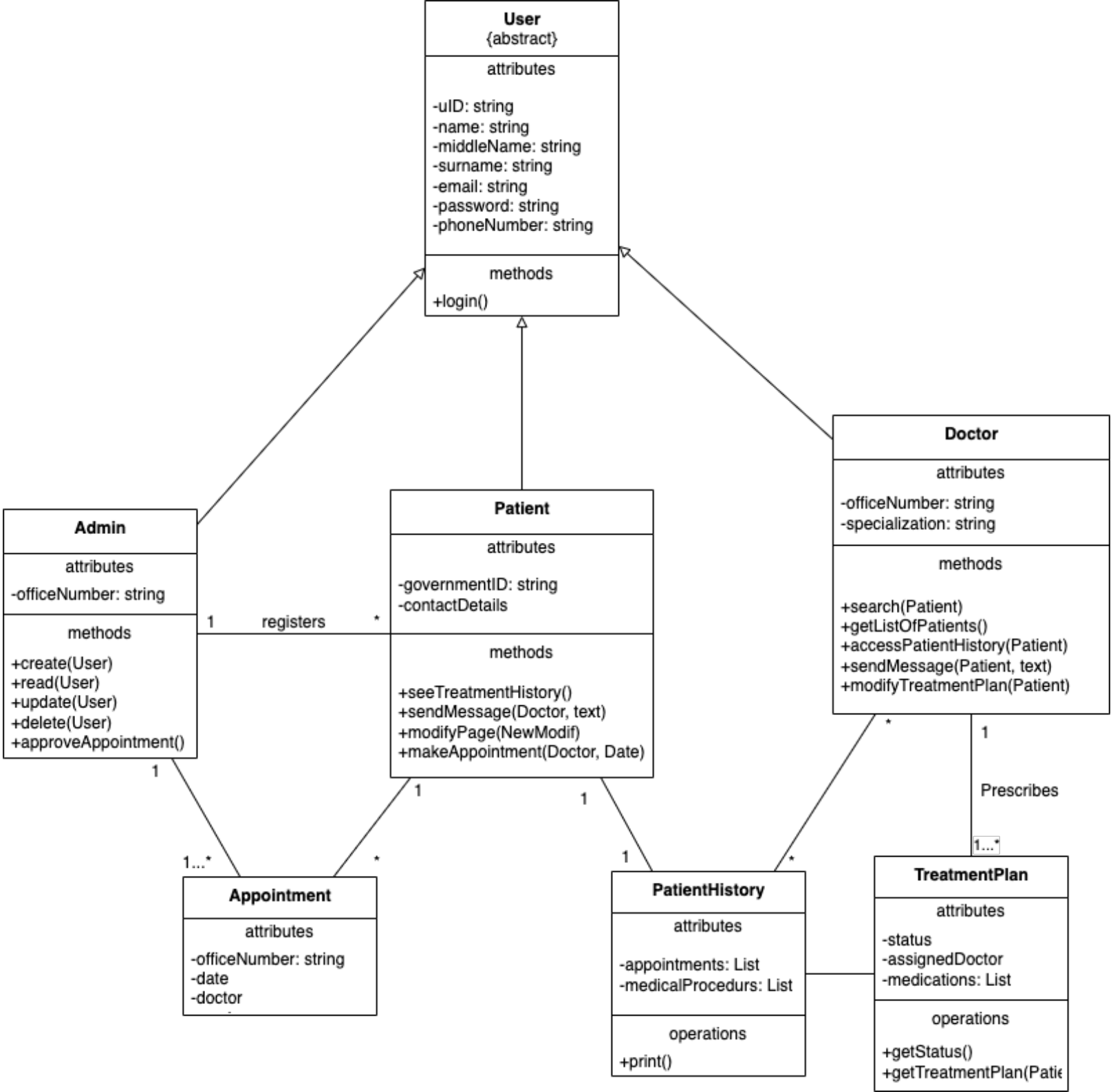
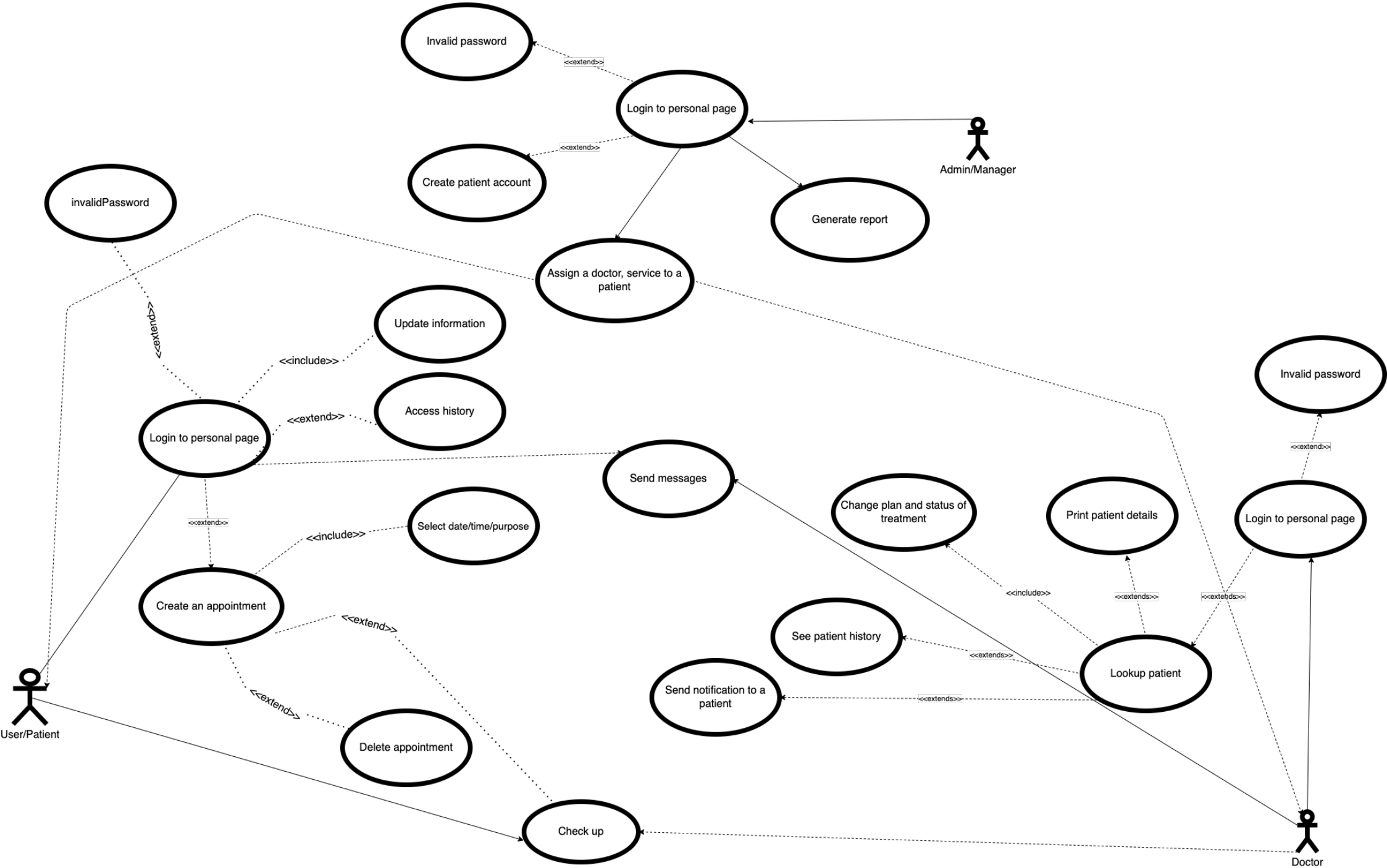
