Design Document

TECH GEEKS
UNIVERSITY OF AUCKLAND

Pengyi Li - PLI552@AUCKLANDUNI.AC.NZ
Chenchen Lyu - CLYU843@AUCKLANDUNI.AC.NZ
Nancy Watta - NWAT700@AUCKLANDUNI.AC.NZ
Ke Wu - WKE918@AUCKLANDUNI.AC.NZ
Yalu You - YYOU311@AUCKLANDUNI.AC.NZ
Zhao Yuan - ZYUA826@AUCKLANDUNI.AC.NZ
Jiabin Zhong - ZJIA684@AUCKLANDUNI.AC.NZ

Table of Contents

1. Architecture Design

1.1. Activity Diagram

This section provides a functional overview of the system by activity diagrams.

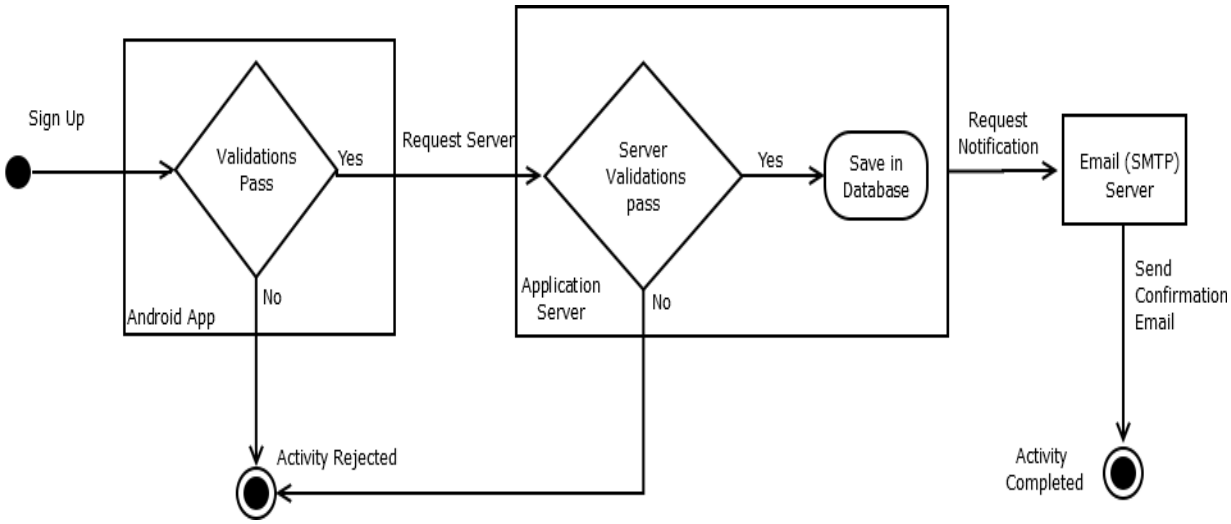


Figure 1

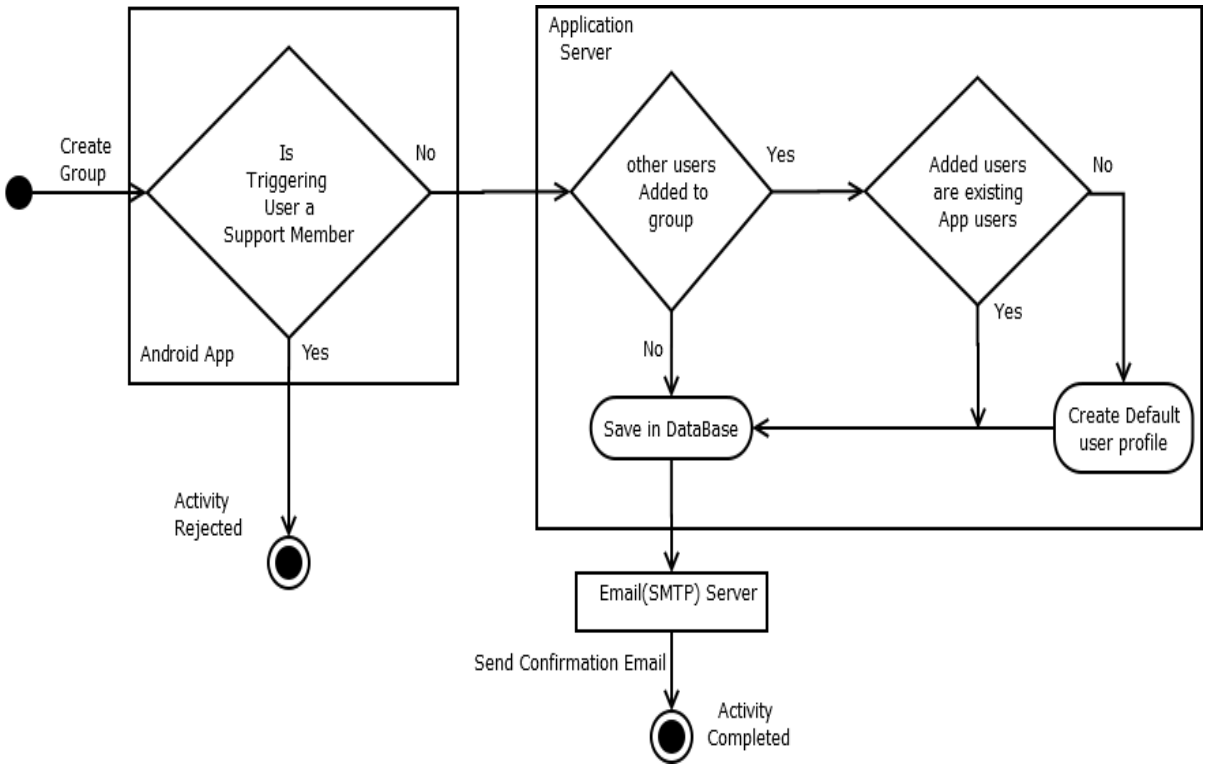


Figure 2

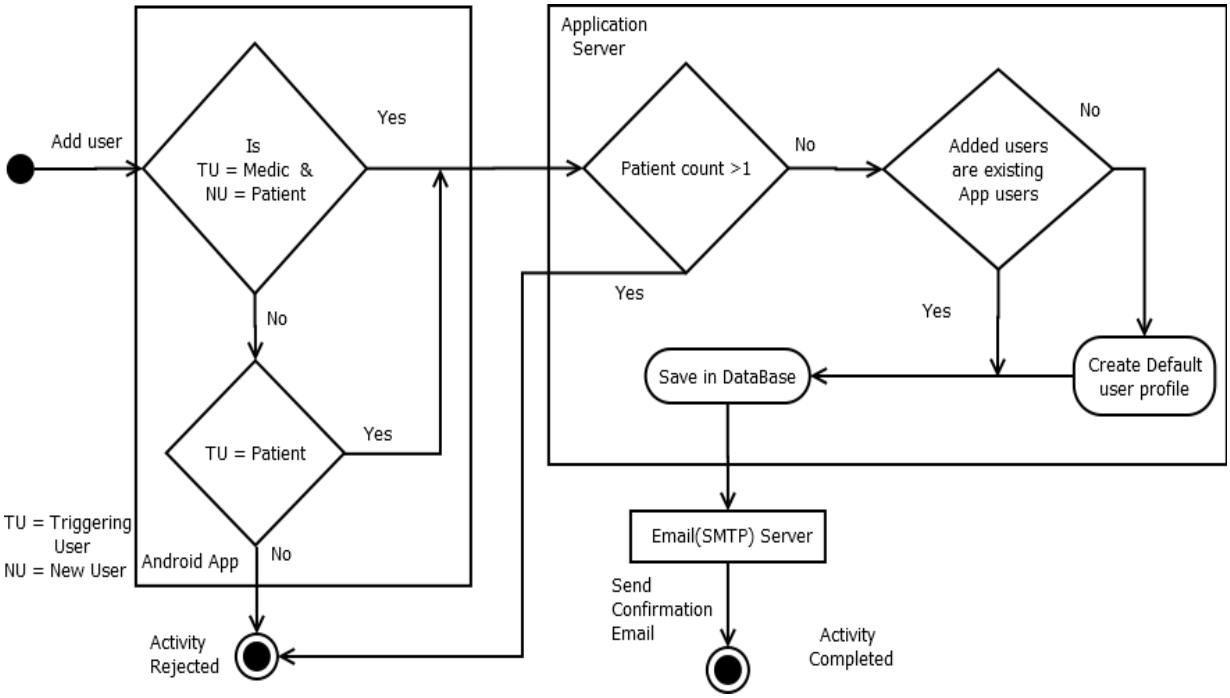


Figure 3

1.2. System Architecture

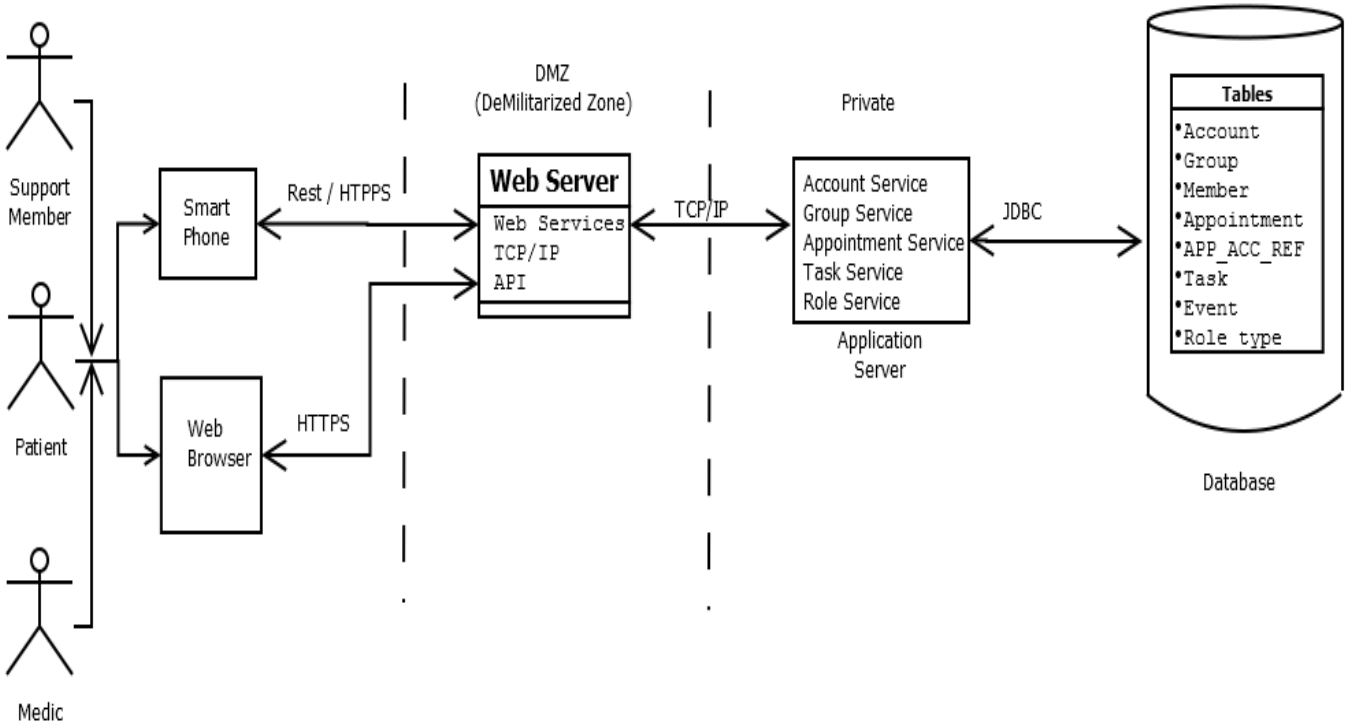


Figure 4

1.3. Logical View

The Health Connect application is divided into layers based on the N-tier architecture. The layering approach is the mostly accepted solution for enterprise applications, which require scalability, modularity and easy maintenance.

