2016

Appointment Management System

2016

**Due Date:** (18 November 2016 | Semester 2)

**Supervisor:** Bahman Sarrafpour [bsarrafpour@unitec.ac.nz](mailto:bsarrafpour@unitec.ac.nz)

**Names: ID:**

Amelia mcneil 1388244

Nawaf Altuwayjiri 1377387

Marzouq Almarzooq 1380949

Oshada Koralage 1434048

Contents

[1 Introduction 5](#_Toc467081997)

[1.1 Background 6](#_Toc467081998)

[1.2 Scope comparison 7](#_Toc467081999)

[1.2.1 Original project scope 7](#_Toc467082000)

[1.2.2 Final project scope 7](#_Toc467082001)

[1.3 Objectives 7](#_Toc467082002)

[2 Design 8](#_Toc467082003)

[2.1 Methodology 8](#_Toc467082004)

[2.2 Database 10](#_Toc467082005)

[2.2.1 Data Model 10](#_Toc467082006)

[2.2.2 Data Dictionary 13](#_Toc467082007)

[2.2.3 UML Diagram 15](#_Toc467082008)

[16](#_Toc467082009)

[3 Implementation 17](#_Toc467082010)

[3.1 Technologies and tools 17](#_Toc467082011)

[3.1.1 Android OS Application Client 17](#_Toc467082012)

[3.1.2 Service Oriented Web Server Application 18](#_Toc467082013)

[3.1.3 MySQL Relational Data Base Management System 18](#_Toc467082014)

[3.1.4 Development Tools 19](#_Toc467082015)

[3.1.5 Version Control Software 19](#_Toc467082016)

[3.2 Database 22](#_Toc467082017)

[3.3 Server Side 22](#_Toc467082018)

[3.4 Client Side 22](#_Toc467082019)

[3.4.1 User Interface Design 22](#_Toc467082020)

[3.4.2 Basic Interface Design 23](#_Toc467082021)

[3.4.3 Final User Interface Design 25](#_Toc467082022)

[4 Testing and performance 28](#_Toc467082023)

[4.1.1 Testing and Performance 28](#_Toc467082024)

[4.1.2 User acceptance testing 31](#_Toc467082025)

[5 User Documentation 33](#_Toc467082026)

[5.1 User Functionality 33](#_Toc467082027)

[5.1.1 Lecturer 33](#_Toc467082028)

[5.1.2 Student 33](#_Toc467082029)

[5.2 Usage 34](#_Toc467082030)

[5.2.1 Lecturer 34](#_Toc467082031)

[5.2.2 Student 38](#_Toc467082032)

[6 Logs 41](#_Toc467082033)

[6.1 Project Planning 41](#_Toc467082034)

[6.2 Meeting Logs 41](#_Toc467082035)

[6.2.1 Supervisor Meetings 41](#_Toc467082036)

[6.2.2 Bi-weekly Meetings 42](#_Toc467082037)

[6.2.3 Group Meetings 42](#_Toc467082038)

[6.3 Supervisor Email records 45](#_Toc467082039)

[7 Conclusion 49](#_Toc467082040)

[7.1 Individual Contribution review 49](#_Toc467082041)

[7.1.1 Amelia 49](#_Toc467082042)

[7.1.2 Marzouq 50](#_Toc467082043)

[7.1.3 Nawaf 51](#_Toc467082044)

[7.1.4 Oshada 52](#_Toc467082045)

[7.2 Future Improvements 53](#_Toc467082046)

[7.3 Summary 55](#_Toc467082047)

[8 Appendix A 56](#_Toc467082048)

[8.1 Client Code 56](#_Toc467082049)

[8.2 Server Code 102](#_Toc467082050)

[8.3 Database SQL Dump 136](#_Toc467082051)

[9 Bibliography 0](#_Toc467082052)

[10 Individual Contribution Review 1](#_Toc467082053)

[11 Project proposal 3](#_Toc467082054)

[1. Executive Summary 2](#_Toc467082055)

[2. Goal 3](#_Toc467082056)

[3. Background 3](#_Toc467082057)

[4. Objectives 3](#_Toc467082058)

[5. Justification 4](#_Toc467082059)

[6. Scope 5](#_Toc467082060)

[6.1 Functional Requirements 5](#_Toc467082061)

[6.2 Exclusions 6](#_Toc467082062)

[7. Project Team 6](#_Toc467082063)

[8. Methodology 7](#_Toc467082064)

[9. Deliverables 8](#_Toc467082065)

[10. Milestones and Tasks 9](#_Toc467082066)

[11. Resource Requirements and Costs 10](#_Toc467082067)

[12. Risk Analysis Checklist 11](#_Toc467082068)

[12.1 Generic Risk Checklist 11](#_Toc467082069)

[12.2 Specific Project Risks 13](#_Toc467082070)

[13. Quality Assurance Process 14](#_Toc467082071)

[14. Project Work Plan 15](#_Toc467082072)

[15. Intellectual Property 15](#_Toc467082073)

[16. Confidentiality 15](#_Toc467082074)

[17. Declaration 16](#_Toc467082075)

[18. Approvals 17](#_Toc467082076)

# Introduction

In this report, we will be able to show the work we have done within the last 5 months for our final project at Unitec, and the ideas we had to make this project a success. The idea we had for our project was to create an effective Appointment Management System. We decided to go with this idea because we wanted to help Students and Lecturers at Unitec.

The Appointment Management System Project was created to develop a multi-tier system to facilitate making out of class appointments between students and lecturers. The multi-tier system comprises of client and server applications supported by a relational database management system. Students and lecturer’s login via an android mobile application using their respective usernames and passwords to arrange appointments with one another. The choice to create a smart phone client application was to provide students and lecturers with greater accessibility.

The original proposal was to create a web- based application however a couple weeks into the project we decided with the workload and amount of time we had for this project we wanted to put all our focus in creating one application either a mobile application or a web-based application. Considering the skills and experience our team had we decided to create a mobile application where we could put all our effort into making it the best possible application that we could.

The main reason behind this idea was based on the current practice where students seek a lecturer by visiting the staff room or office. Because Unitec has no set system in place, students didn’t have required office hour access with lecturers to address and resolve any learning related issues and difficulties or to make arrangements to address the matter at a more suitable time. More often this practice resulted in a continuous flow of students desiring to have out of class access to their lecturers.

Consequently, this caused frustrations for both students and lecturers, since lecturers are interrupted, and students experience delays. To minimize the frustration’s lecturers have adopted their own policies to regulate and manage the flow of students requiring further learning assistance. As these policies vary between lecturers, the system continues to be problematic and confusing.

With a mobile application in place, assisting students and lecturers setting up appointments would save time and effectively illuminate time wastage and interruptions of students showing up unexpectedly with no appointment. We created an Appointment Management System with a common clear standard to reduce inefficiencies arising from inconsistent policies. More so the aim was to provide a useable solution to lessen the burden and frustrations experienced by both students and lecturers. The system is deployed via an android mobile-based application allowing lecturers to advertise their availability to students requiring additional office hour access. The system will also offer both students and lecturers the ability to manage their office hour appointments.

Since the way Lecturers make appointments currently is through their Outlook calendars we decided to sync the system with the lecturers Outlook which means, once an appointment has been made the notification will be sent directly to the Lecturers Outlook inbox and put into their calendars. This is just one feature that has been implemented into our application.

Throughout this report we will introduce the technologies, methodologies and research we have used to complete this project.

## Background

Unitec is the biggest Institute in New Zealand with over 16,000 attending students and over 1000 teaching staff. Students and teaching staff are located across three campuses. All campuses are in Auckland, with the main campus situated in Mt. Albert, the other two situated in Henderson and Albany.

Given the large size of Unitec it is imperative to efficiently manage the logistics for office hour appointments between students and lecturers. Currently there is no set standard policy that defines a best practice, and no system in place to facilitate the process. There are 65 departments at Unitec however we will only be focusing on the Computing department in this project.

## Scope comparison

### Original project scope

Our scope is to create an appointment booking and management web application and a mobile application that will assist students and lecturers in making appointments between each other. The desire of this new process of making an appointment through the web application and mobile application is to provide greater standardization and reduce current inefficiencies arising from inconsistent practices.

The web application will serve as a service for lecturers and student to interact. The web application will assist students in making an appointment through a structured and uniform calendar based system, which will save resources over the unstructured system currently in use. The creation of this web application and the mobile application will encourage a common standard.

### Final project scope

Our scope is to create an appointment booking and management android mobile application that will assist student and lecturers in making appointments between each other. The desire of this new process of making an appointment through the mobile application is to provide greater standardization and reduce current inefficiencies arising from the inconsistent practices.

The mobile application will serve as a service for lecturers and student to interact. The mobile application will assist students in making an appointment through a structured and uniform calendar based system, which will save resources over the unstructured system currently in use. The creation of this android application will encourage a common standard.

## Objectives

1. Building a Mobile android application for making office hour appointments between students and lecturers.
2. Provide students with priori information as to when lecturers are available for an appointment to be made.
3. Minimize the time taken to set up an appointment.
4. Mobile application should also provide some integration with lecturer’s Outlook based calendar.
5. Enable appointments to be amended after these have been made.
6. Provide secure communication.

# Design

## Methodology

**Agile methodology - Scrum Development**

The two-suitable methodologies for our project are Waterfall and Scrum. How we decided which was best to do was to compare them with each other and decide which one would benefit us better as a team for our project.

The waterfall method is a sequential design process which is frequently used for the software development process where the progress is flowing steadily downwards through conception, initiation, analysis, design, construction, testing, implementation and maintenance phases.

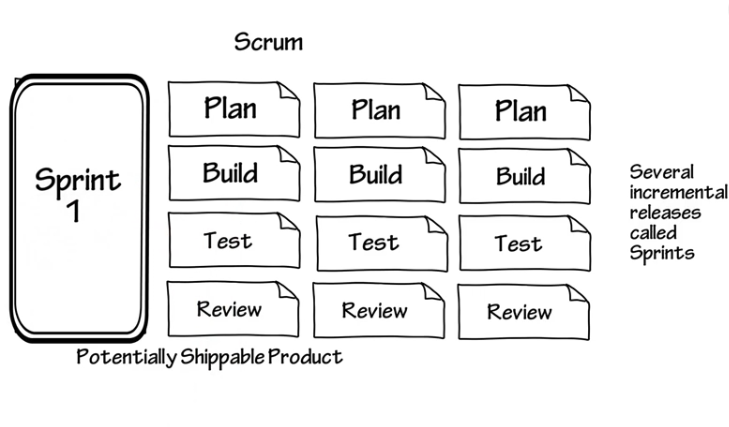
The SCRUM method is a common methodology and is popular among software development teams. The advantage of this methodology is its dynamics and flexibility.

Both these Agile Methodologies were useful to work with in guiding us in this project. Initially we considered the waterfall development so we could work in phases, complete each phase and then move onto the next phase, however this development carried a lot of risk. The main risk was not being able to revisit a phase once it was completed. Due to the high-risk factor, we did not go ahead with this development. We decided to go with the Scrum developmentas we could provide progress of our project daily in our team and weekly to our supervisor. This allowed us to stay on track and show our supervisor our updated application each week.

Scrum is an Agile Project Management Methodology used mainly for software development projects with the goal of delivering new software capability every 2-4 weeks. (what is scrum methodology?)

The Scrum development process is broken up into smaller pieces this allows us to plan, build, test and review a small feature set. This is then repeated time and time again to reduce the time from planning to development to testing each time, through the planning process we are doing just enough planning to complete the next incremental release. You’ll then end up with several releases which are called sprints. Sprints are short duration milestones each sprint takes 2-3 weeks and you keep repeating these sprints until the product is feature complete. (Wawrzyniak, 2014)

After each sprint, we could show our supervisor our completed feature which is noted in the Logs section under the Supervisor Meeting table. In many cases, we could show more features as different team members were working on different tasks at the same time. Because of this development we could stay on track with our project if we were not ready on a specific feature we were working on we recorded it on the Gantt chart.

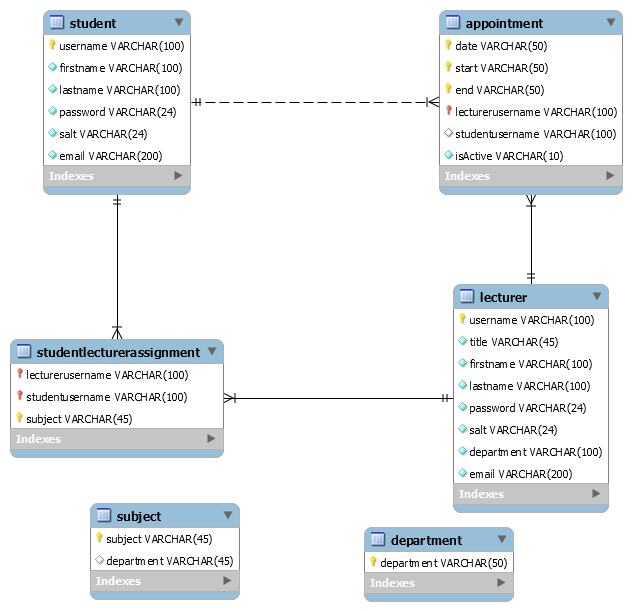


## Database

### Data Model

The Database Software used for this project was MySQL database which was installed onto two team member’s machines who were responsible for the development of the database and the other member whom had the server connected and installed on their computer.

We used RDBMS where there are 6 tables in the completed database. This database was designed to maintain data integrity and to avoid human errors for our android mobile application, many other benefits of using this DB was data safety, scalability and reporting.



1: ERD Diagram for Appointment Management System Application

**STUDENT**

STUDENT(USERNAME,FIRSTNAME,LASTNAME,PASSWORD,SALT,EMAIL)

This STUDENT table has six columns, the primary key being “username” which is a unique ID from Unitecs database. This table holds the student’s information and is used for logging in and creating appointments. For example the students username would be, username: “amal”, lastname: “mcneil”, password(example): “amelia”, Email: “ameliam08@myunitec.co.nz”

**LECTURER**

STUDENT(USERNAME,TITLE,FIRSTNAME,LASTNAME,PASSWORD,SALT,DEPARTMENT,EMAIL)

The LECTURER table has eight columns, here the primary key is ”username”. This table is also used for logging in the lecturer and is used to hold all the details of the lecturers which appear on the student appointment information. It also holds the lecturers email which notifications are sent to when appointments are made.

**STUDENTLECTURERASSIGNMENT**

STUDENTLECTURERASSIGNMENT(LECTUREUSERNAME,STUDENTUSERNAME,SUBJECT)

The STUDENTLECTURERASSIGNMENT table has 3 columns.

**APPOINTMENT**

APPOINTMENT(DATE,START,END,LECTURERUSERNAME,STUDENTUSERNAME,ISACTIVE)

The APPOINTMENT table has 6 columns. Here the primary key is

**SUBJECT**

SUBJECT(SUBJECT,DEPARTMENT)

The SUBJECT table has 2 columns, the primary key is “subject”

**DEPARTMENT**

DEPARTMENT(DEPARTMENT)

The DEPARTMENT table has only one column, the primary key here is “department”.

### Data Dictionary

Here we have the data dictionary this is a collection of descriptions of the data object or the items in a data model. This is for the benefit of the programmers and others who refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects.

#### Appointment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Referenced column |
| date | VARCHAR(50) | NO |  |  |  |
| start | VARCHAR(50) | NO |  |  |  |
| end | VARCHAR(50) | NO |  |  |  |
| lectureusername | VARCHAR(100) | NO |  | lecturer | Username |
| studentusername | VARCHAR(100) | YES |  | student | username |
| isActive | VARCHAR(10) | NO |  |  |  |

#### Lecturer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Comments |
| username | VARCHAR(100) | NO |  |  |  |
| title | VARCHAR(45) | NO |  |  |  |
| firstname | VARCHAR(100) | NO |  |  |  |
| lastname | VARCHAR(100) | NO |  |  |  |
| password | VARCHAR(24) | NO |  |  |  |
| salt | VARCHAR(24) | NO |  |  |  |
| department | VARCHAR(100) | NO |  |  |  |
| email | VARCHAR(200) | NO |  |  |  |

#### Student

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Comments |
| username | VARCHAR(100) | NO |  |  |  |
| firstname | VARCHAR(100) | NO |  |  |  |
| lastname | VARCHAR(100) | NO |  |  |  |
| password | VARCHAR(24) | NO |  |  |  |
| salt | VARCHAR(24) | NO |  |  |  |
| email | VARCHAR(200) | NO |  |  |  |

#### Studentlectureassignment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Referenced column |
| lectureusername | VARCHAR(100) | NO |  | Lecturer | Username |
| studentusername | VARCHAR(100) | NO |  | student | username |
| subject | VARCHAR(45) | NO |  |  |  |

#### Department

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Referenced column |
| department | VARCHAR(45) | NO |  |  |  |

#### Subject

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Links to | Referenced column |
| subject | VARCHAR (45) | NO |  |  |  |
| department | VARCHAR (45) | NO |  |  |  |

### UML Diagram

### 

# Implementation

## Technologies and tools

### Android OS Application Client

Two applications were considered for the Android Development of our project; these were Android Studio and Eclipse. As a group, we had to discuss the pros and cons between the two applications and what would benefit our project best. The main advantage of using Android Studio is that the programmer in our group had previous experience in both applications but found to use Android Studio over Eclipse.

In addition to this the programmers in our group already have experience in using this application. User Interface wise Android Studio was built purposely for Android (Rajput, 2015), While other applications are built to all-purpose IDE that can be used with any language and platform.

Android Studio is a tool for the development of Android platform.

The feature Instant Run can facilitate the pushing of code and resource changes of the running apps. Thus, the developers can access the changes without necessarily restarting the app or rebuilding the APK.

The emulators can install and execute apps in a more rapid manner as compared to the actual devices. It also allows them to test their apps on various Android device settings. The various devices include tablets, phone, and Android TV. Besides, the emulator offers simulation option for different hardware features including network latency, GPS functions, and motion sensors among others.

Developers can write better codes in a faster way thus increasing productivity because of the advanced completion of codes, analysis, and refactoring. This is achieved by the suggestions for the user types in the code editor (Android Studio, 2016)

### Service Oriented Web Server Application

NetBeans is an integrated development environment that is free and can be used to develop applications for various platforms such as personal computers, smartphones, and tablets. It also helps in making HTML5 applications. The NetBeans Editor helps the developer to create smart codes because it indents the lines of the codes and matches the various words and brackets. It also highlights the source code in a syntactic and semantic manner. Furthermore, it is rich with code templates and tips, as well as code generators.

The NetBeans IDE enhance the management of the projects by offering a variety of ways to view the projects. For instance, one can view multiple project windows and few helpful tools thus allowing developers to concentrate on their data faster. Developers can quickly detect errors in their codes using the various tools such as FindBugs tool offered by NetBeans IDE. Besides, they can set breakpoints in their source codes and monitor the executions of the program. (NetBeans, 2016)

### MySQL Relational Data Base Management System

GlassFish is a project that was developed by Sun Microsystems to support Java EE platform. It can enhance the development of scalable and portable applications.

The GlassFish Server reduces the footprint’s size by loading it when necessary only. Thus, the load time is reduced significantly saving time. Besides, it reduces the amount of resources used by the application because of the limited footprint sizes.

The developers can choose the tools they are conformable with to use on GlassFish. For instance, they can adopt either NetBeans or Eclipse to develop their applications with GlassFish. Developers are also able to save time by using GlassFish along other tools such as Eclipse and NetBeans. This is because the combination reduces the typical six steps in development to only three steps. Besides, it uses sessions to store data that eliminate the need to repopulate the session during the development process.

An extra benefit of GlassFish Server is that it is available in many languages. As such, it can be used by developers from around the world comfortably because they can choose the languages they are comfortable with (Oracle)

### Development Tools

A web service is a software that can be found on the internet and send messages through the standardized XML messaging system. It uses XML to transmit messages. It also supports the movement of information from the client to server and back. It also supports the exchange of documents (Saab, Haddad, Coulibaly, Melliti, Moreaux, & Rampacek, 2009)

### Version Control Software

A web server is a system based on the computer and works as a processor of the request through HTTP to access information from the World Wide Web. It allows users to create websites and configure their security. It also lets users transfer files through the File Transfer Protocol. It also permits the creation of virtual directories. (Web Server (IIS) Role Overview, 2016)

XML SOAP is a protocol is lightweight in nature and can be used to support the exchange of information that is structured in a decentralized environment. XML Soap has the extensibility features. The extremes of the application can negotiate the behaviors of the various commands that are supported. Besides, the header element in the XML Soap gives developers the opportunity to include additional information such as credential items that enhance controlled access. XML Soap can also support the exchange of information over various protocols. The independence of the messaging framework from SOAP gives users the freedom to choose from the variety of communications protocols (Server)

For our database, we decided to use MySQL because it can run on many operating systems and is a free and open sourced management system. In addition, it also allows solid data security layers that protect sensitive data from intruders. (MySQL Administration Guide, 2015). It also supports Java language which is the coding language we will be using to develop our application.

MySQL is a database server that is a robust, structured query language, and supports multiple users. It is used to store data, e-commerce, and support applications that require storage of users’ information. As for security, MySQL has a secure and flexible system that supports password and privileges. As such, it can prevent unauthorized access to the data stored. Also, it uses encryption of passwords whenever a user connects to a server.

MySQL offers a rich source of data types that users can use when they store their data. The advanced data types provided by MySQL include BLOB, Enum, and Binary besides the standard data types such as date, varchar, text, and double Float. This is an essential feature that makes developers from around the world have a great experience in the use of MySQL. This is because it can display error messages in many languages. It also supports many character sets from around the world. Furthermore, the time zone of the server can be changed dynamically, and can be adjusted manually as well ("MySQL :: MySQL 5.7 reference manual…", n.d.).

GitHub repository is an online platform that stores tools necessary for software development. It connects over 15 million people around the world to help them share, learn, and work together to build software.

The GitHub repository has several tools that have been tested in the field that can be used for projects. All the tools have these features of being tested before being deployed in the repository. Besides, the tools are free and can be accessed by anyone in the community.

Users have the opportunity to have their problems solved by other members of the repository. As such, GitHub offers a platform that can support the integration of efforts to address problems in the repository.

GitHub workflow allows the review of codes in a cooperative manner. Members can create a branch and make commitment that attracts the Pull requests from people around the world. These people can help to suggest necessary changes for the projects ("Build software better, together", n.d.).

An important aspect of a user account system is how user passwords are protected. To protect our passwords, we decided to use Hash Password, this way we can protect our client’s private information and prevent corruption. Hash Password generates a long random salt using CSPRNG.

The general workflow for account registration and authentication in a hash-based account system is as follows:

1. The user creates an account.
2. Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password ever written to the hard drive.
3. When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database).
4. If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
5. Steps 3 and 4 repeat every time someone tries to login to their account.

(Security, 2016)

SSL(Secure Socket Layers) is a typical security technology for creating an encrypted link between a server and the client. It allows credit card numbers, and login credentials to be transmitted securely. Sensitive information like such normally gets sent between the browsers and web servers in plain text which will leave you vulnerable to eavesdropping. If an attacker can intercept all data being sent between a browser and a web server, they can see and use that information. (digicert, 2016)

## Database

## Server Side

## Client Side

### User Interface Design

The technique we used for designing the user interface for this application was to make the application predictable and simple for the user (Garrett).

**Theme**

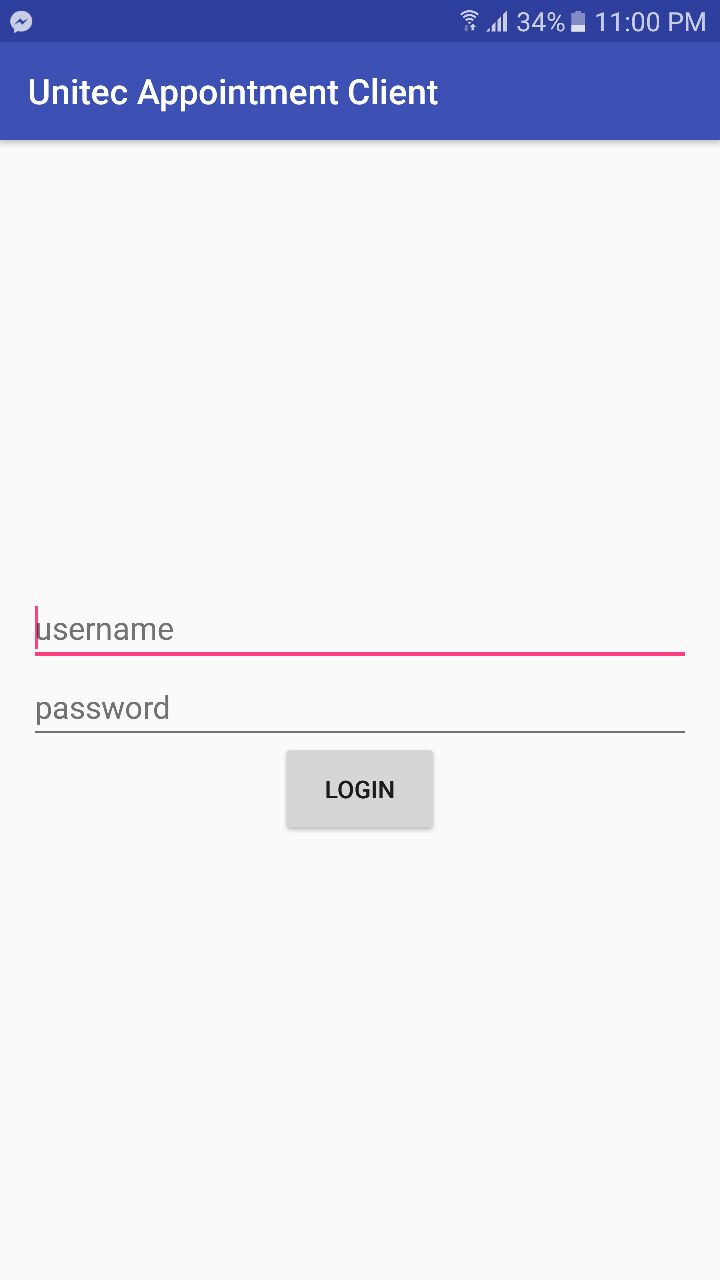
The entire theme is designed using Unitecs theme. We used common UI elements and created consistency throughout the application. We decided to strategically use the colours of Unitec which are green and white. We wanted to incorporate as much of Unitec as we could as we wanted staff and students to be familiar with the new application. As you can see the shades of green will help the user become more familiar with the application they are using. It will also make the application feel authentic that it belongs to Unitec.

