Spring Boot Quick Start

* What is Spring Boot?

→ Spring Boot makes it easy to create stand-alone, production-grade Spring based Application that you can "just run".

* What is Spring?

→ i) Application framework.

ii) Programming and configuration model

iii) Infrastructure Support

* Problem with Spring:

→ i) Huge framework.

ii) Multiple Setup steps.

iii) Multiple configuration steps

iv) Multiple build and deploy steps

* Enter Spring Boot

→ i) Opinioned

ii) Convention Over Configuration

iii) Stand alone.

iv) Production ready.

* Starting Spring Boot

SpringApplication.run (App.class , args);

i) Sets up default configuration.

ii) Starts spring application context.

iii) Perform class path scan.

iv) Start tomcat server.

* Let's add a Controller

→ i) A Java class

ii) Marked with annotations

iii) Has info about

a) What URL access triggers it?

b) What method to run when accessed?

* **Controller Mapping.**

Controller Mapping is the rule that tells Boot which URL should call which method in your controller.

* **Embdded Tomacat Server.**

→ i) Convenience

ii) Servlet container config is now application config.

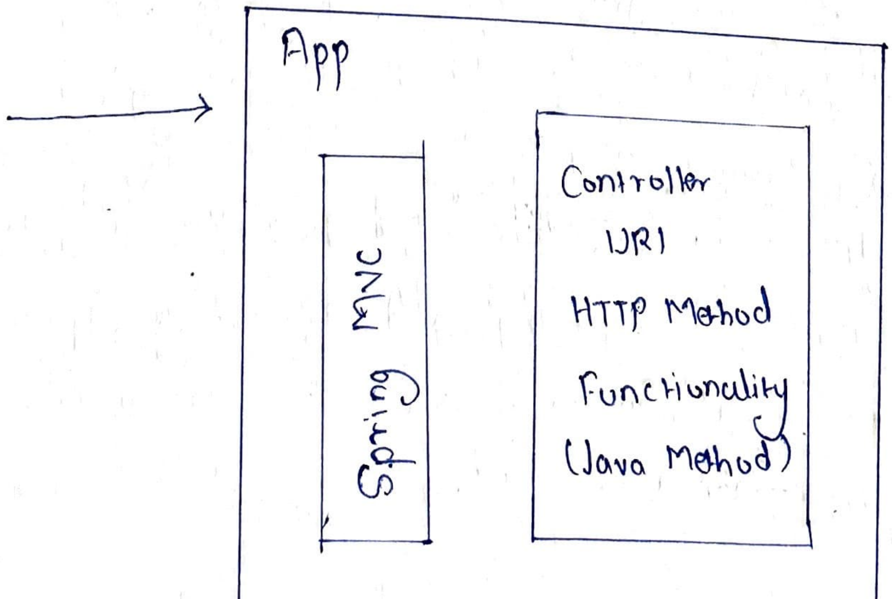
iii) Standalone application

iv) Useful for microservices architecture.

* **Spring MVC Controller.**

Spring MVC is the web Framework inside the spring ecosystem that helps you build web applications and REST API's in a clean structured way.

In simple. ~~a~~ spring mvc is a framework that handles HTTP request and maps them to Java method.



* Course API

1) Resources: i) Topic
ii) Course
iii) Lesson

i) Topics :-

GET	/topics	Get all topics
GET	/topics/id	Gets the topic
POST	/topics	Create a new topic
PUT	/topics/id	Update the topic
DELETE	/topics/id	Deletes the topics

* Booting Spring Boot.

1) Starting a spring Boot App
i) Spring Initializer
ii) Spring Boot CLI
iii) STS IDE
~~iv)~~

2) Configuration

* Spring Data JPA

1) JPA (Java Persistence API)

~~Java~~ JPA is a standred way in java to store data and retreive data from a database using java object instred of SQL everywhere.

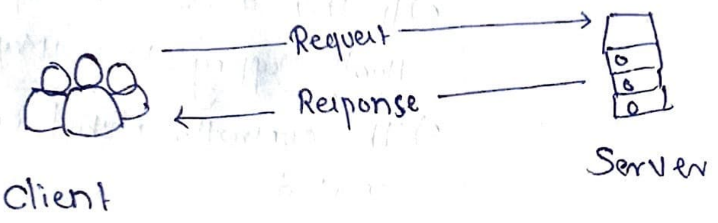
2) Object - Relational Mapping

Object - Relational Mapping (ORM) is a technique that connectes Java Objects with database tables so you can work with data using objects instred of SQL rows.

* API'S (Application Programming Interface)

An API is a set of rules and methods that allows one software application to communicate with another.