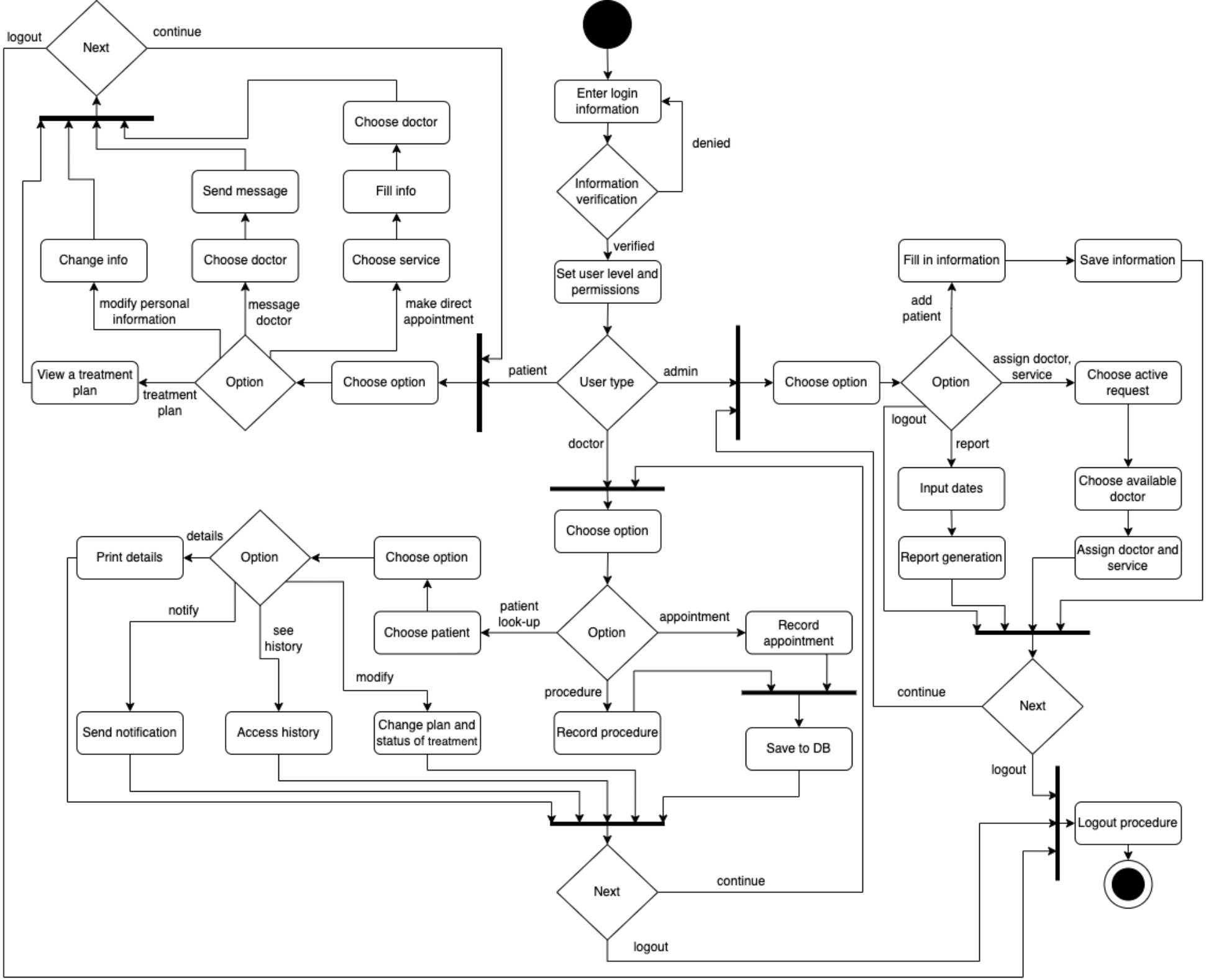
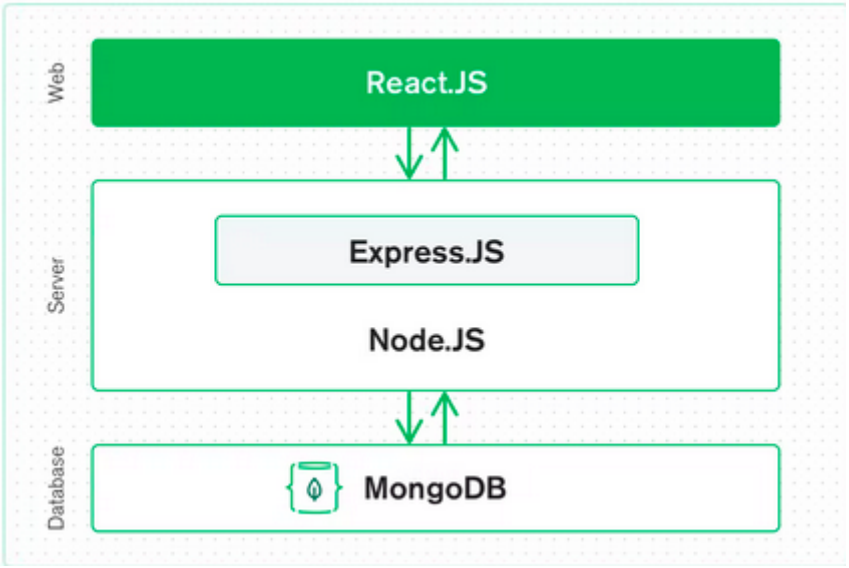

