID>
    <UserName>chamanijs2@gmail.com</UserName>
    <UserType>Customer</UserType>
  </user>
  ▼ <user>
    <Address>No56,289Road</Address>
    <Dob>2/9/9988 12:00:00 AM</Dob>
    <First_name>Jaya</First_name>
    <Gender>Male</Gender>
    <Last_name>Sampath</Last_name>
    <Tp>0774567891</Tp>
    <UserID>3</UserID>
    <UserName>jayakodydw@gmail.com</UserName>
    <UserType>Customer</UserType>
  </user>
</ArrayOfuser>

```

← → ↻ 🏠 ⓘ localhost:3720/Service1.svc/user/Customer

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼ <ArrayOfuser xmlns="http://schemas.datacontract.org/2004/07/WcfService2" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  ▼ <user>
    <Address>No56, Silva Rd, Moratuwa</Address>
    <Dob>2/2/1008 12:00:00 AM</Dob>
    <First_name>Chamani</First_name>
    <Gender>Female</Gender>
    <Last_name>Shiranthika</Last_name>
    <Tp>0111234561</Tp>
    <UserID>2</UserID>
    <UserName>chamanijs2@gmail.com</UserName>
    <UserType>Customer</UserType>
  </user>
  ▼ <user>
    <Address>No56,289Road</Address>
    <Dob>2/9/9988 12:00:00 AM</Dob>
    <First_name>Jaya</First_name>
    <Gender>Male</Gender>
    <Last_name>Sampath</Last_name>
    <Tp>0774567891</Tp>
    <UserID>3</UserID>
    <UserName>jayakodydw@gmail.com</UserName>
    <UserType>Customer</UserType>
  </user>
</ArrayOfuser>

```

← → ↻ 🏠 ⓘ localhost:3720/Service1.svc/user/Customer/chamanijs2@gmail.com

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼ <ArrayOfuser xmlns="http://schemas.datacontract.org/2004/07/WcfService2" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  ▼ <user>
    <Address>DefaultAdd</Address>
    <Dob>Defaultdob</Dob>
    <First_name>DefaultFname</First_name>
    <Gender>DefaultGender</Gender>
    <Last_name>Defaultlname</Last_name>
    <Tp>DefaultTP</Tp>
    <UserID>2</UserID>
    <UserName>chamanijs2@gmail.com</UserName>
    <UserType>Customer</UserType>
  </user>
</ArrayOfuser>

```

## 7) Invoice Calculator



### Main Method-

This GUI is connected with the RESTful web service created in question 6. This code retrieves the booking details as an XML as follows. The result is returned on the console.

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    /**<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
```



```

    }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Invoice.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Invoice.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Invoice.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Invoice.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    }
}
//</editor-fold>
Client client=ClientBuilder.newClient();
WebTarget
target=client.target("http://localhost:3720/Service1.svc/booking");
System.out.println(target.request(MediaType.TEXT_XML).get(String.class)

);

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Invoice().setVisible(true);
    }
});
});

```

### Output (XML result)



```

run:
<ArrayOfBooking xmlns="http://schemas.datacontract.org/2004/07/WcfService2"
xmlns:i="http://www.w3.org/2001/XMLSchema-
instance"><Booking><Booked_Date>9/27/2016 8:17:21
AM</Booked_Date><Booking_Date>9/27/2016 8:17:21
AM</Booking_Date><Booking_ID>1</Booking_ID><CustomerID>1</CustomerID><Duration>3<
/Duration><Instructor_ID/><Maximum_Hourly_Rate>788</Maximum_Hourly_Rate><Subjects
>Chemistry</Subjects></Booking><Booking><Booked_Date>10/18/2016 12:00:00
AM</Booked_Date><Booking_Date>10/18/2016 12:00:00
AM</Booking_Date><Booking_ID>2</Booking_ID><CustomerID>2</CustomerID><Duration>2<
/Duration><Instructor_ID/><Maximum_Hourly_Rate>150</Maximum_Hourly_Rate><Subjects
>Chemistry</Subjects></Booking><Booking><Booked_Date>10/22/2016 12:00:00
AM</Booked_Date><Booking_Date>10/22/2016 12:00:00

