Design Document for Semedicure

Group DO_04

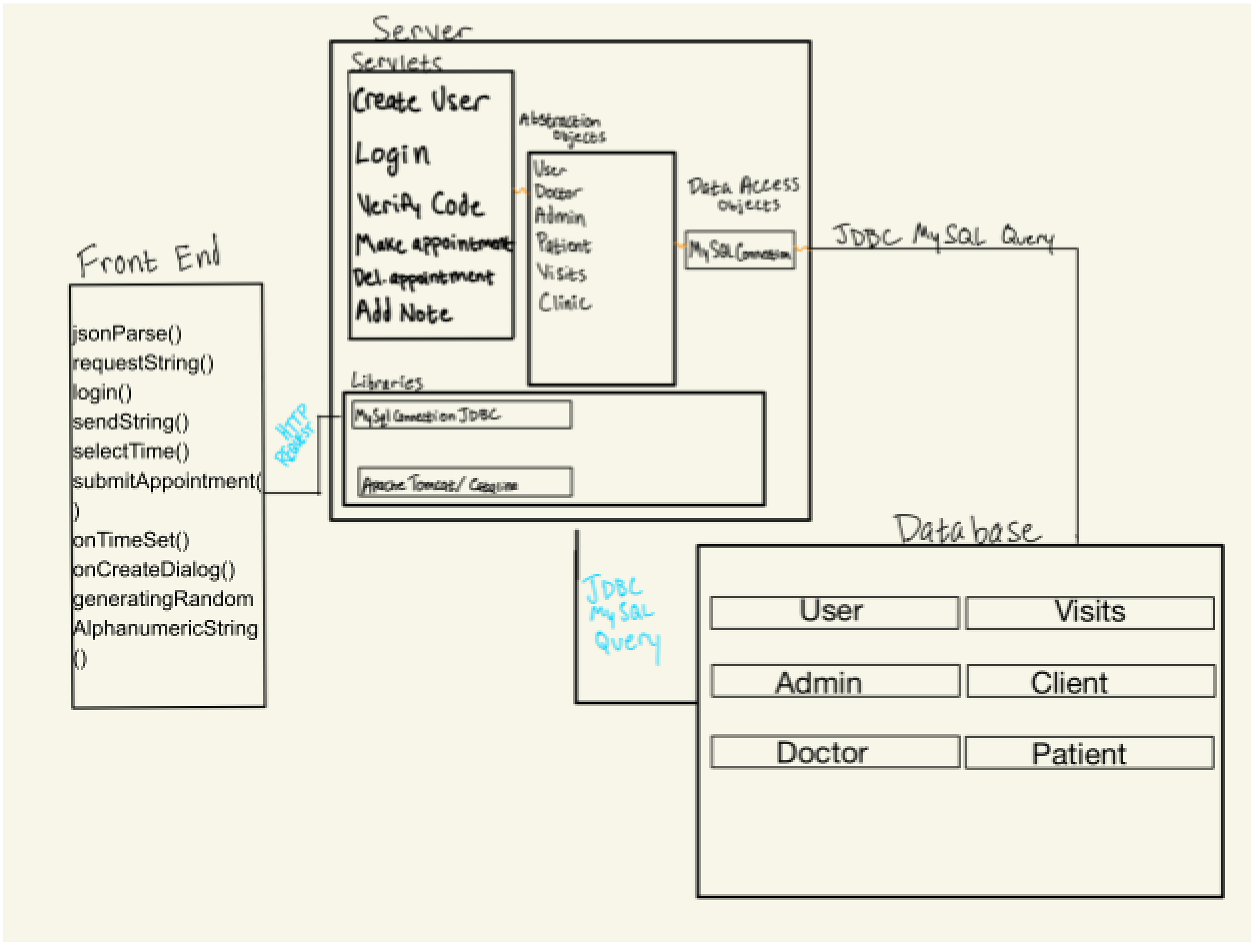
Josue Torres: 25% contribution

Phuoc (Johnny) Nguyen: 25% contribution

Ryan Cambell: 25% contribution

Will Postler: 25% contribution

BLOCK DIAGRAM



Overview of Application

Semedicure is a data app that allows the user to access and store medical information.

Android (Front-End)

For our application, all of the interactions between the frontend and backend follow a sequence of steps. First, the frontend makes an HTTP request to the backend using the volley library. Second, if the frontend needs to pass data to the backend, it will package this data into a hashmap and send it. Finally, the backend sends a response to the frontend, as either a string or json object. Below are the functions which causes this interaction, as well as a detailed description:

1. jsonParse()
   When a response is received, this function parses through the data in order to make it usable for our application. The data is stored in variables which can then be used for whatever it is needed for.

2. requestString()
   Sends an HTTP request and expects a string in response. Also sends data to the backend via a hashmap if the interaction calls for it. Performs a specific action when a response is received, i.e. log the user in for correct credentials or reject the login if password is incorrect.

