

Sacchon Team 3



Members:

Ana Papa
Athanasios Liakos
Chrysa Tsivintzeli
Thomas Tegou



Getting the work done

- Ana Papa:
 - BSc Applied Computer Science, University of Macedonia
 - MSc Artificial Intelligence, University of Utrecht
- For this project:
 - Angular
 - TypeScript
 - Visual Studio Code
 - Postman



Getting the work done

- Chrysa Tsivintzeli:
 - Computer Science, Ionian University
- For this project:
 - Angular
 - TypeScript
 - Visual Studio Code
 - Postman




Getting the work done

- [illegible]



Getting the work done

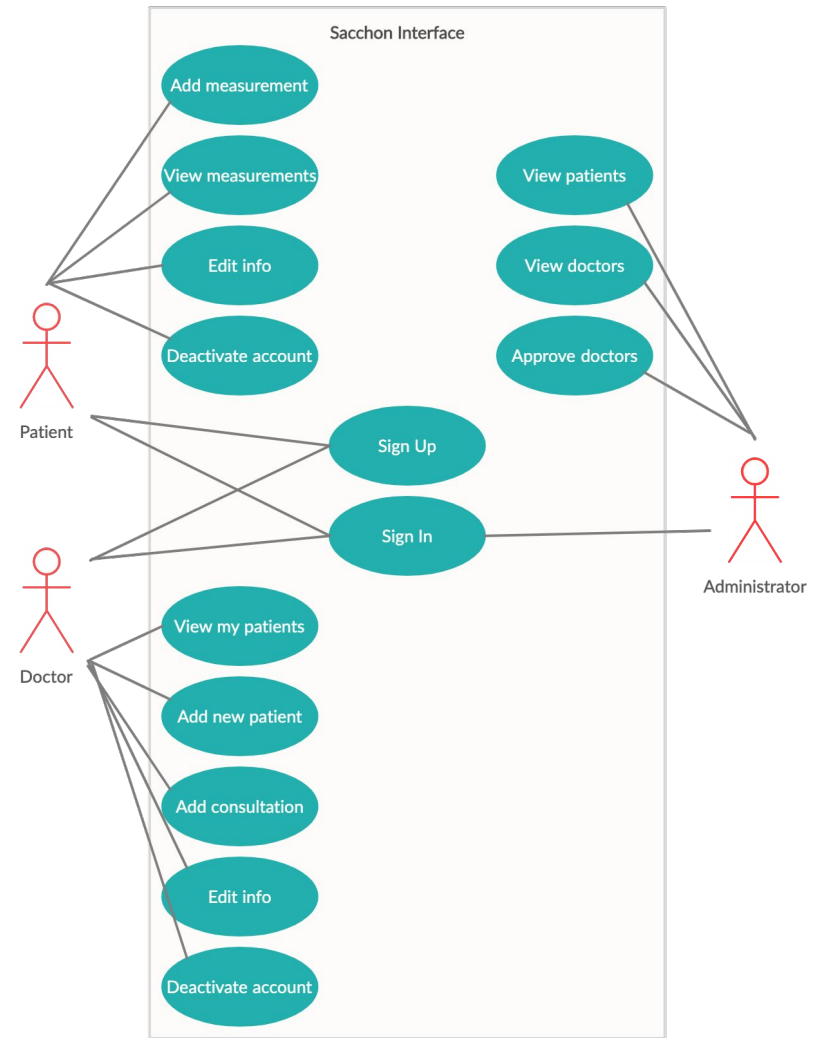
- Athanasios Liakos:
 - Electrical & Computer Engineering, University of Thessaly
 - For this project:
 - Java
 - IntelliJ
 - Hibernate
 - Postman
 - MSSQL Server
- 



User Actions

Administrator:

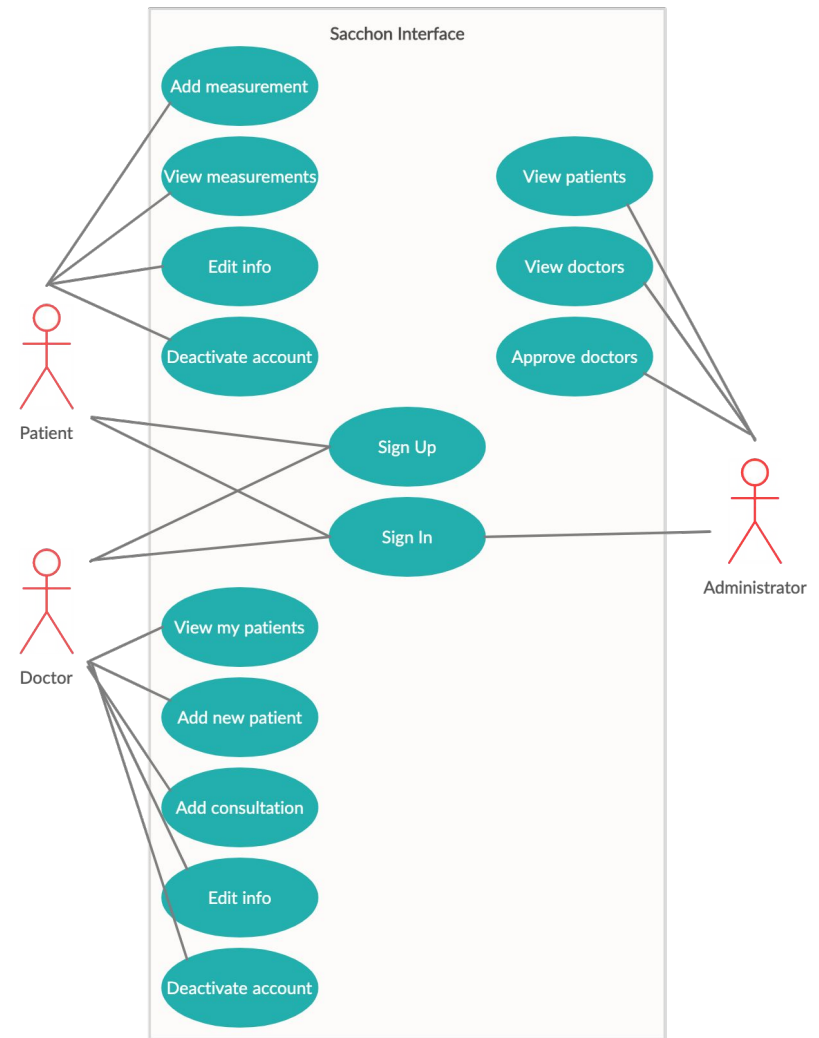
- Sign in
- View patients
- View doctors
- Approve doctors