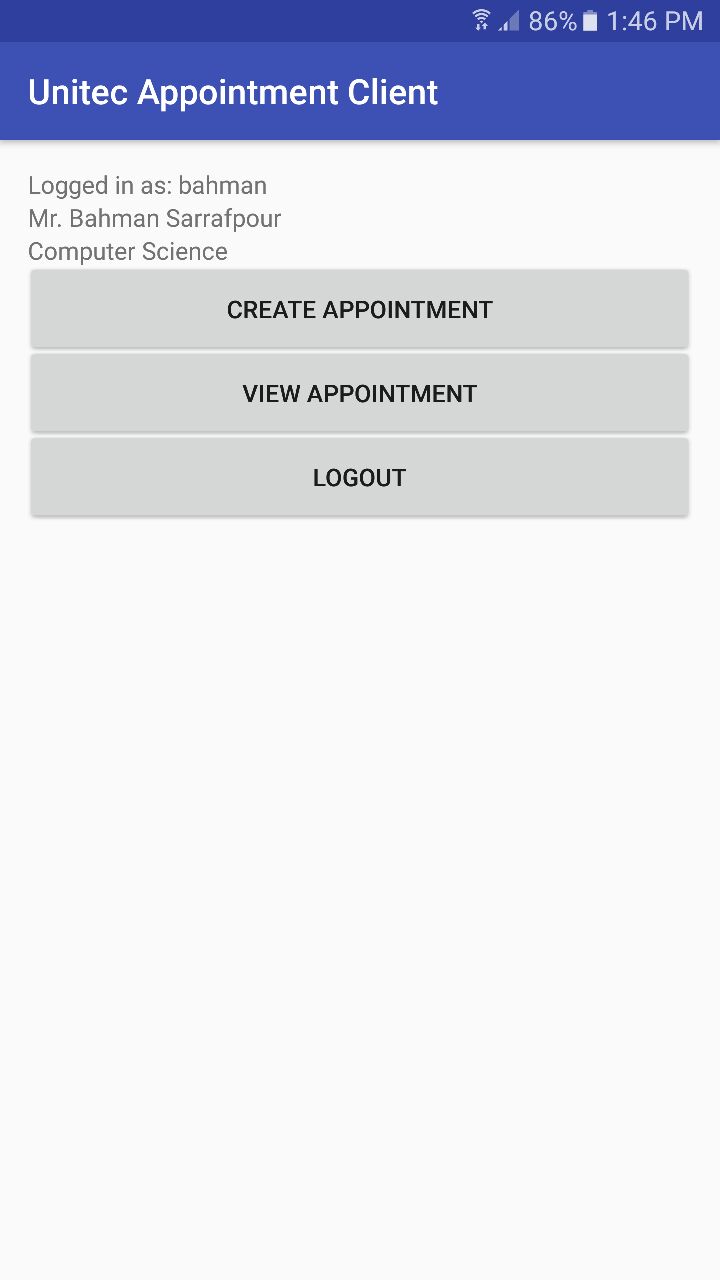


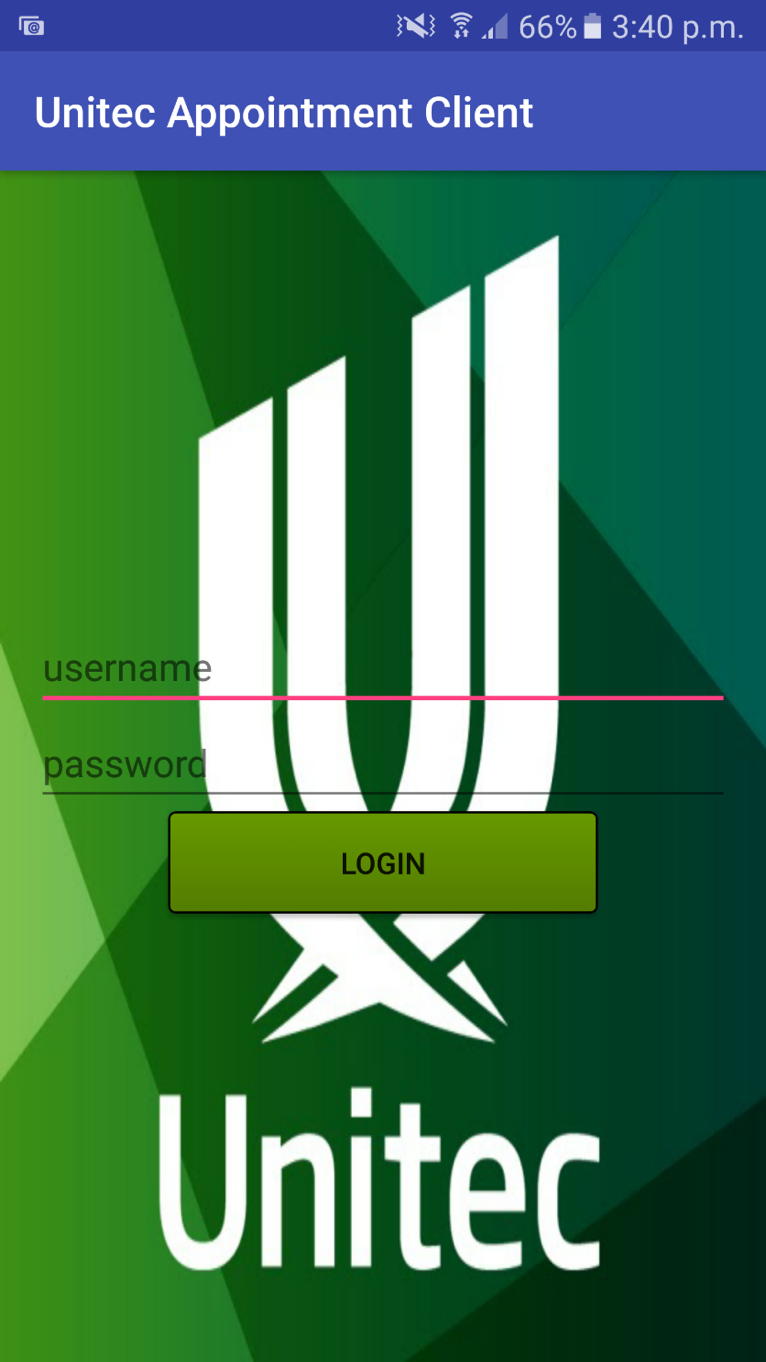
**Target Audience**

The audience we are targeting for the android application would be current students and lecturers who would use this application. Also, any administrators who manage the database.

### Basic Interface Design

Log in page Home page



Create appointment Create Appointment (Input Data)

### Final User Interface Design

Login Homepage

Make appointment(lecturers) Select appointment time slot

Appointment created Cancel Appointment

# Testing and performance

### Testing and Performance

User:

Student

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement to test | Test data input | Expected Outcome | Actual Outcome |
| Student login | Enter username and password. | Student is logged in successfully. | Success |
| Student logout | Select logout tab from homepage | Student is logged out | Success |
| Make an appointment | Select make an appointment from home page.  Select what lecturer whom you want to make an appointment with.  Select available time slot available.  Appointment has been created. | Student has an appointment.  Student has successfully created an appointment. | Success |
| View an appointment | Select view an appointment from home page. | All available appointments are listed.  Student can successfully view appointment. | Success |
| Delete appointment | Select view appointment tab.  Select which appointment you want to delete.  A box will appear to confirm deletion.  Select yes. | Appointment has been deleted. | Success |

User:

Lecturer

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement to test | Test data input | Expected Outcome | Actual Outcome |
| Lecturer login | Enter username and password | Lecturer is logged in successfully | Success |
| Lecturer logout | Select logout tab from homepage | Lecturer is logged out | Success |
| Lecturer add appointment time slot | Enter date.  Enter start time and end time hours and minutes (24hr).  Select create appointment. | Appointment time has been created.  Student can now view lecturer’s appointments. | Success |
| Lecturer view appointment | Lecturer needs to be logged in.  Select view appointment. | Appointments that are booked will be displayed. | Success |
| Notification sent by email when appointment is booked by student | Student makes an appointment. | Email notification will be sent to the lecturer’s email advising new appointment.  Lecturers outlook calendar has been updated. | Success |
| Delete appointment | Select view appointment.  Select which appointment you want to be deleted. | Appointment has been deleted.  Lecturers outlook calendar will update automatically. | Success |
| Notification sent by email when appointment is cancelled by student | Student needs to cancel appointment from their end | Email notification sent to lecturer’s outlook.  Outlook calendar updated. | Success |

### User acceptance testing

**Name:** Abdullah Alhajjaj **Email:** abdullah.alhajjaj@gmail.com

**Date:** 15/11/2016 **Comment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Tasks* | *Poor* | *Fair* | *Good* | *Excellent* |
| *Mobile design* |  |  |  |  |
| *Functionality* |  |  |  |  |
| *Performance* |  |  |  |  |
| *User Interface Design* |  |  |  |  |

**Name:** **Email:**

**Date:** **Comment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Tasks* | *Poor* | *Fair* | *Good* | *Excellent* |
| *Mobile design* |  |  |  |  |
| *Functionality* |  |  |  |  |
| *Performance* |  |  |  |  |
| *User Interface Design* |  |  |  |  |

**Name:** **Email:**

**Date:** **Comment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Tasks* | *Poor* | *Fair* | *Good* | *Excellent* |
| *Mobile design* |  |  |  |  |
| *Functionality* |  |  |  |  |
| *Performance* |  |  |  |  |
| *User Interface Design* |  |  |  |  |

**Name:** **Email:**

**Date:** **Comment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Tasks* | *Poor* | *Fair* | *Good* | *Excellent* |
| *Mobile design* |  |  |  |  |
| *Functionality* |  |  |  |  |
| *Performance* |  |  |  |  |
| *User Interface Design* |  |  |  |  |

**Name:** **Email:**

**Date:** **Comment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Tasks* | *Poor* | *Fair* | *Good* | *Excellent* |
| *Mobile design* |  |  |  |  |
| *Functionality* |  |  |  |  |
| *Performance* |  |  |  |  |
| *User Interface Design* |  |  |  |  |

# User Documentation

## User Functionality

### Lecturer

Can do the following:

* They must be a staff member at Unitec and registered in the database to log into the android application.
* Can create appointments.
* Can cancel appointments.
* Can view, cancel and update appointments using their Outlook calendars.

### Student

Can do the following:

* Must be enrolled in Unitec and registered in the database to log into the android application.
* Students can view appointments if an appointment has been made.
* Students can view lecturers.
* Students can make an appointment with a lecturer.
* Students can cancel appointments.

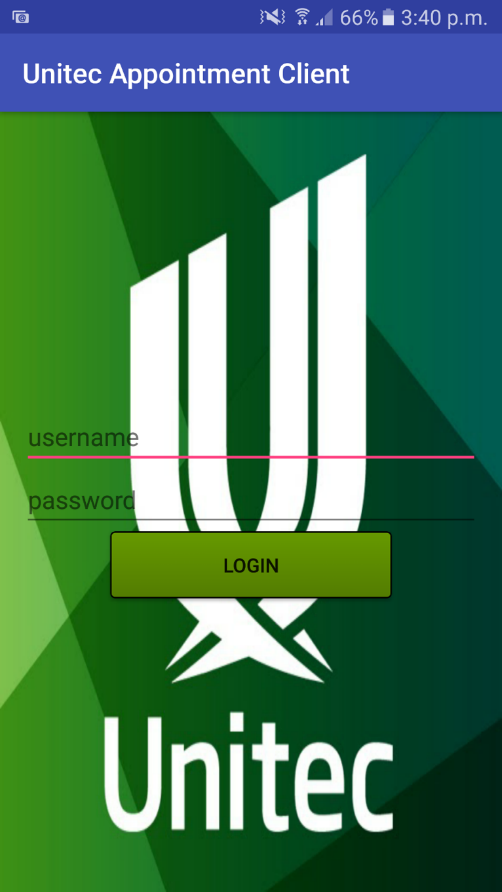
## Usage

Here we have created a guide to help first time users use the application.

### Lecturer

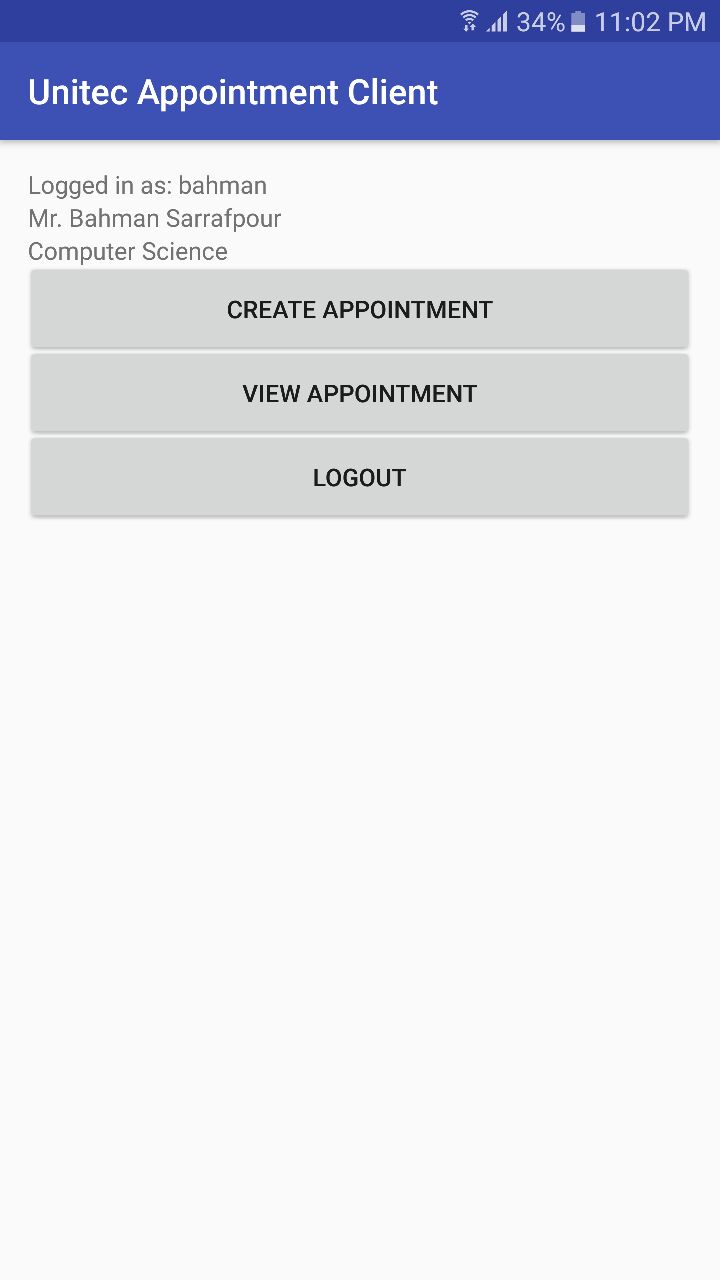
Firstly, as a lecture you need to log into the application by entering your

1. Username
2. Password
3. Select login.

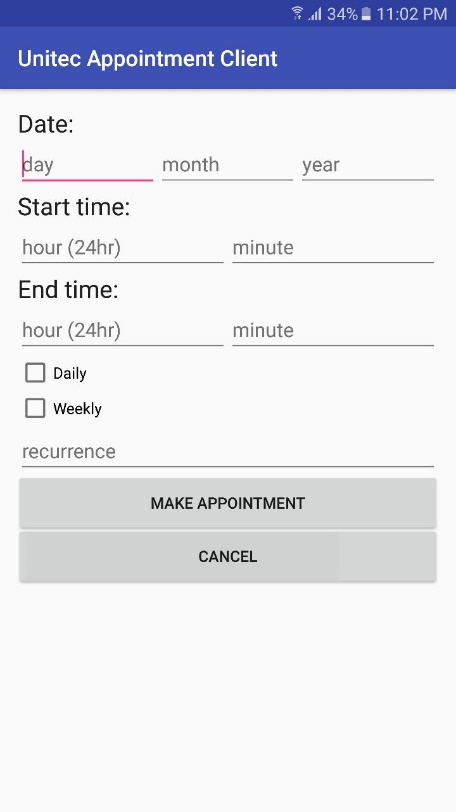
Be sure to enter in the correct details or you will get a message that says password or username not recognized.

Next you will be logged in and directed to the home page and your details will be available in the top of the screen. You will be able to see ‘Create an appointment’ tab, ‘View appointment’ tab and ‘Logout’ tab.

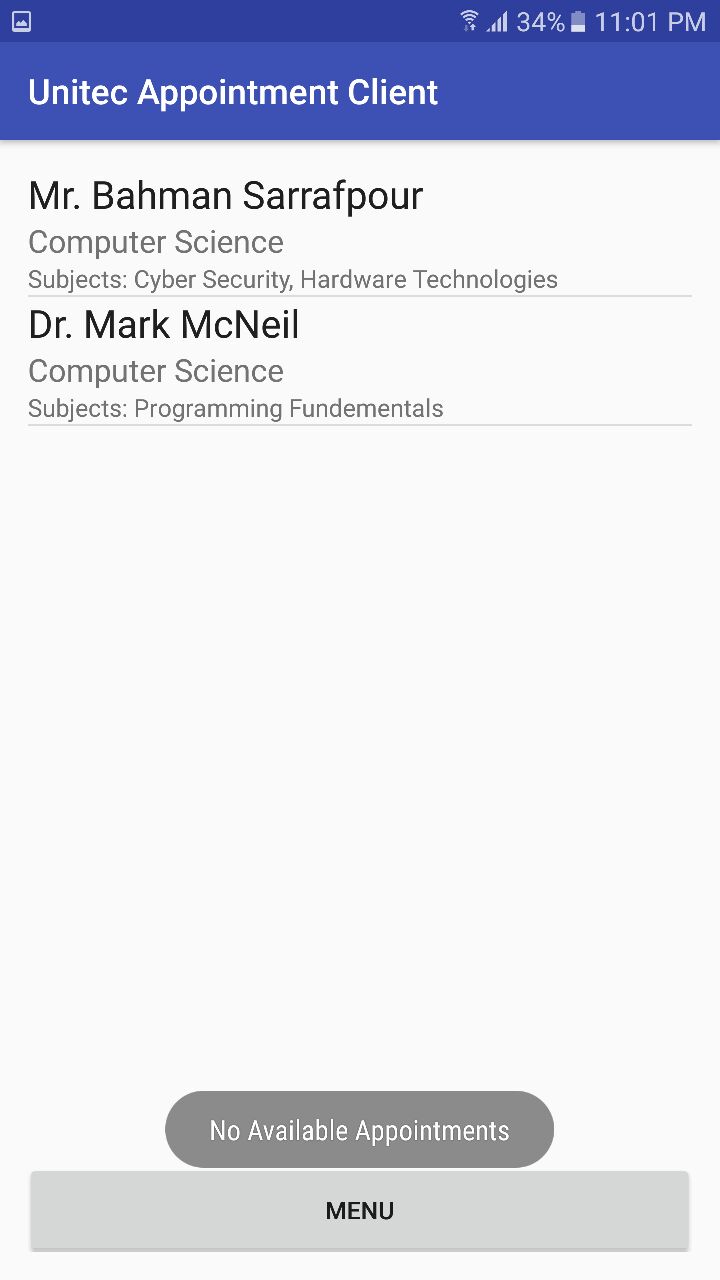
The next step is to select whether you want to create an appointment, view your current appointments or if you want to logout. We will be creating appointment in this next step.

1. Select Create Appointment tab

Once you have selected this tab you will see this page available in the image below. Here you will need to create an appointment time slot in which students will then be able to see as available and then can book an appointment with the lecturer. To create the appointment time, you will need to input all details listed on this page.

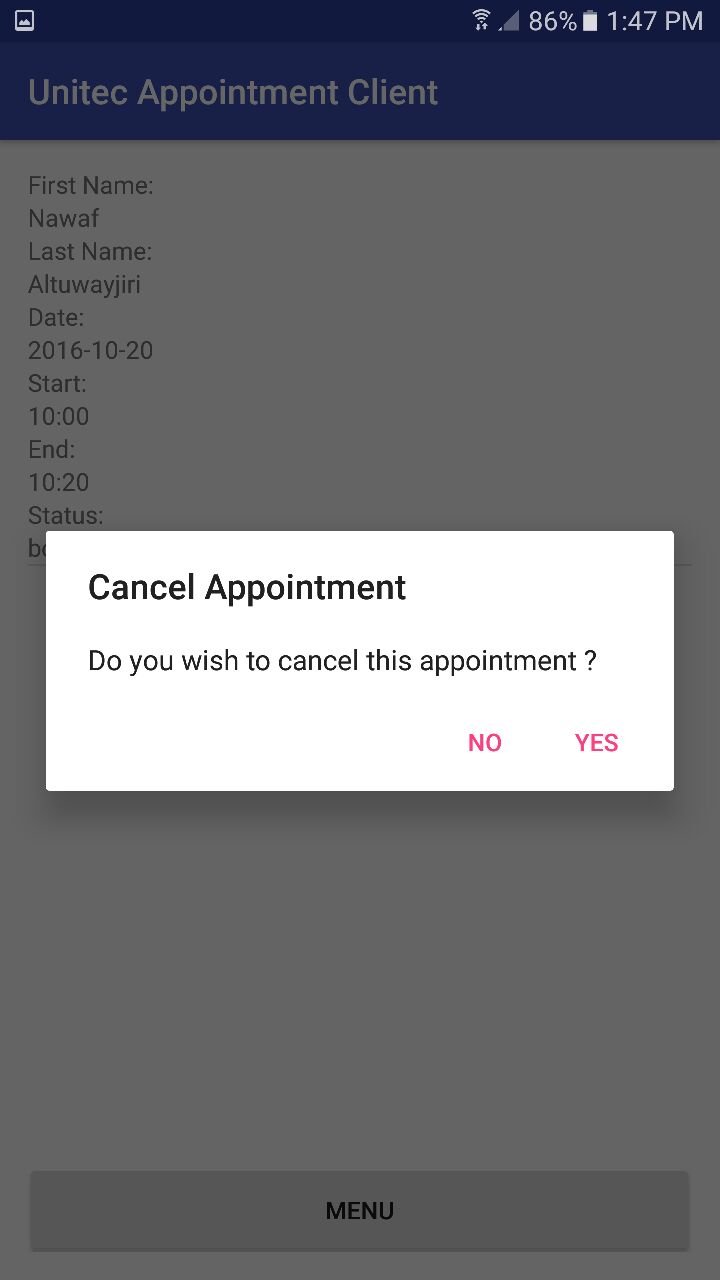
1. Input day/month/year.
2. Input the appointment start time hour (24) and minutes.
3. Input the appointment end time hour (24) and minutes.
4. Select if you want the appointment to recur daily or monthly and enter in the number of how many weeks or days you want it to recur.
5. Select Make Appointment and appointment will then be created.
6. Select cancel appointment and it will not be created.
7. If a message pops up fill in the rest of the details required.

Once the appointment time has been created students will be able to view it and book it. You will then be redirected back to the homepage and you can now select the view appointment tab which will display all the appointments students have made.

1. Select view appointment tab.
2. Appointments booked by students will be listed with all details. (need to change this image)

To cancel an appointment

1. Select appointment you want to cancel.
2. A box will pop up to cancel appointment or to cancel cancelling the appointment.
3. Select cancel appointment
4. An email notification will be sent to your email and the students email saying the appointment has been cancelled.

