**EasyAppointmentSystem Web**

IS2103 Assignment 2 (Team Members: Adele Loh, Teh Jin Woon Valencia, Ma Jieying)

<https://github.com/adelelohzh/NUSIS2103_2021sem2_Group8.git>

**Introduction**

The EasyAppointment System is an e-commerce service for customers to make appointments with different business service providers via a mobile application. There are a total of three user interfaces: administrator interface, service provider interface and customer interface.

**Entity Classes**

There are a total of 5 entity classes: AdminEntity, AppointmentEntity, BusinessCategoryEntity, CustomerEntity and ServiceProviderEntity, each containing key attributes of the users and mappings between the entities.

1. AdminEntity

- (Long) adminId: Key

- (String) email

- (String) password

- (String) name

2. AppointmentEntity

- (Long) appointmentId;

- (String) appointmentNo;

- (String) scheduledTime;

- (String) scheduledDate;

- (Boolean) hasRating

- @ManyToOne mappings to other entities:

- CustomerEntity customerEntity; (Bi-directional)

- ServiceProviderEntity serviceProviderEntity; (Bi-directional)

- BusinessCategoryEntity businessCategoryEntity; (Uni-directional)

The AppointmentEntity is mapped ManyToOne to CustomerEntity because one customer can have unlimited appointments made on the EasyAppointmentSystem Web portal.

Likewise, a service provider can make unlimited appointments on the portal, therefore there is a ManyToOne relationship between AppointmentEntity and ServiceProvideEntity.

Lastly, the AppointmentEntity has a ManyToOne mapping to businessCategoryEntity as the customers are required to select the business categories that they are interested in when making an appointment, hence there are unlimited appointments that can be made per business category. However, this is a unidirectional mapping as the Business Category Session Bean should not be able to retrieve the appointments that are made.

3. BusinessCategoryEntity

- (Long) businessCategoryId;

- (String) category;

- @OneToMany(mappedBy = "mappedByBusinessCategoryEntity")

- List<ServiceProviderEntity> serviceProviderEntities; (Bi-directional)

The BusinessCategoryEntity is mapped OneToMany to ServiceProviderEntity, as one business category can be shared by multiple service providers.

4. CustomerEntity

- (Long) customerId;

- (String) identityNumber;

- (String) password;

- (String) firstName;

- (String) lastName;

- (String) fullName;

- (Character) gender;

- (Integer) age;

- (String) phoneNumber;

- (String) address;

- (String) city;

- (String) emailAddress;

- @OneToMany(mappedBy = "customerEntity")

- List<AppointmentEntity> appointmentEntities; (Bi-directional)

The CustomerEntity is mapped OneToMany to the AppointmentEntity, as one customer can have multiple appointments.

5. ServiceProviderEntity

- (Long) serviceProviderId;

- (String) name;

- (String) businessCategory;

- (String) businessRegistrationNumber;

- (String) city;

- (String) businessAddress;

- (String) emailAddress;

- (String) phoneNumber;

- (String) password;

- (Double) rating;

- (Integer) numberOfRatings;

- (StatusEnum) statusEnum;

- @OneToMany(mappedBy = "serviceProviderEntity")

- List<AppointmentEntity> appointmentEntities; (Bi-directional)

- @ManyToOne mapping to BusinessCategoryEntity

private BusinessCategoryEntity businessCategoryEntity; (Bi-directional)

The ServiceProviderEntity is mapped OneToMany to the AppointmentEntity as one service provider can have multiple appointments.

On the other hand, it has a ManyToOne mapping to BusinessCategoryEntity as many service providers are classified under the same business category.

**Session Beans**

**Stateful Session Bean**

There is one stateful session bean.

1. BookingEntitySessionBean:

The BookingEntitySessionBean retains the conversational state of the customer making use of the EasyAppointmentSystem portal to make appointment(s). The main function of this session bean is to make a booking for a customer, through initialising the state of the session bean to allow a customer to start making a booking and setting all the relevant attributes required for an appointment to be made. At the end of the session, the BookingEntitySessionBean will clear the Booking cart for the next customer to invoke the session bean method.

**Stateless Session Beans**

There are 5 stateless session beans.