User Actions

Patient:

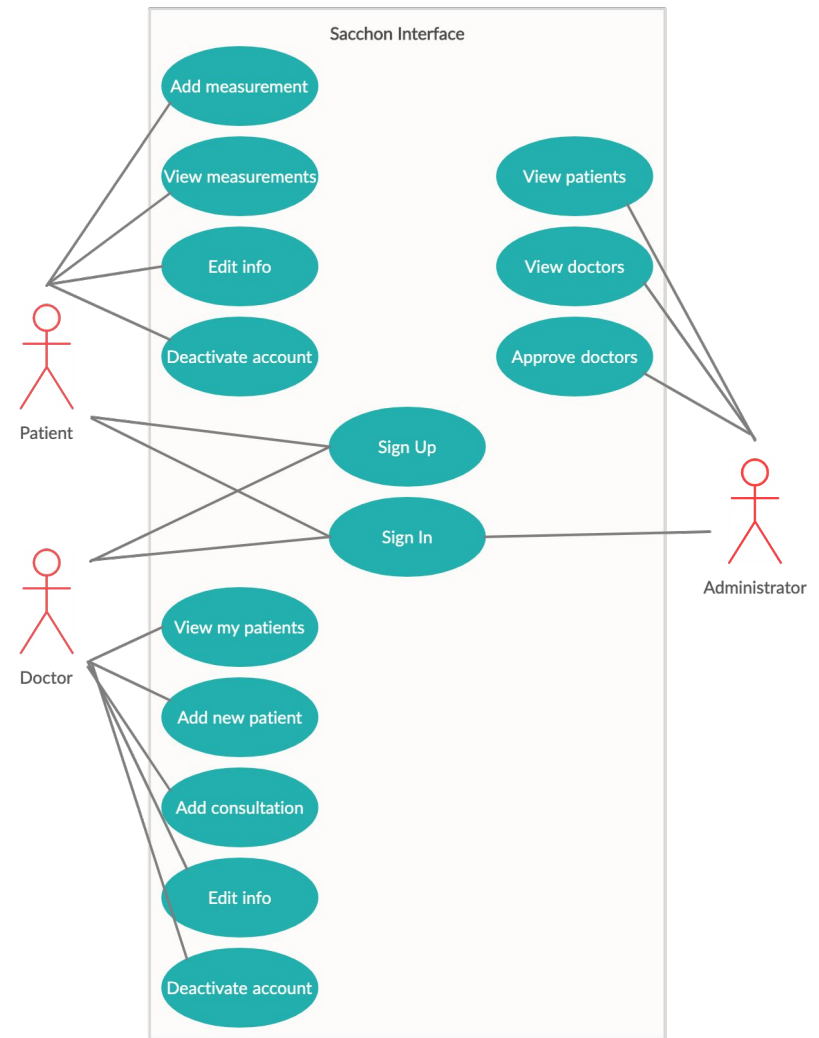
- Sign up
- Sign in
- Add measurement
- View measurements & averages
- View consultations
- Edit info
- Deactivate account