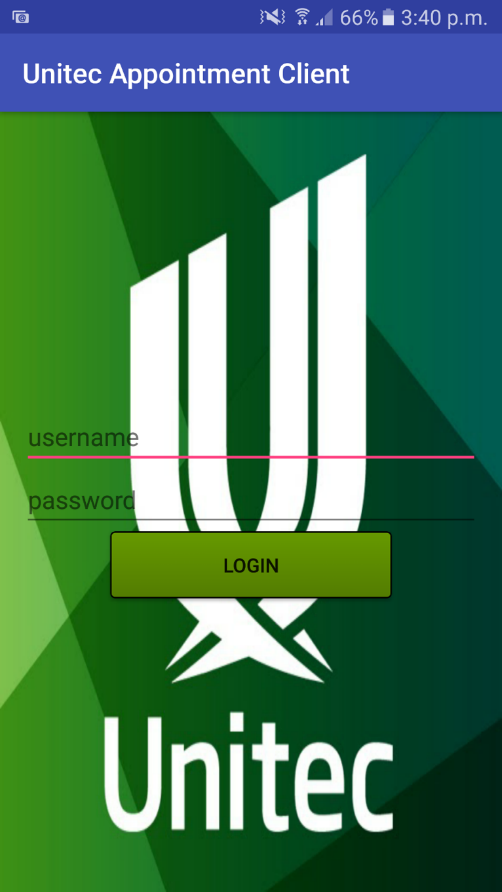


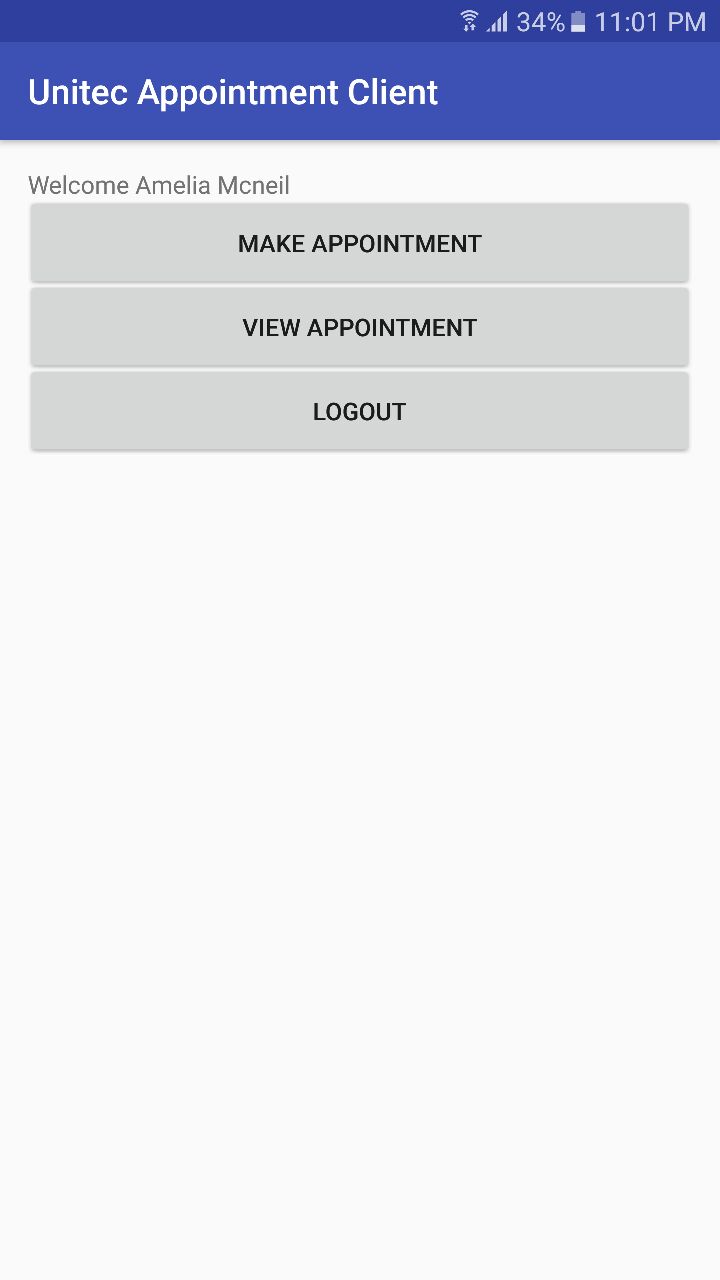
Once the appointment has been cancelled you will be redirected to the main page. The last option is to logout. Once you have finished with managing your appointments you can select logout which will log out of the application and return you back to the login page.

### Student

As a Student, you need to log into the application by entering your

1. Username
2. Password
3. Select login.

Be sure to enter in the correct details or you will get a message that says password or username not recognized.

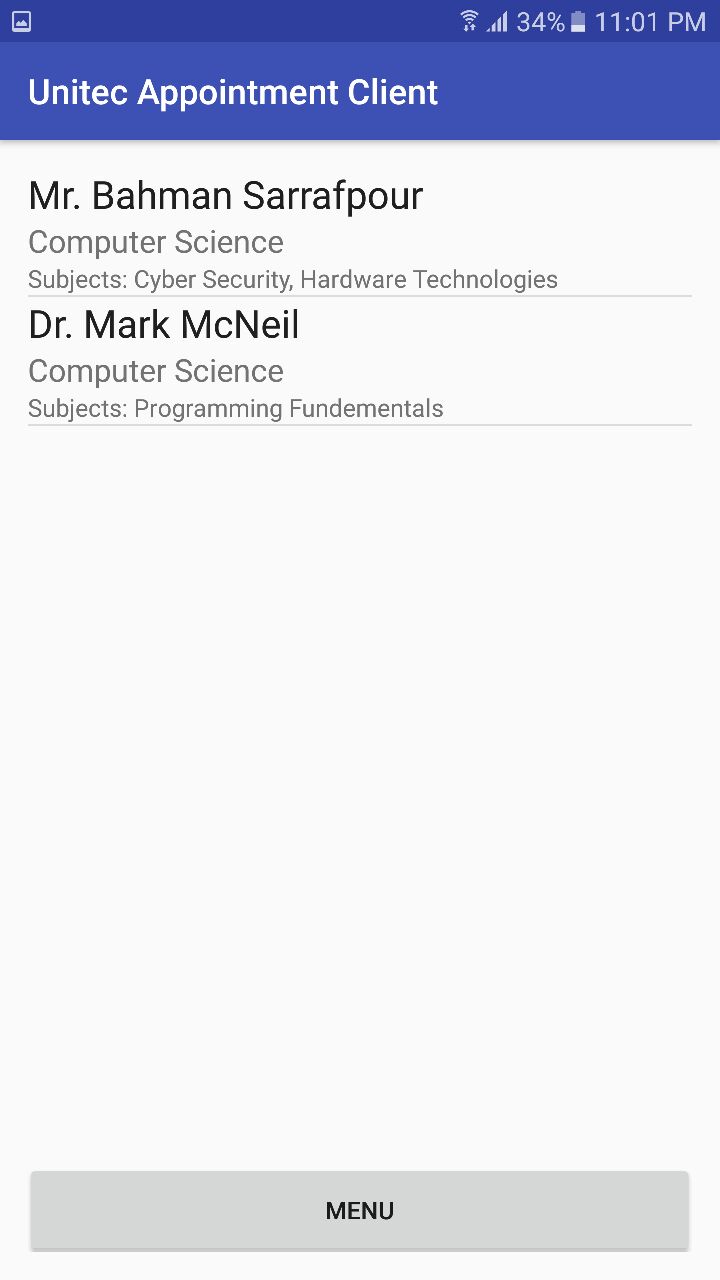
 From here you will be logged in and directed to the home page where your details will be available in the top of the screen and 3 tabs will be available. You will be able to see ‘Make Appointment’ tab, ‘View Appointment’ tab and ‘Logout’ tab.

The next step is to select whether you want to create an appointment, view your current appointments or if you want to logout. We will be creating appointment in this next step.

1. Select Make Appointment tab

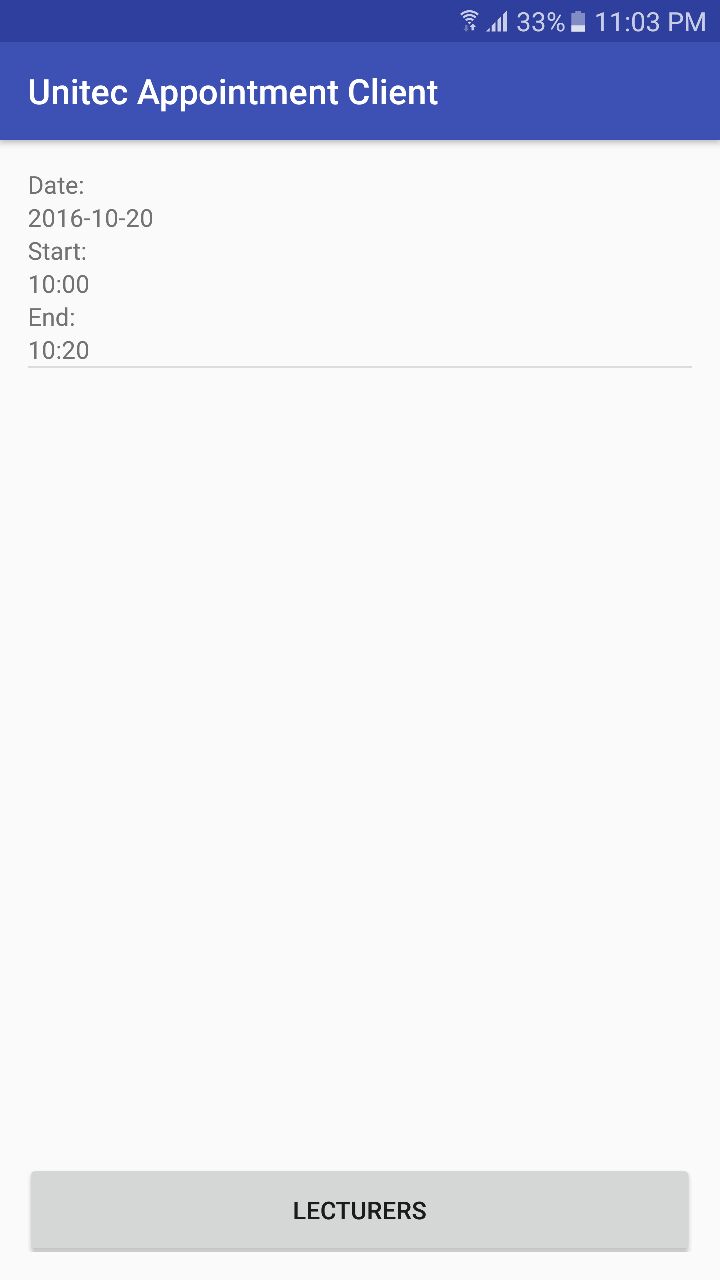
Once you have selected the make an appointment tab you will be able to see all your lectures and all the appointments created by them (note. If a lecturer has no appointments available, then a message will pop up advising no available appointments.)

To create an appointment

1. ****Select what lecturer you want to make an appointment with.

Once you select a lecturer all the available appointments with that lecturer will be listed

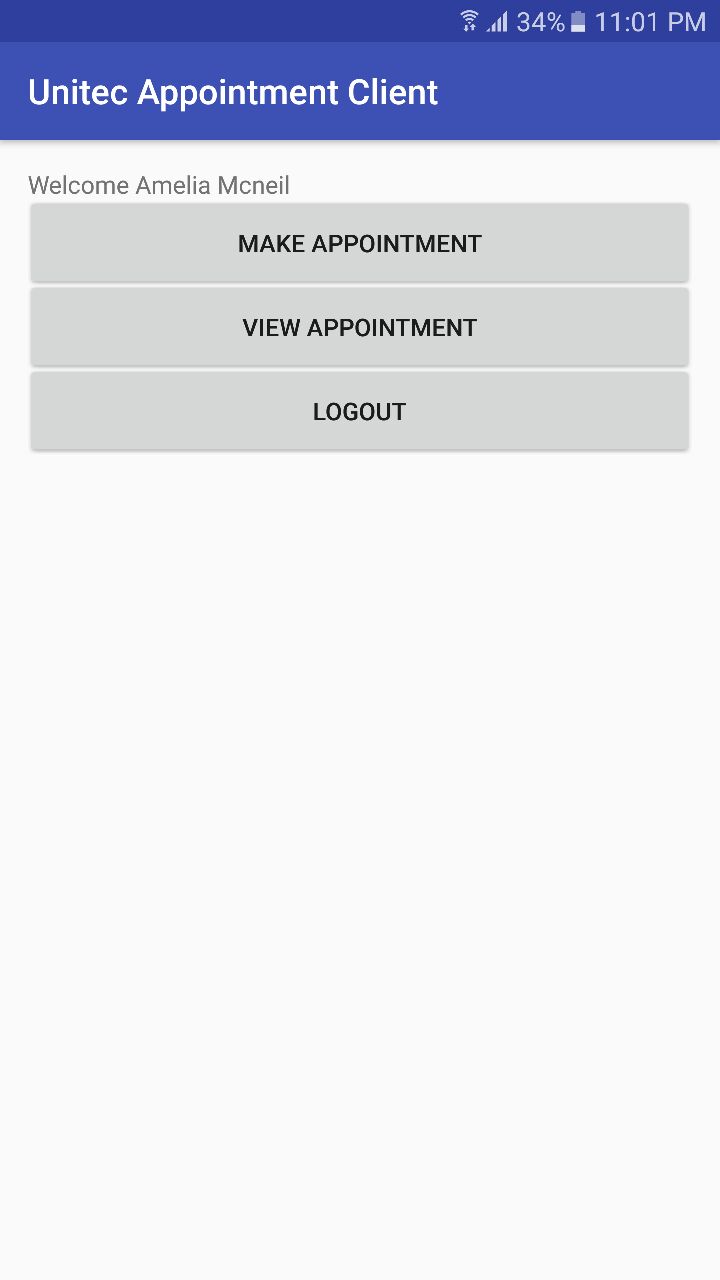
1. Select which time you would like to book.



Now that you have selected an appointment a message will pop up saying appointment has been created. An email notification will then be sent to your lecturer’s email that is registered in the database and updated to their outlook calendar. The student will also receive a message to their email.

You will be then redirected back to the main page; next you will be able to view appointments.

To view appointments

1. select the appointment tab

# Logs

## Project Planning

Gantt Chart

## Meeting Logs

Communication in the project was very important because we were working on different tasks at the same time. We wanted to keep track of the task each member was working on and effectively work together successfully as a group even when we were apart. We had daily meetings with our group members and weekly meetings with our supervisor and in addition to that bi-weekly meetings with all the groups in the course to present our progress.

To document our meetings, we have created three separate table logs which are Supervisor meetings, Group meetings and Bi-weekly meetings. This was to track the purpose of the meetings, the location, time, date and the attendees.

### Supervisor Meetings

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Topic | Time | Attendees |
| 01/08/2016 | Project proposal resubmission | 3pm | Amelia  Marzouq  Nawaf |
| 12/08/2016 | New group member | 2pm | Amelia  Marzouq  Nawaf  Oshada |
| 26/08/2016 | Catch up meeting -Progress check | 2pm | Amelia  Marzouq  Nawaf  Oshada |
| 01/09/2016 | Catch up meeting -Progress check | 12pm | Amelia  Marzouq  Nawaf |
| 09/09/2016 | Catch up meeting -Progress check | 1pm | Amelia  Marzouq  Nawaf  Oshada |
| 19/09/2016 | Catch up meeting -Progress check | 4pm | Amelia  Marzouq  Nawaf |
| 10/10/2016 | Catch up meeting – Progress check | 2:30pm | Amelia  Marzouq  Nawaf |
| 2/11/2016 | Catch up meeting – Progress check | 1:30pm | Amelia  Marzouq |
|  | Catch up meeting – Progress check |  |  |

### Bi-weekly Meetings

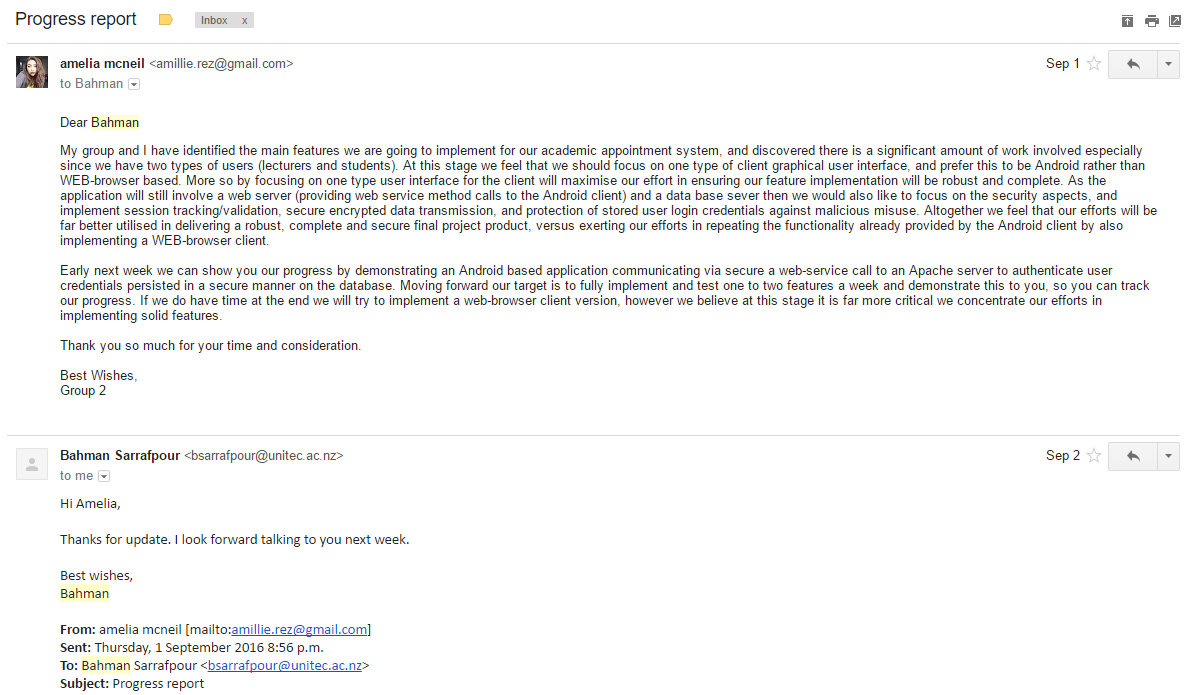
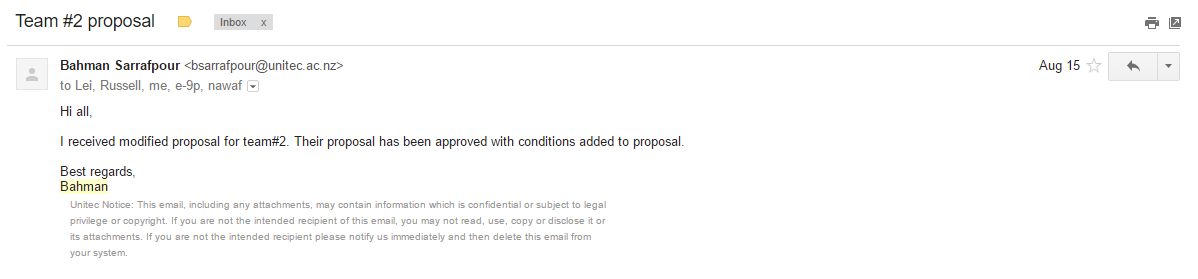
|  |  |  |  |
| --- | --- | --- | --- |
| Date | Topic | Time | Attendees |
| 22/07/2016 | Information Session | 11am |  |
| 11/08/2016 | Proposal Defence | 9:45am – 10:45am | Amelia  Marzouq  Nawaf |
| 26/08/2016 | Meeting 1  Presentation - Progress report 1 | 10am – 11am | Amelia  Marzouq  Nawaf  Oshada |
| 09/09/2016 | Meeting 2  Presentation - Progress report 2 | 10am – 11am | Amelia  Marzouq  Nawaf |
| Did not attend Bi-weekly meeting (Progress report 3) Had meeting prviously | | | |
| 21/10/2016 | Meeting 4  Presentation – Progress report 4 | 10am – 11am | Amelia  Marzouq  Oshada |

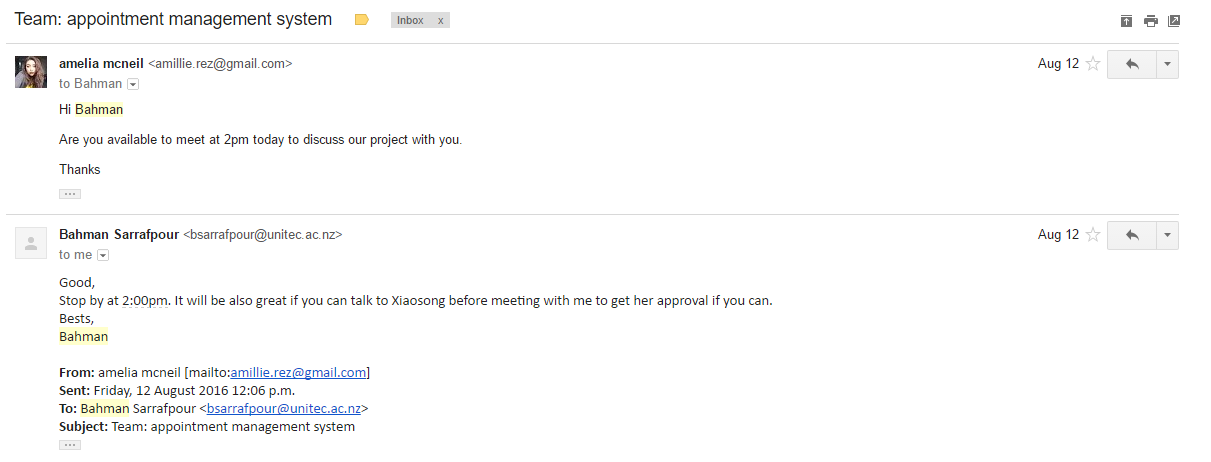
### Group Meetings

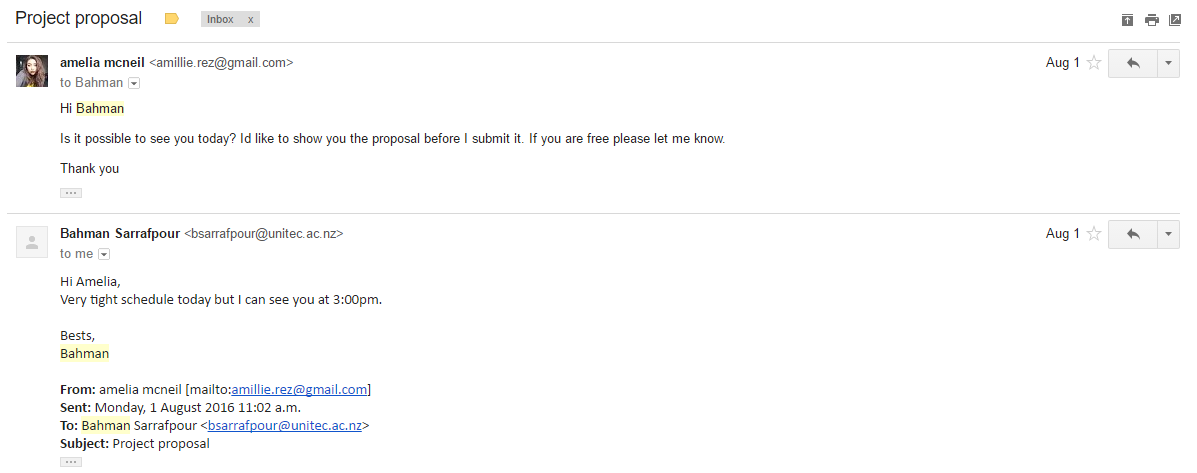
|  |  |  |  |
| --- | --- | --- | --- |
| Date | Topics | Time | Attendees |
| 04/08/2016 | Project proposal write up | 5:30pm - 6:30pm | Amelia  Marzouq  Nawaf |
| 09/08/2016 | Re submission of proposal with new scope |  |  |
| 11/08/2016 | Team catch up/ Progress check up | 3 - 3:30pm | Amelia  Marzouq  Nawaf  Oshada |
| 15/08/2016 | Proposal resubmission  (accepted) | 2:00pm - 3:00pm | Amelia  Marzouq  Nawaf  Oshada |
| 04/09/2016 | Project re evaluation  -Decide to focus on one application | 12pm - 2pm | Amelia  Marzouq  Nawaf  Oshada |
| 07/09/2016 | Team catch up/ Progress check | 11am - 1pm | Amelia  Marzouq  Nawaf  Oshada |
| 08/09/2016 | Team catch up/ Progress check | 2pm - 3pm | Amelia  Marzouq  Nawaf |
| 14/09/2016 | Team catch up/ Progress check | 12pm - 1pm | Amelia  Marzouq  Nawaf  Oshada |
| 25/09/2016 | Team catch up/ Progress check | 5:30pm - 9pm | Amelia  Marzouq  Nawaf  Oshada |
| 26/09/2016 | Team catch up/ Progress check | 5pm - 7pm | Amelia  Marzouq  Nawaf  Oshada |
| 6/10/2016 | Team catch up/ Progress check | 1:30pm - 2:15pm | Amelia  Marzouq  Nawaf  Oshada |
| 12/10/2016 | Team catch up/ Progress check | 12pm - 3pm | Amelia  Marzouq  Nawaf  Oshada |
| 25/10/2016 | Team catch up/ Progress check | 11am - 12pm | Amelia  Marzouq  Nawaf  Oshada |
| 4/10/2016 | Team catch up/ Progress check | 3pm - 5pm | Amelia  Marzouq  Nawaf  Oshada |
| 8/11/2016 | Team catch up/ Progress check | 1pm - 2pm | Amelia  Marzouq  Nawaf |

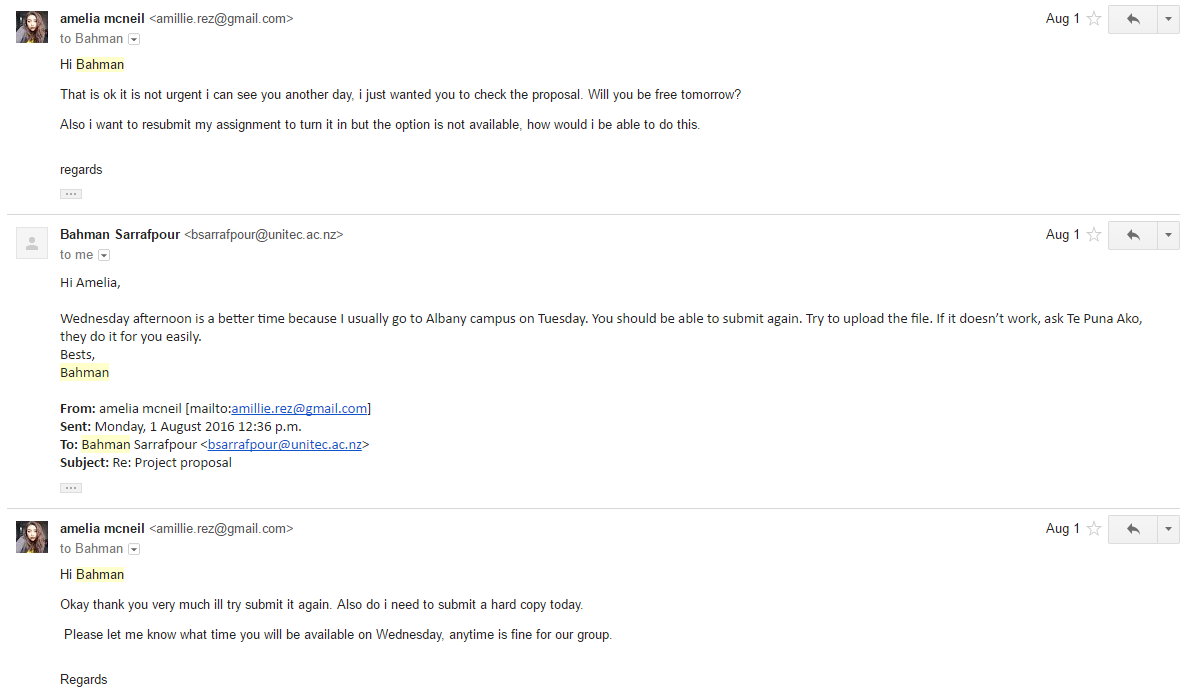
## Supervisor Email records

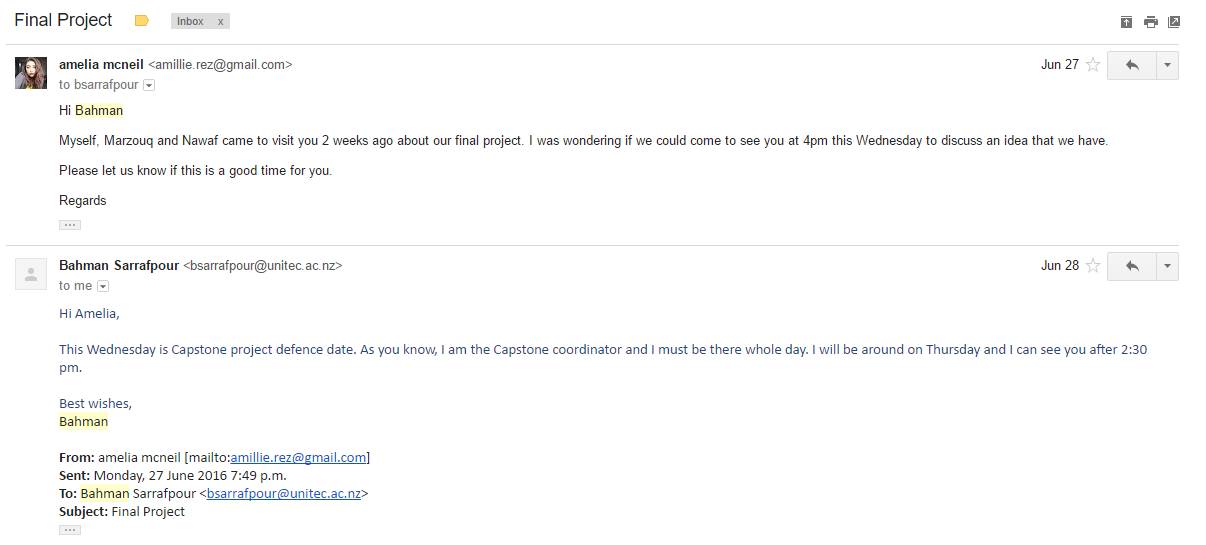
Here we have listed the emails that were sent between the Project supervisor and the project Manager. Most of these emails were to organize meetings which was to show the progress after each sprint, this was part of the methodology process called Scrum Development. After each feature set we organized a meeting with Bahman to show him what we had completed.