1. AppointmentEntitySessionBean

The AppointmentEntitySessionBean contains the basic creation, login, retrieval, updating and deleting functions. The most important functions of this session bean is the creation of new appointments, as well as the retrieval of sorted appointments (in ascending order of scheduled times) given the date and service provider id.

2. BusinessCategoryEntitySessionBean

The BusinessCategoryEntitySessionBean contains the basic creation, login, retrieval, updating and deleting functions. The most important functions of this session bean is the creation and deletion of business category(s).

3. CustomerEntitySessionBean

The CustomerEntitySessionBean contains the basic creation, login, retrieval, updating and deleting functions. The most important function of this session bean is the retrieval of the customer entity by customer id, which was used in the SystemAdministrationModule of the customer portal and the admin portal.

4. EmailEntitySessionBean

The EmailEntitySessionBean contains one method, which is used for the sending of reminder emails to a customer. It makes use of synchronous processing for the sending of email.

5. ServiceProviderEntitySessionBean

The ServiceProviderEntitySessionBean contains the basic creation, login, retrieval, updating and deleting functions. The most important functions of this session bean is the retrieval functions, which were used throughout the SystemAdministrationModules in all client applications.

**General Notes (about the system)**

1. Appointments have to be made 2 hours prior to the actual appointment.

2. Customers are advised to cancel their appointments if they are unable to make it for the appointment.

3. Appointment numbers generated are unique, and are formed by combining the service provider's UIN, followed by the date and appointment type.

4. Any input mismatch/invalid inputs will throw an exception with a relevant exception statement being printed out.

5. Appointments can only be made on Mondays to Saturdays. Time slots available are in 1-hour intervals from 08:30 to 18:30.

6. Customers are not allowed to view nor book appointments from service providers who are pending for approval or blocked by an admin from EasyAppointment.

7. Customers are not allowed to book a separate appointment from another service provider when he already has an appointment on that same day and time slot.

8. Customers should not be able to rate an appointment more than once.

9. Customers can only rate the service from a service provider after the appointment has completed.

10. An admin is unable to approve a service provider who is already blocked previously.

**Different Terminals**

1. EasyAppointmentSystemWebCustomerClient

a) MainApp: For customers to register, log in or exit to terminate session.

- runApp(): Displays the list of operations for the customer: Register, Login and Exit (terminate the session).

- doRegister(): Customer is required to enter their identity number, password, first name, last name, gender, age, phone number, address, city and email. The customerEntity instantiated in this class will set all this attributes if there are no errors with the inputs, and direct customer to the menuMain(customerEntity) which is for logged in customers. The logged in customer will then be able to choose what he/she wants to do with respect to searching for, adding, viewing, cancelling appointments, rating service providers for fulfilled appointments or logging out.

- doLogin(): Customer is required to enter username (email address) and password. Invalid login credentials will result in an error being thrown and the customer being redirected to the runApp() page.

- menuMain(customerEntity): Only customers who have logged in successfully will be directed to menuMain, where he/she will be able to choose what he/she wants to do with respect to searching for, adding, viewing, cancelling appointments, rating service providers for fulfilled appointments or logging out.

b) SystemAdministrationModule: For customers to view, add, cancel appointments and rate service providers.

- viewAppointments(): No input required. Prints out the list of previous and upcoming appointments of logged in customer.

- searchOperation(): Logged in customer enters the business category id, city, date of appointment and will be able to see the list of service providers matching their conditions with their first available time. Exceptions will be thrown if there is input mismatch, or if serviceProviderIds and BusinessCategoryIds entered do not match any in the database.

- addAppointment(): Customers are required to key in the business category id, city, date of appointment and will be able to see the list of service providers matching their conditions with their first available time. Exceptions will be thrown if there is input mismatch, or if serviceProviderIds and BusinessCategoryIds entered do not match any in the database. Customer then gets to enter the selected service provider and view their associated free timings. If they enter a valid timing that the service provider is available for, the program will check if the appointment scheduling is made at least 2 hours before the actual one. This is done by comparing the years, months, date, or time, depending on what is necessary. If valid, BookingSessionBean will perform booking for customer.