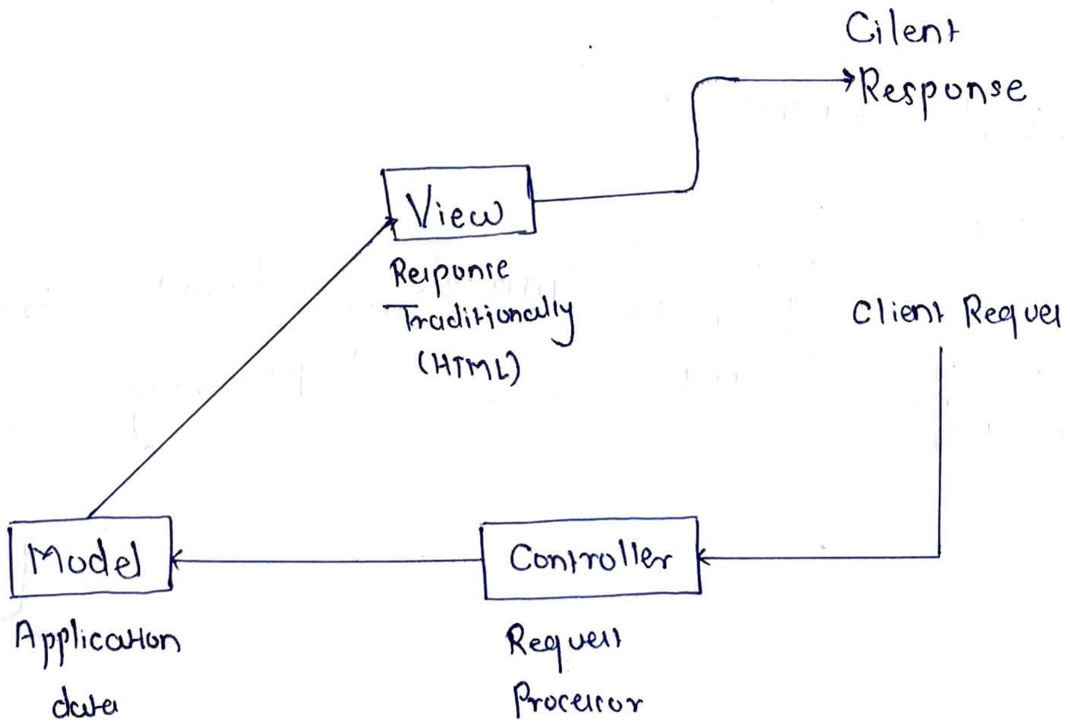
In short:-
API is a bridge that lets different programs talks to each other



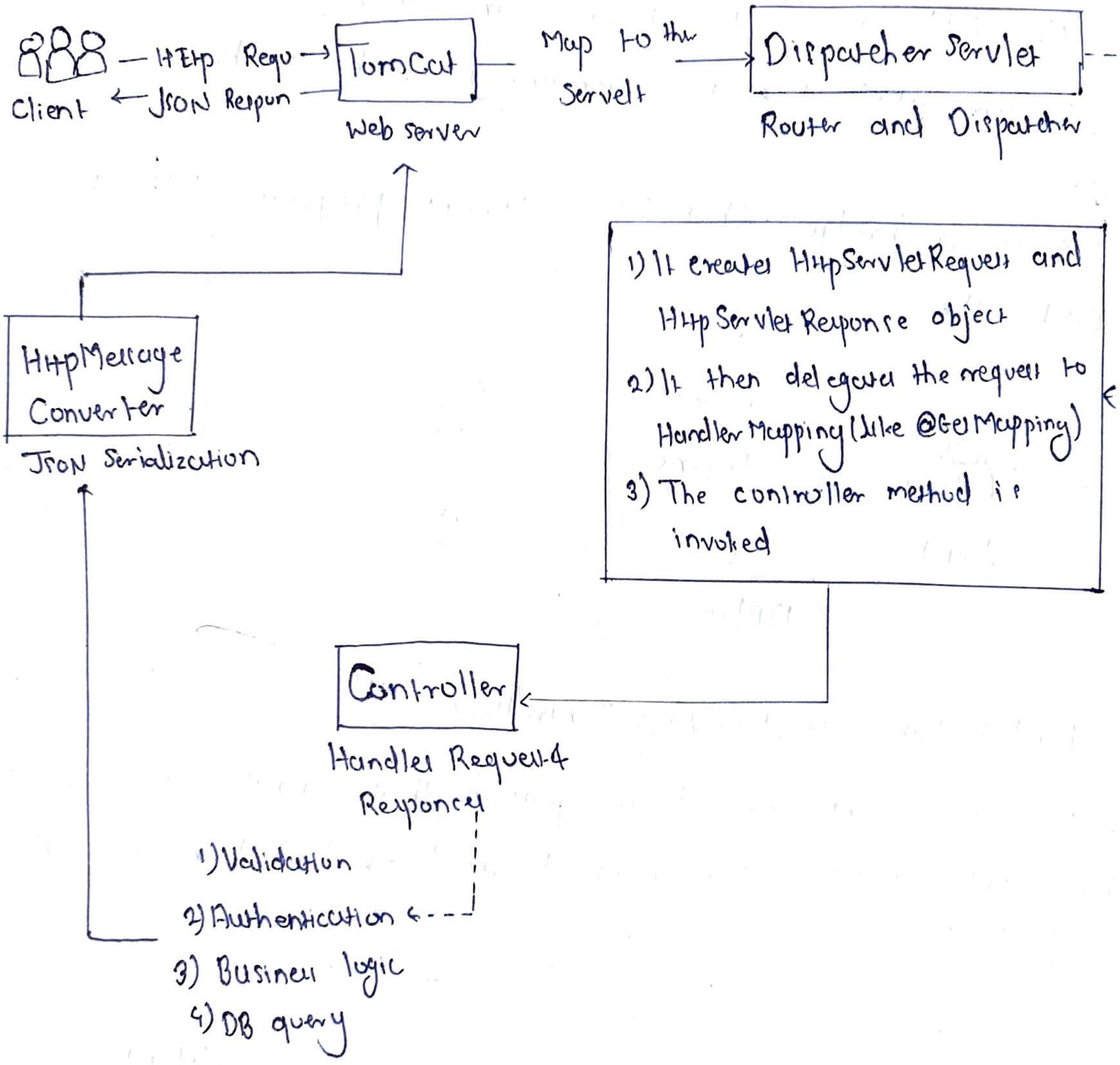
API's

* MVC Architecture (Model-View-Controller)

MVC is a software design pattern that seprate an application into three interconnected components to make code clean, maintainable and scalable.