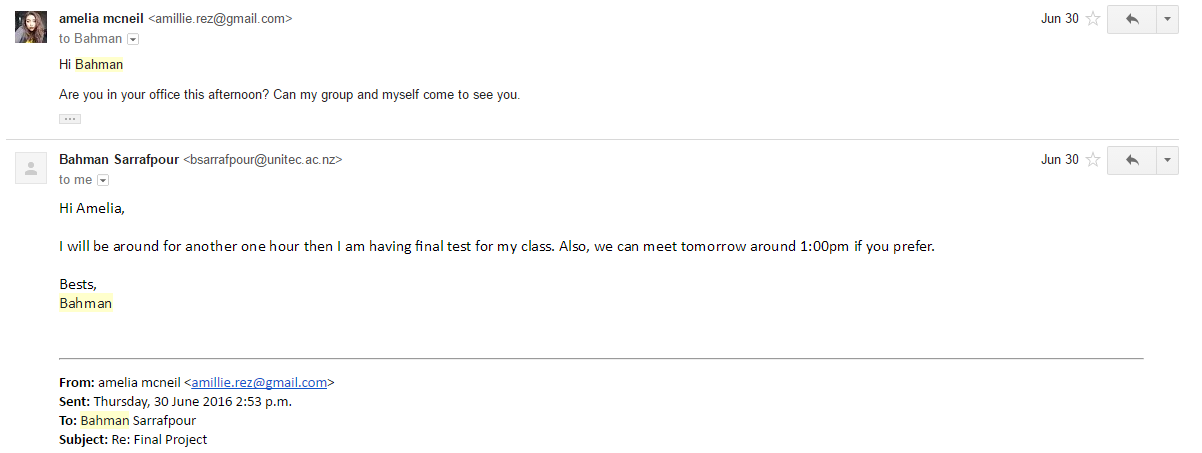












# Conclusion

## Individual Contribution review

### Amelia

The Appointment Management System has been the most challenging project in my bachelor’s degree but very rewarding. I have learnt many skills including organizational and leadership skills and gained experience by learning new technologies. In this project, my role was developing the database, Testing and Documentation. I also organized team meetings and made progress checks.

My group was made up of 4 members with different backgrounds in IT, this was an advantage because with the experience we already had we could create a great project. I have gained experience from my team members because of the background experience we all had we could help each other and explain technical parts of the project that we may not have understood.

Our project supervisor was Bahman Sarrafpour, he played a big role in our project and we are very thankful to have had him as our supervisor. He was very supportive and kept us positive throughout the project. He was always available when we needed advice from him about problems we were experiencing and has guided us all the way.

As a group, we were all able to accomplish our goals and objectives as we had very good communication skills. This was important as a group, during the development phase we were all working on different tasks and had to be updated on each other’s work. We met a few times a week to review our tasks and helped each other if we had any problems. Our group was very friendly and

Overall I enjoyed my time working on this project and I am very happy with the outcome. I felt I was always allowed to have an input and my ideas were always taken into consideration. I never felt out casted in this project and I’m glad I got to work on this project with students who started this Bachelors course with me. I know the experience I gained from this project will be very useful in my future when I want to progress my career.

### Marzouq

### Nawaf

In this project, my role was Implementing the Server. Personally, I enjoyed the project and I think my group worked well together because we all got involved and communicated well. My group worked well in dividing tasks between each other and always helped each other in overcoming any challenging obstacles.

Our supervisor in this project was Bahman Sarrafpour, he played a big role in supporting us through this project and guided us when we needed advice. He was always available when wanted to meet him, we are very thankful to have had him as our supervisor because he led us to the successful completion of our project.

Communication is very important when in a team because everyone is consistently working on different tasks. I believe our project was successful because of the good communication in our group. We had daily group meetings to keep on top of the tasks we were all working on and we always let each other know if we had a problem. Because of the backgrounds we had in Networking, Programming, Database development and Business Intelligence we could learn new skills from each other and then work on tasks where some team members needed more help.

During this project, I have learned new skills and experience I can take with me into my professional career. The most exciting part of this project was being included in the design of the application and coming up with ideas and watching the application come together into a fully functional android mobile application.

Overall I enjoyed working on this project as my last final project in my BCS and learning new technologies that I had never used before. This has been the biggest project I have done in my life and now I can apply the skills I have gained in my professional career.

### Oshada

## Future Improvements

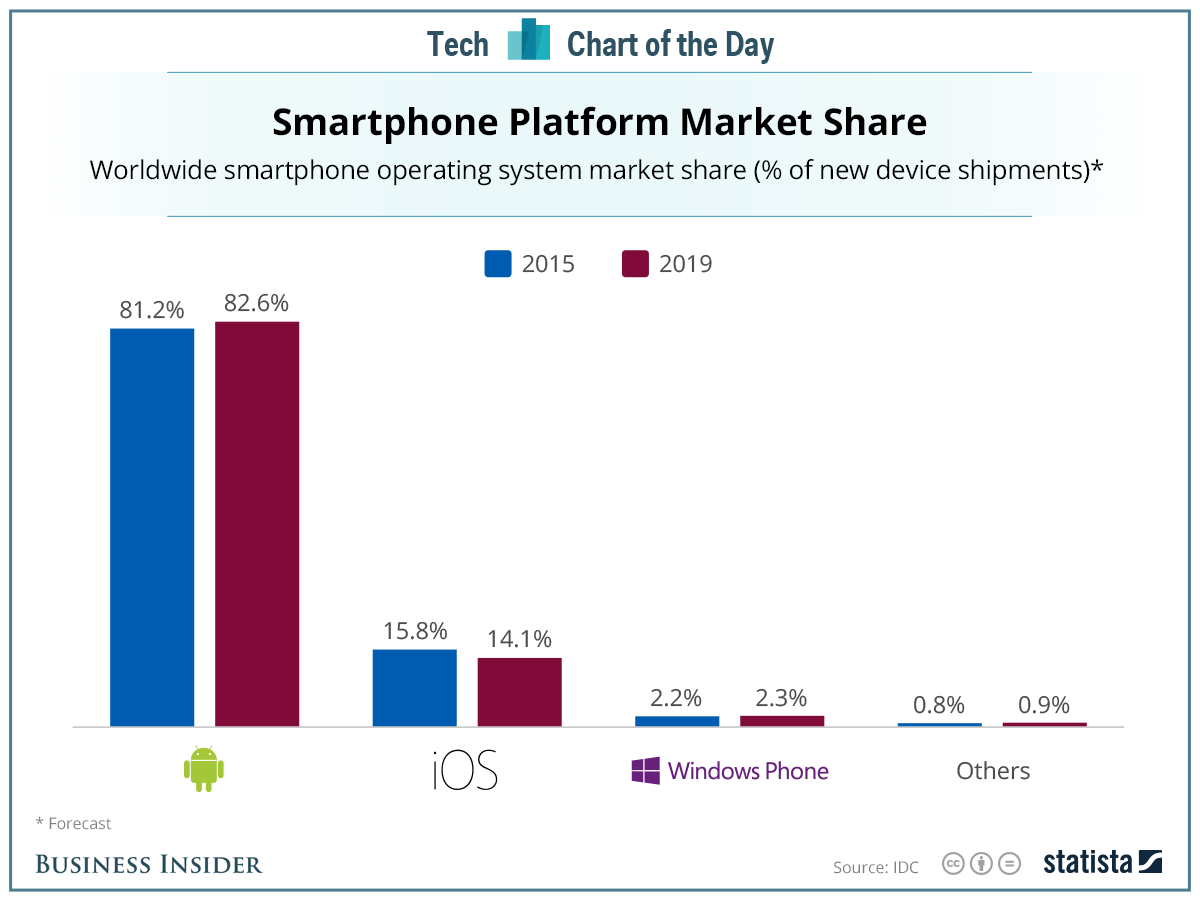
In this project, we’ve presented an Android Mobile Application which creates and manages appointments for Students and Lecturers at Unitec within the Computing Department. As we have completed our project we have decided that as a group we have many ideas for future improvements.

**Unitec Access**

Firstly, we would like all departments at Unitec to have access to this application. For this project, we focused solely on Students and Lecturers from the Computing Department as our focus was mainly on the development and success of the application. After testing the application on the Computing Department, we would look at the feedback from the users and implement the access of all departments.

**Availability on Platforms**

Another improvement would be to make this app available on more platforms for example, Apple iOS. Currently Apple is the second leading Smartphone Operating System on the market and developing an iOS application will allow students whom do not have an android Smartphone to have access to the application. (Rosoff, 2015)



**Adding a Web application**

The last and final improvement for our application would be to add a website and link it on the Unitec website. Originally when researching the project, we were considering doing a website however, the experience we had in a group we decided it would be better to develop an android mobile application and then consider the web based application in the future.

Once the application is finished we would start the development of a website and link it on the Unitecs website so when students are on a computer or do not have access to their phones they will be able to make the appointment through the website.

So far the project has been completed but in the future if Unitec wants to pursue the

## Summary

In this report, we have documented the process of developing the Appointment Management System Application. We created this report to document the projects timeline and the technologies used and the steps we have gone through to create this project, including research, design, implementation and testing. Here we have clarified in depths the work we have put into this system to create an effective Appointment Management System

This application was designed to effectively help Students and Lectures create and manage their appointments. We have succeeded in this as we have improved the communication between the Students and Lecturers by assisting them in the process of making their appointments through this application.

As a group, we could help each other in the tasks we were assigned, we also learnt new skills from each other in the implementation phase. Since the project has finished we can use these skills as we pursue our careers or even further our education.

# Appendix A

## Client Code

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.util.Log;  
  
**import** org.json.JSONObject;  
**import** org.ksoap2.SoapEnvelope;  
**import** org.ksoap2.serialization.PropertyInfo;  
**import** org.ksoap2.serialization.SoapObject;  
**import** org.ksoap2.serialization.SoapPrimitive;  
**import** org.ksoap2.serialization.SoapSerializationEnvelope;  
**import** org.ksoap2.transport.HttpTransportSE;  
  
  
**import** java.security.KeyManagementException;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.security.SecureRandom;  
  
**import** javax.net.ssl.HostnameVerifier;  
**import** javax.net.ssl.HttpsURLConnection;  
**import** javax.net.ssl.SSLContext;  
**import** javax.net.ssl.SSLSession;  
**import** javax.net.ssl.X509TrustManager;  
**import** java.security.cert.X509Certificate;  
  
*/\*\*  
 \* Class for making web service calls and obtaining results to and from web-server  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** WebService {  
  
 *//services available form web server* **public static** String *LOGIN\_METHOD* = **"login"**;  
 **public static** String *LOGIN\_PARAMETER* = **"userLoginDetails"**;  
 **public static** String *CREATE\_APPOINTMENT\_METHOD* = **"createAppointment"**;  
 **public static** String *CREATE\_APPOINTMENT\_PARAMETER* = **"appointmentDetails"**;  
 **public static** String *MAKE\_APPOINTMENT\_METHOD* = **"makeAppointment"**;  
 **public static** String *MAKE\_APPOINTMENT\_PARAMETER* = **"userDetails"**;  
 **public static** String *GET\_AVAILABLE\_APPOINTMENT\_METHOD* = **"getAvailableAppointment"**;  
 **public static** String *GET\_AVAILABLE\_APPOINTMENT\_PARAMETER* = **"userDetails"**;  
 **public static** String *BOOK\_APPOINTMENT\_METHOD* = **"bookAppointment"**;  
 **public static** String *BOOK\_APPOINTMENT\_PARAMETER* = **"appointmentDetails"**;  
 **public static** String *VIEW\_APPOINTMENT\_METHOD* = **"viewAppointment"**;  
 **public static** String *VIEW\_APPOINTMENT\_PARAMETER* = **"userDetails"**;  
 **public static** String *CANCEL\_APPOINTMENT\_METHOD* = **"cancelAppointment"**;  
 **public static** String *CANCEL\_APPOINTMENT\_PARAMETER* = **"appointmentDetails"**;  
  
 *//Namespace of the Webservice - can be found in WSDL* **private static** String *NAMESPACE* = **"http://unitecappointmentserver/"**;  
 *//Webservice URL - WSDL File location* **private static** String *URL* = **"https://192.168.1.134:8181/Unitec\_Appointment\_Server/AppointementServices?WSDL"**;  
 *//SOAP Action URI again Namespace + Web method name* **private static** String *SOAP\_ACTION* = **"http://unitecappointmentserver/AppointementServices/"**;  
  
 */\*\*  
 \* Create and initialise the activity view  
 \*  
 \** ***@param methodName*** *web service name  
 \** ***@param parameterName*** *web service parameter name  
 \** ***@param data*** *parameter data  
 \*/* **public static** JSONObject invokeWebService(String methodName, String parameterName, JSONObject data) {  
  
 JSONObject result = **null**;  
  
 **try** {  
 *// Create request* SoapObject request = **new** SoapObject(*NAMESPACE*, methodName);  
 *// Property which holds input parameters* PropertyInfo propertyInfo = **new** PropertyInfo();  
 *// Set Name* propertyInfo.setName(parameterName);  
 *// Set Value* propertyInfo.setValue(data.toString());  
 *// Set dataType* propertyInfo.setType(String.**class**);  
 *// Add the property to request object* request.addProperty(propertyInfo);  
 *// Create envelope* SoapSerializationEnvelope envelope = **new** SoapSerializationEnvelope(  
 SoapEnvelope.***VER11***);  
 *// Set output SOAP object* envelope.setOutputSoapObject(request);  
  
 *// Create HTTP call object* SSLConection.*allowAllSSL*();  
 HttpTransportSE httpTransportSE = **new** HttpTransportSE(*URL*);  
  
 *// Invoke web service* httpTransportSE.call(*SOAP\_ACTION* + methodName, envelope);  
 *// Get the response* SoapPrimitive response = (SoapPrimitive) envelope.getResponse();  
 *// Assign it to result variable static variable* result = **new** JSONObject(response.toString());  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 *//Return result to calling object* **return** result;  
 }  
  
 */\*\*  
 \* private class for application certificate trust manager, defined to trust all certificates  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private static class** SSLConection {  
  
 **private static** TrustManager[] *trustManagers*;  
  
 **public static class** TrustManager **implements** X509TrustManager {  
  
  
 @Override  
 **public void** checkClientTrusted(java.security.cert.X509Certificate[] x509Certificates, String s) {  
  
 }  
  
 @Override  
 **public void** checkServerTrusted(java.security.cert.X509Certificate[] x509Certificates, String s) {  
  
 }  
  
 @Override  
 **public** X509Certificate[] getAcceptedIssuers() {  
 **return new** X509Certificate[0];  
 }  
  
  
 }  
  
 */\*\*  
 \* Allows self-signed certifcates to be accepted  
 \*/* **public static void** allowAllSSL() {  
 HttpsURLConnection.*setDefaultHostnameVerifier*(**new** HostnameVerifier() {  
 @Override  
 **public boolean** verify(String hostname, SSLSession session) {  
 **return true**;  
 }  
 });  
  
 SSLContext context;  
 **if** (*trustManagers* == **null**) {  
 *trustManagers* = **new** TrustManager[]{**new** TrustManager()};  
 }  
 **try** {  
 context = SSLContext.*getInstance*(**"TLS"**);  
 context.init(**null**, *trustManagers*, **new** SecureRandom());  
 HttpsURLConnection.*setDefaultSSLSocketFactory*(context.getSocketFactory());  
 } **catch** (NoSuchAlgorithmException e) {  
 Log.*e*(**"allowAllSSL"**, e.toString());  
 } **catch** (KeyManagementException e) {  
 Log.*e*(**"allowAllSSL"**, e.toString());  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
*/\*\*  
 \* login activity, used by students and lecturers  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** LoginActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 **private** EditText **txtUsername**; *//login username* **private** EditText **txtPassword**; *//user password* **private** Button **btnLogin**; *//button to submit login details  
  
 /\*\*  
 \* Create and initialise the login activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_login***);  
  
 *//obtain references to activity sub-views* **txtUsername** = (EditText) findViewById(R.id.***txtUsername***);  
 **txtPassword** = (EditText) findViewById(R.id.***txtPassword***);  
 **btnLogin** = (Button) findViewById(R.id.***btnLogin***);  
  
 *// attach listener to handle button selection touch events* **btnLogin**.setOnClickListener(**this**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click evevnt  
 \*/* @Override  
 **public void** onClick(View v) {  
 *// create JSON object with user login details* JSONObject loginDetails = **new** JSONObject();  
 **try** {  
 loginDetails.put(**"username"**, **txtUsername**.getText().toString());  
 loginDetails.put(**"password"**, **txtPassword**.getText().toString());  
 *// pass Json object to asynchronous task thread responsible for web service call* **new** AsyncLoginTask(loginDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for logging in  
 \*  
 \** ***@author*** *Marzouq Almarzooq (1380949)  
 \** ***@author*** *Nawaf Altuwayjiri (1377387)  
 \*/* **private class** AsyncLoginTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userLoginDetails**; *// Json object with login details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userLoginDetails*** *Json object with user login details  
 \*/* **public** AsyncLoginTask(JSONObject userLoginDetails) {  
 *// initialise task object attributes* **this**.**userLoginDetails** = userLoginDetails;  
 **progressDialog** = **new** ProgressDialog(LoginActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing login details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*LOGIN\_METHOD*, WebService.*LOGIN\_PARAMETER*, **userLoginDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(LoginActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"false"**) == 0) {  
 *//invalid login* toast = Toast.*makeText*(LoginActivity.**this**, **"Invalid login details"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *//valid login* String user = **response**.getString(**"user"**);  
 String username = **userLoginDetails**.getString(**"username"**);  
 String firstName = **response**.getString(**"firstName"**);  
 String lastName = **response**.getString(**"lastName"**);  
 **if** (user.compareToIgnoreCase(**"lecturer"**) == 0) {  
 *//lecturer user valid login* Intent intent = **new** Intent(LoginActivity.**this**, MainLecturerActivity.**class**);  
 intent.putExtra(**"username"**, username);  
 intent.putExtra(**"firstName"**, firstName);  
 intent.putExtra(**"lastName"**, lastName);  
 intent.putExtra(**"title"**, **response**.getString(**"title"**));  
 intent.putExtra(**"department"**, **response**.getString(**"department"**));  
 startActivity(intent);  
 } **else** {  
 *//student user valid login* Intent intent = **new** Intent(LoginActivity.**this**, MainStudentActivity.**class**);  
 intent.putExtra(**"username"**, username);  
 intent.putExtra(**"firstName"**, firstName);  
 intent.putExtra(**"lastName"**, lastName);  
 startActivity(intent);  
 }  
 }  
 }  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.*makeText*(LoginActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.TextView;  
  
**import** java.util.List;  
  
*/\*\*  
 \* A view adapter class to customize the list view for displaying available appointments  
 \*  
 \* @author Oshada Koralage (1434048)  
 \*/***public class** AppointmentAdapter **extends** ArrayAdapter<AppointmentModel> {  
  
 */\*\*  
 \* Constructor for the adapter class  
 \*  
 \** ***@param context*** *context from parent display  
 \** ***@param appointments*** *collection of available appointments to populate the adapted list view  
 \*/* **public** AppointmentAdapter(Context context, List<AppointmentModel> appointments) {  
 **super**(context, 0, appointments);  
 }  
  
 */\*\*  
 \* Returns an adapted view for a given appointment item in the list  
 \*  
 \** ***@param position*** *index of item in the list view  
 \** ***@param convertView*** *list item view to be adapted  
 \** ***@param parent*** *parent view of all adapted list view items  
 \** ***@return*** *adapted list view item  
 \*/* @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) {  
  
 *//obtain an appointment for a given index position in the list view* AppointmentModel appointment = getItem(position);  
  
 *//if a adapted list item view has not been created, then inflate one from its respective layout template* **if** (convertView == **null**) {  
 convertView = LayoutInflater.*from*(getContext()).inflate(R.layout.***appointment\_item***, parent, **false**);  
 }  
  
 *//obtain references to the sub-view items contained within the adapted list view item* TextView lblDate = (TextView) convertView.findViewById(R.id.***lblDate***);  
 TextView lblStart = (TextView) convertView.findViewById(R.id.***lblStart***);  
 TextView lblEnd = (TextView) convertView.findViewById(R.id.***lblEnd***);  
  
 *//populate the sub-view items with details for an available appointment* lblDate.setText(appointment.getDate());  
 lblStart.setText(appointment.getStart());  
 lblEnd.setText(appointment.getEnd());  
  
 **return** convertView;  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Lecturer main menu activity, used by lecturers to access application functionality  
 \*  
 \* @author Oshada Koralage (1434048)  
 \*/***public class** MainLecturerActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 **private** TextView **lblLogin**; *//username display* **private** TextView **lblName**; *//full name display* **private** TextView **lblDepartment**; *//department display* **private** Button **btnCreateAppointment**; *//button to create appointments* **private** Button **btnViewAppointment**; *//button to view booked appointments* **private** Button **btnLogout**; *//button to log out* **private** String **username**; *//username* **private** String **firstName**; *//first name* **private** String **lastName**; *//last name* **private** String **title**; *//title* **private** String **department**; *//department  
  
 /\*\*  
 \* Create and initialise the menu activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main\_lecturer***);  
  
 *//obtain references to activity sub-views* **lblLogin** = (TextView) findViewById(R.id.***lblLogin***);  
 **lblName** = (TextView) findViewById(R.id.***lblName***);  
 **lblDepartment** = (TextView) findViewById(R.id.***lblDepartment***);  
  
 **btnCreateAppointment** = (Button) findViewById(R.id.***btnCreateAppointment***);  
 *// attach listener to handle button selection touch events* **btnCreateAppointment**.setOnClickListener(**this**);  
 **btnViewAppointment** = (Button) findViewById(R.id.***btnViewAppointment***);  
 *// attach listener to handle button selection touch events* **btnViewAppointment**.setOnClickListener(**this**);  
 **btnLogout** = (Button) findViewById(R.id.***btnLogout***);  
 *// attach listener to handle button selection touch events* **btnLogout**.setOnClickListener(**this**);  
  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 *// obtain extra information passed from previous activity via intent* **username** = intent.getStringExtra(**"username"**);  
 **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **title** = intent.getStringExtra(**"title"**);  
 **department** = intent.getStringExtra(**"department"**);  
  