```



```
AM</Booking_Date><Booking_ID>3</Booking_ID><CustomerID>1</CustomerID><Duration>1<
/Duration><Instructor_ID/><Maximum_Hourly_Rate>200</Maximum_Hourly_Rate><Subjects
>Chemistry</Subjects></Booking><Booking><Booked_Date>10/27/2016 12:00:00
AM</Booked_Date><Booking_Date>10/27/2016 12:00:00
AM</Booking_Date><Booking_ID>7</Booking_ID><CustomerID>1</CustomerID><Duration>4<
/Duration><Instructor_ID/><Maximum_Hourly_Rate>300</Maximum_Hourly_Rate><Subjects
>Western Music</Subjects></Booking></ArrayOfBooking>
BUILD SUCCESSFUL (total time: 25 seconds)
```

## I. “View Lesson Manager” button click results-

### Code:

```
public void fetch_data1(){
    try{
        connection1();
        String sql="select * from lessonmanager";
        ResultSet rs=stmt.executeQuery(sql);
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(Exception ex){
        JOptionPane.showMessageDialog(null,ex);
    }
}
```

**Invoice Calculator**

**E-Explore**  
Online E-Learning Academy

Log_ID	Start_Time	Duration	Notes	Instructor_ID
1	4	1	Conducted Soc...	1
2	3	2	Conducted Soc...	2
3	7	11	Conducted	3
4	3	3	Success	1
8	5	2	Not good	5
9	3	1	Successful	1
10	7	2	Successful led...	1
11	6	1	null	3
12	6	1	Good	2
13	4	1	Good	2
14	4	1	Good	5
15	5	60	Success	2

**Booking No**

**Maximum Hourly Rate**

**Duration**

**Total Amount**

**Calculate Total In ID**

**View Lesson Manager**

**View Calculated Invoices**

**Calculate Total & Generate XML**

**Save XML Data to database**

**Lesson Manager Report**

**Generate Invoice Report**

## II. “View Calculated Invoices” button click results-

**Code:**

```
public void fetch_data2(){
    try{
        connection1();
        String sql="select * from invoice";
        ResultSet rs=stmt.executeQuery(sql);
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(Exception ex){
        JOptionPane.showMessageDialog(null,ex);
    }
}
```

The screenshot shows a Java Swing window titled "E-Explore Online E-Learning Academy". The window has a light blue background. On the left, there is a green circular icon with a white document labeled "INVOICE". The main title "Invoice Calculator" is in a large, bold, blue font. Below the title is a table with 5 columns: Invoice\_ID, Hourly\_Rate, Duration, Total, and Payment\_Status. The table contains 5 rows of data. To the right of the table is a vertical stack of buttons: "View Lesson Manager", "View Calculated Invoices", "Calculate Total & Generate XML", "Save XML Data to database", "Lesson Manager Report", and "Generate Invoice Report". Below the table is a section for "Booking No" with a dropdown menu labeled "Select Booking ID". Below that are three input fields for "Maximum Hourly Rate", "Duration", and "Total Amount", each with a green "Calculate Total In ID" button. At the bottom right, there is a graphic of a document with a pencil.

Invoice_ID	Hourly_Rate	Duration	Total	Payment_Status
1	200.0	3.0	600.0	Not Paid
2	200.0	3.0	600.0	Not Paid
3	200.0	3.0	600.0	Not Paid
4	200.0	3.0	600.0	Not Paid
5	200.0	3.0	600.0	Not Paid

### III. “Calculate Total & Generate XML” button click results-

The Invoice Total of each invoice will be calculated and saved as a XML file. In the console the results will be displayed.

