-

**WIREFRAMES**

reports

* # apt types by month
* Schedule for each consultant
* Login/logout

Customer records

* Add record
* Delete record
* Update record

Appointment

* Add appointment
  + Type of appointment
  + Link toCustomer in database
* Delete appointment
* Update appointment

Calendar

* View by month
* View by week

Login

* Ability to determine user’s location
* Translate in two languages

Menu

* Customer Records
* Appointment
* Calendar
* reports

|  |
| --- |
| Appointment |
| * appointmentID : int * title : String * description : String * location : String * contact : String * type : String * start : DATETIME * end : DATETIME |
| + Appointment(customerID : int, userID : int, appointmentID : int, title : String, description : String, location : String, contact : String, type : String, start : DATETIME, end : DATETIME)  + getAppointmentID() : int  + getTitle() : int  + getDescription() : String  + getLocation() : String  + getContact() : String  + getType() : String  + getStart() : DATETIME  + getEnd() : DATETIME  + setAppointmentID(int) : void  + setTitle(int) : void  + setDescription(String) : void  + setLocation(String) :void  + setContact(String) : void  + setType(String) : void  + setStart(DATETIME) : void  + setEnd(DATETIME) : void |

|  |
| --- |
| AppointmentList |
| * appointmentList : ObservableList <Appointment> |
| + addAppointment(Appointment) : void  + updateAppointment(Appointment) : void  + deleteAppointment(Appointment) : void  + getAppointmentsByWeek(Date) : ObservableList  + getAppointmentsByMonth(Date):ObservableList |

**UML**

|  |
| --- |
| Customer |
| * customerID : int * customerName: String |
| + Customer( countryID : int, country : String. CityID : int, city: String, int : addressID, String : address, String : address2, String : postalCode, String : Phone, int : CustomerID, String : customerName)  + getCustomerID() : int  + getCustomerName(): String  + setCustomerID(int) : void  + setCustomerName(String) : void  + updateCustomer |

|  |
| --- |
| User |
| * userID : int * userName : String * password : String |
| + User(userID : int, userName : String, password : String)  + getUserID() : int  + getUserName() : String  + getPassword() : String  + setUserID(int : id) : void  + setUserName(String : username) : void  + setPassword(String : password) : void |

|  |
| --- |
| Address |
| * addressID: int * address: String * address2: String * postalCode: String * phone: String |
| + Address( countryID : int, country : String. CityID : int, city: String, int : addressID, String : address, String : address2, String : postalCode, String : Phone)  + getAddressID() : int  + getAddress() : String  + getAddress2() : String  + getPostalCode() : String  + getPhone() : String  + setAddressID(addressID : int) : void  + setAddress(address : String) : void  + setAddress2(address2 : String) : void  + setPostalCode(postalCode : String) : void  + setPhone(phone : String) : void |

|  |
| --- |
| City |
| * cityID: int * city: String |
| + City( countryID : int, country : String. CityID : int, city: String)  + getCityID(): int  + getCity(): String  + setCityID(int: id): void  + setCity(String: city): void |

|  |
| --- |
| Country |
| * countyID: int * country: String |
| + Country(countryID : int, country : String)  + getCountryID():int  + getCountry():String  + setCountyID(countryID: int):void  + setCountry(country: String):voide |

SOFTWARE FUNCTIONALITIES

A.   Create a log-in form that can determine the user’s location and translate log-in and error control messages (e.g., “The username and password did not match.”) into **two** languages.

B.   Provide the ability to add, update, and delete customer records in the database, including name, address, and phone number.

C.   Provide the ability to add, update, and delete appointments, capturing the type of appointment and a link to the specific customer record in the database.

D.   Provide the ability to view the calendar by month and by week.

E.    Provide the ability to automatically adjust appointment times based on user time zones and daylight saving time.

F.   Write exception controls to prevent each of the following. You may use the same mechanism of exception control more than once, but you must incorporate at least  **two** different mechanisms of exception control.

•   scheduling an appointment outside business hours

•   scheduling overlapping appointments

•   entering nonexistent or invalid customer data

•   entering an incorrect username and password

G.  Write two or more lambda expressions to make your program more efficient, justifying the use of each lambda expression with an in-line comment.

H.   Write code to provide an alert if there is an appointment within 15 minutes of the user’s log-in.

I.   Provide the ability to generate each  of the following reports:

•   number of appointment types by month

•   the schedule for each consultant

•   one additional report of your choice

J.   Provide the ability to track user activity by recording timestamps for user log-ins in a .txt file. Each new record should be appended to the log file, if the file already exists.

K. Demonstrate professional communication in the content and presentation of your submission.

**Log-in**

The log-in form has functionality to determine the user’s location and translate log-in and error control messages into 2 languages. The code is complete and functions properly

**CUSTOMER RECORDS**

The application has functionality to add, update, and delete customer records in the database, including name, address, and phone number. The code is complete and functions properly.

**APPOINTMENT RECORDS & LINK APPOINTMENT TO CUSTOMER**

The application code has functionality to add, update, and delete appointments, capture the type of appointment, and link the appointments to the specific customer record in the database. The code is complete and functions properly.

**APPOINTMENTS VIEW**

The application has functionality to view the calendar by month and by week. The code is complete and functions properly.

**TIME ZONES**

The application has functionality to automatically adjust appointment times based on user time zones and daylight saving time. The code is complete and functions properly.

**EXCEPTION CONTROLES**

The application code includes exception controls to prevent each of the given points and uses at least 2 different mechanisms. The code is complete and functions properly.

**LAMBDA EXPRESSIONS**

The application code includes two or more appropriate lambda expressions to make the program more efficient and provides a logical justification of the use of each lambda expression with in-line comments. The code is complete and functions properly.

**appointment alert**

The application has functionality for an alert if there is an appointment within 15 minutes of the user’s log-in. The code is complete and functions properly.

**generating reports**

The application has functionality to generate each of the given reports. The code is complete and functions properly.

**track user activity**

The application has functionality to track user activity by recording timestamps for user log-ins in a .txt file, and each new record is appended to the log file if the file already exists. The code is complete and functions properly.

**COMPETENT**

Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding.

**ERD**