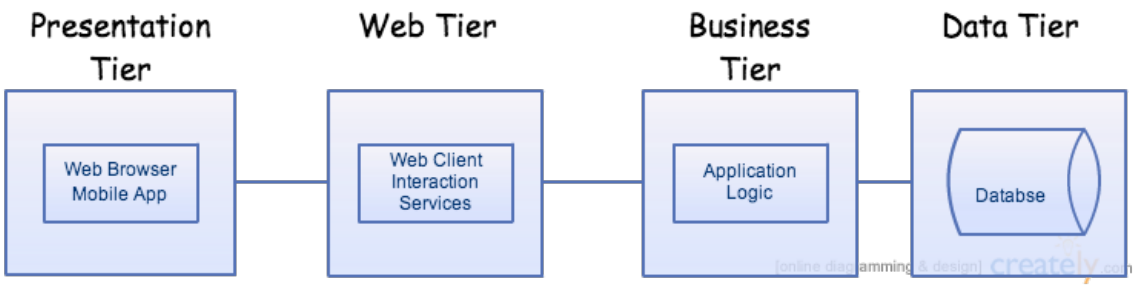


Figure 5

1.4. Database Design

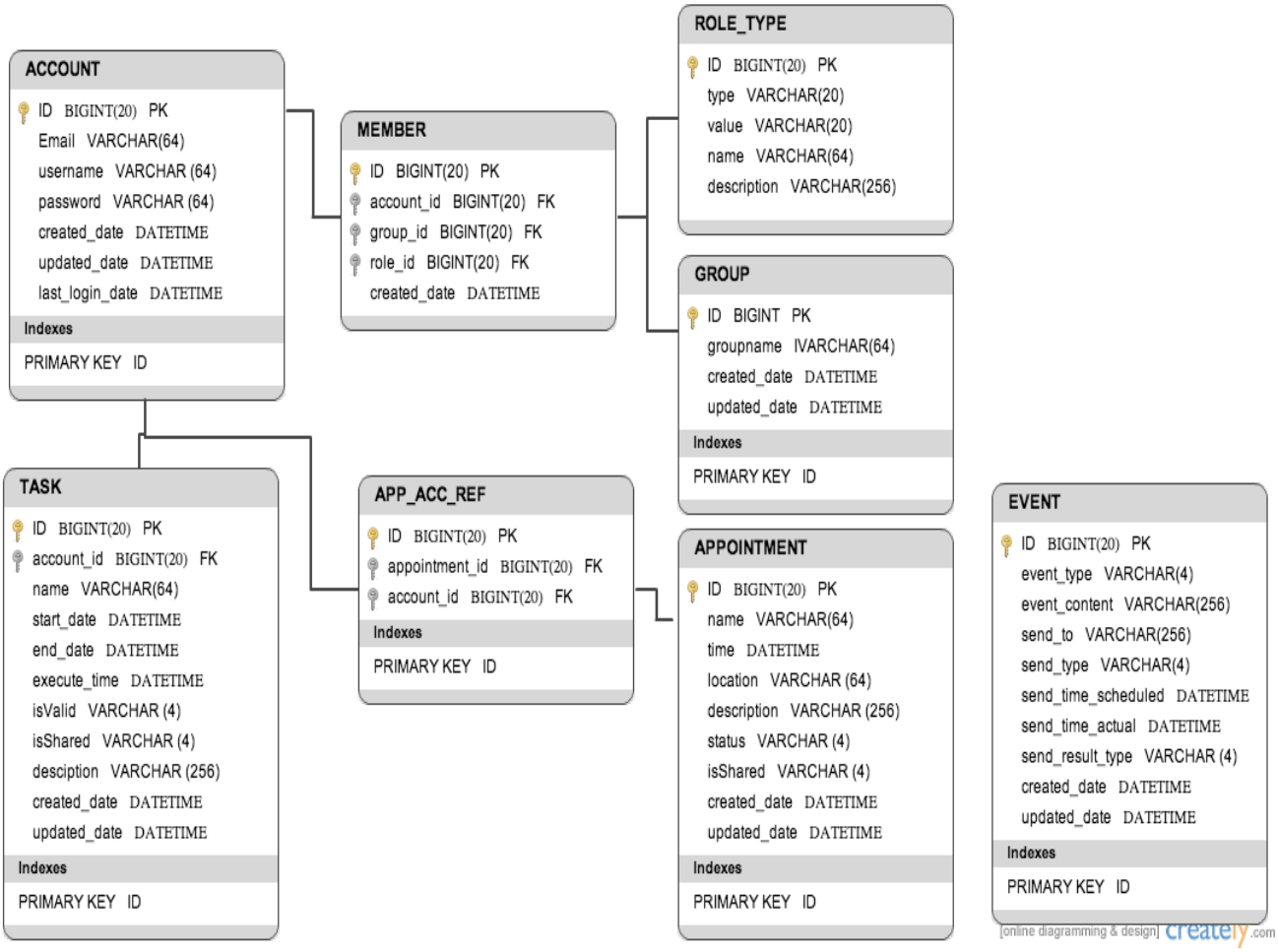


Figure 6

2. Class Diagram