#### Code:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    try {  
        // TODO add your handling code here:  
        //double total_amount=Double.parseDouble(textField1.getText());  
  
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  
        DocumentBuilder builder = factory.newDocumentBuilder();  
        org.w3c.dom.Document doc =  
builder.parse("http://localhost:3720/Service1.svc/booking");  
        XPathFactory xPathfactory = XPathFactory.newInstance();  
        XPath xpath = xPathfactory.newXPath();  
        XPathExpression expr1= xpath.compile("//ArrayOfBooking/Booking/Duration");  
        XPathExpression expr2=  
xpath.compile("//ArrayOfBooking/Booking/Maximum_Hourly_Rate");  
        XPathExpression expr3= xpath.compile("//ArrayOfBooking/Booking");  
  
        NodeList nl = (NodeList) expr1.evaluate(doc, XPathConstants.NODESET);  
        NodeList n2 = (NodeList) expr2.evaluate(doc, XPathConstants.NODESET);  
        NodeList n3 = (NodeList) expr3.evaluate(doc, XPathConstants.NODESET);  
  
        AllInvoices inv = new AllInvoices();  
        AllInvoiceList invlist = new AllInvoiceList();  
  
        int n4=n3.getLength();  
        for (int i = 0; i < n4; i++)  
  
        {  
            org.w3c.dom.Node node = nl.item(i);  
            String key1 = node.getTextContent();  
            dur=Double.parseDouble(key1);  
  
            org.w3c.dom.Node node1 = n2.item(i);  
            String key2 = node1.getTextContent();  
            hr=Double.parseDouble(key2);  
  
            System.out.println("Duration : "+dur);  
            inv.setDuration(dur);  
            System.out.println("Maximum Hourly Rate :"+hr);  
            inv.setHourly_rate(hr);  
        }  
    }  
}
```

```

        System.out.println("Invoice Total :"+hr*dur);
        inv.setTotal(hr*dur);

        invlist.add(inv);
    }

    try {
        File file = new File("D:\\InvoiceList.xml");
        JAXBContext jaxbContext = JAXBContext.newInstance(AllInvoiceList.class);
        Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
        jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        jaxbMarshaller.marshal(invlist, file);
        jaxbMarshaller.marshal(invlist, System.out);
    } catch (JAXBException e) {
        e.printStackTrace();
    }

    } catch (ParserConfigurationException ex) {
        Logger.getLogger(Invoice.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SAXException ex) {
        Logger.getLogger(Invoice.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(Invoice.class.getName()).log(Level.SEVERE, null, ex);
    } catch (XPathExpressionException ex) {
        Logger.getLogger(Invoice.class.getName()).log(Level.SEVERE, null, ex);
    }

}

```

### **AllInvoices.java**

```

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

```

```

@XmlRootElement

```

```

public class AllInvoices {

```

```

    double duration = 0;
    double hourly_rate = 0;
    double total=0;

```

```

    public double getDuration() {
        return duration;
    }

```

```

    @XmlElement
    public void setDuration(double duration) {

```

```

        this.duration = duration;
    }

    public double getHourly_rate() {
        return hourly_rate;
    }

    @XmlElement
    public void setHourly_rate(double hourly_rate) {
        this.hourly_rate = hourly_rate;
    }

    public double getTotal() {
        return total;
    }

    @XmlElement
    public void setTotal(double total) {
        this.total = total;
    }
}

```

### **AllInvoiceList.java**

```

import java.util.List;
import java.util.ArrayList;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="InvoiceList")
public class AllInvoiceList {
    List<AllInvoices> allInvoiceList;

    /**
     *
     * @param allInvoiceList
     */

    @XmlElement( name = "Invoice" )
    public void setAllInvoiceList(List<AllInvoices> allInvoiceList)
    {
        this.allInvoiceList = allInvoiceList;
    }

    public List<AllInvoices> getAllInvoiceList()

```

```

    {
        return allInvoiceList;
    }

    public void add( AllInvoices allInvoices )
    {
        if( this.allInvoiceList == null )
        {
            this.allInvoiceList = new ArrayList<AllInvoices>();
        }

        this.allInvoiceList.add( allInvoices );
    }
}

```

### Output (in console):

run:

```

Duration : 3.0
Maximum Hourly Rate :788.0
Invoice Total :2364.0

```

```

Duration : 2.0
Maximum Hourly Rate :150.0
Invoice Total :300.0
Duration : 1.0

```

```

Maximum Hourly Rate :200.0
Invoice Total :200.0
Duration : 4.0

```

```

Maximum Hourly Rate :300.0
Invoice Total :1200.0
Duration : 3.0

```

```

Maximum Hourly Rate :200.0
Invoice Total :600.0

```

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InvoiceList>
    <Invoice>
        <duration>3.0</duration>
        <hourly_rate>200.0</hourly_rate>
        <total>600.0</total>
    </Invoice>
    <Invoice>
        <duration>3.0</duration>
        <hourly_rate>200.0</hourly_rate>
        <total>600.0</total>
    </Invoice>
</InvoiceList>

```

```

        <duration>3.0</duration>
        <hourly_rate>200.0</hourly_rate>
        <total>600.0</total>
    </Invoice>
    <Invoice>
        <duration>3.0</duration>
        <hourly_rate>200.0</hourly_rate>
        <total>600.0</total>
    </Invoice>
    <Invoice>
        <duration>3.0</duration>
        <hourly_rate>200.0</hourly_rate>
        <total>600.0</total>
    </Invoice>
</InvoiceList>

```

#### IV. “Save XML Data to Database” button click results-

##### Code:

```

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        File fXmlFile = new File("D:\\\\InvoiceList.xml");
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(fXmlFile);

        doc.getDocumentElement().normalize();
        NodeList nList = doc.getElementsByTagName("Invoice");
        for (int temp = 0; temp < nList.getLength(); temp++) {
            org.w3c.dom.Node nNode = nList.item(temp);
            if (nNode.getNodeType() == org.w3c.dom.Node.ELEMENT_NODE) {
                Element eElement = (Element) nNode;
                String
n1=eElement.getElementsByTagName("hourly_rate").item(0).getTextContent();
                String
n2=eElement.getElementsByTagName("duration").item(0).getTextContent();
                String n3=eElement.getElementsByTagName("total").item(0).getTextContent();

                connection1();
                int r=stmt.executeUpdate("insert into
invoices(Hourly_rate,Duration,Total,Payment_Status) values('"+n1+"','"+n2+"','"+n3+"','"+Not
Paid+"')");
            }
        }
        JOptionPane.showMessageDialog(null,"Invoice Data Saved Successfully");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

}}  
**Output:**



	Invoice_ID	Hourly_Rate	Duration	Total	Payment_Status
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	200	3	600	Not Paid
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	200	3	600	Not Paid
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	200	3	600	Not Paid
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	200	3	600	Not Paid
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	200	3	600	Not Paid

## V. “Lesson Manager Report” button click results-

This will display the records of Lesson Manager as a Jasper report.

### Code:

```
public void Print(){
    try{
        //Class.forName("com.mysql.jdbc.Driver");
        // Connection
        c=DriverManager.getConnection("jdbc:mysql://localhost:3306/sowpassignmentnew",
        "root","123");

        String report =
        "C:\\Users\\User\\Documents\\NetBeansProjects\\Invoice_Client\\src\\java\\forms\\report3.jrxml
        ";

        JRXmlDataSource xmlDataSource = new
        JRXmlDataSource("D:\\LessonManagerApp.xml","/Users/User");
        JasperReport JASP_REP =JasperCompileManager.compileReport(report);