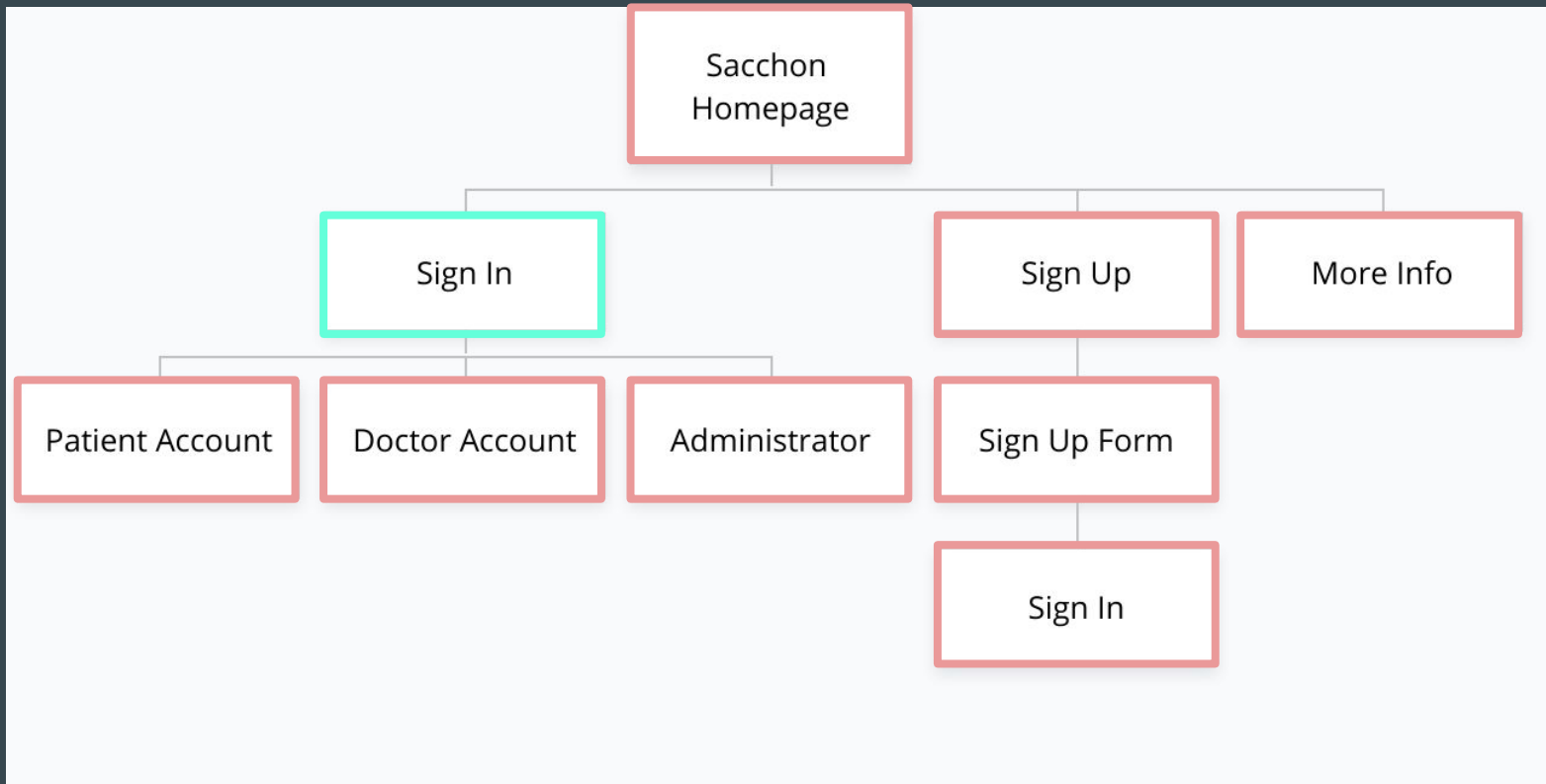


User Actions

Doctor:

- Sign up (waiting for approval)
- Sign in
- View their patients
- Add new patient
- Add consultation
- Edit info
- Deactivate account





Patient Account



```
graph TD; PA[Patient Account] --> AM[Add measurement]; PA --> VM[View measurement]; PA --> VC[View consultations]; PA --> EI[Edit info]; PA --> DA[Deactivate account]; AM --> AMF[Add measurement form]; EI --> EIF[Edit info form];
```