 **lblLogin**.setText(**"Logged in as: "** + **username**);  
 **lblName**.setText(**title** + **" "** + **firstName** + **" "** + **lastName**);  
 **lblDepartment**.setText(**department**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click evevnt  
 \*/* @Override  
 **public void** onClick(View v) {  
 **if** (v == **btnCreateAppointment**) {  
 *//create appointment function selected* Intent intent = **new** Intent(MainLecturerActivity.**this**, CreateAppointmentActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"title"**, **title**);  
 intent.putExtra(**"department"**, **department**);  
 startActivity(intent);  
 } **else if** (v == **btnViewAppointment**){  
 *// create JSON object with user details to obtain booked appointments* JSONObject userDetails = **new** JSONObject();  
 **try** {  
 userDetails.put(**"username"**, **username**);  
 userDetails.put(**"type"**, **"lecturer"**);  
 **new** AsyncViewAppointmentTask(userDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 } **else** {  
 *//logout* Intent intent = **new** Intent(MainLecturerActivity.**this**, LoginActivity.**class**);  
 startActivity(intent);  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for obtaining booked appointments  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncViewAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userDetails**; *// Json object with user details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userDetails*** *Json object with user details  
 \*/* **public** AsyncViewAppointmentTask(JSONObject userDetails) {  
 *// initialise task object attributes* **this**.**userDetails** = userDetails;  
 **progressDialog** = **new** ProgressDialog(MainLecturerActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 **response** = WebService.*invokeWebService*(WebService.*VIEW\_APPOINTMENT\_METHOD*,  
 WebService.*VIEW\_APPOINTMENT\_PARAMETER*, **userDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(MainLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *//obtain booked appointments and pass these to view booked appointment activity via intent* JSONArray appointments = **response**.getJSONArray(**"appointments"**);  
 List<BookedAppointmentModel> bookedAppointmentsList = **new** ArrayList<>();  
 **for** (**int** index = 0; index < appointments.length(); index++) {  
 JSONObject appointment = appointments.getJSONObject(index);  
 BookedAppointmentModel bookedAppointmentModelItem = **new** BookedAppointmentModel(  
 appointment.getString(**"username"**),  
 appointment.getString(**"firstName"**),  
 appointment.getString(**"lastName"**),  
 appointment.getString(**"date"**),  
 appointment.getString(**"start"**),  
 appointment.getString(**"end"**),  
 appointment.getString(**"status"**));  
 bookedAppointmentsList.add(bookedAppointmentModelItem);  
 }  
 *//create intent and start next activity* Intent intent = **new** Intent(MainLecturerActivity.**this**, BookedAppointmentActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"title"**, **title**);  
 intent.putExtra(**"department"**, **department**);  
 intent.putExtra(**"type"**, **"lecturer"**);  
 intent.putExtra(**"bookedAppointments"**, (ArrayList<BookedAppointmentModel>)bookedAppointmentsList);  
 startActivity(intent);  
 } **else** {  
 *// no booked appointments* toast = Toast.*makeText*(MainLecturerActivity.**this**, **"No Booked Appointments"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.*makeText*(MainLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Student main menu activity, used by students to access application functionality  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** MainStudentActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 **private** TextView **lblWelcome**; *//welcome text display* **private** Button **btnMakeAppointment**; *//button to book appointments* **private** Button **btnViewAppointment**; *//button to view booked appointments* **private** Button **btnLogout**; *//button to logout* **private** String **username**; *//student username* **private** String **firstName**; *//last name* **private** String **lastName**; *//first name  
  
 /\*\*  
 \* Create and initialise the menu activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main\_student***);  
 *//obtain references to activity sub-views* **lblWelcome** = (TextView) findViewById(R.id.***lblWelcome***);  
 **btnMakeAppointment** = (Button) findViewById(R.id.***btnMakeAppointment***);  
 *// attach listener to handle button selection touch events* **btnMakeAppointment**.setOnClickListener(**this**);  
 **btnViewAppointment** = (Button) findViewById(R.id.***btnViewAppointment***);  
 *// attach listener to handle button selection touch events* **btnViewAppointment**.setOnClickListener(**this**);  
 **btnLogout** = (Button) findViewById(R.id.***btnLogout***);  
 *// attach listener to handle button selection touch events* **btnLogout**.setOnClickListener(**this**);  
  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 *// obtain extra information passed from previous activity via intent* **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **username** = intent.getStringExtra(**"username"**);  
 **lblWelcome**.setText(**"Welcome "** + **firstName** + **" "** + **lastName**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click evevnt  
 \*/* @Override  
 **public void** onClick(View v) {  
 **if** (v == **btnMakeAppointment**) {  
 *// create JSON object with user details to obtain available appointments for booking* JSONObject userDetails = **new** JSONObject();  
 **try** {  
 userDetails.put(**"username"**, **username**);  
 **new** AsyncMakeAppointmentTask(userDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 } **else if** (v == **btnViewAppointment**) {  
 *// create JSON object with user details to obtain booked appointments* JSONObject userDetails = **new** JSONObject();  
 **try** {  
 userDetails.put(**"username"**, **username**);  
 userDetails.put(**"type"**, **"student"**);  
 **new** AsyncViewAppointmentTask(userDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 } **else if** (v == **btnLogout**) {  
 *//logout* Intent intent = **new** Intent(MainStudentActivity.**this**, LoginActivity.**class**);  
 startActivity(intent);  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for obtaining appointments available for booking  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncViewAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userDetails**; *// Json object with user details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userDetails*** *Json object with user details  
 \*/* **public** AsyncViewAppointmentTask(JSONObject userDetails) {  
 *// initialise task object attributes* **this**.**userDetails** = userDetails;  
 **progressDialog** = **new** ProgressDialog(MainStudentActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing login details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*VIEW\_APPOINTMENT\_METHOD*,  
 WebService.*VIEW\_APPOINTMENT\_PARAMETER*, **userDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(MainStudentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *//json result of appointments available for booking* JSONArray appointments = **response**.getJSONArray(**"appointments"**);  
 List<BookedAppointmentModel> bookedAppointmentsList = **new** ArrayList<>();  
 **for** (**int** index = 0; index < appointments.length(); index++) {  
 JSONObject appointment = appointments.getJSONObject(index);  
 BookedAppointmentModel bookedAppointmentModelItem = **new** BookedAppointmentModel(  
 appointment.getString(**"username"**),  
 appointment.getString(**"firstName"**),  
 appointment.getString(**"lastName"**),  
 appointment.getString(**"date"**),  
 appointment.getString(**"start"**),  
 appointment.getString(**"end"**),  
 appointment.getString(**"status"**));  
 bookedAppointmentsList.add(bookedAppointmentModelItem);  
 }  
 *//pass list of available appointments in intent to booked appointment view* Intent intent = **new** Intent(MainStudentActivity.**this**, BookedAppointmentActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"type"**, **"student"**);  
 intent.putExtra(**"bookedAppointments"**, (ArrayList<BookedAppointmentModel>)bookedAppointmentsList);  
 startActivity(intent);  
 } **else** {  
 *//no appointments available for booking* toast = Toast.*makeText*(MainStudentActivity.**this**, **"No Booked Appointments"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.*makeText*(MainStudentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for obtaining booked appointments  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncMakeAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userDetails**; *// Json object with login details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userDetails*** *Json object with user login details  
 \*/* **public** AsyncMakeAppointmentTask(JSONObject userDetails) {  
 *// initialise task object attributes* **this**.**userDetails** = userDetails;  
 **progressDialog** = **new** ProgressDialog(MainStudentActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing login details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*MAKE\_APPOINTMENT\_METHOD*,  
 WebService.*MAKE\_APPOINTMENT\_PARAMETER*, **userDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(MainStudentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *//Json result returns a list lecturers asscoiated with the student* JSONArray lecturers = **response**.getJSONArray(**"lecturers"**);  
 List<LecturerModel> lecturesList = **new** ArrayList<>();  
 **for** (**int** index = 0; index < lecturers.length(); index++) {  
 JSONObject lecturer = lecturers.getJSONObject(index);  
 LecturerModel lecturerItem = **new** LecturerModel(lecturer.getString(**"username"**),  
 lecturer.getString(**"title"**), lecturer.getString(**"firstName"**),  
 lecturer.getString(**"lastName"**), lecturer.getString(**"department"**));  
 JSONArray subjects = lecturer.getJSONArray(**"subjects"**);  
 **for** (**int** index1 = 0; index1 < subjects.length(); index1++) {  
 lecturerItem.addSubject(subjects.getJSONObject(index1).getString(**"subject"**));  
 }  
 lecturesList.add(lecturerItem);  
 }  
 *//pass list of lecturers to appointment booking view* Intent intent = **new** Intent(MainStudentActivity.**this**, StudentLecturerActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"lecturers"**, (ArrayList<LecturerModel>)lecturesList);  
 startActivity(intent);  
 } **else** {  
 toast = Toast.*makeText*(MainStudentActivity.**this**, **"No Assigned Lecturers"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.*makeText*(MainStudentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.AdapterView;  
**import** android.widget.AdapterView.OnItemClickListener;  
**import** android.widget.Button;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Student associated lecturers activity, used by students to select a lecturer to book an appointment with  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** StudentLecturerActivity **extends** AppCompatActivity **implements** OnClickListener,  
 OnItemClickListener {  
  
 **private** ListView **lstLecturers**; *//list display of associated lecturers* **private** Button **btnMenu**; *//button to go back to main menu* **private** String **username**; *//student username* **private** String **firstName**; *//first name* **private** String **lastName**; *//last name* **private** List<LecturerModel> **lecturers**; *//list model of available lecturers  
  
 /\*\*  
 \* Create and initialise the activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_student\_lecturer***);  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 *// obtain extra information passed from previous activity via intent* **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **username** = intent.getStringExtra(**"username"**);  
 **lecturers** = (ArrayList<LecturerModel>)intent.getSerializableExtra(**"lecturers"**);  
 *//create adapter using intent obtained model data for lecturer list display* LecturerAdapter lecturerAdapter = **new** LecturerAdapter(**this**, **lecturers**);  
 **lstLecturers** = (ListView) findViewById(R.id.***lstLecturers***);  
 **lstLecturers**.setAdapter(lecturerAdapter);  
 *// attach listener to handle selection touch events* **lstLecturers**.setOnItemClickListener(**this**);  
 **btnMenu** = (Button) findViewById(R.id.***btnMenu1***);  
 *// attach listener to handle button selection touch events* **btnMenu**.setOnClickListener(**this**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click event  
 \*/* @Override  
 **public void** onClick(View v) {  
 **if** (v == **btnMenu**) {  
 *//back to main menu, pass back menu view information via intent* Intent intent = **new** Intent(StudentLecturerActivity.**this**, MainStudentActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 startActivity(intent);  
 }  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event list view listener  
 \*  
 \** ***@param parent*** *parent adapted list view  
 \** ***@param view*** *list item view responsible for touch click event  
 \** ***@param position*** *position of list item view  
 \** ***@param id*** *unique id for list item  
 \*/* @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **try** {  
 *// create JSON object with details from the list lecturer model to book  
 // view available appointments for a particular lecturer* JSONObject userDetails = **new** JSONObject();  
 LecturerModel lecturer = (LecturerModel) **lstLecturers**.getItemAtPosition(position);  
 userDetails.put(**"username"**, lecturer.getUsername());  
 *// pass Json object to asynchronous task thread responsible for web service call* **new** AsyncAvailableAppointmentTask(userDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for obtaining available appointments for a lecturer  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncAvailableAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userDetails**; *// Json object with lecturer details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userDetails*** *Json object with lecturer details  
 \*/* **public** AsyncAvailableAppointmentTask(JSONObject userDetails) {  
 *// initialise task object attributes* **this**.**userDetails** = userDetails;  
 **progressDialog** = **new** ProgressDialog(StudentLecturerActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 *// display progress dialog* **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing appointment details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*GET\_AVAILABLE\_APPOINTMENT\_METHOD*,  
 WebService.*GET\_AVAILABLE\_APPOINTMENT\_PARAMETER*, **userDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *// Json result returns available appointments for a lecturer* JSONArray appointments = **response**.getJSONArray(**"appointments"**);  
 List<AppointmentModel> appointmentsList = **new** ArrayList<>();  
 **for** (**int** index = 0; index < appointments.length(); index++) {  
 JSONObject appointment = appointments.getJSONObject(index);  
 AppointmentModel appointmentItem = **new** AppointmentModel(appointment.getString(**"date"**),  
 appointment.getString(**"start"**), appointment.getString(**"end"**));  
 appointmentsList.add(appointmentItem);  
 }  
 *//pass avaialble appointments via intent to appointment booking view* Intent intent = **new** Intent(StudentLecturerActivity.**this**, AvailableAppointmentsActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"lecturers"**, (ArrayList<LecturerModel>)**lecturers**);  
 intent.putExtra(**"lecturerUsername"**, **userDetails**.getString(**"username"**));  
 intent.putExtra(**"appointments"**, (ArrayList<AppointmentModel>)appointmentsList);  
 startActivity(intent);  
 } **else** {  
 *// available appointments for lecturer* toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"No Available Appointments"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid Json* e.printStackTrace();  
 toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.AdapterView;  
**import** android.widget.AdapterView.OnItemClickListener;  
**import** android.widget.Button;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Student associated lecturers activity, used by students to select a lecturer to book an appointment with  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** StudentLecturerActivity **extends** AppCompatActivity **implements** OnClickListener,  
 OnItemClickListener {  
  
 **private** ListView **lstLecturers**; *//list display of associated lecturers* **private** Button **btnMenu**; *//button to go back to main menu* **private** String **username**; *//student username* **private** String **firstName**; *//first name* **private** String **lastName**; *//last name* **private** List<LecturerModel> **lecturers**; *//list model of available lecturers  
  
 /\*\*  
 \* Create and initialise the activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_student\_lecturer***);  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 *// obtain extra information passed from previous activity via intent* **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **username** = intent.getStringExtra(**"username"**);  
 **lecturers** = (ArrayList<LecturerModel>)intent.getSerializableExtra(**"lecturers"**);  
 *//create adapter using intent obtained model data for lecturer list display* LecturerAdapter lecturerAdapter = **new** LecturerAdapter(**this**, **lecturers**);  
 **lstLecturers** = (ListView) findViewById(R.id.***lstLecturers***);  
 **lstLecturers**.setAdapter(lecturerAdapter);  
 *// attach listener to handle selection touch events* **lstLecturers**.setOnItemClickListener(**this**);  
 **btnMenu** = (Button) findViewById(R.id.***btnMenu1***);  
 *// attach listener to handle button selection touch events* **btnMenu**.setOnClickListener(**this**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click event  
 \*/* @Override  
 **public void** onClick(View v) {  
 **if** (v == **btnMenu**) {  
 *//back to main menu, pass back menu view information via intent* Intent intent = **new** Intent(StudentLecturerActivity.**this**, MainStudentActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 startActivity(intent);  
 }  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event list view listener  
 \*  
 \** ***@param parent*** *parent adapted list view  
 \** ***@param view*** *list item view responsible for touch click event  
 \** ***@param position*** *position of list item view  
 \** ***@param id*** *unique id for list item  
 \*/* @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **try** {  
 *// create JSON object with details from the list lecturer model to book  
 // view available appointments for a particular lecturer* JSONObject userDetails = **new** JSONObject();  
 LecturerModel lecturer = (LecturerModel) **lstLecturers**.getItemAtPosition(position);  
 userDetails.put(**"username"**, lecturer.getUsername());  
 *// pass Json object to asynchronous task thread responsible for web service call* **new** AsyncAvailableAppointmentTask(userDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for obtaining available appointments for a lecturer  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncAvailableAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **userDetails**; *// Json object with lecturer details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param userDetails*** *Json object with lecturer details  
 \*/* **public** AsyncAvailableAppointmentTask(JSONObject userDetails) {  
 *// initialise task object attributes* **this**.**userDetails** = userDetails;  
 **progressDialog** = **new** ProgressDialog(StudentLecturerActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 *// display progress dialog* **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing appointment details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*GET\_AVAILABLE\_APPOINTMENT\_METHOD*,  
 WebService.*GET\_AVAILABLE\_APPOINTMENT\_PARAMETER*, **userDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *// Json result returns available appointments for a lecturer* JSONArray appointments = **response**.getJSONArray(**"appointments"**);  
 List<AppointmentModel> appointmentsList = **new** ArrayList<>();  
 **for** (**int** index = 0; index < appointments.length(); index++) {  
 JSONObject appointment = appointments.getJSONObject(index);  
 AppointmentModel appointmentItem = **new** AppointmentModel(appointment.getString(**"date"**),  
 appointment.getString(**"start"**), appointment.getString(**"end"**));  
 appointmentsList.add(appointmentItem);  
 }  
 *//pass avaialble appointments via intent to appointment booking view* Intent intent = **new** Intent(StudentLecturerActivity.**this**, AvailableAppointmentsActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"lecturers"**, (ArrayList<LecturerModel>)**lecturers**);  
 intent.putExtra(**"lecturerUsername"**, **userDetails**.getString(**"username"**));  
 intent.putExtra(**"appointments"**, (ArrayList<AppointmentModel>)appointmentsList);  
 startActivity(intent);  
 } **else** {  
 *// available appointments for lecturer* toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"No Available Appointments"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid Json* e.printStackTrace();  
 toast = Toast.*makeText*(StudentLecturerActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.TextView;  
  
**import** java.util.List;  
  
*/\*\*  
 \* A view adapter class to customize the list view for displaying booked appointments  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** BookedAppointmentAdapter **extends** ArrayAdapter<BookedAppointmentModel> {  
  
 */\*\*  
 \* Constructor for the adapter class  
 \*  
 \** ***@param context*** *context from parent display  
 \** ***@param bookedAppointments*** *collection of booked appointments to populate the adapted list view  
 \*/* **public** BookedAppointmentAdapter(Context context, List<BookedAppointmentModel> bookedAppointments) {  
 **super**(context, 0, bookedAppointments);  
 }  
  
  
 */\*\*  
 \* Returns an adapted view for a given booked appointment item in the list  
 \*  
 \** ***@param position*** *index of item in the list view  
 \** ***@param convertView*** *list item view to be adapted  
 \** ***@param parent*** *parent view of all adapted list view items  
 \** ***@return*** *adapted list view item  
 \*/* **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 BookedAppointmentModel bookedAppointment = getItem(position);  
  
 *//obtain a booked appointment for a given index position in the list view* **if** (convertView == **null**) {  
 convertView = LayoutInflater.*from*(getContext()).inflate(R.layout.***booked\_appointment\_item***, parent, **false**);  
 }  
  
 *//if a adapted list item view has not been created, then inflate one from its repective layout template* TextView lblFirstName = (TextView) convertView.findViewById(R.id.***lblFirstName***);  
 TextView lblLastName = (TextView) convertView.findViewById(R.id.***lblLastName***);  
 TextView lblDate = (TextView) convertView.findViewById(R.id.***lblDate***);  
 TextView lblStart = (TextView) convertView.findViewById(R.id.***lblStart***);  
 TextView lblEnd = (TextView) convertView.findViewById(R.id.***lblEnd***);  
 TextView lblStatus = (TextView) convertView.findViewById(R.id.***lblStatus***);  
  
 *//populate the sub-view items with details for a booked appointment* lblFirstName.setText(bookedAppointment.getFirstName());  
 lblLastName.setText(bookedAppointment.getLastName());  
 lblDate.setText(bookedAppointment.getDate());  
 lblStart.setText(bookedAppointment.getStart());  
 lblEnd.setText(bookedAppointment.getEnd());  
 lblStatus.setText(bookedAppointment.getStatus());  
  
 **return** convertView;  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** java.io.Serializable;  
  
*/\*\*  
 \* A class representing the data model for a booked appointment. This class is also serializable  
 \* so its objects can be passed as extra information between Activities via Intents.  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** BookedAppointmentModel **implements** Serializable {  
  
 **private final** String **username**; *//username of the person the appointment is made with (student for lecturer booking view, lecturer for student booking view)* **private final** String **firstName**; *//user's first name* **private final** String **lastName**; *//user's last name* **private final** String **date**; *//appointment date* **private final** String **start**; *//appointment start time* **private final** String **end**; *//appointment end time* **private final** String **status**; *//appointment status (booked, or cancelled)  
  
 /\*\*  
 \* Constructor for a booked appointment data model  
 \*  
 \** ***@param username*** *username of the person the appointment is made with (student for lecturer booking view, lecturer for student booking view)  
 \** ***@param firstName*** *user's first name  
 \** ***@param lastName*** *user's last name  
 \** ***@param date*** *appointment date  
 \** ***@param start*** *appointment start time  
 \** ***@param end*** *appointment end time  
 \** ***@param status*** *appointment status (booked, or cancelled)  
 \*/* **public** BookedAppointmentModel(String username, String firstName, String lastName, String date, String start, String end, String status) {  
 **this**.**username** = username;  
 **this**.**firstName** = firstName;  
 **this**.**lastName** = lastName;  
 **this**.**date** = date;  
 **this**.**start** = start;  
 **this**.**end** = end;  
 **this**.**status** = status;  
 }  
  
 */\*\*  
 \** ***@return*** *the username  
 \*/* **public** String getUsername() {  
 **return username**;  
 }  
  
 */\*\*  
 \** ***@return*** *the date  
 \*/* **public** String getDate() {  
 **return date**;  
 }  
  
 */\*\*  
 \** ***@return*** *the start  
 \*/* **public** String getStart() {  
 **return start**;  
 }  
  
 */\*\*  
 \** ***@return*** *the end  
 \*/* **public** String getEnd() {  
 **return end**;  
 }  
  
 */\*\*  
 \** ***@return*** *the firstName  
 \*/* **public** String getFirstName() {  
 **return firstName**;  
 }  
  
 */\*\*  
 \** ***@return*** *the lastName  
 \*/* **public** String getLastName() {  
 **return lastName**;  
 }  
  
 */\*\*  
 \** ***@return*** *the status  
 \*/* **public** String getStatus() {  
 **return status**;  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.CheckBox;  
**import** android.widget.EditText;  
**import** android.widget.LinearLayout;  
**import** android.widget.Toast;  
  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
*/\*\*  
 \* Used by lecturers to create appointments for students to book  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** CreateAppointmentActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 **private** String **username**; *//username* **private** String **firstName**; *//first name* **private** String **lastName**; *//last name* **private** String **title**; *//title (Dr., Prof., Ms., etc...)* **private** String **department**; *//school department* **private** EditText **txtDay**; *//appointment day of week* **private** EditText **txtMonth**; *//appointment month* **private** EditText **txtYear**; *//appointment year* **private** EditText **txtStartHour**; *//appointment start hour* **private** EditText **txtStartMinute**; *//appointment start minute* **private** EditText **txtEndHour**; *//appointment end hour* **private** EditText **txtEndMinute**; *//appointment minute hour* **private** Button **btnMakeAppointment**; *//button to submit creating an appointment* **private** Button **btnCancel**; *//button to cancel creating an appointment* **private** CheckBox **chkDaily**; *//check box indicate appointment will reoccur daily* **private** CheckBox **chkWeekly**; *//check box indicate appointment will reoccur weekly* **private** EditText **txtRecurrence**; *//number of appointment recurrences  
  
 /\*\*  
 \* Create and initialise the create appointment activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_create\_appointment***);  
  
 *//obtain references to activity sub-views* ((LinearLayout) findViewById(R.id.***background***)).getBackground().setAlpha(127);  
 **txtDay** = (EditText) findViewById(R.id.***txtDay***);  
 **txtMonth** = (EditText) findViewById(R.id.***txtMonth***);  
 **txtYear** = (EditText) findViewById(R.id.***txtYear***);  
 **txtStartHour** = (EditText) findViewById(R.id.***txtStartHour***);  
 **txtStartMinute** = (EditText) findViewById(R.id.***txtStartMinute***);  
 **txtEndHour** = (EditText) findViewById(R.id.***txtEndHour***);  
 **txtEndMinute** = (EditText) findViewById(R.id.***txtEndMinute***);  
 **btnMakeAppointment** = (Button) findViewById(R.id.***btnCreateAppointment***);  
 **btnMakeAppointment**.setOnClickListener(**this**);  
 **btnCancel** = (Button) findViewById(R.id.***btnCancel***);  
 **btnCancel**.setOnClickListener(**this**);  
 **chkDaily** = (CheckBox) findViewById(R.id.***chkDialy***);  
 **chkDaily**.setOnClickListener(**this**);  
 **chkWeekly** = (CheckBox) findViewById(R.id.***chkWeekly***);  
 **chkWeekly**.setOnClickListener(**this**);  
 **txtRecurrence** = (EditText) findViewById(R.id.***txtRecurrence***);  
  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 **username** = intent.getStringExtra(**"username"**);  
 **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **title** = intent.getStringExtra(**"title"**);  
 **department** = intent.getStringExtra(**"department"**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click event  
 \*/* @Override  
 **public void** onClick(View v) {  
 **if** (v == **btnMakeAppointment**) {  
 *//submits details to create an appointment* JSONObject appointmentDetails = **new** JSONObject();  
 **try** {  
 appointmentDetails.put(**"username"**, **username**);  
 appointmentDetails.put(**"day"**, **txtDay**.getText().toString());  
 appointmentDetails.put(**"month"**, **txtMonth**.getText().toString());  
 appointmentDetails.put(**"year"**, **txtYear**.getText().toString());  
 appointmentDetails.put(**"startHour"**, **txtStartHour**.getText().toString());  
 appointmentDetails.put(**"startMinute"**, **txtStartMinute**.getText().toString());  
 appointmentDetails.put(**"endHour"**, **txtEndHour**.getText().toString());  
 appointmentDetails.put(**"endMinute"**, **txtEndMinute**.getText().toString());  
 appointmentDetails.put(**"isDaily"**, **chkDaily**.isChecked() ? **"true"**:**"false"**);  
 appointmentDetails.put(**"isWeekly"**, **chkWeekly**.isChecked() ? **"true"**:**"false"**);  
 appointmentDetails.put(**"recurrence"**, **txtRecurrence**.getText().toString());  
 **new** AsyncCreateAppointmentTask(appointmentDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 *//cancels and goes back to previous activity* } **else if** (v == **btnCancel**) {  
 Intent intent = **new** Intent(CreateAppointmentActivity.**this**, MainLecturerActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"title"**, **title**);  
 intent.putExtra(**"department"**, **department**);  
 startActivity(intent);  
 } **else if** (v == **chkDaily**) {  
 *// daily indicated then weekly should be deactivated* **if** (**chkDaily**.isChecked()) {  
 **chkWeekly**.setChecked(**false**);  
 }  
 } **else if** (v == **chkWeekly**) {  
 *// weekly indicated then daily should be deactivated* **if**(**chkWeekly**.isChecked()) {  
 **chkDaily**.setChecked(**false**);  
 }  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for creating appointments  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncCreateAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **appointmentDetails**; *// Json object with appointment creation details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param appointmentDetails*** *Json object with appointment creation details  
 \*/* **public** AsyncCreateAppointmentTask(JSONObject appointmentDetails) {  
 *// initialise task object attributes* **this**.**appointmentDetails** = appointmentDetails;  
 **progressDialog** = **new** ProgressDialog(CreateAppointmentActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing appointment details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*CREATE\_APPOINTMENT\_METHOD*,  
 WebService.*CREATE\_APPOINTMENT\_PARAMETER*, **appointmentDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (**response** == **null** || **response**.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.*makeText*(CreateAppointmentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// successful response result  
 // appointment cannot be created as there's a class* **if** (**response**.getString(**"result"**).compareToIgnoreCase(**"false"**) == 0) {  
 toast = Toast.*makeText*(CreateAppointmentActivity.**this**, **"This appointment clashes with an existing appointment"**, Toast.***LENGTH\_LONG***);  
 } **else** {  
 *// appointment created* toast = Toast.*makeText*(CreateAppointmentActivity.**this**, **"Appointment created"**, Toast.***LENGTH\_LONG***);  
 }  
 }  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.*makeText*(CreateAppointmentActivity.**this**, **"Problem communicating with server"**, Toast.***LENGTH\_LONG***);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.TextView;  
  