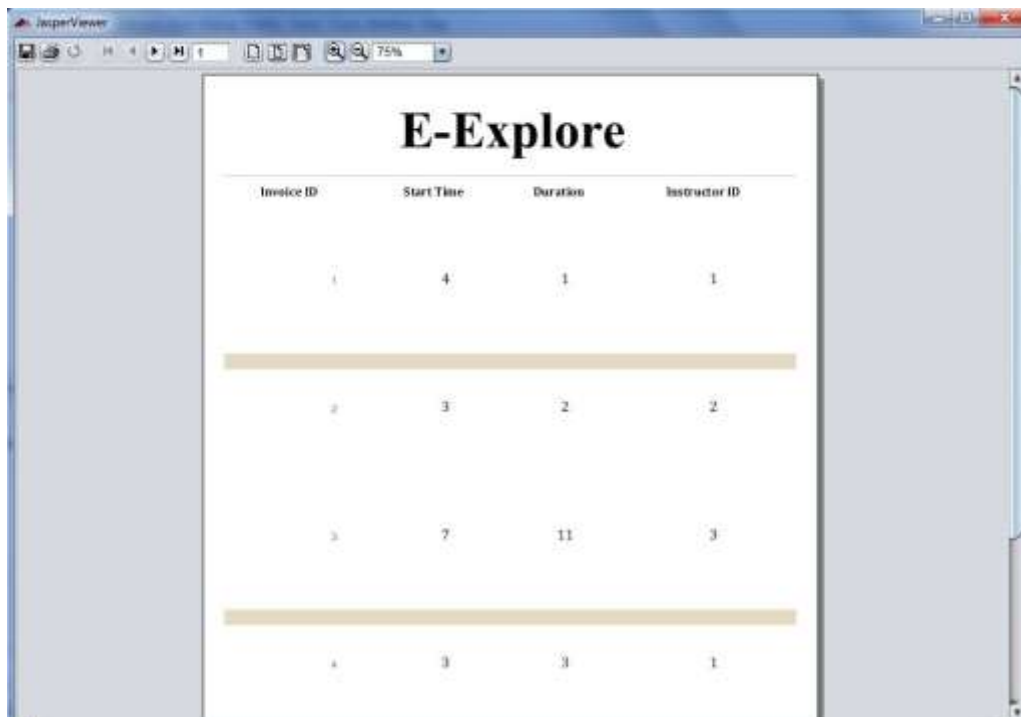
        JasperPrint JASP_PRINT = JasperFillManager.fillReport(JASP_REP,new
        HashMap(),xmlDataSource);
```



```

JasperViewer.viewReport(JASP_PRINT);
//JasperExportManager.exportReportToPdfFile(JASP_PRINT, "sample.pdf");
}
catch(Exception ex)
{
    System.out.println(ex);
}
}

```



The screenshot shows a window titled 'JasperViewer' displaying a report titled 'E-Explore'. The report contains a table with four columns: 'Invoice ID', 'Start Time', 'Duration', and 'Instructor ID'. The table has four data rows. The first row is highlighted with a light blue background. The second row is highlighted with a light orange background. The third row is highlighted with a light blue background. The fourth row is highlighted with a light orange background.

Invoice ID	Start Time	Duration	Instructor ID
1	4	1	1
2	3	2	2
3	7	11	3
4	3	3	1

## VI. “Generate Invoice Report” button click results-

This will display all the Invoices as a Jasper report.

### Code:

```

public void Print_sheet(){
    try{
        //Class.forName("com.mysql.jdbc.Driver");
        // Connection
        c=DriverManager.getConnection("jdbc:mysql://localhost:3306/sowpassignmentnew",
        "root","123");

        String report =
        "C:\\Users\\User\\Documents\\NetBeansProjects\\Invoice_Client\\src\\java\\forms\\report1.jr
        xml";
    }
}