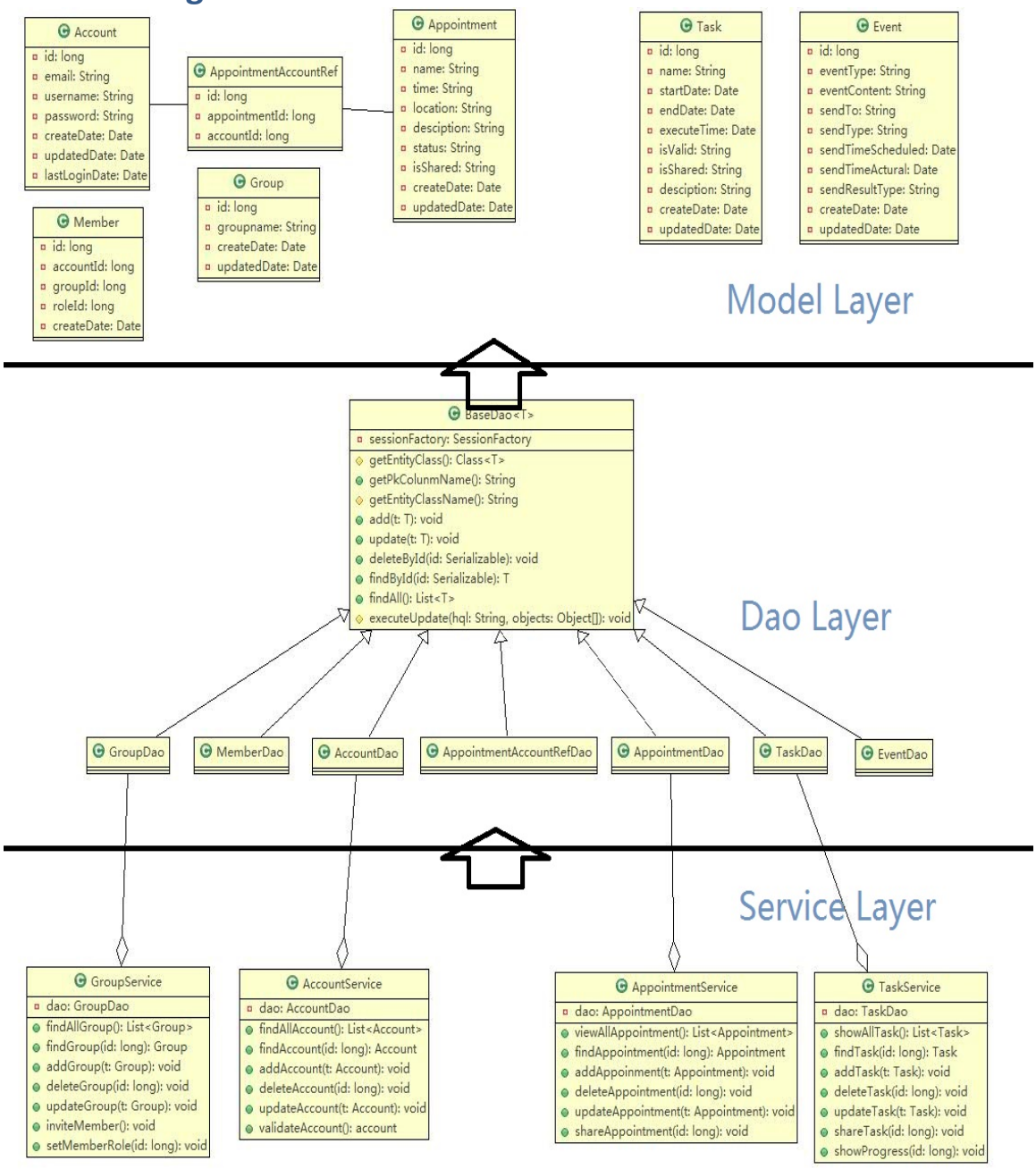
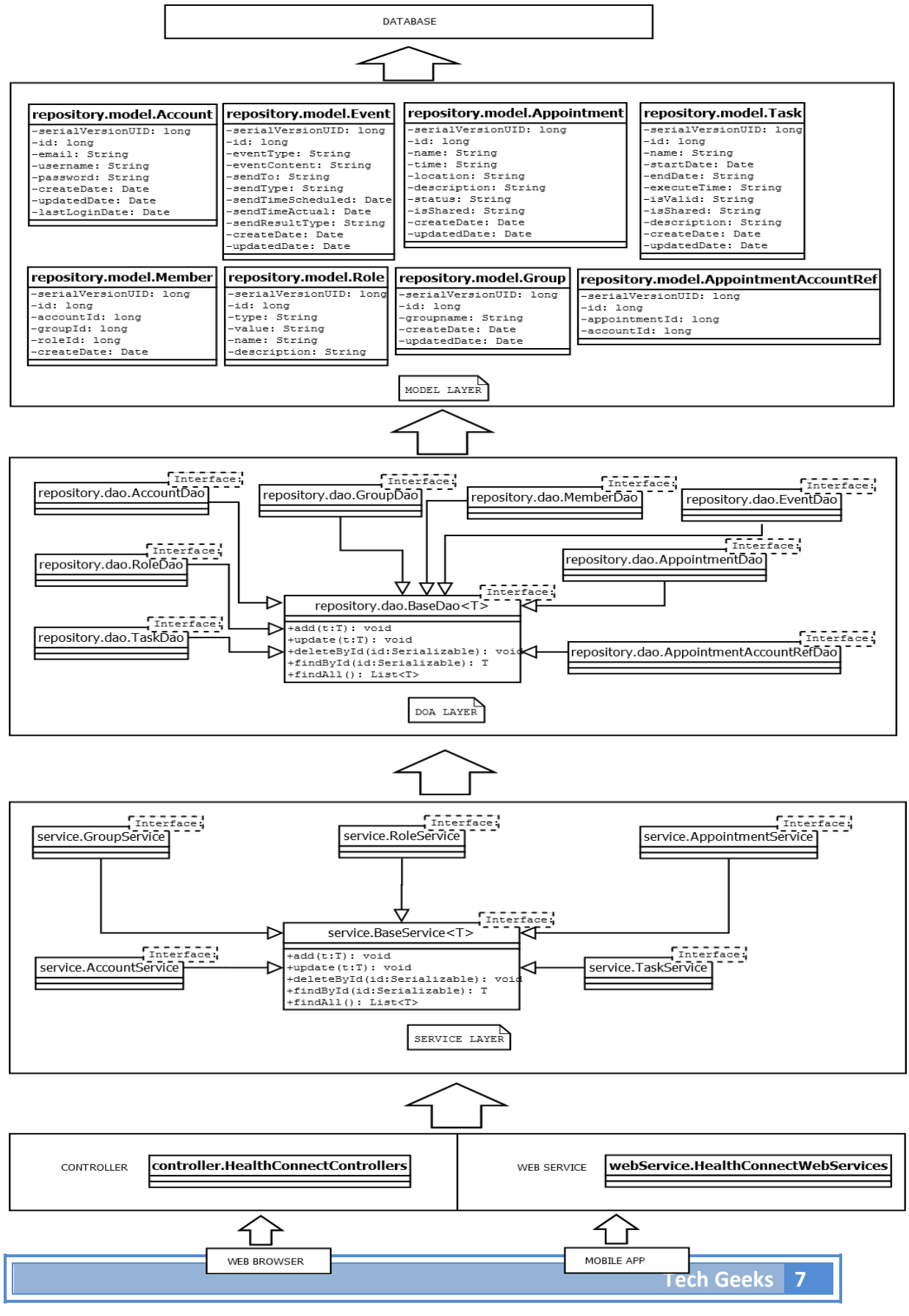