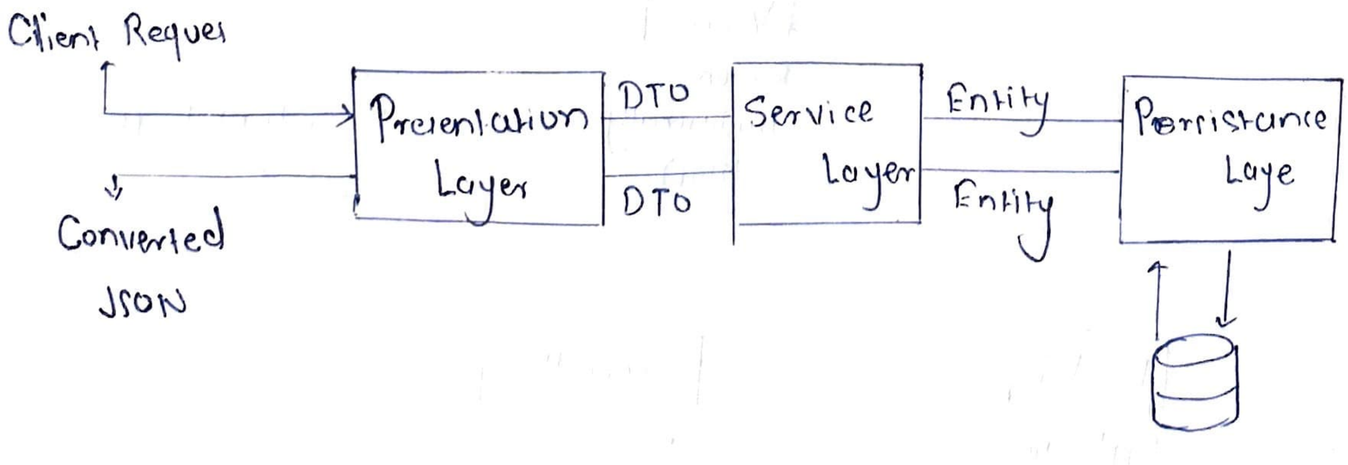
* How does Web Server works in Spring Boot ?



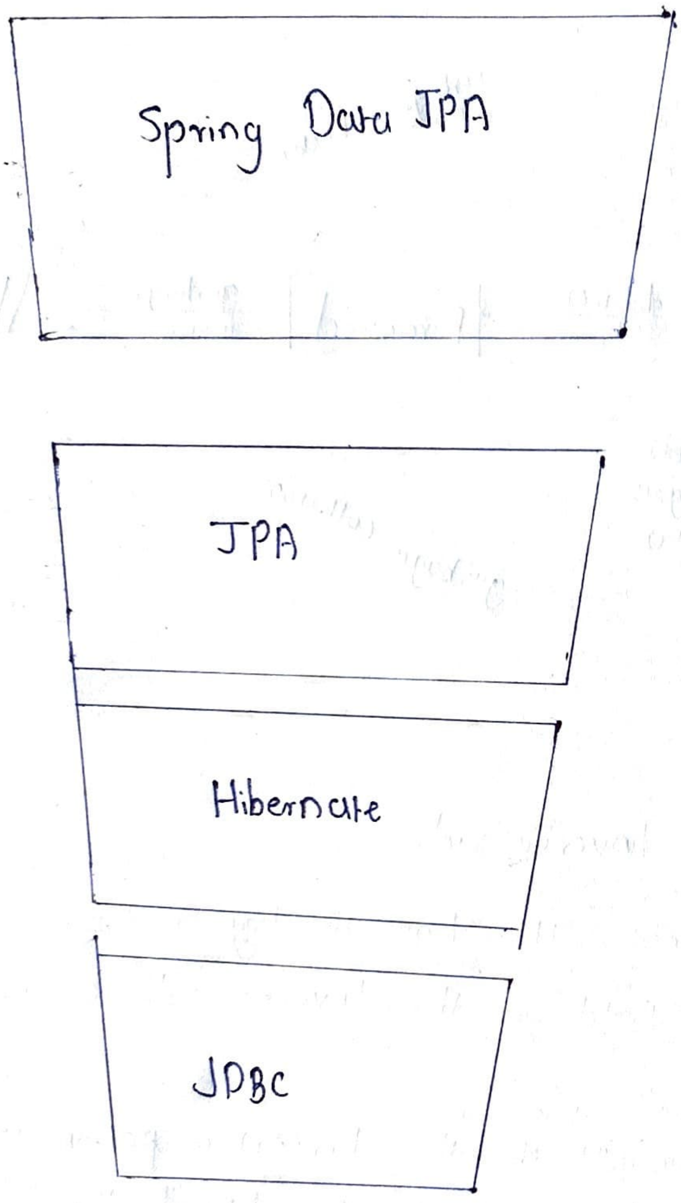
1) It creates HttpServletRequest and HttpServletResponse object
2) It then delegates the request to Handler Mapping (Like @GetMapping)
3) The controller method is invoked