**import** java.util.List;  
  
*/\*\*  
 \* A view adapter class to customize the list view for displaying lecturers assigned to a particular student user  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** LecturerAdapter **extends** ArrayAdapter<LecturerModel> {  
  
 */\*\*  
 \* Constructor for the adapter class  
 \*  
 \** ***@param context*** *context from parent display  
 \** ***@param lecturers*** *collection of lecturers to populate the adapted list view  
 \*/* **public** LecturerAdapter(Context context, List<LecturerModel> lecturers) {  
 **super**(context, 0, lecturers);  
 }  
  
 */\*\*  
 \* Returns an adapted view for a given lecturer item in the list  
 \*  
 \** ***@param position*** *index of item in the list view  
 \** ***@param convertView*** *list item view to be adapted  
 \** ***@param parent*** *parent view of all adapted list view items  
 \** ***@return*** *adapted list view item  
 \*/* **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 *//obtain a lecturer for a given index position in the list view* LecturerModel lecturer = getItem(position);  
  
 *//if a adapted list item view has not been created, then inflate one from its repective layout template* **if** (convertView == **null**) {  
 convertView = LayoutInflater.*from*(getContext()).inflate(R.layout.***lecturer\_item***, parent, **false**);  
 }  
  
 *//obtain references to the sub-view items contained within the adapted list view item* TextView lblLecturerTitleName = (TextView) convertView.findViewById(R.id.***lblLecturerTitleName***);  
 TextView lblDepartment = (TextView) convertView.findViewById(R.id.***lblDepartment***);  
 TextView lblSubjects = (TextView) convertView.findViewById(R.id.***lblSubjects***);  
  
 *//populate the sub-view items with details for an available appointment* lblLecturerTitleName.setText(lecturer.getTitle() + **" "** + lecturer.getFirstName() + **" "** +  
 lecturer.getLastName());  
  
 lblDepartment.setText(lecturer.getDepartment());  
  
 String subjects = **"Subjects: "**;  
 **for** (String subject : lecturer.getSubjects()) {  
 subjects = subjects.concat(subject + **", "**);  
 }  
 subjects = subjects.substring(0, subjects.length() - 2);  
 lblSubjects.setText(subjects);  
  
 **return** convertView;  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** java.io.Serializable;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* A class representing the data model for an available appointment. This class is also serializable  
 \* so its objects can be passed as extra information between Activities via Intents.  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** LecturerModel **implements** Serializable {  
  
 **private** String **username**; *//lecturer username* **private** String **title**; *//title* **private** String **firstName**; *//first name* **private** String **lastName**; *//last name* **private** String **department**; *//department* **private** List<String> **subjects**; *//subjects taught by lecturer  
  
 /\*\*  
 \* Constructor for a lecturer data model  
 \*  
 \** ***@param username*** *//lecturer username  
 \** ***@param title*** *//title  
 \** ***@param firstName*** *//first name  
 \** ***@param lastName*** *//last name  
 \** ***@param department*** *//department  
 \*/* **public** LecturerModel(String username, String title, String firstName,  
 String lastName, String department) {  
 **this**.**username** = username;  
 **this**.**title** = title;  
 **this**.**firstName** = firstName;  
 **this**.**lastName** = lastName;  
 **this**.**department** = department;  
 **subjects** = **new** ArrayList<>();  
 }  
  
 */\*\*  
 \** ***@return*** *the lecturer username  
 \*/* **public** String getUsername() {  
 **return username**;  
 }  
  
 */\*\*  
 \** ***@return*** *the lecturer title  
 \*/* **public** String getTitle() {  
 **return title**;  
 }  
  
 */\*\*  
 \** ***@return*** *the lecturer firstName  
 \*/* **public** String getFirstName() {  
 **return firstName**;  
 }  
  
 */\*\*  
 \** ***@return*** *the lecturer lastName  
 \*/* **public** String getLastName() {  
 **return lastName**;  
 }  
  
 */\*\*  
 \** ***@return*** *the lecturer department  
 \*/* **public** String getDepartment() {  
 **return department**;  
 }  
  
 */\*\*  
 \** ***@return*** *the list of lecturer subjects  
 \*/* **public** List<String> getSubjects() {  
 **return subjects**;  
 }  
  
 */\*\*  
 \** ***@param subject*** *subject to add  
 \*/* **public void** addSubject(String subject) {  
 **subjects**.add(subject);  
 }  
  
 */\*\*  
 \* String representation of a lecturer data model  
 \*  
 \** ***@return*** *string containing lecturer details  
 \*/* **public** String toString() {  
 **return title** + **" "** + **firstName** + **" "** + **lastName**;  
 }  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** java.io.Serializable;  
  
*/\*\*  
 \* A class representing the data model for an available appointment. This class is also serializable  
 \* so its objects can be passed as extra information between Activities via Intents.  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** AppointmentModel **implements** Serializable {  
  
 **private final** String **date**; *// available appointment date* **private final** String **start**; *// available appointment start time* **private final** String **end**; *// available appointment end time  
  
 /\*\*  
 \* Constructor for an available appointment data model  
 \*  
 \** ***@param date*** *appointment date  
 \** ***@param start*** *appointment start time  
 \** ***@param end*** *appointment end time  
 \*/* **public** AppointmentModel(String date, String start, String end) {  
 **this**.**date** = date;  
 **this**.**start** = start;  
 **this**.**end** = end;  
 }  
  
 */\*\*  
 \** ***@return*** *the appointment date  
 \*/* **public** String getDate() {  
 **return date**;  
 }  
  
 */\*\*  
 \** ***@return*** *the appointment start  
 \*/* **public** String getStart() {  
 **return start**;  
 }  
  
 */\*\*  
 \** ***@return*** *the appointemnt end  
 \*/* **public** String getEnd() {  
 **return end**;  
 }  
  
 */\*\*  
 \* String representation of an appointment data model  
 \*  
 \** ***@return*** *string containing appointment details  
 \*/* **public** String toString() {  
 **return date** + **" "** + **start** + **" "** + **end**;  
 }  
  
}

**package** unitec.ac.nz.unitecappointmentclient;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.AdapterView;  
**import** android.widget.AdapterView.OnItemClickListener;  
**import** android.widget.Button;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Available appointment activity, used by students, and responsible for displaying available  
 \* appointments for a specific lecturer, and selecting available appointments for booking  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/***public class** AvailableAppointmentsActivity **extends** AppCompatActivity **implements** OnClickListener,  
 OnItemClickListener {  
  
 **private** ListView **lstAppointments**; *// list view for displaying all available lecturer appointments* **private** Button **btnLecturers**; *// button for going back to lecturer list activity* **private** String **username**; *// student's username* **private** String **firstName**; *// student's first name* **private** String **lastName**; *// student's last name* **private** String **lecturerUsername**; *// lecture username* **private** List<LecturerModel> **lecturers**; *// lecturer data model collection* **private** List<AppointmentModel> **appointments**; *// available appointment data model collection  
  
 /\*\*  
 \* Create and initialise the available appointment activity view  
 \*  
 \** ***@param savedInstanceState*** *any state information persisted from activity's previous life cycle  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 *// call ancestor constructor for proper initialisation, assign the respective layout for  
 // this activity's view* **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_available\_appointments***);  
  
 *// obtain intent passed from previous activity* Intent intent = getIntent();  
 *// obtain extra information passed from previous activity via intent* **firstName** = intent.getStringExtra(**"firstName"**);  
 **lastName** = intent.getStringExtra(**"lastName"**);  
 **username** = intent.getStringExtra(**"username"**);  
 **lecturers** = (ArrayList<LecturerModel>)intent.getSerializableExtra(**"lecturers"**);  
 **lecturerUsername** = intent.getStringExtra(**"lecturerUsername"**);  
 **appointments** = (ArrayList<AppointmentModel>)intent.getSerializableExtra(**"appointments"**);  
  
 *// create list view adapter from appointment data model passed via intent, then pass  
 // adapter to customize and populate the list view* AppointmentAdapter appointmentAdapter = **new** AppointmentAdapter(**this**, **appointments**);  
 **lstAppointments** = (ListView) findViewById(R.id.***lstAppointments***);  
 **lstAppointments**.setAdapter(appointmentAdapter);  
 *// attach listener to handle list view item selection touch events* **lstAppointments**.setOnItemClickListener(**this**);  
  
 *// obtain button used to go back to the previous lecturer list view, and assign listener  
 // to handle button touch events* **btnLecturers** = (Button) findViewById(R.id.***btnLecturers***);  
 *// attach listener to handle button selection touch events* **btnLecturers**.setOnClickListener(**this**);  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event listener  
 \*  
 \** ***@param v*** *view responsible for raising the touch click evevnt  
 \*/* @Override  
 **public void** onClick(View v) {  
 *// if touch click event was raised by back to lecturer list activity* **if** (v == **btnLecturers**) {  
 *// create intent and pass extra information required for initializing activity about to be started* Intent intent = **new** Intent(AvailableAppointmentsActivity.**this**, StudentLecturerActivity.**class**);  
 intent.putExtra(**"username"**, **username**);  
 intent.putExtra(**"firstName"**, **firstName**);  
 intent.putExtra(**"lastName"**, **lastName**);  
 intent.putExtra(**"lecturers"**, (ArrayList<LecturerModel>)**lecturers**);  
 startActivity(intent);  
 }  
 }  
  
 */\*\*  
 \* Call back handler for events arising from touch click event list view listener  
 \*  
 \** ***@param parent*** *parent adapted list view  
 \** ***@param view*** *list item view responsible for touch click event  
 \** ***@param position*** *position of list item view  
 \** ***@param id*** *unique id for list item  
 \*/* @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **try** {  
 *// create JSON object with details from the list item appointment model to book  
 // a particular available appointment* JSONObject appointmentDetails = **new** JSONObject();  
 AppointmentModel appointment = (AppointmentModel) **lstAppointments**.getItemAtPosition(position);  
 appointmentDetails.put(**"studentUsername"**, **username**);  
 appointmentDetails.put(**"lecturerUsername"**, **lecturerUsername**);  
 appointmentDetails.put(**"date"**, appointment.getDate());  
 appointmentDetails.put(**"start"**, appointment.getStart());  
 appointmentDetails.put(**"end"**, appointment.getEnd());  
 *// pass Json object to asynchronous task thread responsible for web service call* **new** AsyncBookAppointmentTask(appointmentDetails).execute();  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Private helper class to handle asynchronous web service call for booking appointments  
 \*  
 \** ***@author*** *Oshada Koralage (1434048)  
 \*/* **private class** AsyncBookAppointmentTask **extends** AsyncTask<Void, Void, Void> {  
  
 **private** JSONObject **appointmentDetails**; *// Json object with appointment booking details* **private** ProgressDialog **progressDialog**; *// dialog for progress of web service call* **private** JSONObject **response**; *// web service response Json object  
  
 /\*\*  
 \* Constructor for asynchronous task  
 \*  
 \** ***@param appointmentDetails*** *Json object with appointment booking details  
 \*/* **public** AsyncBookAppointmentTask(JSONObject appointmentDetails) {  
 *// initialise task object attributes* **this**.**appointmentDetails** = appointmentDetails;  
 **progressDialog** = **new** ProgressDialog(AvailableAppointmentsActivity.**this**);  
 }  
  
 */\*\*  
 \* Call back handler called prior to asynchronous task execution  
 \*\*/* @Override  
 **protected void** onPreExecute() {  
 *// display progress dialog* **progressDialog**.setMessage(**"Please wait..."**);  
 **progressDialog**.show();  
 }  
  
 */\*\*  
 \* Asynchronous task execution  
 \*  
 \** ***@param params*** *//extra parameters can be optionally passed  
 \*\*/* @Override  
 **protected** Void doInBackground(Void... params) {  
 *// make web service call passing appointment details json object, and obtain  
 // response back from web service call* **response** = WebService.*invokeWebService*(WebService.*BOOK\_APPOINTMENT\_METHOD*,  
 WebService.*BOOK\_APPOINTMENT\_PARAMETER*, **appointmentDetails**);  
 **return null**;  
 }  
  
 */\*\*  
 \* Call back handler called after asynchronous task execution is complete  
 \*  
 \** ***@param arg*** *//optional arguments passed from asynchronous task completion  
 \*\*/* @Override  
 **protected void** onPostExecute(Void arg) {  
 *// close progress dialog* **progressDialog**.dismiss();  
 Toast toast = **null**;  
 **try** {  
 *// Json response object availability and result* **if** (response == **null** || response.getString(**"result"**).compareToIgnoreCase(**"error"**) == 0) {  
 *// if unavailable or error result then problem occurred at web server* toast = Toast.makeText(AvailableAppointmentsActivity.**this**, **"Problem communicating with server"**, Toast.LENGTH\_LONG);  
 } **else** {  
 *// successful response result* **if** (response.getString(**"result"**).compareToIgnoreCase(**"true"**) == 0) {  
 *// appointment successfully booked  
 // go back to previous activity passing any initialisation information needed via intent* Intent intent = **new** Intent(AvailableAppointmentsActivity.**this**, StudentLecturerActivity.**class**);  
 intent.putExtra(**"username"**, username);  
 intent.putExtra(**"firstName"**, firstName);  
 intent.putExtra(**"lastName"**, lastName);  
 intent.putExtra(**"lecturers"**, (ArrayList<LecturerModel>)lecturers);  
 startActivity(intent);  
 } **else** {  
 *// unable to book appointment as it is no longer available* toast = Toast.makeText(AvailableAppointmentsActivity.**this**, **"Unable to make appointment"**, Toast.LENGTH\_LONG);  
 }  
 }  
  
 } **catch** (JSONException e) {  
 *//invalid json response* e.printStackTrace();  
 toast = Toast.makeText(AvailableAppointmentsActivity.**this**, **"Problem communicating with server"**, Toast.LENGTH\_LONG);  
 }  
 **if** (toast != **null**) {  
 toast.show();  
 }  
 }  
 }  
}

## Server Code

/\*\*

\* SOAP Web Services Class

\*

\* @author Marzouq Almarzooq (1380949)

\* @author Nawaf Altuwayjiri (1377387)

\*/

package unitecappointmentserver;

import java.io.IOException;

import java.io.StringReader;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.security.spec.KeySpec;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Properties;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.json.Json;

import javax.json.JsonObject;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import java.util.Base64;

import java.util.Calendar;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.activation.DataHandler;

import javax.annotation.Resource;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.PBEKeySpec;

import javax.json.JsonArrayBuilder;

import javax.mail.BodyPart;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Multipart;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;

import javax.mail.util.ByteArrayDataSource;

/\*\*

\* Web Service definition and implementation class

\*

\* @author Marzouq Almarzooq (1380949)

\* @author Nawaf Altuwayjiri (1377387)

\*/

@WebService(serviceName = "AppointementServices") //web services name

public class AppointementServices {

@Resource(lookup = "mail/AppointmentMailSession") //mail session lookup from server

private Session mailSession; //mail session object

private final Connection connection; //db connection

//SQL prepared statements

private final PreparedStatement getStudent; //get students

private final PreparedStatement getLecturer; //get lecturer

private final PreparedStatement getLecturerAppointmentsByDate; //get lecturer appointments by date

private final PreparedStatement addLecturerAppointment; //get lecturer appointments

private final PreparedStatement getAssignedLecturers; //get student assigned lecturers

private final PreparedStatement getAvailableAppointments; //get available appointments

private final PreparedStatement getAppointment; //get specific appointment

private final PreparedStatement bookAppointment; //book specific appointment

private final PreparedStatement getStudentBookedAppointment; //get booked appointments for a student

private final PreparedStatement getLecturerBookedAppointment; //get booked appointments for a lecturer

private final PreparedStatement studentCancelAppointment; //cancel a booked appointment for a student

private final PreparedStatement lecturerCancelAppointment; //cancel a booked appointment for a lecturer

//DB table names and table attributes

private final String dbStudentTable;

private final String dbLecturerTable;

private final String dbAppointmentTable;

private final String dbStudentLecturerAssignmentTable;

private final String dbUsernameAtt;

private final String dbTitleAtt;

private final String dbFirstNameAtt;

private final String dbLastNameAtt;

private final String dbDepartmentAtt;

private final String dbPasswordAtt;

private final String dbSaltAtt;

private final String dbDateAtt;

private final String dbStartAtt;

private final String dbEndAtt;

private final String dbLecturerUserNameAtt;

private final String dbStudentUserNameAtt;

private final String dbIsActiveAtt;

private final String dbSubjectAtt;

private final String dbEmailAtt;

/\*\*

\* Web services constructor, connects to db, and prepares all SQL statements

\*

\* @throws java.io.IOException

\* @throws java.lang.ClassNotFoundException

\* @throws java.sql.SQLException

\*/

public AppointementServices() throws IOException, ClassNotFoundException, SQLException {

//load db properties file

Properties properties = new Properties();

properties.loadFromXML(getClass().getResourceAsStream("DBConfig.xml"));

//db driver and name

String dbDriver = properties.get("dbDriver").toString();

String dbUrl = properties.get("dbUrl").toString();

//obtain sb table names and table attributes from loaded db properties file

dbStudentTable = properties.get("dbStudentTable").toString();

dbLecturerTable = properties.get("dbLecturerTable").toString();

dbAppointmentTable = properties.get("dbAppointmentTable").toString();

dbStudentLecturerAssignmentTable = properties.get("dbStudentLecturerAssignmentTable").toString();

dbUsernameAtt = properties.get("dbUsernameAtt").toString();

dbTitleAtt = properties.get("dbTitleAtt").toString();

dbFirstNameAtt = properties.get("dbFirstNameAtt").toString();

dbLastNameAtt = properties.get("dbLastNameAtt").toString();

dbDepartmentAtt = properties.get("dbDepartmentAtt").toString();

dbPasswordAtt = properties.get("dbPasswordAtt").toString();

dbSaltAtt = properties.get("dbSaltAtt").toString();

dbDateAtt = properties.get("dbDateAtt").toString();

dbStartAtt = properties.get("dbStartAtt").toString();

dbEndAtt = properties.get("dbEndAtt").toString();

dbLecturerUserNameAtt = properties.get("dbLecturerUserNameAtt").toString();

dbStudentUserNameAtt = properties.get("dbStudentUserNameAtt").toString();

dbIsActiveAtt = properties.get("dbIsActiveAtt").toString();

dbSubjectAtt = properties.getProperty("dbSubjectAtt");

dbEmailAtt = properties.getProperty("dbEmailAtt");

//connect to db using driver, location, and db admin user details

String dbUserName = properties.get("dbusername").toString();

String dbPassword = properties.get("dbpassword").toString();

Class.forName(dbDriver);

connection = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);

//sql prepared statement definition

getStudent = connection.prepareStatement("SELECT \* FROM " + dbStudentTable + " WHERE " + dbUsernameAtt + " = ?");

getLecturer = connection.prepareStatement("SELECT \* FROM " + dbLecturerTable + " WHERE " + dbUsernameAtt + " = ?");

getLecturerAppointmentsByDate = connection.prepareStatement("SELECT \* FROM "

+ dbAppointmentTable + " WHERE " + dbLecturerUserNameAtt + " = ? AND " + dbDateAtt + " = ?");

addLecturerAppointment = connection.prepareStatement("INSERT INTO " + dbAppointmentTable + " ("

+ dbDateAtt + ", " + dbStartAtt + ", " + dbEndAtt + ", " + dbLecturerUserNameAtt + ", "

+ dbIsActiveAtt + ") VALUES (?, ?, ?, ?, ?)");

getAssignedLecturers = connection.prepareStatement("SELECT \* FROM "

+ dbStudentLecturerAssignmentTable + " WHERE " + dbStudentUserNameAtt + " = ?");

getAvailableAppointments = connection.prepareStatement("SELECT \* FROM "

+ dbAppointmentTable + " WHERE " + dbLecturerUserNameAtt + " = ? "

+ "AND " + dbIsActiveAtt + " = 'available'");

getAppointment = connection.prepareStatement("SELECT \* FROM " + dbAppointmentTable + " WHERE "

+ dbDateAtt + " = ? AND " + dbStartAtt + " = ? AND " + dbEndAtt + " = ? AND "

+ dbLecturerUserNameAtt + " = ?");

bookAppointment = connection.prepareStatement("UPDATE " + dbAppointmentTable

+ " SET " + dbIsActiveAtt + " = 'booked', "

+ dbStudentUserNameAtt + " = ? WHERE " + dbLecturerUserNameAtt + " = ? AND "

+ dbDateAtt + " = ? AND " + dbStartAtt + " = ? AND " + dbEndAtt + " = ?");

getStudentBookedAppointment = connection.prepareStatement("SELECT \* FROM "

+ dbAppointmentTable + " WHERE " + dbStudentUserNameAtt + " = ? "

+ "AND (" + dbIsActiveAtt + " = 'booked' OR " + dbIsActiveAtt + " = 'cancelled')");

getLecturerBookedAppointment = connection.prepareStatement("SELECT \* FROM "

+ dbAppointmentTable + " WHERE " + dbLecturerUserNameAtt + " = ? "

+ "AND " + dbIsActiveAtt + " = 'booked'");

studentCancelAppointment = connection.prepareStatement("UPDATE " + dbAppointmentTable

+ " SET " + dbIsActiveAtt + " = 'available' WHERE " + dbLecturerUserNameAtt + " = ? AND "

+ dbDateAtt + " = ? AND " + dbStartAtt + " = ? AND " + dbEndAtt + " = ?");

lecturerCancelAppointment = connection.prepareStatement("UPDATE " + dbAppointmentTable

+ " SET " + dbIsActiveAtt + " = 'cancelled' WHERE " + dbStudentUserNameAtt + " = ? AND "

+ dbDateAtt + " = ? AND " + dbStartAtt + " = ? AND " + dbEndAtt + " = ?");

}

/\*\*

\* Web service operation for logging in user

\*

\* @param userLoginDetails

\* @return login result

\*/

@WebMethod(operationName = "login")

public String login(@WebParam(name = "userLoginDetails") String userLoginDetails) {

//obtain username and password from json

JsonObject loginDetails = Json.createReader(new StringReader(userLoginDetails)).readObject();

JsonObject result;

String username = loginDetails.getString("username");

String password = loginDetails.getString("password");

boolean isStudentFound = false;

boolean isLecturerFound = false;

ResultSet resultSet = null;

//lookup user record using username

try {

getStudent.setString(1, username);

resultSet = getStudent.executeQuery();

isStudentFound = resultSet.first();

if (!isStudentFound) {

getLecturer.setString(1, username);

resultSet = getLecturer.executeQuery();

isLecturerFound = resultSet.first();

}

} catch (SQLException ex) {

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

}

if ((isStudentFound || isLecturerFound) && resultSet != null) {

//if user name found

try {

//obtain user details

String firstName = resultSet.getString(dbFirstNameAtt);

String lastName = resultSet.getString(dbLastNameAtt);

String userPassword = resultSet.getString(dbPasswordAtt);

String userSalt = resultSet.getString(dbSaltAtt);

//obtain salt for user and decode from string to bytes

Base64.Decoder decoder = Base64.getDecoder();

byte[] salt = decoder.decode(userSalt);

//hash supplied password and salt

KeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, 65536, 128);

SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

byte[] hash = secretKeyFactory.generateSecret(keySpec).getEncoded();

//encode hash back to string

Base64.Encoder encoder = Base64.getEncoder();

String hashedPassword = encoder.encodeToString(hash);

//compare string hash with password hash from db

if (hashedPassword.compareTo(userPassword) == 0) {

//password correct

if (isStudentFound) {

//student user succesfyul login

result = Json.createObjectBuilder()

.add("result", "true")

.add("user", "student")

.add("firstName", firstName)

.add("lastName", lastName).build();

} else {

//lecturer user successful login

String title = resultSet.getString(dbTitleAtt);

String department = resultSet.getString(dbDepartmentAtt);

result = Json.createObjectBuilder()

.add("result", "true")

.add("user", "lecturer")

.add("title", title)

.add("firstName", firstName)

.add("lastName", lastName)

.add("department", department).build();

}

} else {

//password incorrect

result = Json.createObjectBuilder()

.add("result", "false").build();

}

} catch (SQLException | NoSuchAlgorithmException | InvalidKeySpecException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

result = Json.createObjectBuilder()

.add("result", "error").build();