Figure 7


DATABASE
repository.model.Account
-serialVersionUID: long
-id: long
-email: String
-username: String
-password: String
-createDate: Date
-updatedDate: Date
-lastLoginDate: Date
repository.model.Event
-serialVersionUID: long
-id: long
-eventType: String
-eventContent: String
-sendTo: String
-sendType: String
-sendTimeScheduled: Date
-sendTimeActual: Date
-sendResultType: String
-createDate: Date
-updatedDate: Date
repository.model.Appointment
-serialVersionUID: long
-id: long
-name: String
-time: String
-location: String
-description: String
-status: String
-isShared: String
-createDate: Date
-updatedDate: Date
repository.model.Task
-serialVersionUID: long
-id: long
-name: String
-startDate: Date
-endDate: String
-executeTime: String
-isValid: String
-isShared: String
-description: String
-createDate: Date
-updatedDate: Date
repository.model.Member
-serialVersionUID: long
-id: long
-accountId: long
-groupId: long
-roleId: long
-createDate: Date
repository.model.Role
-serialVersionUID: long
-id: long
-type: String
-value: String
-name: String
-description: String
repository.model.Group
-serialVersionUID: long
-id: long
-groupname: String
-createDate: Date
-updatedDate: Date
repository.model.AppointmentAccountRef
-serialVersionUID: long
-id: long
-appointmentId: long
-accountId: long
MODEL LAYER
Interface
repository.dao.AccountDao
repository.dao.GroupDao
repository.dao.MemberDao
repository.dao.EventDao
repository.dao.RoleDao
repository.dao.AppointmentDao
repository.dao.BaseDao<T>
+add(t:T): void
+update(t:T): void
+deleteById(id:Serializable): void
+findById(id:Serializable): T
+findAll(): List<T>
repository.dao.TaskDao
repository.dao.AppointmentAccountRefDao
DOA LAYER
service.GroupService
service.RoleService
service.AppointmentService
service.BaseService<T>
+add(t:T): void
+update(t:T): void
+deleteById(id:Serializable): void
+findById(id:Serializable): T
+findAll(): List<T>
service.AccountService
service.TaskService
SERVICE LAYER
CONTROLLER
controller.HealthConnectControllers
WEB SERVICE
webService.HealthConnectWebServices
WEB BROWSER
MOBILE APP