1) Validation
2) Authentication
3) Business logic
4) DB query

* 3- level archeteture



* Spring Data JPA



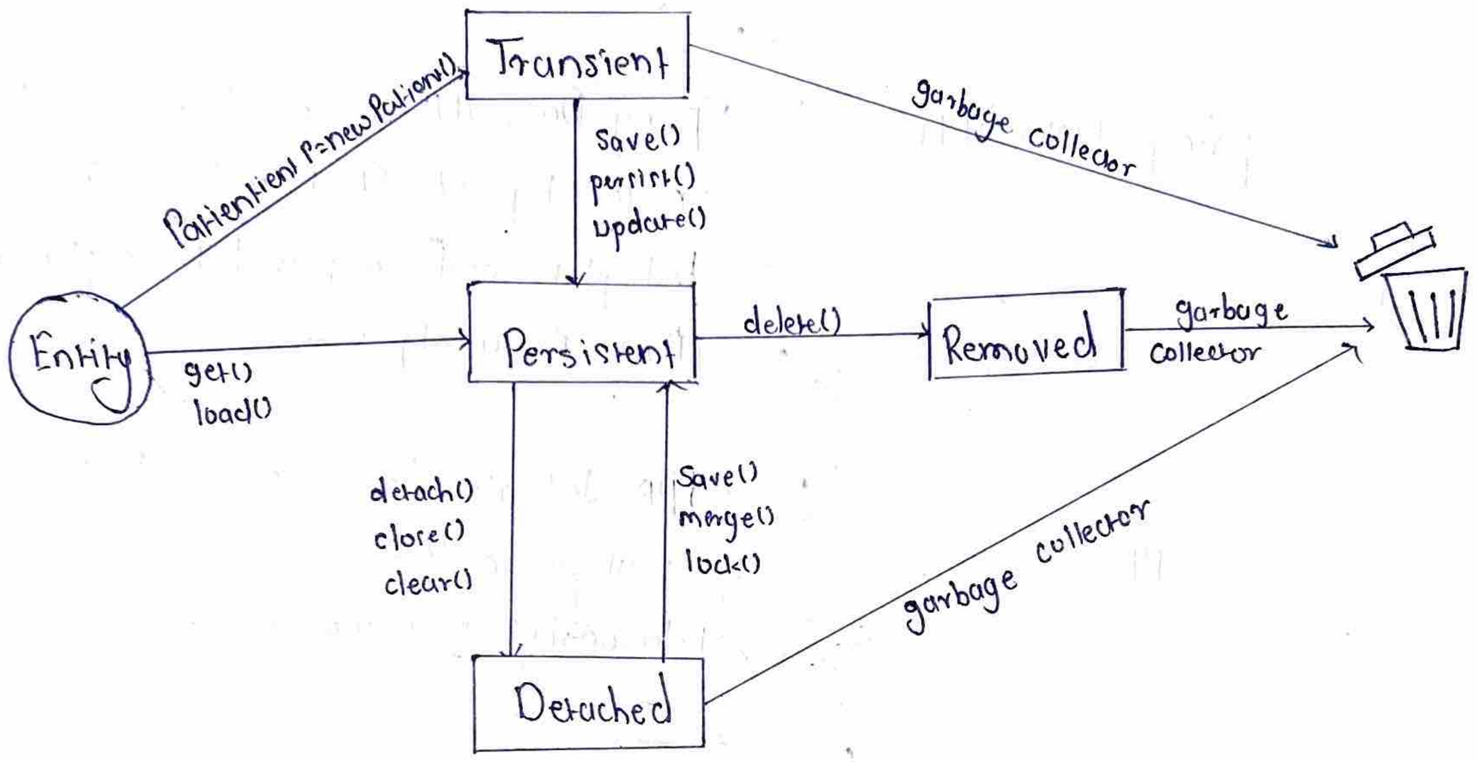
Spring Data JPA is an abstraction layer on the top of JPA to reduce the boilerplate code required to implement Data Access object

JPA (Jakarta Persistance API) is a specification that facilitates object Relationship mapping in JPA