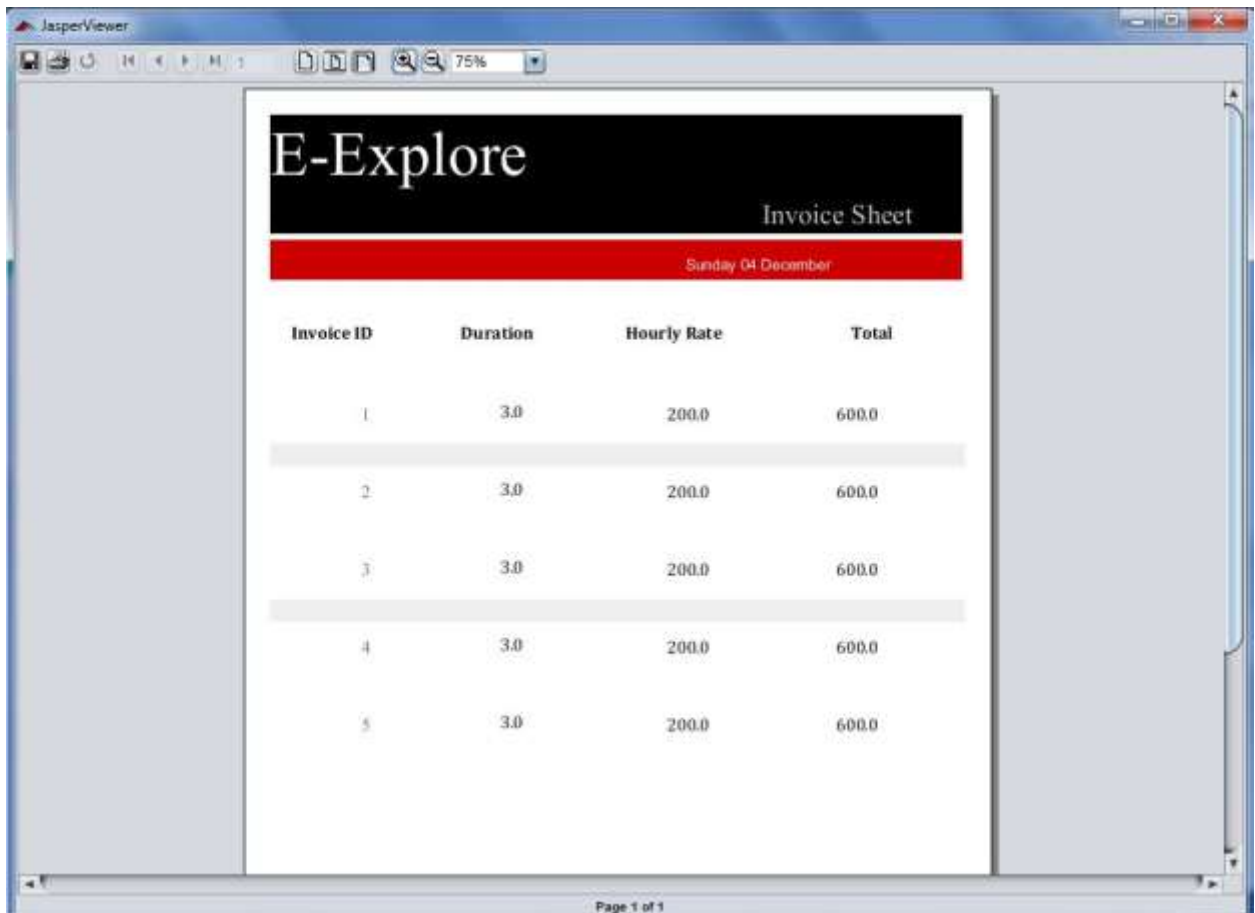
```

```

JRXmlDataSource xmlDataSource = new
JRXmlDataSource("D:\\InvoiceList.xml","/InvoiceList/Invoice");
JasperReport JASP_REP =JasperCompileManager.compileReport(report);