Add
measurement

View
measurement

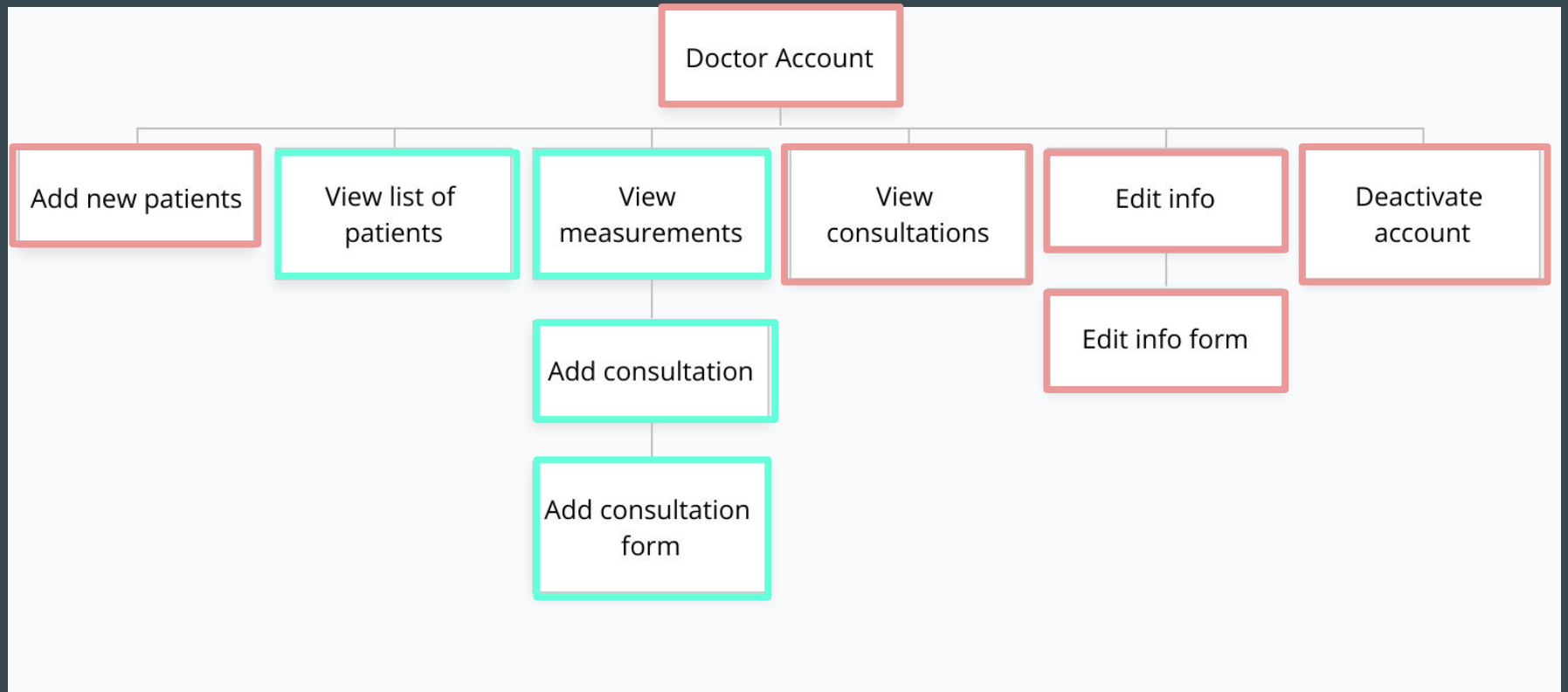
View
consultations

Edit info

Deactivate
account

Add
measurement
form

Edit info form



Administrator

```
graph TD; Admin[Administrator] --- ViewPatients[View patients]; Admin --- ViewDoctors[View doctors]; Admin --- ApproveDoctors[Approve doctors];
```

View patients

View doctors

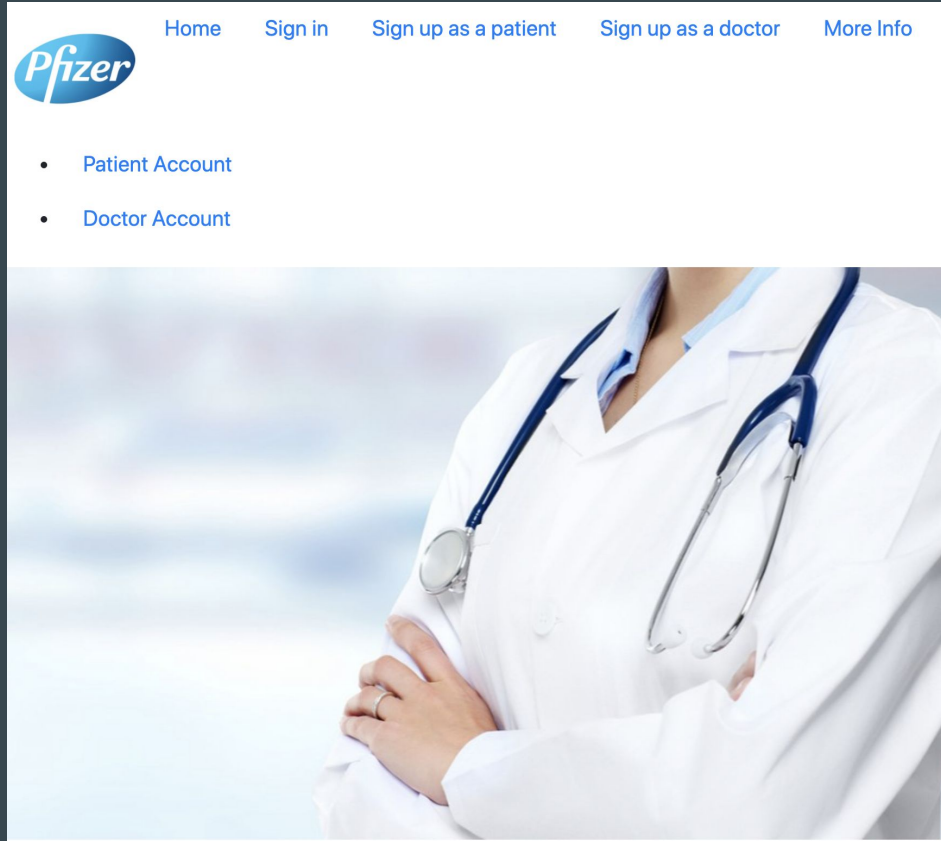
Approve doctors

Development Flow

- Step 1: Listing all the parts of the app
- Step 2: Drawing state & action diagrams
- Step 3: Defining API actions
- Step 4: Writing some of the code
- Step 5: Developing the back-end architecture
- Step 6: Writing some more of the code
- Step 7: Designing and developing the front-end architecture