Hibernate is an implementation of JPA, and it generates SQL queries

SQL Queries are executed by JDBC which connectes to the Database

* Hibernate = Entity LifeCycle

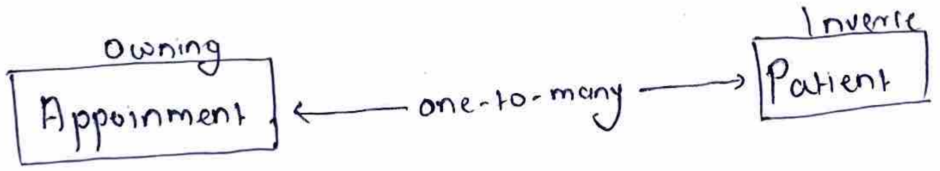


* Relationship Owning side and Inverse side

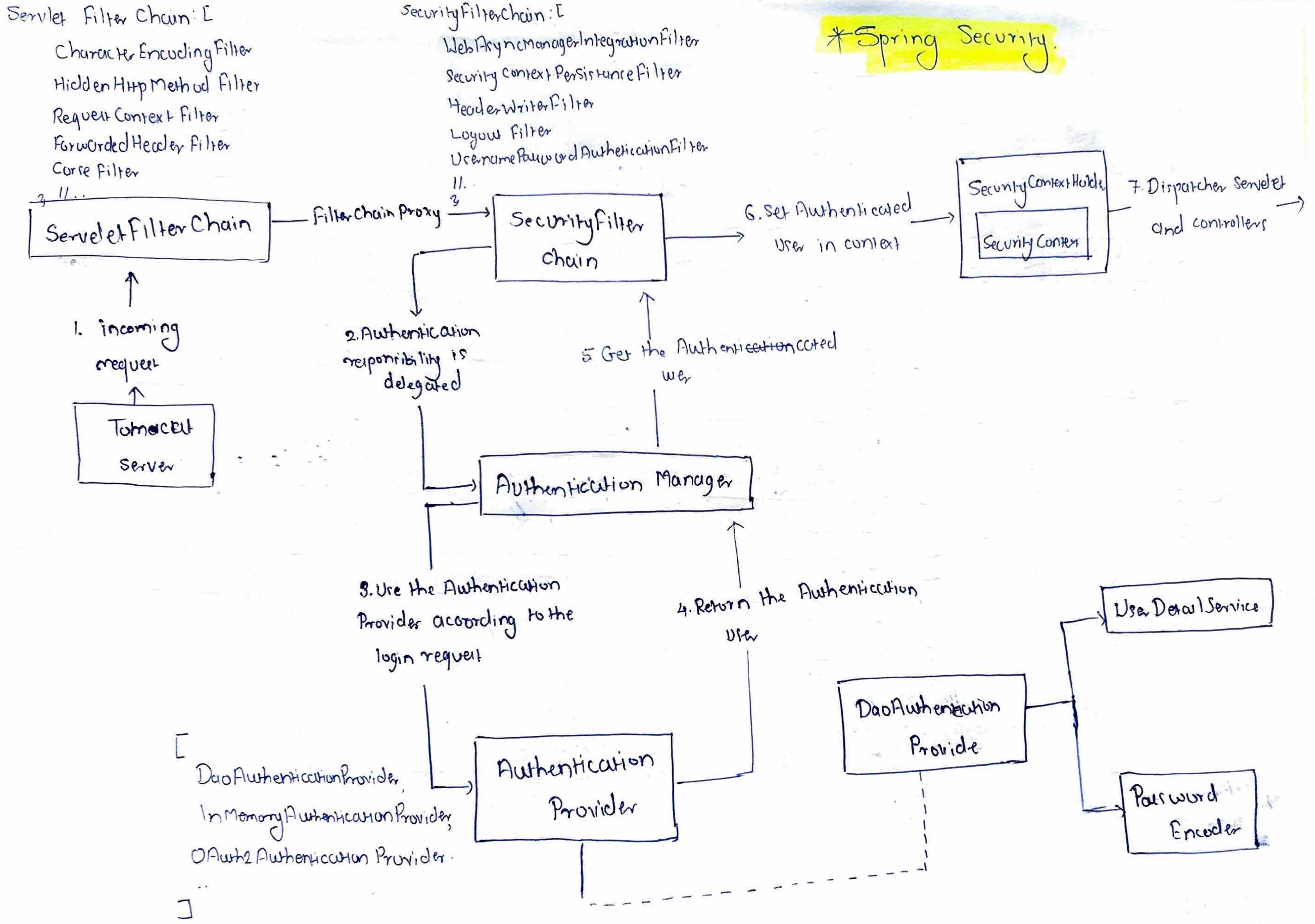
key points :- i) The owning side dicates the foreign key updates.

ii) Updates to the mapped field on the Inverse side cannot update the forign key.