JasperPrint JASP_PRINT = JasperFillManager.fillReport(JASP_REP,new
HashMap(),xmlDataSource);
JasperViewer.viewReport(JASP_PRINT);
//JasperExportManager.exportReportToPdfFile(JASP_PRINT, "sample.pdf");
}
catch(Exception ex)
{
    System.out.println(ex);
}
}

```




Invoice ID	Duration	Hourly Rate	Total
1	3.0	200.0	600.0
2	3.0	200.0	600.0
3	3.0	200.0	600.0
4	3.0	200.0	600.0
5	3.0	200.0	600.0

## VII. “Calculate Total In ID” button click results-

### Code:

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    double x;  
    double y;  
    String total;  
    try {  
        connection1();  
  
        ResultSet r=stmt.executeQuery("select Maximum_Hourly_Rate, Duration from booking  
where Booking_ID=" + """+jComboBox1.getSelectedItem()+"");  
  
        if(r.next()){  
            textField2.setText(r.getString("Maximum_Hourly_Rate"));  
            textField3.setText(r.getString("Duration"));  
            x = Double.parseDouble(textField2.getText());  
            y = Double.parseDouble(textField3.getText());  
            double total1=x*y;  
            total = String.valueOf(total1);  
            textField4.setText(total);  
        }  
    } catch (Exception ex) {  
        ex.printStackTrace();  
    }  
}
```

When a booking ID is selected and this button is clicked this will retrieve the maximum hourly rate and duration of that invoice and calculate the invoice total.



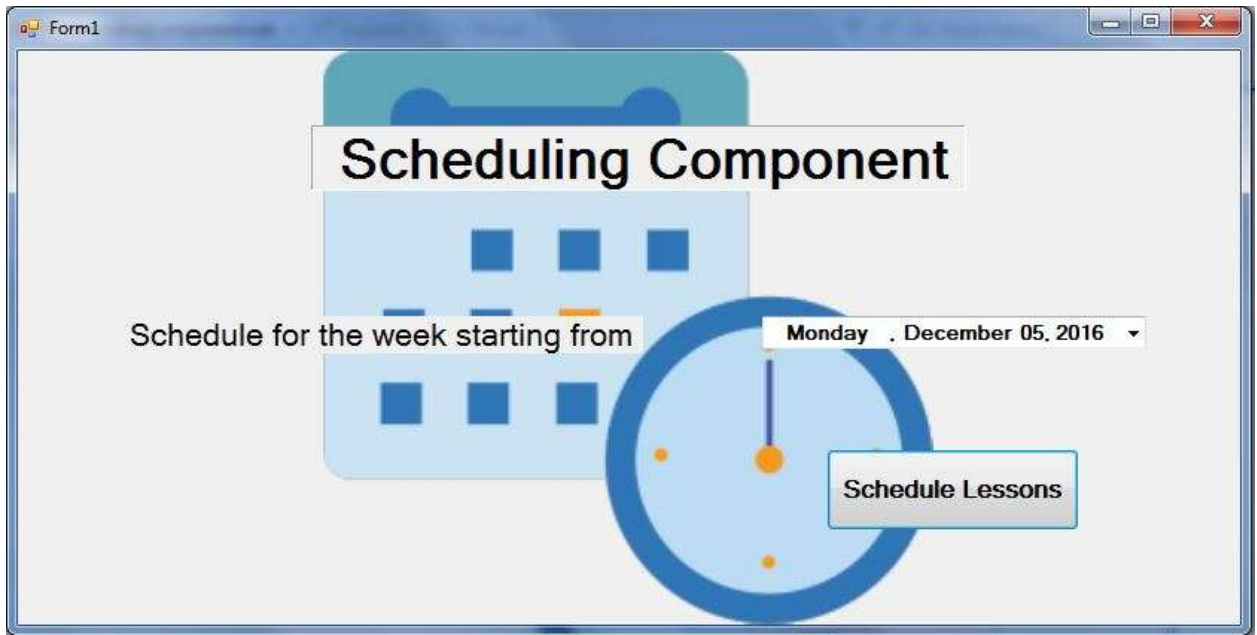
Label	Value
Booking No	7
Maximum Hourly Rate	300
Duration	4
Total Amount	1200.0

Calculate Total In ID

## 8) Scheduling Component

I implemented this component in both C#.NET and PHP.