Live demo

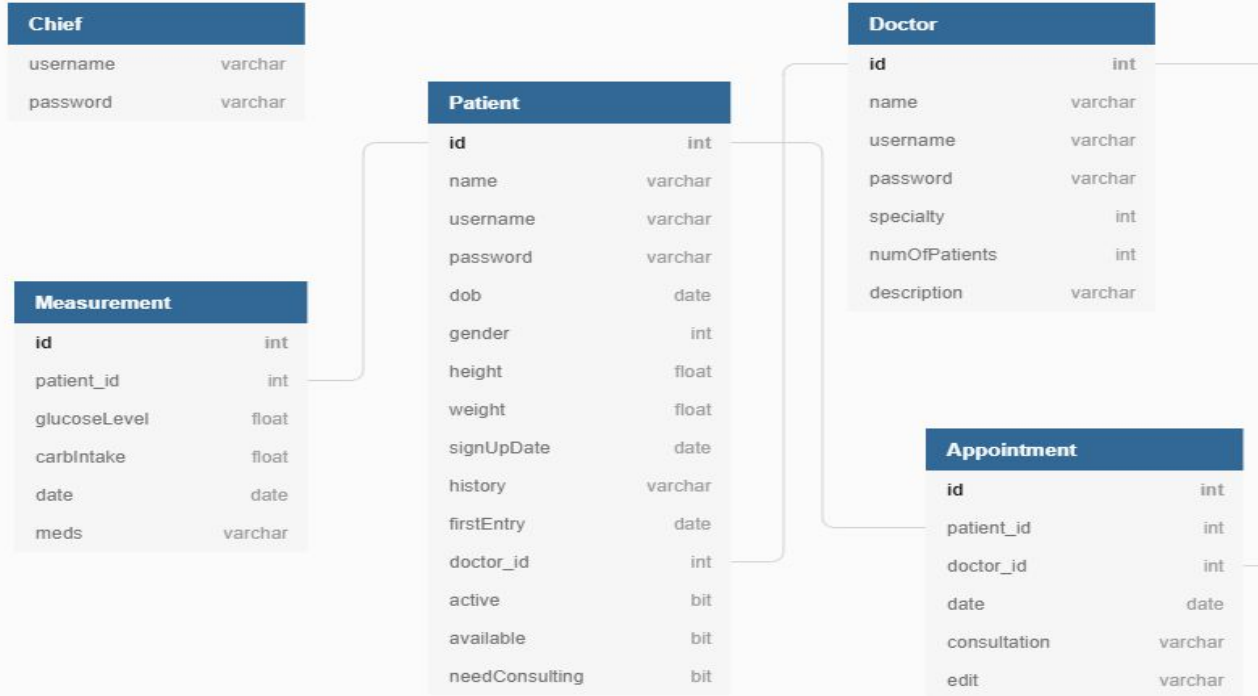


Getting the work done

- Database design
- API actions



Database Architecture



Patient's Actions (Backend)



- Patient can add/view/edit his/her personal data

URI post/get/put implementation:
`/app/patient/{patientID}`



- Patient can deactivate his/her account

URI put implementation:
`/patient/softRemove/{patientID}`

- This action sets his/her active field false
- Certain patient is removed from assigned doctor

```
{  
  "name": "Dimitris",  
  "username": "dimi28",  
  "password": "dimt2345@",  
  "dob": "1987-01-10",  
  "height": 1.89,  
  "weight": 87.0,  
  "gender": "MALE",  
  "history": "test history",  
  "active": true,  
  "doctorID": 1  
}
```

Patient's Actions (Backend)

- Patient enters daily measurement for Carbohydrates consumption and glucose level

URI post : app/measurement

- Patient can update a specific measurement

URI put : app/measurement

```
{  
  "glucoseLevel": 110.0,  
  "carbIntake": 566.0,  
  "meds": "test med0",  
  "date": "2020-10-03",  
  "patientID": 1,  
  "measurementID": 1  
}
```

- Patient can view past measurements for specific period or all of them by id or username

URI GET (*examples*)

/app/measurement/?patientUsername=dimi28&fromDate=2020-10-10&toDate=2020-10-11

/app/measurement/?patientUsername=dimi28&fromDate=2020-10-10

/app/measurement/?patientUsername=dimi28&fromDate=2020-10-10&toDate=2020-10-12

/app/measurement/?patientID=1&fromDate=2020-10-19&toDate=2020-10-23

Patient's Actions (Backend)



- Patient can view average glucose level and consumed carbohydrates over a certain period

URI GET implementation: `app/measurement/average` (*examples*)

`/app/measurement/average/?patientID=1&fromDate=2020-10-03&toDate=2020-10-04`



- Patient can view his/her appointments and doctor's consultations by patient ID

URI GET implementation: `/app/appointment` (*examples*)