3. Design Decisions

3.1. Reusability

Logic is divided into Tiers (dao, services etc.) with the intention of promoting reuse.

- In **AccountDao** class, **findById** (long id) is a method that will return specific account model matching the given id parameter. In our solution, there will be **AppointmentService**, **TaskService**, **GroupService** that also need to find a related account when creating a new appointment or inviting a member to a group. Using separate tiers can help to reuse the same method like AppointmentService.createAppointment

```
{
    …
    Account owner = accountDao.findById(id);
    …
}
```

- Interfaces, generic type and inheritance have been implemented for reusability **BaseDaoImpl**<T> implements **BaseDao**<T> so that in each actual business dao, we can just add a new class like AccountDaoImpl extends BaseDaoImpl<Account> implements AccountDao{…}

3.2. Reliability

The application provides reliability properties with respect to the delivery of data to the intended recipient(s).

- Email Notifications will be sent to the users on successful creation of a group, account or adding an appointment. In case of any failure, useful messages will be displayed to the user.
- The default setting for connection timeout is 1000 and read timeout is 5000. If it is time out, system will reset the status of these messages in Event table from "successfully send" to "failure". Then the back-up thread will scan these tasks to find out "to-send" messages and send again. If one message has been sent 10 times, the system will give up trying to send and mark this as 'giveup' status.
- J2EE application server supports load balancing through clusters.

3.3. Extensibility

Currently the application allows users to take three different roles i.e. patient, nurse and support member. But in future, the application would be extended to allow more roles such as pharmacist, GP or specialist. Thus services like Appointment, Events, Group, Account will remain the same and data flow is minimally affected.

- System has a '**RoleType**' table to store default-setting parameters, which includes the different role(s) the user holds in a group. For example, the rows in the **RoleType** table will be

"id:001; type:1; value:Nurse"
"id:002; type:1; value:Patient"
"id:003; type:1; value:Support"
'TYPE' represents this is a role. In future, if there is any requirement to add a new role, it will be handled by inserting a new row into this table e.g.
"id:004; type:1; value:Pharamacist".

3.4. Security

The HTTPS connection will be used to service requests from the mobile app and web browser.

- Business logic and data will be in server end for security.
- Http server and application server will be deployed separately to avoid application being attacked.
- Notification server will be deployed separately and set domain trust between application server and itself for security.
- J2EE native security mechanisms will be reused.

3.5. Maintainability and Scalability

The project will use a three-tier architecture model corresponding to logical layers of the application. The services created will be deployed on a server to help keep up with changes, and can also redeploy them as growth of the application's user base, data, and transaction volume increases.

- The presentation tier, or user services layer, will give user access to the application. This layer presents data to the user and optionally permits data manipulation and data entry.
- The middle tier, or business services layer, will consist of business and data rules. Because these components will not be tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require.
- The data tier, or data services layer, will interact with persistent data stored in the database.

3.6. Interoperability

- The team will focus on having Web Services, which will allow different platform applications to interact with our server. Web services will make the application platform and technology independent.
  For example, **HealthConnectWebService** offers a RESTful API :
  url: GET HealthConect/service/account/1
  Response: return accountService.findById(1);

  Hence either browser or mobile phone can post Http request and get response.
- The system is fully J2EE compliant and thus can be deployed onto any J2EE application server.