3. login()
   Sends a request to the server, and packages email and password in a hashmap for the backend to interpret. Logs the user in and prompts them to the correct screen depending on the type of user, if given correct credentials. Rejects login otherwise.

4. sendString()
   Called when sending a request to the backend. Sends data in a string format to then be interpreted by the backend.

5. selectTime()
   Generates a time dialog when scheduling an appointment that allows the user to select what time they would like their appointment. Saves the minute and the hour in a variable to then be packaged and sent to the backend.

6. submitAppointment()
   Send the appointment date to the backend. This consists of the time, day, and year the user would like to set an appointment.

7. onTimeSet()
   Used in conjunction with selectTime(). When the time is selected and the dialog closes, a request is sent to the server, and the variable data generated from selectTime() are sent.

8. onCreateDialog()
   An abstract version of a dialog. Builds the dialog to be used on screen, and saves the data requested from the user through the dialog in variables. These variables can then be sent to the backend.

9. generatingRandomAlphanumericString()
   Generates a randomized alphanumeric token to be used to initiate the transfer of patient data. When called, this function provides a random string that is then sent to the backend. This string is then compared with the string provided by the doctor when requesting patient data. If they match, the backend will transfer the patient data to the frontend.

<u>Server (Back-End)</u>

Due to the nature of Semedicure, all of the data being passed via an HTTP request from the front-end to the server are JSON packets. Below are more detailed descriptions of the interactions that the back-end will receive and from there any sub-interactions that take place to ensure the right data is being retrieved.

1. Registering a new user:
   The backend will receive data from the front end identifying the type of user they are registering (Patient, Admin, Provider) and then send the data to the correct table within the MySQL database to store the data.

2. Logging a user into their account:
   The backend will receive the login data from an existing user, request that same data from the MySQL database, and compare it to verify it is the correct login information. Once access is granted they will be directed to the appropriate page.

3. Uploading information to a new office:

   When a patient wishes to upload their medical information to a new office. They will be given a temporary alphanumeric code to give to an administrator at the new office. Once the administrator enters the patient's code the on their profile data is sent to the server. Upon verification that the code is correct, the administration and providers at that clinic will be able to access that patient's information.

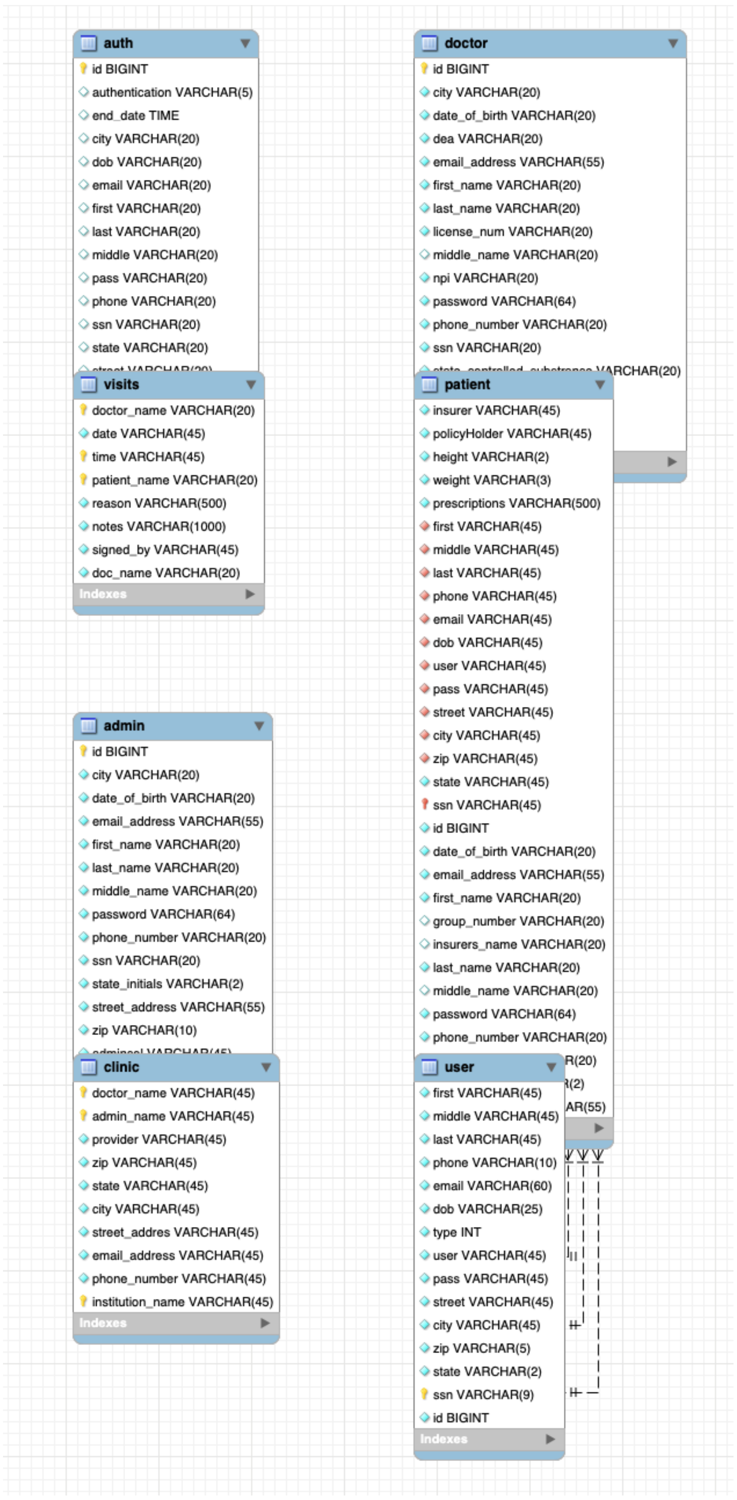
4. Requesting an appointment with a Provider:

   When a patient wishes to request an appointment with a provider the data requesting the appointment will be sent to the server. The server will store the pending appointment in the patient's database and send a notification to the administrator of the office asking them to accept or decline the request. At which point the data will go back through the server and notify the patient of the response.

5. Adding medical notes to the Patient’s chart:

   When a provider updates a patient's chart by adding notes from the appointment that information is sent to the server which will then direct these notes to the database where the information is stored.

TABLE RELATIONSHIPS DIAGRAM


auth
id BIGINT
authentication VARCHAR(5)
end_date TIME
city VARCHAR(20)
dob VARCHAR(20)
email VARCHAR(20)
first VARCHAR(20)
last VARCHAR(20)
middle VARCHAR(20)
pass VARCHAR(20)
phone VARCHAR(20)
ssn VARCHAR(20)
state VARCHAR(20)
doctor
id BIGINT
city VARCHAR(20)
date_of_birth VARCHAR(20)
dea VARCHAR(20)
email_address VARCHAR(55)
first_name VARCHAR(20)
last_name VARCHAR(20)
license_num VARCHAR(20)
middle_name VARCHAR(20)
npi VARCHAR(20)
password VARCHAR(64)
phone_number VARCHAR(20)
ssn VARCHAR(20)
VARCHAR(20)
visits
doctor_name VARCHAR(20)
date VARCHAR(45)
time VARCHAR(45)
patient_name VARCHAR(20)
reason VARCHAR(500)
notes VARCHAR(1000)
signed_by VARCHAR(45)
doc_name VARCHAR(20)
Indexes
patient
insurer VARCHAR(45)
policyHolder VARCHAR(45)
height VARCHAR(2)
weight VARCHAR(3)
prescriptions VARCHAR(500)
first VARCHAR(45)
middle VARCHAR(45)
last VARCHAR(45)
phone VARCHAR(45)
email VARCHAR(45)
dob VARCHAR(45)
user VARCHAR(45)
pass VARCHAR(45)
street VARCHAR(45)
city VARCHAR(45)
zip VARCHAR(45)
state VARCHAR(45)
ssn VARCHAR(45)
id BIGINT
date_of_birth VARCHAR(20)
email_address VARCHAR(55)
first_name VARCHAR(20)
group_number VARCHAR(20)
insurers_name VARCHAR(20)
last_name VARCHAR(20)
middle_name VARCHAR(20)
password VARCHAR(64)
phone_number VARCHAR(20)
R(20)
R(2)
AR(55)
admin
id BIGINT
city VARCHAR(20)
date_of_birth VARCHAR(20)
email_address VARCHAR(55)
first_name VARCHAR(20)
last_name VARCHAR(20)
middle_name VARCHAR(20)
password VARCHAR(64)
phone_number VARCHAR(20)
ssn VARCHAR(20)
state_initials VARCHAR(2)
street_address VARCHAR(55)
zip VARCHAR(10)
clinic
doctor_name VARCHAR(45)
admin_name VARCHAR(45)
provider VARCHAR(45)
zip VARCHAR(45)
state VARCHAR(45)
city VARCHAR(45)
street_addres VARCHAR(45)
email_address VARCHAR(45)
phone_number VARCHAR(45)
institution_name VARCHAR(45)
Indexes
user
first VARCHAR(45)
middle VARCHAR(45)
last VARCHAR(45)
phone VARCHAR(10)
email VARCHAR(60)
dob VARCHAR(25)
type INT
user VARCHAR(45)
pass VARCHAR(45)
street VARCHAR(45)
city VARCHAR(45)
zip VARCHAR(5)
state VARCHAR(2)
ssn VARCHAR(9)
id BIGINT
Indexes