`/app/appointment/?patientID=2`

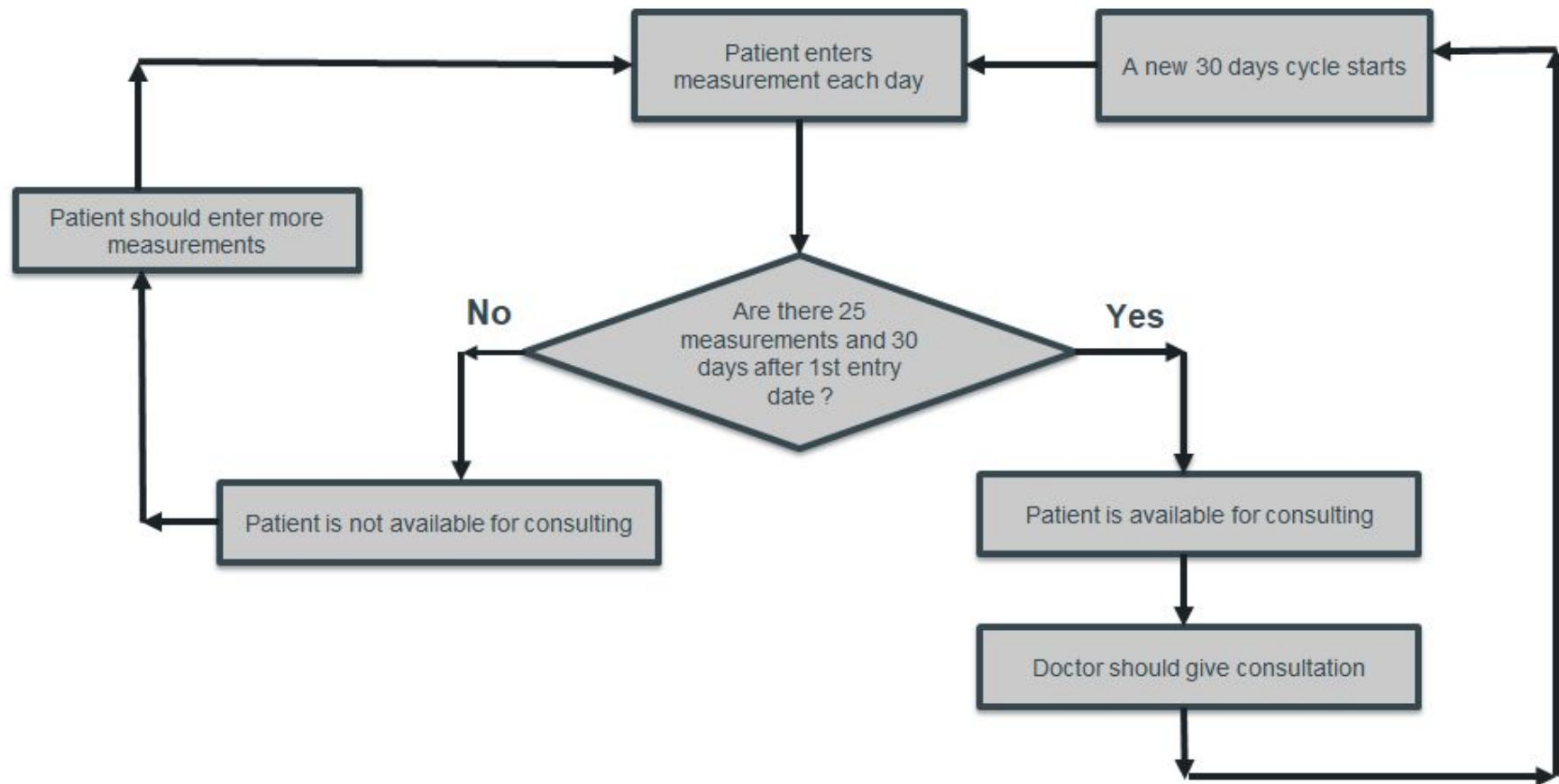
`/app/appointment/?patientID=1&doctorID=1`

```
{  
  "date": "2020-10-18",  
  "consultation": "Consultation",  
  "edit": "Edit test ",  
  "doctorID": 1,  
  "patientUsername": "Username",  
}
```

Average measurement json

```
{  
  "glucoseLevel": 87.6666667,  
  "carbIntake": 175.333333334  
}
```