- viewAppointments(): Logged in customer enters own id to view all previous and upcoming appointments.

- cancelAppointments(): Logged in customer enters appointment number to be cancelled. Only appointments that are at least 24 hours before the actual thing can be cancelled. If not, the program throws an exception.

- rateServiceProvider(): Customers will only be able to rate service providers once per appointment, and the appointment must be completed and must exist. Else, exceptions will be thrown.

2. EasyAppointmentSystemWebAdminClient

a) MainApp: For administrators to log in or exit to terminate session. Invalid login credentials will result in an error being thrown and the customer being redirected to the runApp() page.

- runApp(): Displays the list of operations for the administrator: Login and Exit (terminate the session).

- doLogin(): Administrator is required to enter email address and password.

- menuMain(): Only admins who have logged in successfully will be directed to menuMain, where he/she will be able to choose what he/she wants to do with respect to searching for, viewing appointments, approving and blocking service providers and business categories.

b) SystemAdministrationModule: For administrators to view appointments, approve and block service providers, send reminder emails and approve new business categories.

- viewCustomerAppointments(): Administrator can view the list of customer appointments after inputting a correct, valid customer id. An exception will be thrown if the customer or appointment is not found, or if an invalid input is keyed in to the system.

- viewServiceProviderAppointments(): Administrator can view the list of service provider appointments after inputting a correct, valid service provider id. An exception will be thrown if the service provider or appointment is not found, or if an invalid input is keyed in to the system.

- viewServiceProviders(): Administrator will be able to view the list of service providers along with their associated details without the need of any inputs. If there are no service providers within the system, an exception will be thrown.

- approveServiceProviders(): The service providers that are currently pending will be approved by the administrators once valid service provider id(s) have been added. Otherwise, of there are no matching service providers with the input service id or there is an input mismatch, an exception will be thrown.

- blockServiceProviders(): Administrator can input in a valid service provider id in order to block the service provider. A list of service providers will first be retrieved, before the administrator enters the input. Before deleting the service provider, the list of service provider's appointments (if any) will be deleted.

- addBusinessCategory(): The administrator can add business categories, but there will be a retrieval of all the existing business categories from the database first so that the system can ensure that the business category to be added does not already existed (i.e. there will not be a duplicate entry).

- removeBusinessCategory(): The administrator can remove the existing business categories by inputting a valid business category id. To do so, the system will first print out all the business categories with their associated ids and details, before prompting administrator for an input. The system will then attempt to delete the business category, but if it cannot be deleted, an exception will be thrown.

- sendReminderEmail(): The administrator can send a reminder email to the customer for their most upcoming appointment. This is done through making use of the compareTo methods in LocalDate and LocalTime, where the system iterates through all the appointments for the customer and will skip all those that are previous appointments (if the compareTo method returns -1). Once the first upcoming appointment is found, the iteration will terminate and the email is sent via asynchronous processing to the customer's email.

3. EasyAppointmentSystemWebServiceProviderClient

a) MainApp: For service providers to register, log in or exit to terminate session.

- runApp(): Displays the list of operations for the service provider: Register, Login and Exit (terminate the session). If the service provider logs in, he/she will be directed to the menu() on the MainMenu terminal.

- registerServiceProvider(): Service provider enters their name, business category ids (in accordance to the list of business categories and their associated business category id displayed). If the inputs are valid, the service provider will have to input their business registration number, which will be checked to ensure that no two service providers registerd have the same business registration number, followed by phone, business address, email and password. All the attributes and their associated values will then be set and persisted to the database.

b) MainMenu: For service providers to view and edit profile, view and cancel appointment(s).

- viewProfile(): A service provider can view their own profile after logging in. No input is required to view the profile.

- editProfile(): A service provider can edit their own profile's attributes (city, business address, email address, phone number and password). If there are no changes for certain attributes, service provider can leave them blank. Exceptions will be thrown if the updating is not completed successfully or if the email entered is repeated.

- viewAppointment(): A service provider can view their list of appointments without the need for any inputs.

- cancelAppointment(): A service provider can cancel their appointment as long as the appointment date is at least one day from today. This is done through making use of the compareTo methods for both LocalDate and LocalTime.