### I. C#.NET implementation



#### Code for the button

```
private void button1_Click(object sender, EventArgs e)
{
    string connetionString = null;
    MySqlConnection connection;
    MySqlCommand command;
    String sql1 = null;

    MySqlDataAdapter adpter = new MySqlDataAdapter();
    DataSet ds = new DataSet();

    connetionString = "Database=sowpassignmentnew; Data Source=localhost; User
Id=root; Password=123";

    connection = new MySqlConnection(connetionString);
    connection.Open();

    sql1="INSERT INTO lessonmanagerapp(Booking_ID,Start_Time,Duration,Day,Subject)
SELECT Booking_ID,Start_Time,Duration,Day,Subjects FROM booking ; INSERT into
lessonmanagerapp (Instructor_ID) select Instructor_ID from instructor_available_days
where Available_Days IN (select Day from booking where
booking.Day=instructor_available_days.Available_Days)";

    command = new MySqlCommand(sql1, connection);
    adpter.InsertCommand = command;
    adpter.InsertCommand.ExecuteNonQuery();
}
```

```
MessageBox.Show("Successfully Scheduled");  
connection.Close();  
}
```

When the button is clicked automatically lessons are scheduled according to the given considerations under question 8.



## II. PHP implementation

A screenshot of a web application interface for scheduling lessons. The header is light blue and features a graduation cap icon on the left, the title 'Schedule Lessons' in large blue font in the center, and the subtitle 'Assign the schedule' in pink font below it. On the right of the header is a 'Log out' button. Below the header is a light gray form area. On the left of the form, there is a red asterisk and the text '\* required fields'. Below this, it says 'Schedule for the week starting from :'. To the right of this text is a text input field containing the date '10/03/2016'. On the right side of the form, there are two buttons: a green 'View All Schedules' button and a blue 'Generate Schedule' button.

## PHP Code for schedule.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "123";
$dbname = "sowpassignmentnew";

$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$ScDate = "";
$date1Err = "";

if (isset($_POST['submit'])) {

    if (empty($_POST["ScDate"])) {
        $date1Err = "Date is required";
    } else {
        $date1 = mysqli_real_escape_string($conn, $_POST['ScDate']);
    }
}

if (!empty($_POST["ScDate"]) && !($date1Err))
{
    $sql10="INSERT INTO
    lessonmanagerapp(Booking_ID,Start_Time,Duration,Day,Subject)
    SELECT Booking_ID,Start_Time,Duration,Day,Subjects FROM booking";

    $sql9="INSERT into lessonmanagerapp (Instructor_ID)
    select Instructor_ID from instructor_available_days where Available_Days IN (select
    Day from booking where booking.Day=instructor_available_days.Available_Days)";

    if (mysqli_query($conn, $sql9) && mysqli_query($conn, $sql10)) {
        echo "New Schedule created successfully";
    } else {
        echo "Could not able to execute $sql9. " . mysqli_error($conn);
    }
}

if (isset($_POST['submitIns'])) {

    $sql14="INSERT into lessonmanagerapp (Instructor_ID)
```

```
select Instructor_ID from instructor_available_days where Available_Days IN (select
Day from booking)";
```

```
if (mysqli_query($conn, $sql14)) {
    echo "New Instructor scheduled successfully";
} else {
    echo "Could not able to execute $sql14. " . mysqli_error($conn);
}
}
```

```
mysqli_close($conn);
```

?>

Success message after scheduling lessons for a given week



The screenshot shows a web application interface for scheduling lessons. The header is light blue and contains a graduation cap icon on the left, the title "Schedule Lessons" in large blue font in the center, and a "Log out" button on the right. Below the title is the subtitle "Assign the schedule" in pink. The main content area is white and displays a success message: "New Schedule created successfully". To the right of this message is a green button labeled "View All Schedules". Below the message, there is a red asterisk followed by the text "\* required fields". Underneath, it says "Schedule for the week starting from :" followed by a text input field containing the date "10/09/2016". To the right of the input field is a blue button labeled "Generate Schedule".