}

} else {

//no user found

result = Json.createObjectBuilder()

.add("result", "false").build();

}

return result.toString();

}

/\*\*

\* Web service operation for creating appointment

\*

\* @param appointmentDetails

\* @return make appointment result

\*/

@WebMethod(operationName = "createAppointment")

public String createAppointment(@WebParam(name = "appointmentDetails") String appointmentDetails) {

//obtain create appointment detials

JsonObject makeAppointmentDetails = Json.createReader(new StringReader(appointmentDetails)).readObject();

String username = makeAppointmentDetails.getString("username");

String day = makeAppointmentDetails.getString("day");

String month = makeAppointmentDetails.getString("month");

String year = makeAppointmentDetails.getString("year");

String startHour = makeAppointmentDetails.getString("startHour");

String startMinute = makeAppointmentDetails.getString("startMinute");

String endHour = makeAppointmentDetails.getString("endHour");

String endMinute = makeAppointmentDetails.getString("endMinute");

String isDaily = makeAppointmentDetails.getString("isDaily");

String isWeekly = makeAppointmentDetails.getString("isWeekly");

String recurrence = makeAppointmentDetails.getString("recurrence");

//number of recurrences if creating appointment has multiple recurrences

int repeat;

if (recurrence.isEmpty()) {

repeat = 1;

} else {

repeat = Integer.parseInt(recurrence);

}

//date time formating to store in db

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");

//repeat for number of ocurrences

for (int i = 0; i < repeat; i++) {

try {

//obtain date objects for caluclations from string date and time given format

Date appointmentDate = dateFormat.parse(year + "-" + month + "-" + day);

Date appointmentStart = timeFormat.parse(startHour + ":" + startMinute);

Date appointmentEnd = timeFormat.parse(endHour + ":" + endMinute);

//determine type of reocuurence weekly or dialy

Calendar calendar = Calendar.getInstance();

if (isDaily.compareTo("true") == 0) {

calendar.setTime(appointmentDate);

calendar.add(Calendar.DATE, i);

appointmentDate = calendar.getTime();

} else if (isWeekly.compareTo("true") == 0) {

calendar.setTime(appointmentDate);

calendar.add(Calendar.DATE, i \* 7);

appointmentDate = calendar.getTime();

}

// get lecturer current appointments

getLecturerAppointmentsByDate.setString(1, username);

getLecturerAppointmentsByDate.setString(2, dateFormat.format(appointmentDate));

ResultSet resultSet = getLecturerAppointmentsByDate.executeQuery();

boolean isFound = resultSet.first();

if (isFound) {

//if lecturer has existing appointments then check for clash

do {

Date startTime = timeFormat.parse(resultSet.getString(dbStartAtt));

Date endTime = timeFormat.parse(resultSet.getString(dbEndAtt));

if (startTime.before(appointmentEnd) && appointmentStart.before(endTime)) {

//clash exist cannot make appointment

return Json.createObjectBuilder().add("result", "false").build().toString();

}

} while (resultSet.next());

}

//no clash intialise prepare statement with new appointment details

addLecturerAppointment.setString(1, dateFormat.format(appointmentDate));

addLecturerAppointment.setString(2, timeFormat.format(appointmentStart));

addLecturerAppointment.setString(3, timeFormat.format(appointmentEnd));

addLecturerAppointment.setString(4, username);

addLecturerAppointment.setString(5, "available");

//create appointment

addLecturerAppointment.executeUpdate();

} catch (ParseException | SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

//appoitment created return conformation result

return Json.createObjectBuilder().add("result", "true").build().toString();

}

/\*\*

\* Web service operation for obtaining lecturers for a given student

\*

\* @param userDetails

\* @return list of assigned lecturers to student

\*/

@WebMethod(operationName = "makeAppointment")

public String makeAppointment(@WebParam(name = "userDetails") String userDetails) {

try {

//obtain student details

JsonObject makeAppointmentDetails = Json.createReader(new StringReader(userDetails)).readObject();

String username = makeAppointmentDetails.getString("username");

//get lecturers for student from db

getAssignedLecturers.setString(1, username);

ResultSet resultSet = getAssignedLecturers.executeQuery();

if (resultSet.first()) {

//student has assigned lecturers

Map<String, LecturerModel> lecturers = new HashMap<>();

//obtain details for each assigned lecturer

do {

String lecturerUsername = resultSet.getString(dbLecturerUserNameAtt);

LecturerModel lecturer;

if (!lecturers.containsKey(lecturerUsername)) {

getLecturer.setString(1, lecturerUsername);

ResultSet resultSet1 = getLecturer.executeQuery();

resultSet1.first();

lecturer = new LecturerModel(

lecturerUsername, resultSet1.getString(dbTitleAtt),

resultSet1.getString(dbFirstNameAtt),

resultSet1.getString(dbLastNameAtt),

resultSet1.getString(dbDepartmentAtt));

lecturers.put(lecturerUsername, lecturer);

}

lecturer = lecturers.get(lecturerUsername);

lecturer.addSubject(resultSet.getString(dbSubjectAtt));

} while (resultSet.next());

//build json with all lecturer details

JsonArrayBuilder jsonLecturersArrayBuilder = Json.createArrayBuilder();

for (String lecturer : lecturers.keySet()) {

JsonArrayBuilder jsonSubjectsArrayBuilder = Json.createArrayBuilder();

for (String subject : lecturers.get(lecturer).getSubjects()) {

jsonSubjectsArrayBuilder.add(Json.createObjectBuilder().

add("subject", subject));

}

jsonLecturersArrayBuilder.add(Json.createObjectBuilder().

add("username", lecturers.get(lecturer).getUsername()).

add("title", lecturers.get(lecturer).getTitle()).

add("firstName", lecturers.get(lecturer).getFirstName()).

add("lastName", lecturers.get(lecturer).getLastName()).

add("department", lecturers.get(lecturer).getDepartment()).

add("subjects", jsonSubjectsArrayBuilder.build()));

}

//return json string with assigned lecturers

return Json.createObjectBuilder().

add("result", "true").

add("lecturers", jsonLecturersArrayBuilder.build()).build().toString();

} else {

//student has no assigned lectures

return Json.createObjectBuilder().

add("result", "false").build().toString();

}

} catch (SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

/\*\*

\* Web service operation to obtain available appointments for a specific lecturer

\*

\* @param userDetails

\* @return list of available appointments for a specific lecturer

\*/

@WebMethod(operationName = "getAvailableAppointment")

public String getAvailableAppointment(@WebParam(name = "userDetails") String userDetails) {

try {

//obtain lecturer details

JsonObject makeAppointmentDetails = Json.createReader(new StringReader(userDetails)).readObject();

String username = makeAppointmentDetails.getString("username");

//look up db for avaialable appointment for lecturer

getAvailableAppointments.setString(1, username);

ResultSet resultSet = getAvailableAppointments.executeQuery();

if (resultSet.first()) {

//lecturer has available appointments

List<AppointmentModel> appointments = new ArrayList<>();

//obtain available appointments

do {

appointments.add(new AppointmentModel(

resultSet.getString(dbDateAtt),

resultSet.getString(dbStartAtt),

resultSet.getString(dbEndAtt)));

} while (resultSet.next());

//build json result containing available appointment details

JsonArrayBuilder jsonAppointmentArrayBuilder

= Json.createArrayBuilder();

for (AppointmentModel appointment : appointments) {

jsonAppointmentArrayBuilder.add(Json.createObjectBuilder().

add("date", appointment.getDate()).

add("start", appointment.getStart()).

add("end", appointment.getEnd()).build());

}

//return json string with avaialble appointments

return Json.createObjectBuilder().add("result", "true").

add("appointments", jsonAppointmentArrayBuilder.build()

).build().toString();

} else {

//no availbale appointments for given lecturer

return Json.createObjectBuilder().

add("result", "false").build().toString();

}

} catch (SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

/\*\*

\* Web service operation to book an appointment

\*

\* @param appointmentDetails

\* @return book appointment result

\*/

@WebMethod(operationName = "bookAppointment")

public String bookAppointment(@WebParam(name = "appointmentDetails") String appointmentDetails) {

try {

//obtain appointment details for booking

JsonObject makeAppointmentDetails = Json.createReader(new StringReader(appointmentDetails)).readObject();

String studentUsername = makeAppointmentDetails.getString("studentUsername");

String lecturerUserName = makeAppointmentDetails.getString("lecturerUsername");

String date = makeAppointmentDetails.getString("date");

String start = makeAppointmentDetails.getString("start");

String end = makeAppointmentDetails.getString("end");

//look up available appointemnt

getAppointment.setString(1, date);

getAppointment.setString(2, start);

getAppointment.setString(3, end);

getAppointment.setString(4, lecturerUserName);

ResultSet resultSet = getAppointment.executeQuery();

if (resultSet.first()) {

String isActive = resultSet.getString(dbIsActiveAtt);

//check appointment is available

if (isActive.compareTo("available") == 0) {

//appointment is available then proceed with booking by SQL update statement

bookAppointment.setString(1, studentUsername);

bookAppointment.setString(2, lecturerUserName);

bookAppointment.setString(3, date);

bookAppointment.setString(4, start);

bookAppointment.setString(5, end);

//get lecturer and student details for booked appointment

int result = bookAppointment.executeUpdate();

getLecturer.setString(1, lecturerUserName);

ResultSet resultSet1 = getLecturer.executeQuery();

getStudent.setString(1, studentUsername);

ResultSet resultSet2 = getStudent.executeQuery();

if (result != 0 && resultSet1.first() && resultSet2.first()) {

//send booked appointment, student and lecturer details to lecturer email

String lecturerEmail = resultSet1.getString(dbEmailAtt);

String studentFirstName = resultSet2.getString(dbFirstNameAtt);

String studentLastName = resultSet2.getString(dbLastNameAtt);

MimeMessage message = new MimeMessage(mailSession);

try {

//create an email appointment with booking details

InternetAddress[] recipientAddress = {new InternetAddress(lecturerEmail)};

InternetAddress senderAddress = new InternetAddress(mailSession.getProperty("mail.from"));

message.addHeaderLine("method=REQUEST");

message.addHeaderLine("charset=UTF-8");

message.addHeaderLine("component=VEVENT");

message.setFrom(senderAddress);

message.setRecipients(Message.RecipientType.TO, recipientAddress);

message.setSubject("Appointment Request From " + studentFirstName + " "

+ studentLastName);

Calendar calendar = Calendar.getInstance();

message.setSentDate(calendar.getTime());

StringBuilder stringBuilder = new StringBuilder();

StringBuilder buffer = stringBuilder.append("BEGIN:VCALENDAR\n"

+ "PRODID:-//Microsoft Corporation//Outlook 9.0 MIMEDIR//EN\n"

+ "VERSION:2.0\n"

+ "METHOD:REQUEST\n"

+ "BEGIN:VEVENT\n"

+ "ATTENDEE;ROLE=REQ-PARTICIPANT;RSVP=FALSE:MAILTO:" + recipientAddress[0].toString() + "\n"

+ "ORGANIZER:MAILTO:" + senderAddress.toString() + "\n"

+ "DTSTART:" + date.replace("-", "") + "T" + start.replace(":", "") + "00Z\n"

+ "DTEND:" + date.replace("-", "") + "T" + end.replace(":", "") + "00Z\n"

+ "LOCATION:Lecturer office\n"

+ "TRANSP:OPAQUE\n"

+ "SEQUENCE:0\n"

+ "UID:040000008200E00074C5B7101A82E00800000000002FF466CE3AC5010000000000000000100\n"

+ " 000004377FE5C37984842BF9440448399EB02\n"

+ "DTSTAMP:" + calendar.get(Calendar.YEAR) + (calendar.get(Calendar.MONTH) + 1) + calendar.get(Calendar.DAY\_OF\_MONTH) + "T"

+ calendar.get(Calendar.HOUR\_OF\_DAY) + calendar.get(Calendar.MINUTE) + calendar.get(Calendar.SECOND) + "Z\n"

+ "CATEGORIES:Meeting\n"

+ "DESCRIPTION:Student meeting.\n\n"

+ "SUMMARY:Student meeting request\n"

+ "PRIORITY:5\n"

+ "CLASS:PUBLIC\n"

+ "BEGIN:VALARM\n"

+ "TRIGGER:PT1440M\n"

+ "ACTION:DISPLAY\n"

+ "DESCRIPTION:Reminder\n"

+ "END:VALARM\n"

+ "END:VEVENT\n"

+ "END:VCALENDAR");

// Create the message part

BodyPart messageBodyPart = new MimeBodyPart();

// Fill the message

messageBodyPart.setHeader("Content-Class", "urn:content- classes:calendarmessage");

messageBodyPart.setHeader("Content-ID", "calendar\_message");

messageBodyPart.setDataHandler(new DataHandler(

new ByteArrayDataSource(buffer.toString(), "text/calendar")));// very important

// Create a Multipart

Multipart multipart = new MimeMultipart();

// Add part one

multipart.addBodyPart(messageBodyPart);

// Put parts in message

message.setContent(multipart);

Transport.send(message);

} catch (MessagingException | IOException ex) {

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

}

//return confirmation result that appointment is booked

return Json.createObjectBuilder().add("result", "true").build().toString();

} else {

return Json.createObjectBuilder().

add("result", "false").build().toString();

}

} else {

//appointment cannot be booked result since it is not available

return Json.createObjectBuilder().add("result", "booked").build().toString();

}

} else {

//server error

return Json.createObjectBuilder().add("result", "error").build().toString();

}

} catch (SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

/\*\*

\* Web service operation to obtain booked appointments

\*

\* @param userDetails

\* @return list of booked appointments for a user

\*/

@WebMethod(operationName = "viewAppointment")

public String viewAppointment(@WebParam(name = "userDetails") String userDetails) {

try {

//obtain user detail

JsonObject viewAppointmentDetails = Json.createReader(new StringReader(userDetails)).readObject();

String username = viewAppointmentDetails.getString("username");

String type = viewAppointmentDetails.getString("type");

ResultSet resultSet;

//determine if user is student or lecturer

if (type.compareToIgnoreCase("student") == 0) {

//get booked appointments for student from db

getStudentBookedAppointment.setString(1, username);

resultSet = getStudentBookedAppointment.executeQuery();

} else {

//get booked appointments for lecturer from db

getLecturerBookedAppointment.setString(1, username);

resultSet = getLecturerBookedAppointment.executeQuery();

}

if (resultSet.first()) {

//booked appointments exist

List<BookedAppointmentModel> bookedAppointments = new ArrayList<>();

if (type.compareToIgnoreCase("student") == 0) {

//create data model of booked appointments for student

do {

String lecturerUsername = resultSet.getString(dbLecturerUserNameAtt);

getLecturer.setString(1, lecturerUsername);

ResultSet resultSet1 = getLecturer.executeQuery();

resultSet1.first();

BookedAppointmentModel bookedAppointment = new BookedAppointmentModel(

lecturerUsername, resultSet1.getString(dbFirstNameAtt),

resultSet1.getString(dbLastNameAtt), resultSet.getString(dbDateAtt), resultSet.getString(dbStartAtt),

resultSet.getString(dbEndAtt), resultSet.getString(dbIsActiveAtt));

bookedAppointments.add(bookedAppointment);

} while (resultSet.next());

} else {

//create data model of booked appointments for student

do {

String studentUsername = resultSet.getString(dbStudentUserNameAtt);

getStudent.setString(1, studentUsername);

ResultSet resultSet1 = getStudent.executeQuery();

resultSet1.first();

BookedAppointmentModel bookedAppointment = new BookedAppointmentModel(

studentUsername, resultSet1.getString(dbFirstNameAtt),

resultSet1.getString(dbLastNameAtt), resultSet.getString(dbDateAtt), resultSet.getString(dbStartAtt),

resultSet.getString(dbEndAtt), resultSet.getString(dbIsActiveAtt));

bookedAppointments.add(bookedAppointment);

} while (resultSet.next());

}

//build result json string from booked appointment data model

JsonArrayBuilder jsonBookedAppointmentsArrayBuilder = Json.createArrayBuilder();

for (BookedAppointmentModel appointment : bookedAppointments) {

jsonBookedAppointmentsArrayBuilder.add(Json.createObjectBuilder().

add("username", appointment.getUsername()).

add("firstName", appointment.getFirstName()).

add("lastName", appointment.getLastName()).

add("date", appointment.getDate()).

add("start", appointment.getStart()).

add("end", appointment.getEnd()).

add("status", appointment.getStatus()).build());

}

//successful result returns json with booked appointments

return Json.createObjectBuilder().

add("result", "true").

add("appointments", jsonBookedAppointmentsArrayBuilder.build()).build().toString();

} else {

//no booked appointments

return Json.createObjectBuilder().

add("result", "false").build().toString();

}

} catch (SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

/\*\*

\* Web service operation to cancel appointment

\*

\* @param appointmentDetails

\* @return

\*/

@WebMethod(operationName = "cancelAppointment")

public String cancelAppointment(@WebParam(name = "appointmentDetails") String appointmentDetails) {

try {

//obtain appointement details from json string

JsonObject cancelAppointmentDetails = Json.createReader(new StringReader(appointmentDetails)).readObject();

String username = cancelAppointmentDetails.getString("username");

String date = cancelAppointmentDetails.getString("date");

String start = cancelAppointmentDetails.getString("start");

String end = cancelAppointmentDetails.getString("end");

String firstName = cancelAppointmentDetails.getString("firstName");

String lastName = cancelAppointmentDetails.getString("lastName");

String type = cancelAppointmentDetails.getString("type");

int result;

ResultSet resultSet;

//check if student or lecturer is cancelling

if (type.compareTo("student") == 0) {

//update db with cancel, making appointment avaialble for other students

studentCancelAppointment.setString(1, username);

studentCancelAppointment.setString(2, date);

studentCancelAppointment.setString(3, start);

studentCancelAppointment.setString(4, end);

result = studentCancelAppointment.executeUpdate();

getLecturer.setString(1, username);

resultSet = getLecturer.executeQuery();

} else {

//update db with cancel, appointment no longer available since lecturer has cancelled

lecturerCancelAppointment.setString(1, username);

lecturerCancelAppointment.setString(2, date);

lecturerCancelAppointment.setString(3, start);

lecturerCancelAppointment.setString(4, end);

result = lecturerCancelAppointment.executeUpdate();

getStudent.setString(1, username);

resultSet = getStudent.executeQuery();

}

if (result != 0 && resultSet != null && resultSet.first()) {

MimeMessage message = new MimeMessage(mailSession);

try {

//send cancel appointment, student and lecturer details to lecturer email

String lecturerEmail = resultSet.getString(dbEmailAtt);

InternetAddress[] recipientAddress = {new InternetAddress(lecturerEmail)};

InternetAddress senderAddress = new InternetAddress(mailSession.getProperty("mail.from"));

message.addHeaderLine("method=CANCEL");

message.addHeaderLine("charset=UTF-8");

message.addHeaderLine("component=VEVENT");

message.setFrom(senderAddress);

message.setRecipients(Message.RecipientType.TO, recipientAddress);

message.setSubject("Appointment Cancel From " + firstName + " "

+ lastName);

Calendar calendar = Calendar.getInstance();

message.setSentDate(calendar.getTime());

StringBuilder stringBuilder = new StringBuilder();

StringBuilder buffer = stringBuilder.append("BEGIN:VCALENDAR\n"

+ "PRODID:-//Microsoft Corporation//Outlook 9.0 MIMEDIR//EN\n"

+ "VERSION:2.0\n"

+ "METHOD:CANCEL\n"

+ "BEGIN:VEVENT\n"

+ "ATTENDEE;ROLE=REQ-PARTICIPANT;RSVP=FALSE:MAILTO:" + recipientAddress[0].toString() + "\n"

+ "ORGANIZER:MAILTO:" + senderAddress.toString() + "\n"

+ "DTSTART:" + date.replace("-", "") + "T" + start.replace(":", "") + "00Z\n"

+ "DTEND:" + date.replace("-", "") + "T" + end.replace(":", "") + "00Z\n"

+ "LOCATION:Lecturer office\n"

+ "TRANSP:OPAQUE\n"

+ "SEQUENCE:0\n"

+ "UID:040000008200E00074C5B7101A82E00800000000002FF466CE3AC5010000000000000000100\n"

+ " 000004377FE5C37984842BF9440448399EB02\n"

+ "DTSTAMP:" + calendar.get(Calendar.YEAR) + (calendar.get(Calendar.MONTH) + 1) + calendar.get(Calendar.DAY\_OF\_MONTH) + "T"

+ calendar.get(Calendar.HOUR\_OF\_DAY) + calendar.get(Calendar.MINUTE) + calendar.get(Calendar.SECOND) + "Z\n"

+ "CATEGORIES:Meeting\n"

+ "DESCRIPTION:Student meeting.\n\n"

+ "SUMMARY:Student meeting request\n"

+ "PRIORITY:5\n"

+ "CLASS:PUBLIC\n"

+ "BEGIN:VALARM\n"

+ "TRIGGER:PT1440M\n"

+ "ACTION:DISPLAY\n"

+ "DESCRIPTION:Reminder\n"

+ "END:VALARM\n"

+ "END:VEVENT\n"

+ "END:VCALENDAR");

// Create the message part

BodyPart messageBodyPart = new MimeBodyPart();

// Fill the message

messageBodyPart.setHeader("Content-Class", "urn:content- classes:calendarmessage");

messageBodyPart.setHeader("Content-ID", "calendar\_message");

messageBodyPart.setDataHandler(new DataHandler(

new ByteArrayDataSource(buffer.toString(), "text/calendar")));// very important

// Create a Multipart

Multipart multipart = new MimeMultipart();

// Add part one

multipart.addBodyPart(messageBodyPart);

// Put parts in message

message.setContent(multipart);

Transport.send(message);

} catch (MessagingException | IOException ex) {

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

}

//successful result

return Json.createObjectBuilder().

add("result", "true").build().toString();

} else {

//server error

return Json.createObjectBuilder().

add("result", "error").build().toString();

}

} catch (SQLException ex) {

//server error

Logger.getLogger(AppointementServices.class.getName()).log(Level.SEVERE, null, ex);

return Json.createObjectBuilder().add("result", "error").build().toString();

}

}

}

package unitecappointmentserver;

/\*\*

\* Data model for an available appointment

\*

\* @author Marzouq Almarzooq (1380949)

\* @author Nawaf Altuwayjiri (1377387)

\*/

public class AppointmentModel {

private final String date;

private final String start;

private final String end;

/\*\*

\* Constructor

\*

\* @param date appointment date

\* @param start start time

\* @param end end time

\*/

public AppointmentModel(String date, String start, String end) {

this.date = date;

this.start = start;

this.end = end;

}

/\*\*

\* @return the date

\*/

public String getDate() {

return date;

}

/\*\*

\* @return the start

\*/

public String getStart() {

return start;

}

/\*\*

\* @return the end

\*/

public String getEnd() {

return end;

}

}

package unitecappointmentserver;

/\*\*

\* Data model for a booked appointment

\*

\* @author Marzouq Almarzooq (1380949)

\* @author Nawaf Altuwayjiri (1377387)

\*/

public class BookedAppointmentModel {

private final String username;

private final String firstName;

private final String lastName;

private final String date;

private final String start;

private final String end;

private final String status;

/\*\*

\* Constructor

\*

\* @param username booking username

\* @param firstName booking user first name

\* @param date appointment date

\* @param lastName booking user last name

\* @param start start time

\* @param end end time

\* @param status status of booking

\*/

public BookedAppointmentModel(String username, String firstName, String lastName, String date, String start, String end, String status) {

this.username = username;

this.firstName = firstName;

this.lastName = lastName;

this.date = date;

this.start = start;

this.end = end;

this.status = status;

}

/\*\*

\* @return the date

\*/

public String getDate() {

return date;

}

/\*\*

\* @return the start

\*/

public String getStart() {

return start;

}

/\*\*

\* @return the end

\*/

public String getEnd() {

return end;

}

/\*\*

\* @return the firstName

\*/

public String getFirstName() {

return firstName;

}

/\*\*

\* @return the lastName

\*/

public String getLastName() {

return lastName;

}

/\*\*

\* @return the status

\*/

public String getStatus() {

return status;

}

/\*\*

\* @return the username

\*/

public String getUsername() {

return username;

}

}

package unitecappointmentserver;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Data model for a lecturer

\*

\* @author Marzouq Almarzooq (1380949)

\* @author Nawaf Altuwayjiri (1377387)

\*/

public class LecturerModel {

private final String username;

private final String title;

private final String firstName;

private final String lastName;

private final String department;

private final List<String> subjects;

/\*\*

\* Constructor

\*

\* @param username lecturer username

\* @param title lecturer title

\* @param firstName lecturer first name

\* @param lastName lecturer last name

\* @param department lecturer department

\*/

public LecturerModel(String username, String title, String firstName,

String lastName, String department) {

this.username = username;

this.title = title;

this.firstName = firstName;

this.lastName = lastName;

this.department = department;

subjects = new ArrayList<>();

}

/\*\*

\* @return the username

\*/

public String getUsername() {

return username;

}

/\*\*

\* @return the title

\*/

public String getTitle() {

return title;

}

/\*\*

\* @return the first name

\*/

public String getFirstName() {

return firstName;

}

/\*\*

\* @return the last name

\*/

public String getLastName() {

return lastName;

}

/\*\*

\* @return the department

\*/

public String getDepartment() {

return department;

}

/\*\*

\* @return the list of lecturer subjects

\*/

public List<String> getSubjects() {

return subjects;

}

/\*\*

\* @param subject subject to add to the list of subjects

\*/

public void addSubject(String subject) {

subjects.add(subject);

}

}

## Database SQL Dump

-- MySQL dump 10.13 Distrib 5.7.12, for osx10.9 (x86\_64)

-- Author: Amelia Mcneil (1388244)

-- Host: localhost Database: unitecappointmentdb

-- ------------------------------------------------------

-- Server version 5.7.14

/\*!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 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `appointment`

--

DROP TABLE IF EXISTS `appointment`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `appointment` (