3.7. Object oriented Design

- The application will use hibernate which is an open source Java persistence

framework project instead of manually creating traditional JDBC connections. This enables having persistent data; cache data in model layer so as to reduce the bottleneck happening in DB and enhance the performance of retrieving data. It provides an object-oriented way for programmers to manipulate tables and data as objects.
For example, system has **BaseDao** and **SessionFactory** to access database and 'models' to represent actual tables. Using these objects, database operations will be performed.

- Also will be using spring which is JEE open source framework instead of create web application from scratch thus coping up with many pitfalls/changes that are common in web development. For example,
  - ***Annotation***: we use annotation to automatically distinguish our classes like "@Repository, @Entity, @Service and @Controller" and simplify our configuration.
  - ***IoC***: we use spring container to manage and inject dependent classes.
  - ***Aop***: we use Proxy and AspectJ to implement user authority check and write log.

3.8. Data storage for future reference

Server-side processing is used to interact with permanent storage like databases or files. For example, create account, create group, invite user, set appointment etc. will call server-side methods to save data on the server. This will help to manage and even analyze unified data. It is also useful in cases when users change or even lose their devices.

3.9. Interface Segregation Principle (ISP)

We have designed many contracts (interfaces) and created classes implementing these interfaces. This will provide a loose connection between concrete classes and keeps the system decoupled. Interfaces provide layers of abstraction that facilitate conceptual explanation of the code and create a barrier preventing dependencies. For example, we have interfaces called GroupService, TaskService, and Appointment Service. So the classes can implement these interfaces to know only about the methods that are of interest to them.

4. Low Fi Prototypes

SE761-TechGeeks-Prototype.pdf is an interactive PDF document to demonstrate how the application will run when live. All the UI screens designed have been labeled with the user stories numbered as per the priority mentioned below.

Priority	Item	Description
1	As a user, I want to create an account, So that I can use the application.	The users need to register an account to use the application.
2	As a nurse, I want to create a group,	A group contains only one patient and multiple nurses

	So that I can manage my patients.	and support members. Only patient and nurse can create groups. The group is used to share appointments among the members.
3	As a patient, I want to create a group, So that I can invite support people/nurse to view my calendar or actions.	
4	As a nurse, I want to invite my patient to the group, So that I can help my patient manage his time.	If the account being invited does not exist in the database, this invite action will automatically create an account; the person who has been invited can use this account to login the application.
5	As a patient, I want to invite nurse/family member, So that I can get the support from the group.	
6	As a nurse, I want to create a calendar event, So that I can help my patient remind of the appointment.	When a nurse creates a calendar event, the patient in the same group will receive a notification and view the appointment.
7	As a patient, I want to create a calendar event, So that I can manage my time efficiently.	When a patient creates a calendar event, he/she can choose the members in the group whom he/she wants to share with.
8	As a nurse, I want to view all my appointments, So that I can manage my schedules.	This is a function like normal Google calendar, users can view their timetable by day, week or month.
9	As a patient, I want to view all my appointments, So that I can manage my time efficiently.	
10	As a nurse, I want to filter my clinical appointments, So that I can have a better view.	Calendar events can have different types. When the event is created, it is assigned with a type; nurse and patient can choose to view only clinical events.
11	As a patient, I want to filter my clinical appointments, So that I can have a manageable view.	
12	As a patient, I want to share my calendar with specific people, So that I can have privacy related to my events.	The patient will have some privacy related to whom he/she wants to share the appointments with.
13	As a support member, I want to view my friend appointments, So that I can provide them health support.	The support member should be able to see what is going on with the patient

		in the group.
14	As a nurse, I want to get reminder of my patient appointments, So that I can provide them the health support.	Nurse, patient, as well as support member can get notification when certain important event is going to happen. In this case they won't forget their appointment and can provide support and be connected with each other.
15	As a patient, I want to get reminder of my appointment and action event, So that I do not forget.	
16	As a support member, I want to get reminder of the friend's appointments and action events, So that I can track their progress.	
17	As a nurse, I want to send a message to the patient, So that I can communicate more effectively	It is better to provide a direct communication with the app. The preferred means of communication will be chosen by the patient i.e. email, message or fax.
18	As a patient, I want to create action events, So that I can plan my day.	Action even means the long-term goal that the patient wants to achieve, such as "loose 10kg". This can be added to their calendar. Other people should be able to see the action event, to understand the daily progress of the patient.
19	As a patient, I want to share my action events, So that I can help others benefit or know my progress.	