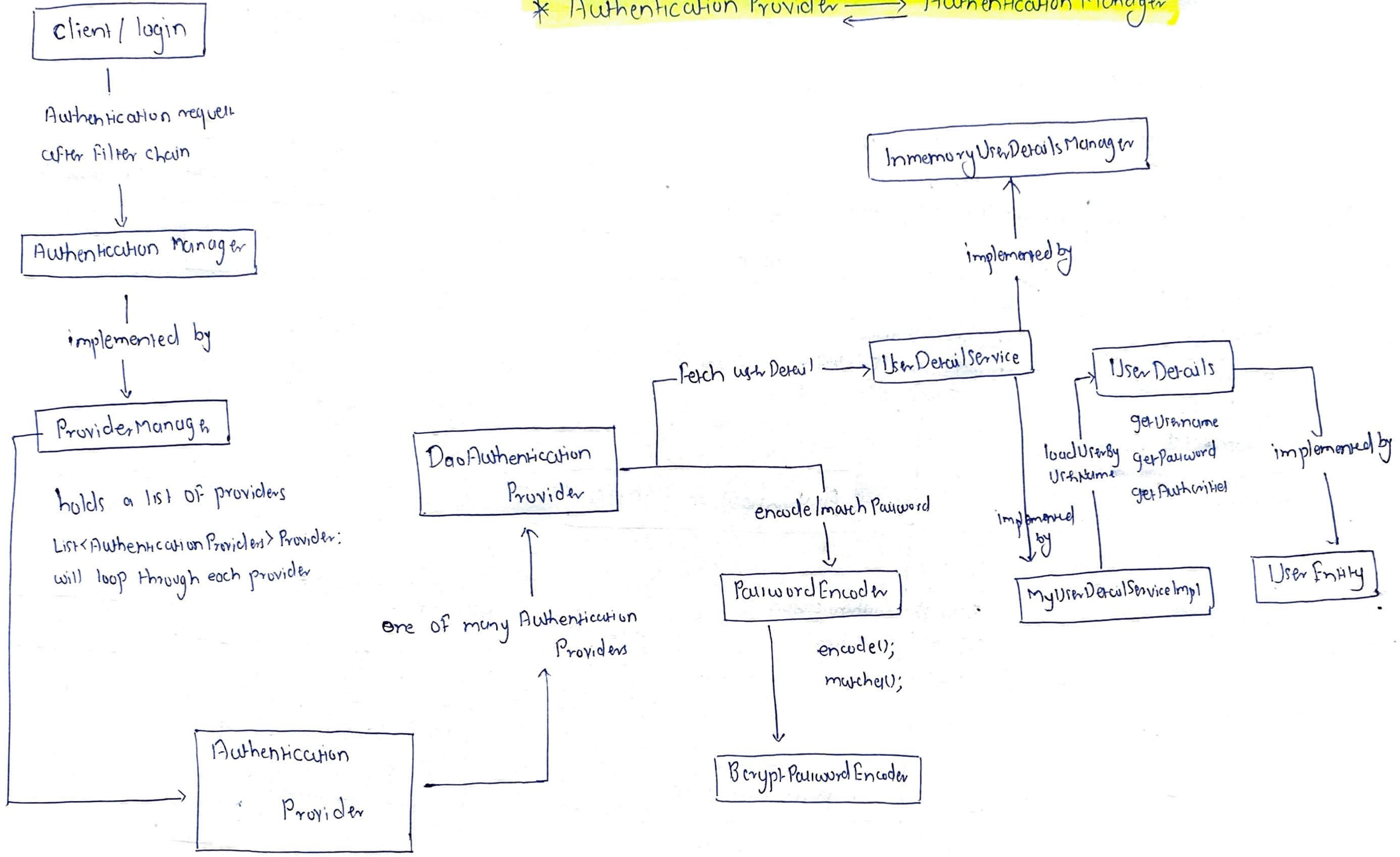
iii) Parent controlls the lifecycle of other, here if a patient is deleted, their appointments should also be deleted. Hence Patient is parent.



*Spring Security.


Servlet Filter Chain: [
CharacterEncodingFilter
HiddenHttpMethod Filter
Request Context Filter
ForwordedHeader Filter
Corse Filter
//..
]
SecurityFilterChain: [
WebAsyncManagerIntegrationFilter
Security Context PersistenceFilter
HeaderWriterFilter
Logout Filter
UsernamePassword AutheticationFilter
//..
]
ServeletFilterChain
FilterChain Proxy
SecurityFilter chain
6. Set Authenticated User in context
SecurityContextHolder
Security Contex
7. Dispatcher Servelet and controllers
1. incoming request
Tomcat server
2. Authentication responsibility is delegated
5 Get the Auth~~entication~~cated user
Authentication Manager
3. Use the Authentication Provider according to the login request
4. Return the Authentication User
[
DaoAuthenticationProvider,
InMemoryAuthenticationProvider,
OAuth2 Authentication Provider.
..
]
Authentication Provider
DaoAuthentication Provide
User Detail Service
Password Encoder

* Authentication Provider ⟷ Authentication Manager



* Cascading in JPA Mappings

If cascade = CascadeType.PERSIST or ALL, and you've added appointment object to patient.getAppoinment() and set appoinment.setPatient (patient), then:

i) Saving the Patient automatically saves the Appoinments.

ii) Deleting the patient automatically deletes all Appoinments. (because of REMOVE and orphanRemoval = true).

iii) No need to explicitly save or delete Appoinment.

* JWT (JSON Web Token)