`date` varchar(50) NOT NULL,

`start` varchar(50) NOT NULL,

`end` varchar(50) NOT NULL,

`lecturerusername` varchar(100) NOT NULL,

`studentusername` varchar(100) DEFAULT NULL,

`isActive` varchar(10) NOT NULL,

PRIMARY KEY (`start`,`end`,`lecturerusername`,`date`),

KEY `studentFK\_idx` (`studentusername`),

KEY `lecturerFK\_idx` (`lecturerusername`),

CONSTRAINT `lecturerFK1` FOREIGN KEY (`lecturerusername`) REFERENCES `lecturer` (`username`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `studentFK1` FOREIGN KEY (`studentusername`) REFERENCES `student` (`username`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `appointment`

--

LOCK TABLES `appointment` WRITE;

/\*!40000 ALTER TABLE `appointment` DISABLE KEYS \*/;

INSERT INTO `appointment` VALUES ('2016-11-02','02:00','02:30','bahman ','amal','booked'),('2016-11-10','09:00','09:15','bahman','amal','booked'),('2016-10-20','10:00','10:20','bahman ','nawaf','cancelled'),('2016-10-26','15:00','15:30','bahman','amal','cancelled');

/\*!40000 ALTER TABLE `appointment` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `department`

--

DROP TABLE IF EXISTS `department`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `department` (

`department` varchar(50) NOT NULL,

PRIMARY KEY (`department`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `department`

--

LOCK TABLES `department` WRITE;

/\*!40000 ALTER TABLE `department` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `department` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `lecturer`

--

DROP TABLE IF EXISTS `lecturer`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `lecturer` (

`username` varchar(100) NOT NULL,

`title` varchar(45) NOT NULL,

`firstname` varchar(100) NOT NULL,

`lastname` varchar(100) NOT NULL,

`password` varchar(24) NOT NULL,

`salt` varchar(24) NOT NULL,

`department` varchar(100) NOT NULL,

`email` varchar(200) NOT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `lecturer`

--

LOCK TABLES `lecturer` WRITE;

/\*!40000 ALTER TABLE `lecturer` DISABLE KEYS \*/;

INSERT INTO `lecturer` VALUES ('bahman','Mr.','Bahman','Sarrafpour','HFBtxKIudUuiYjZd8j8Hxw==','DBrzy1ANbFtIA4OFzq5Epg==','Computer Science','e-9p@hotmail.com'),('mark','Dr.','Mark','McNeil','COA8GE6VlxUHiVor8JYBrA==','FBR1RhlXfqHayEaK/R+CJg==','Computer Science','e-9p@hotmail.com');

/\*!40000 ALTER TABLE `lecturer` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `student`

--

DROP TABLE IF EXISTS `student`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `student` (

`username` varchar(100) NOT NULL,

`firstname` varchar(100) NOT NULL,

`lastname` varchar(100) NOT NULL,

`password` varchar(24) NOT NULL,

`salt` varchar(24) NOT NULL,

`email` varchar(200) NOT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `student`

--

LOCK TABLES `student` WRITE;

/\*!40000 ALTER TABLE `student` DISABLE KEYS \*/;

INSERT INTO `student` VALUES ('amal','Amelia','Mcneil','HxAt3y+V3BbOFGZoEUnTUA==','ZUc8wfZiMmYf6dbbgRT55w==','amelia-mcneil@hotmail.com'),('nawaf','Nawaf','Altuwayjiri','DvAJmOVXSY0uwE1KIvRTqw==','g1+6K8PbP47GlDO2S8br3g==','amelia-mcneil@hotmail.com'),('osh','Oshada','Koralage','TDpwmgX20ZydRo3r4MLRTQ==','WzOXJktbn/QxU2JN2qCk5Q==','amelia-mcneil@hotmail.com'),('zezo','Marzouq','Almarzooq','VNdTKuubW/vZH6E36EM9Vw==','lxcs1/6QcpM4LJV6Syoc0g==','e-9p@hotmail.com');

/\*!40000 ALTER TABLE `student` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `studentlecturerassignment`

--

DROP TABLE IF EXISTS `studentlecturerassignment`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `studentlecturerassignment` (

`lecturerusername` varchar(100) NOT NULL,

`studentusername` varchar(100) NOT NULL,

`subject` varchar(45) NOT NULL,

PRIMARY KEY (`lecturerusername`,`studentusername`,`subject`),

KEY `studentfk\_idx` (`studentusername`),

CONSTRAINT `lecturerfk` FOREIGN KEY (`lecturerusername`) REFERENCES `lecturer` (`username`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `studentfk` FOREIGN KEY (`studentusername`) REFERENCES `student` (`username`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `studentlecturerassignment`

--

LOCK TABLES `studentlecturerassignment` WRITE;

/\*!40000 ALTER TABLE `studentlecturerassignment` DISABLE KEYS \*/;

INSERT INTO `studentlecturerassignment` VALUES ('bahman','amal','Cyber Security'),('bahman','amal','Hardware Technologies'),('mark','amal','Programming Fundementals'),('bahman','nawaf','Hardware Technologies'),('bahman','osh','Hardware Technologies'),('mark','osh','Programming Fundementals');

/\*!40000 ALTER TABLE `studentlecturerassignment` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `subject`

--

DROP TABLE IF EXISTS `subject`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `subject` (

`subject` varchar(45) NOT NULL,

`department` varchar(45) DEFAULT NULL,

PRIMARY KEY (`subject`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `subject`

--

LOCK TABLES `subject` WRITE;

/\*!40000 ALTER TABLE `subject` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `subject` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping events for database 'unitecappointmentdb'

--

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!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 \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2016-11-15 17:31:24

# Bibliography

*Android Studio*. (2016, October 24). Retrieved from Android Studio Features: https://developer.android.com/studio/features.html

digicert. (2016). *What Is SSL (Secure Sockets Layer) and What Are SSL Certificates?* Retrieved from digicert: https://www.digicert.com/ssl.htm

Garrett, J. J. (n.d.). *User Interface Design Basics*. Retrieved from usability.gov: https://www.usability.gov/what-and-why/user-interface-design.html

*MySQL Administration Guide*. (2015). Retrieved from Novell: http://www.novell.com/documentation/nw65/web\_mysql\_nw/data/aj5bj52.html

NetBeans. (2016). *Netbeans IDE - The smart and faster way to code*. Retrieved from Netbeans: https://netbeans.org/features/

Oracle. (n.d.). *Oracle* . Retrieved from Oracle Glassfish Server: http://www.oracle.com/us/products/middleware/application-server/050870.pdf

Rajput, M. (2015, May 20). *Why Android Studio Is Better For Android Developers Instead Of Eclipse*. Retrieved from DZone/Mobile Zone: https://dzone.com/articles/why-android-studio-better

Rosoff, M. (2015, December 8). *The research firm that once thought Microsoft would beat the iPhone has given up on Windows Phone*. Retrieved from Business Insider: http://www.businessinsider.com.au/idc-smartphone-os-market-share-2015-12?r=US&IR=T

Saab, C. B., Haddad, S., Coulibaly, D., Melliti, T., Moreaux, P., & Rampacek, S. (2009). An Integrated Framework for Web services orchestration. (L.-J. Zhang, Ed.) *International Journal of Web Services Research, Volume 6, Issue 4* , 3.

Security, D. (2016, September 26). *Salted Password Hashing - Doing it Right*. Retrieved from CrackStation: https://crackstation.net/hashing-security.htm#normalhashing

Server, W. (n.d.). *Microsoft*. Retrieved from Web Server (IIS) Role Overview: https://technet.microsoft.com/en-us/library/cc770634(v=ws.11).aspx

Wawrzyniak, P. (2014, June 29). *Scrum vs Waterfall – two different approaches to software development*. Retrieved from turingsman: http://turingsman.net/my-blog-list/183-scrum-vs-waterfall-two-different-approaches-to-software-development

*Web Server (IIS) Role Overview*. (2016). Retrieved from Microsoft: https://technet.microsoft.com/en-us/library/cc770634(v=ws.11).aspx

*what is scrum methodology?* (n.d.). Retrieved from VersionOne: https://www.versionone.com/agile-101/what-is-scrum/

# Individual Contribution Review

This form is to be completed **once** by the group and included with your documentation.  You may use several forms if necessary. If the form is missing or empty it will be assumed that all students contributed equally.

**Each group member must sign this form as evidence that they agree with its content.**

**Name:** Amelia McNeil **Signature:**

**Name:** Marzouq Almarzooq **Signature:**

**Name:** Nawaf Altuwayjiri **Signature:**

**Name:** Oshada Koralage **Signature:**

Enter the task and/or sub-topic and the name of the student to whom it was assigned. Should a task have been shared, then enter an agreed estimate of percentage involvement beside each name. If only one name appears then it is assumed that the individual contributed 100%. Each task should require approximately the same amount of effort.

|  |  |  |
| --- | --- | --- |
| **Task/Sub-topic** | **Names** | **%** |
| Project Manager | Amelia | 100% |
| Documentation | Amelia | 70% |
| Documentation | Marzouq | 10% |
| Documentation | Oshada | 10% |
| Documentation | Nawaf | 10% |
| Programming (Android implementation) | Oshada | 100% |
| Database Development | Amelia | 100% |
| Server (Implementation) | Marzouq | 50% |
| Server (Implementation) | Nawaf | 50% |
| Android Interface Design | Oshada | 100% |
|  |  |  |
|  |  |  |
|  |  |  |

# Project proposal

Appointment Management System

Project Team

2016

Marzouq 1380949 | Amelia 1388244 | Nawaf 1377387 | Oshada 1434048

Table of Contents

[1. Executive Summary 1](#_Toc457819229)

[2. Goal 2](#_Toc457819230)

[3. Background 2](#_Toc457819231)

[4. Objectives 2](#_Toc457819232)

[5. Justification 3](#_Toc457819233)

[6. Scope 3](#_Toc457819234)

[6.1 Functional Requirements 4](#_Toc457819235)

[6.2 Exclusions 4](#_Toc457819236)

[7. Project Team 5](#_Toc457819237)

[8. Methodology 5](#_Toc457819238)

[9. Deliverables 6](#_Toc457819239)

[10. Milestones and Tasks 7](#_Toc457819240)

[11. Resource Requirements and Costs 8](#_Toc457819241)

[12. Risk Analysis Checklist 9](#_Toc457819242)

[12.1 Generic Risk Checklist 9](#_Toc457819243)

[12.2 Specific Project Risks 11](#_Toc457819244)

[13. Quality Assurance Process 12](#_Toc457819245)

[14. Project Work Plan 13](#_Toc457819246)

[15. Intellectual Property 13](#_Toc457819247)

[16. Confidentiality 13](#_Toc457819248)

[17. Declaration 14](#_Toc457819249)

[18. Approvals 15](#_Toc457819250)

### Executive Summary

From time to time students require office hour access with lecturers to address and resolve any learning related issues and difficulties. Currently the practice is for students to usually seek a particular lecturer by visiting the staff room or office to either deal with a learning matter or make arrangements to address the matter at a more suitable time. More often this practice results in a continuous flow of students desiring to have out of class access to their lecturers. Consequently, this causes frustrations for both students and lecturers, since lecturers are interrupted, and students experience delays. To minimize the frustration’s lecturers have adopted their own policies to regulate and manage the flow of students requiring further learning assistance. As these policies vary between lecturers, the system continues to be problematic and confusing.

The proposal is to set up a computerised system with a common clear standard to reduce inefficiencies arising from inconsistent policies. More so the aim is to provide a useable solution to lessen the burden and frustrations experienced by both students and lecturers. The system will be deployed via a web-based application and mobile application allowing lecturers to advertise their availability to students requiring additional office hour access. The system will also offer both students and lecturers the ability to manage their office hour appointments.

### Goal

The target of the project is to deliver an efficient and more effective way of making appointments for both the students and lecturers. The web-based application and mobile application must provide a useable interface that facilitates the appointment process and management for both students and lecturers.

### Background

Unitec is the biggest Institute in New Zealand with over 16,000 attending students and over 1000 teaching staff. Students and teaching staff are located across three campuses. All campuses are located in Auckland, with the main campus situated in Mt. Albert, the other two situated in Henderson and Albany.

Given the large size of Unitec it is imperative to efficiently manage the logistics for office hour appointments between students and lecturers. Currently there is no set standard policy that defines a best practice, and no system in place to facilitate the process.

### Objectives

1. Building a web application and mobile application for making office hour appointments between students and lecturers.
2. Provide students with priori information as to when lecturers are available for an appointment to be made.
3. Minimise the time taken to set up an appointment.
4. Both the web-application and mobile application should also provide some integration with lecturer’s Outlook based calendar.
5. Enable appointments to be amended after these have been made.
6. Provide secure communication.

### Justification

The project was considered as many students were finding it difficult to get an appointment with their lecturers. Based on feedback provided by students at Unitec there is no standard clear way of contacting a lecturer. Many staff members have resorted to defining their own policies for students wishing to meet with them causing inadvertent frustrations, confusion, and complications.

At times a lecturer could be busy and cannot check all their emails to see the student emailed them to make an appointment, consequently students will just show up unexpectedly and interrupt the staff members. Another example is sometimes a student will show up and there are many students lined up waiting to see the same lecturer causing human traffic. On other occasions students may find a sign on the door indicating the member of staff is only vacant on particular times.

Both students and the staff need an organized and consistent way of arranging and managing meetings. We have researched the project in terms of what skills it takes to create a web application as this can provide convenient wide access across all campuses. Using this idea as our final project, we intended to improve the current system for making appointments between lecturer and student by facilitating the interaction and reduce inconsistencies.

This project will provide an opportunity to create a useful web application, and further develop our programming skills across a wide area of ICT technologies span ranging from client-side to server-side as well as database design and implementation. As the system we are developing is distributed, hence having a background in computer networking will be very useful.

### Scope

Our scope is to create an appointment booking and management web application and also a mobile application that will assist students and lecturers in making appointments between each other. The desire of this new process of making an appointment through the web application and mobile application is to provide greater standardisation and reduce current inefficiencies arising from inconsistent practices.

The web application will serve as a service for lecturers and student to interact. The web application will assist students in making an appointment through a structured and uniform calendar based system, which will save resources over the unstructured system currently in use. The creation of this web application and the mobile application will encourage a common standard.

### Functional Requirements

**Web Contents**

The web application will have several user views. There will be a common navigation interface across all user views to assist with usability of the web-application. The user views will include:

Home Page

* Information about the website
* Demo how to set up appointments

Profile Page

* Student profile view /Lecturer profile view
* Upcoming appointments
* Saved lecturers/students based on classes

Appointment Page

* Calendar view
* Booking appointments

Lecturers Page

* Preview of the lecturer’s name and picture if available
* Description of the lecturer’s details including their position in Unitec and the papers they teach

Contact

* Contact details
* Email for trouble logging in

### Exclusions

1. There will be no desktop application, as the client interaction is web-based.
2. There will be no IOS compatible application. However, the aforementioned devices can access the system via their respective web browsers.

### Project Team

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unitec Team members** | **Name** | **Student ID** | **Email** | **Phone** |
| Marzouq Almarzooq | 1380949 | [e-9p@hotmail.com](mailto:e-9p@hotmail.com) | 0222584444 |
| Amelia McNeil | 1388244 | [Amillie.rez@gmail.com](mailto:Amillie.rez@gmail.com) | 0221654743 |
| Nawaf Altuwayjiri | 1377387 | [nawaf.s.h@hotmail.com](mailto:nawaf.s.h@hotmail.com) | 0223914544 |
| Oshada | 1434048 | [Oshada78@gmail.com](mailto:Oshada78@gmail.com) | 02040851845 |

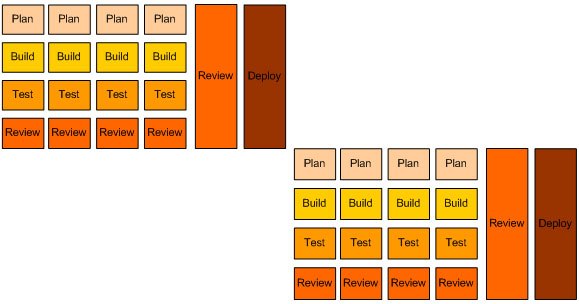
|  |  |  |  |
| --- | --- | --- | --- |
| **Project Supervisor from sponsoring company.** | Bahman  Sarrafpour | Department of computing  Unitec Institute of Technology | 098154321 ext.6043  bsarrafpour@unitec.ac.nz |

### Methodology

**Agile methodology - Scrum Development**

Initially we looked at a waterfall method so that we could complete a phase before moving onto the next phase this would keep the project organized, however this development carries a lot of risk because in this method we cannot revisit a phase, once it is completed you are water falling you cannot go back. We decided that the waterfall method was not suitable due to high risk factor.

A more risk averse approach is to use an Agile methodology such as Scrum. Scrum development is achieved with sprints. Sprints would allow us to focus on delivering independent, tested features within manageable workloads, hence minimising risk as shown in the diagram below. This approach will allow us to plan, build, test and review each feature and then work on the next feature in a more structured and reliable fashion.



### Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **Description** | **Contribution to the project** |
| **Project proposal** | First documentation, proposing the idea. | 10% |
| **Project resources** | Gathering resources needed for the project software, hardware etc. | 20% |
| **Project requirements** | Computers, software, hardware | 10% |
| **Client-Side Application Development** | Development of the web application and the mobile application | 10% |
| **Server-Side Web Development** | Development using NetBeans | 10% |
| **Relational Database Management System Development {Multi-Tier System Development}** | Development using MySQL | 10% |
| **Final Web and mobile application** | Final stages on building the web application and mobile application. | 20% |
| **Final Documentation** | Documents ready for submission. | 5% |
| **Project Presentation** | The final presentation. | 5% |

### Milestones and Tasks

|  |  |
| --- | --- |
| Milestone and Tasks | Description |
| Project meeting 1 | First project meeting |
| Hand-in project proposal | Submit final proposal |
| Proposal defence | Presentation and defence in front of panel |
| User Stories | Create User stories |
| Scrum Sprints | Developing the applications in individual sprints |
| Testing final project | Test completed applications |
| Hand-in project documentation | Finalise projects final documentation |
| Project presentation and defence | Final Presentation |
| Close project | Project is finished |

### 

### Resource Requirements and Costs

|  |  |  |
| --- | --- | --- |
| **Resources provided by Unitec** | | |
| **Resource** | **Quantity** | **Status** |
| PC | 3 | Available |
| Swipe Cards | 4 | Available |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software needed** | | | | |
| **Resource** | **Quantity** | **Status** | **Price** | **Reference** |
| Microsoft Office (Word, PowerPoint, Excel etc.) | 3 | Available | Provided by Unitec | |
| MySQL | 3 | Available | Free | <https://www.mysql.com/downloads/> |
| J Query | 3 | Available | Free | <https://jquery.com/download/> |
| NetBeans | 3 | Available | Free | <https://netbeans.org/> |
| Adobe -Dreamweaver | 3 | Available | Free | Provided by Unitec |
| VisualStudio | 3 | Pending | Free | Provided by Unitec |
| AndroidStudio | 3 | Pending | Free | Provided by Unitec |

### Risk Analysis Checklist

### 12.1 Generic Risk Checklist

The following risk checklist is a generic model and is used to give an overall (non-specific) picture of the project’s risk factors. It can be used to compare the relative risks to the organisation of a number of different projects.

|  | **Low (Yes)** | | | | **RISK** | | | | **(No) High** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 9 |
| **Inherent Risks** |  |  |  |  | |  |  |  | |  |  |  |
| **Project Objectives** |  |  |  |  | |  |  |  | |  |  |  |
| Is the project small? |  |  |  |  | |  |  |  | |  | **√** |  |
| Is the project of minor importance to the business? |  |  |  |  | |  | **√** |  | |  |  |  |
| Is the project functionally straightforward? |  |  |  |  | |  | **√** |  | |  |  |  |
| Are several parties able to define the requirements? |  |  |  |  | |  |  | **√** | |  |  |  |
| Is the subject area well documented? |  |  |  |  | | **√** |  |  | |  |  |  |
| Are preceding projects well documented? |  |  |  |  | |  |  | **√** | |  |  |  |
| **User Organisation** |  |  |  |  | |  |  |  | |  |  |  |
| Does the project maintain existing user procedures? |  |  |  |  | |  |  |  | | **√** |  |  |
| Is other organisational change unlikely during the project? |  |  |  |  | |  |  |  | | **√** |  |  |
| Are the users grouped in one location? |  | **√** |  |  | |  |  |  | |  |  |  |
| **Technology** |  |  |  |  | |  |  |  | |  |  |  |
| Is tried hardware being used? |  |  |  | **√** | |  |  |  | |  |  |  |
| Is tried software being used? | **√** |  |  |  | |  |  |  | |  |  |  |
| Can custom programming be avoided? |  |  |  |  | |  | **√** |  | |  |  |  |
| Is the project technically straightforward? |  |  |  |  | |  |  | **√** | |  |  |  |
| Is the quality of existing data good? |  |  |  |  | | **√** |  |  | |  |  |  |
| **Acquired Risks** |  |  |  |  | |  |  |  | |  |  |  |
| **Scope and Approach** |  |  |  |  | |  |  |  | |  |  |  |
| Is the project scope well defined and agreed? |  |  | **√** |  | |  |  |  | |  |  |  |
| Is the project approach well defined and agreed? |  |  | **√** |  | |  |  |  | |  |  |  |
| **Project Organisation** |  |  |  |  | |  |  |  | |  |  |  |
| Are people’s roles clearly defined? |  |  |  |  | | **√** |  |  | |  |  |  |
| Are users committed to the project? |  |  | **√** |  | |  |  |  | |  |  |  |
| Are staff able to commit sufficient time to the project? | **√** |  |  |  | |  |  |  | |  |  |  |
| Are the required skills available? |  | **√** |  |  | |  |  |  | |  |  |  |
| Does backup exist for all members of the project? |  | **√** |  |  | |  |  |  | |  |  |  |
| Are political and personal relationships good? | **√** |  |  |  | |  |  |  | |  |  |  |
| Is the project independent of third parties? |  | **√** |  |  | |  |  |  | |  |  |  |
| Can a small group achieve the design? |  | **√** |  |  | |  |  |  | |  |  |  |
| Can a “Big Bang” implementation be avoided? |  |  |  |  | | **√** |  |  | |  |  |  |
| **Experience, Training and Support** |  |  |  |  | |  |  |  | |  |  |  |
| Does the IT team know the technology? |  | **√** |  |  | |  |  |  | |  |  |  |
| Do the users know the technology? |  |  |  | **√** | |  |  |  | |  |  |  |
| Is the technology well supported? |  |  | **√** |  | |  |  |  | |  |  |  |

### 

### 12.2 Specific Project Risks

This section identifies risks that are specific to this project and are in addition to the generic risks considered above.

| Issue | Probability | Impact | Schedule | Issue/Action |
| --- | --- | --- | --- | --- |
| Misplaced documentation | Low | Low |  | Make sure to save regularly and make multiple copies on USB’s, emails and computers to be cautious. |
| Project sponsor changes the original scope | Medium | High |  | Project will be delayed. |
| Team member does not contribute or is unable to work due to unexpected circumstances | Medium | Low |  | Some case the member could be slacking and not paying full attention with helping the in the project due to lack of motivation.  Family emergency arises or member falls ill and cannot help the team until said person’s health has improved.  There are no ways to avoid unexpected circumstances. In this situation the other group members will contribute work for that individual. |
| Security breach (computer virus) | Low | Medium |  | Security breaches can occur unexpectedly to prevent this we need to make sure the computers we are working on have anti-virus security installed. If we do get a virus and we are not protected it can affect our whole project. |
| Proposal gets rejected | Low | Medium |  | The project could get rejected due to many reasons including not enough supporting facts.  Before submitting the proposal, a lot of research is required stating the many benefits of the applications and how it could impact the students and lecturers.  We will make sure all the key points are clearly stated in the proposal. |
| PC cannot connect to server | Low | Low |  | We will have trouble shoot programs in backup to prevent such scenarios and to make sure there is always an available server for the PC to connect to. |

### 

### Quality Assurance Process

To assure quality of the project we will have iterative tests, this way we won’t be leaving testing to the last stage, we will be building a feature and testing it to assure quality.

Another technique to maintain quality assurance we will use will be pilot testing by using our fellow students to try the feature and provide us with feedback of how the feature worked for them was it easy to use and why, what problems they experienced etc. Pilot testing.

### Project Work Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task/Milestone** | **July** | **August** | **September** | **October** | **November** |
| Proposal |  |  |  |  |  |
| Proposal Presentation |  |  |  |  |  |
| User stories/ Use Cases |  |  |  |  |  |
| Scrum Sprints |  |  |  |  |  |
| Final Report Write-up |  |  |  |  |  |
| Final Presentation/  Demo |  |  |  |  |  |

### 

### Intellectual Property

The deliverables created in this project will belong to Bahman Sarrafpour the project supervisor and the project team.

### Confidentiality

UNITEC is not permitted to retain a copy of the project documentation for academic purposes including: assessment, accreditation and peer review, without permission.

### Declaration

I am not employed by the sponsoring organization.

The people involved with this project from the sponsoring organization are not related to me or close friends.

I declare that the information in this proposal is, to the best of my knowledge, correct.

Team member name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team member name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team member name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team member name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Approvals

Date approval in principle received:

I agree that the scope, objectives, resource estimates and plans given in this project proposal describe my general requirements for the project. I confirm that I have the authority to approve the expenditure outlined in this proposal. I understand that this is a student project and that UNITEC and the students will endeavour to provide the services described but for whatever reason may not be able to do so.

Project Sponsor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_\_\_\_/\_\_\_\_/\_\_\_\_\_\_\_\_

I agree that the project described in this project proposal is generally suitable to meet the learning outcomes for the course ISCG 7431 Capstone Project.

UNITEC Project Coordinator \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_\_\_\_/\_\_\_\_/\_\_\_\_\_\_\_\_