User
{abstract}
attributes
-uID: string
-name: string
-middleName: string
-surname: string
-email: string
-password: string
-phoneNumber: string
methods
+login()
Admin
attributes
-officeNumber: string
methods
+create(User)
+read(User)
+update(User)
+delete(User)
+approveAppointment()
Patient
attributes
-governmentID: string
-contactDetails
methods
+seeTreatmentHistory()
+sendMessage(Doctor, text)
+modifyPage(NewModif)
+makeAppointment(Doctor, Date)
Doctor
attributes
-officeNumber: string
-specialization: string
methods
+search(Patient)
+getListOfPatients()
+accessPatientHistory(Patient)
+sendMessage(Patient, text)
+modifyTreatmentPlan(Patient)
1
registers
*
1
1
1
*
1
*
1
*
1
Prescribes
1...*
1...*
Appointment
attributes
-officeNumber: string
-date
-doctor
PatientHistory
attributes
-appointments: List
-medicalProcedurs: List
operations
+print()
TreatmentPlan
attributes
-status
-assignedDoctor
-medications: List
operations
+getStatus()
+getTreatmentPlan(Patie


Invalid password
<<extend>>
Login to personal page
Admin/Manager
<<extend>>
Create patient account
Generate report
invalidPassword
Assign a doctor, service to a patient
<<extend>>
Update information
<<include>>
Access history
<<extend>>
Login to personal page
Send messages
<<extend>>
Select date/time/purpose
<<include>>
Create an appointment
<<extend>>
<<extend>>
Delete appointment
User/Patient
Check up
Invalid password
<<extend>>
Login to personal page
Change plan and status of treatment
Print patient details
<<include>>
<<extends>>
<<extends>>
See patient history
<<extends>>
Lookup patient
Send notification to a patient
<<extends>>
Doctor