① eyJhbGhad dbeFGhNop. ② eyJZNOrlpsTr2295Fd. ③ xbHFNRTmY92

① Header

```
{"alg": "HS256"
 "typ": "JWT"
}
```

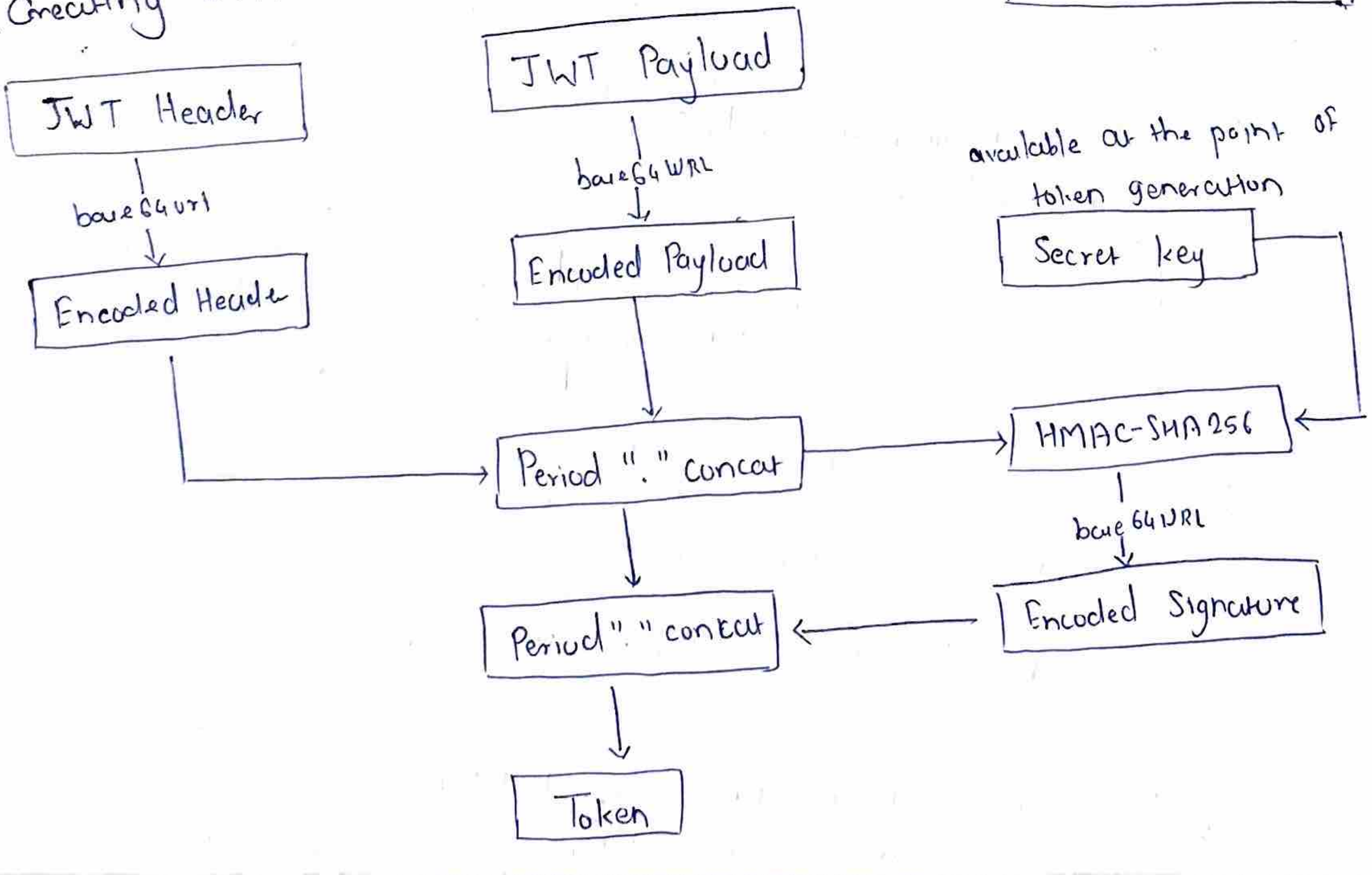
② Payload

```
{"sub": "1234567890",
 "name": "John Deo",
 "iat": 15680
}
```