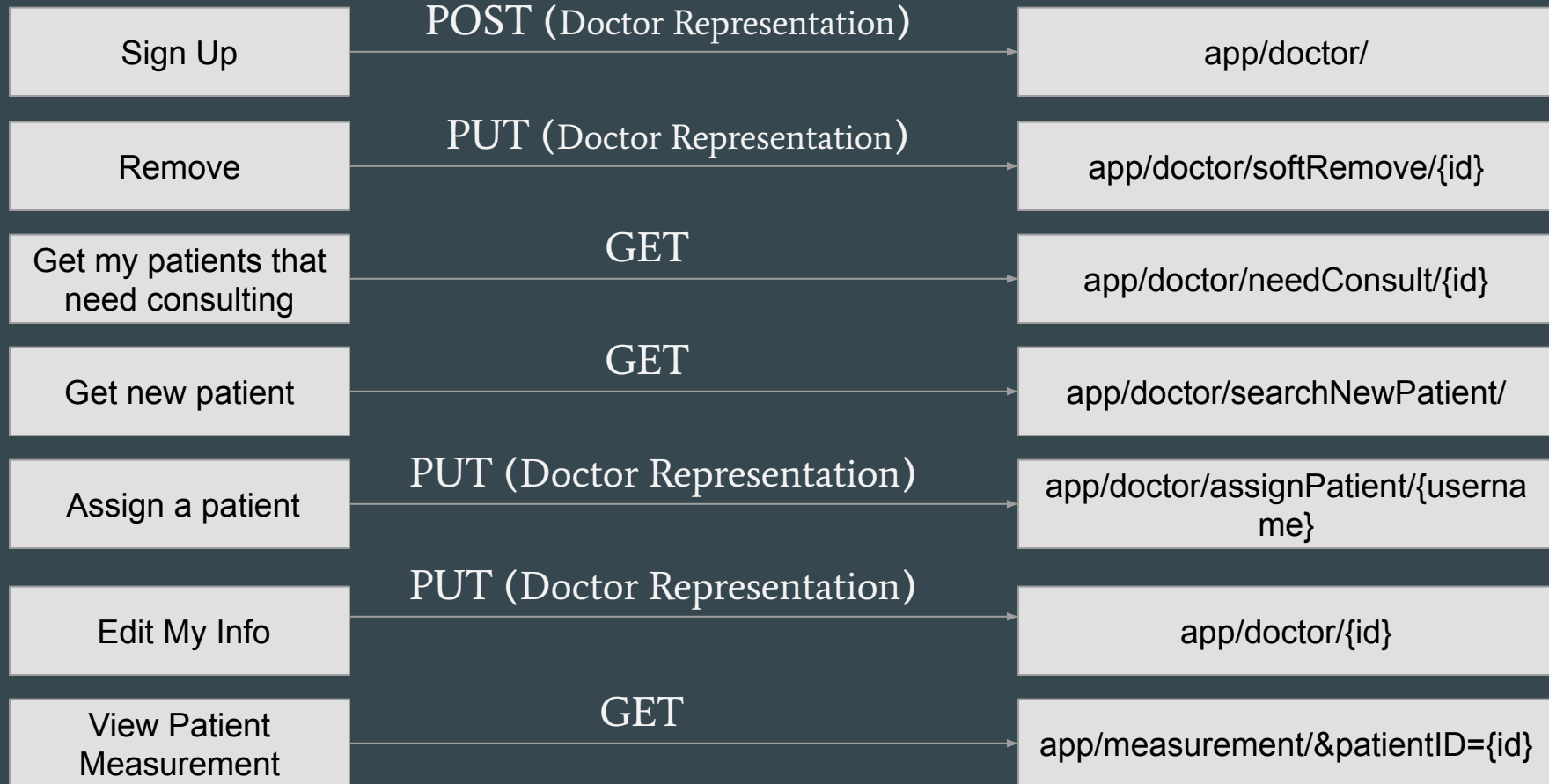
Patient's availability update for doctor's consultation



Doctor Rest API

Actions:

End Points:



Chief Rest API

Actions:

End Points:

View All Patients

GET

app/patient/

View Doctors

GET

app/doctor/

Approve Doctor

PUT (Doctor Representation)

app/doctor/approveDoctor/{username}

Patient Representation

```
{
  "name": "David",
  "username": "dav1d",
  "password": "1234!@#$$",
  "dob": "1987-01-10",
  "height": 1.89,
  "weight": 87.0,
  "gender": "MALE",
  "history": "Sleep Paralysis",
  "active": true,
  "doctorID": 0,
  "uri":
"http://localhost:9000/app/patient/5"
}
```

Doctor Representation

```
{
  "name": "Eva",
  "username": "eva1989",
  "password": "eva1989!eva",
  "specialty": "CARDIOLOGY",
  "numOfPatients": 2,
  "description": "Specialized in ...",
  "active": true,
  "uri":
"http://localhost:9000/app/doctor/1"
}
```

Measurement Representation

```
{
  "glucoseLevel": 35.0,
  "carbIntake": 157.0,
  "meds": "test med0",
  "date": "2020-08-25",
  "patientID": 1,
  "measurementID": 1,
  "uri":
"http://localhost:9000/app/measurements/1" ,
}
```