Enter login information
denied
Information verification
verified
Set user level and permissions
User type
patient
admin
Choose option
Option
View a treatment plan
treatment plan
modify personal information
Change info
message doctor
Choose doctor
Send message
make direct appointment
Choose service
Fill info
Choose doctor
logout
Next
continue
Choose option
Option
assign doctor, service
Choose active request
Choose available doctor
Assign doctor and service
add patient
Fill in information
Save information
report
Input dates
Report generation
doctor
Choose option
Option
details
Print details
notify
Send notification
see history
Access history
modify
Change plan and status of treatment
Choose option
Choose patient
patient look-up
Option
appointment
Record appointment
Save to DB
procedure
Record procedure
Next
continue
logout
Next
continue
logout
Logout procedure

For this project, we will use the MERN stack, which stands for MongoDB, ExpressJS, ReactJS, and Node.js.



We use this stack because MERN is a full-stack solution, meaning that it includes a stack for both the back-end and front-end. Moreover, these technologies synchronize with each other well.

React.JS is for the frontend part and serves for creating dynamic pages with reusable components. React is also designed for handling errors, events, etc.

Express.JS serve for creating a server-side platform inside the Node.JS server. In meanwhile, **Node.JS** is responsible for creating a Javascript web server. Express has powerful functions that handle HTTP requests and responses. Meanwhile

MongoDB is a database system. MongoDB is perfectly suited for working with Node.JS and has great functionality with JSON data.

The stacks mentioned above, allow a natural flow of JSON from *front* to *back*. Also, it requires knowing only javascript instead of learning a bunch of different languages for each tier.

The given project involves a lot of database management and login-registering of new users. For that purpose, MongoDB suits well.

The drawback of MERN is despite the fact that MERN uses only javascript language, it involves a deep knowledge of javascript, which would require a lot of time to get to know it. Instead of using MongoDB, there is a possibility of using backend-as-a-service Firebase, which would significantly decrease the development process.
Though, the choice of Database document will be fully decided later after the first prototype.