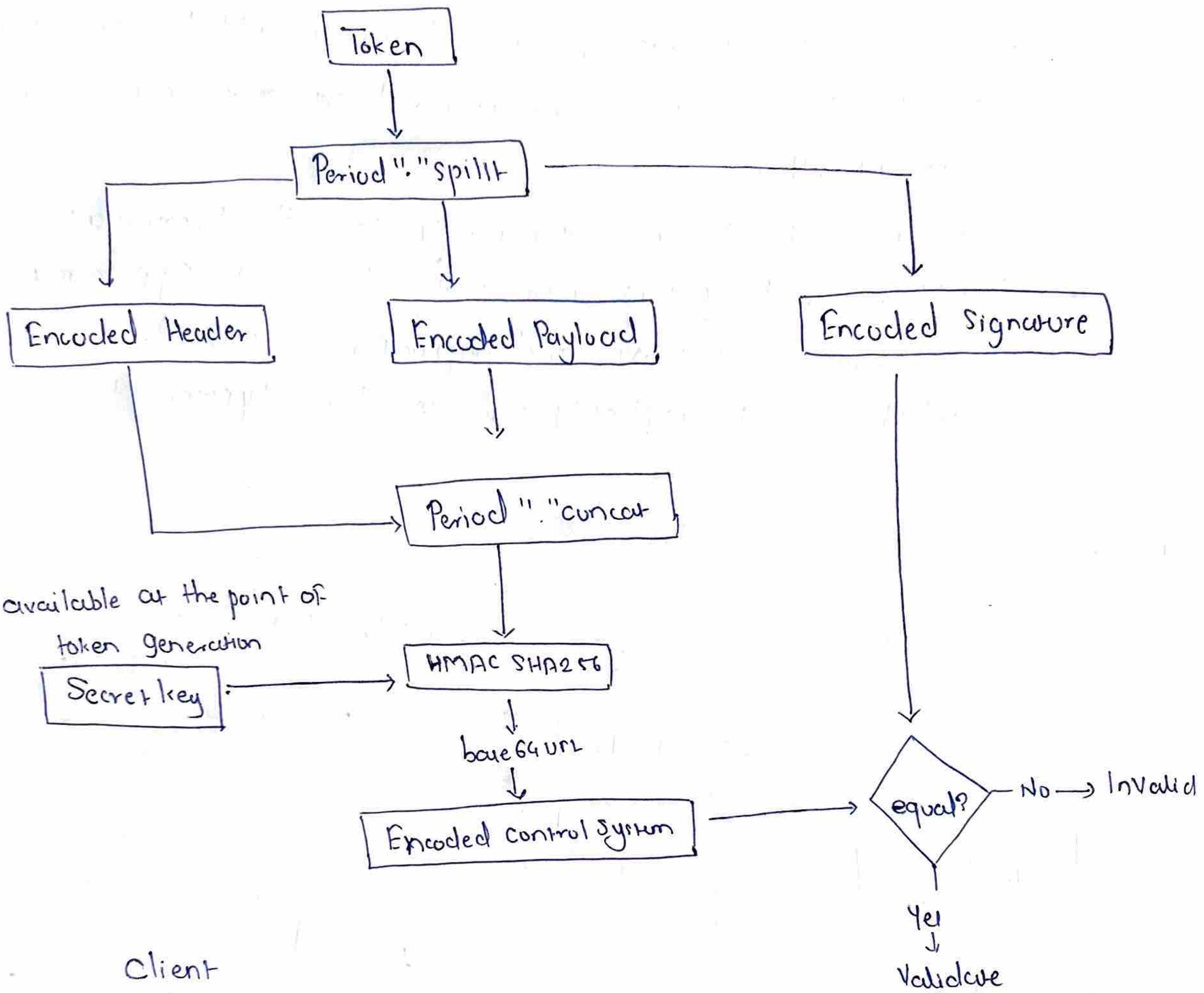
③ Signature

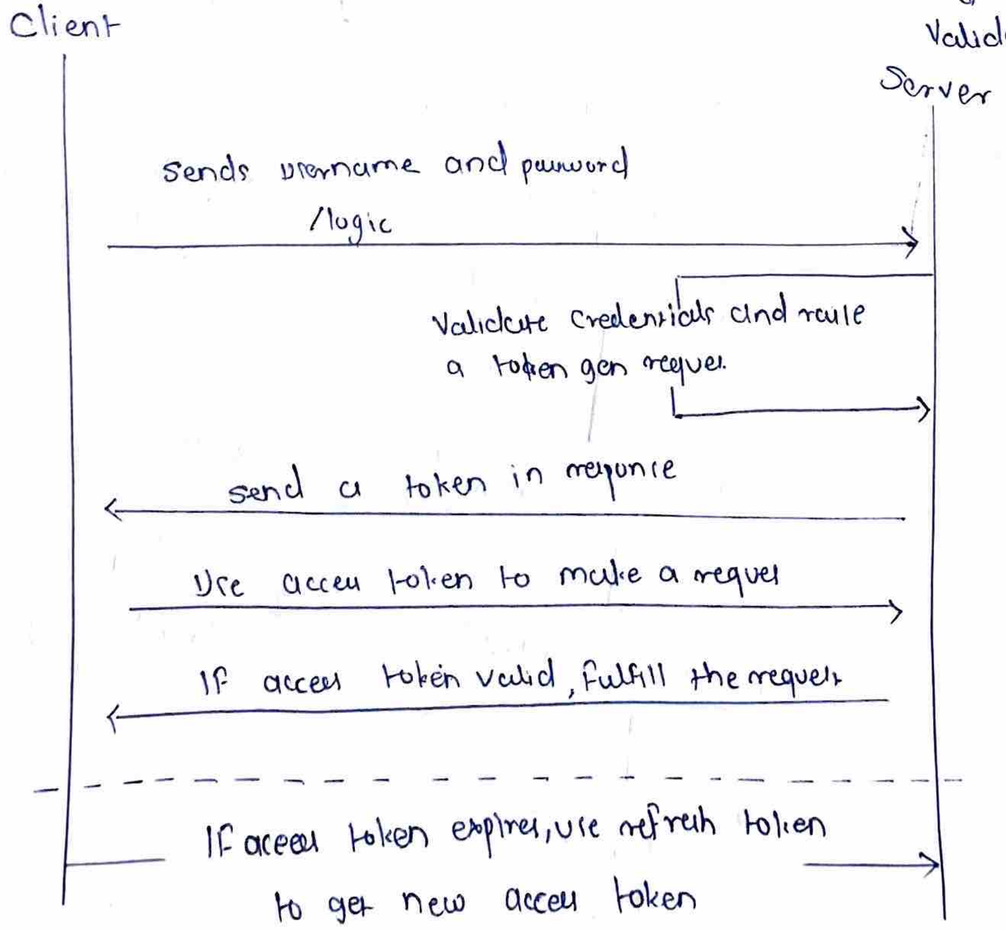
```
HMACSHA256(
BASE64URL(header)
BASE64URL(Payload)
Secret)
```

* Creating JWT



* Validation a JWT





Client
Server
sends username and password /login
Validate credentials and raise a token gen request
send a token in response
Use access token to make a request
If access token valid, fulfill the request
If access token expires, use refresh token to get new access token