Appointment Representation

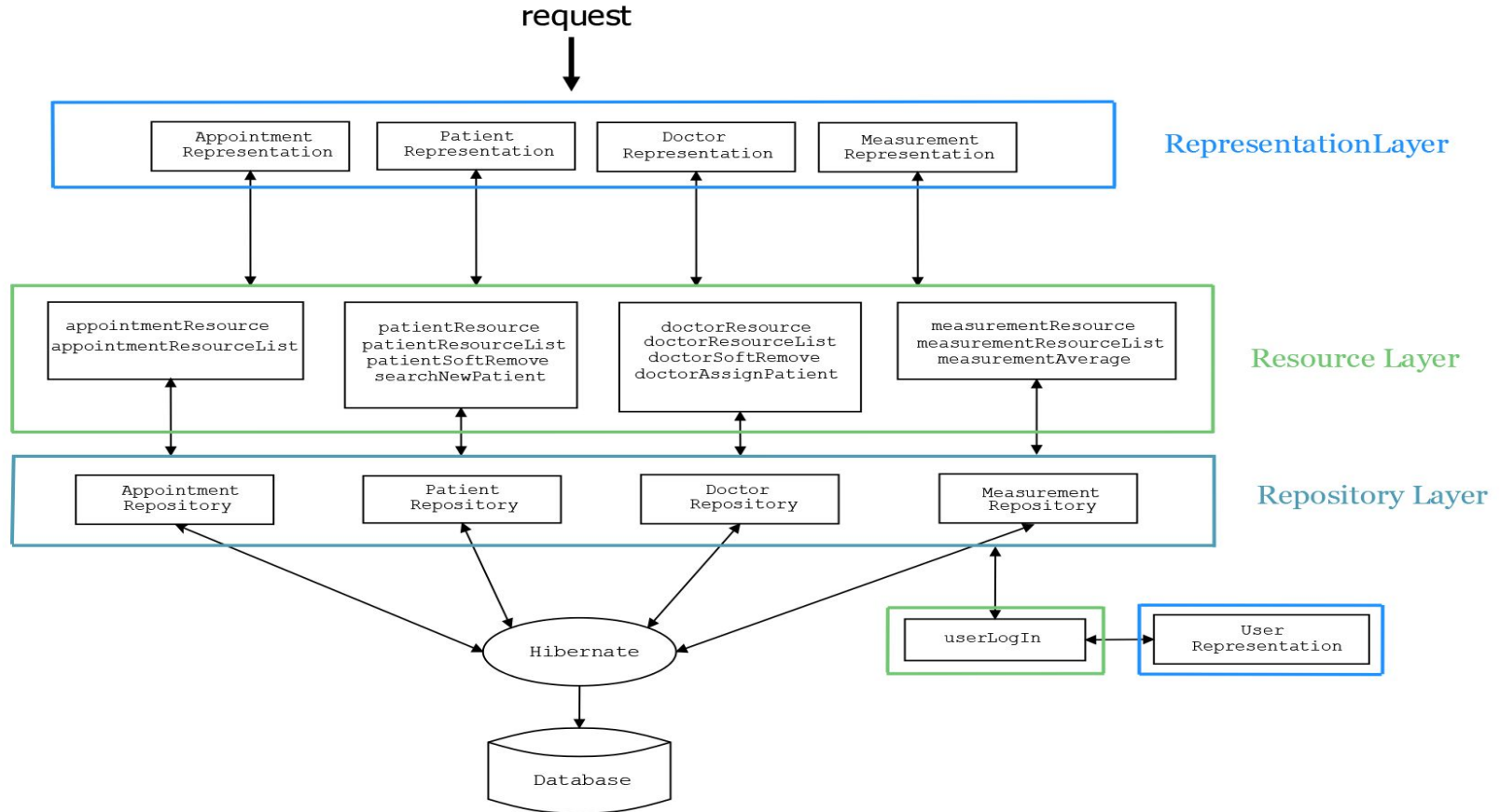
```
{
  "date": "2020-09-28",
  "consultation": "Nothing to mention.",
  "edit": "",
  "doctorID": 1,
  "patientUsername": "david",

  "uri":"http://localhost:9000/app/appointment/1"
}
```

User Login Representation

```
{
  "username": "david",
  "password": "1234!@#$"
}
```


Back End Layers



Security

Every request processed under security by authentication and authorization check

Roles: Admin (Chief)

Credentials: Username

Doctor

Password

Patient



Trello

Boards

Sacchon

Sacchon Free

Team Visible

Invite

Board

Sacchon

Team Visible

Invite

to do list

database - hibernate

rest.api

sitemap / front end design

προσθέστε και εσείς ό,τι χρειαστεί στο μελλον

Hibernate additional functions merging

+ Add another card

Hibernate implementation

Creation of database

Patient table created

Doctor table created

Appointment table created

Measurement table created

+ Add another card

Hibernate testing

Test Records created for all tables

Automatic patient functionality added taking into account number and date of measurements

+ Add another card

REST Representations

Keep password field in Patient and Doctor Representation so user can change it

Add fields -private Patient patient; -private Doctor doctor; -private int appt_id; in AppointmentRepresentation to get id's in ApplicationResourceImpl

+ Add another card

REST Resources

Test Patient Resource

Implementation of Doctor, Measurement and Appointment Resource (get by id)

Implementation of Patient, Doctor, Measurement and Appointment resource verbs: (Delete, Put, Get(List), Post) +Patch!!

(Optimization) Make the Resource Interfaces "Abstractly" (if it's possible)

Implementations for Patient, PatientList, Doctor and Appointment

+ Add another card

REST Routing

Create Custom Router & add endpoints for each resource implementation

REST api list functionalities added, examples:
<http://localhost:9000/app/appointment?doctorID=2&patientID=2>
<http://localhost:9000/app/measurement?patientID=1>
<http://localhost:9000/app/patient?doctorID=1>

+ Add another card

File exchange and Git

- Creating repositories
- Adding collaborators
- Checking for conflicts
- Ready, set... commit, push, pull, go!



What we would add

- Robust security
- Improved styling
- Better responsiveness
- Better user experience & feel
- Better structure on front-end code

Our best work

- API design
- Database design
- Simple interface

What we omitted

- Edit of measurements & consultations
- Add consultation back-end implementation



Thank you for your attention

Ready for questions!