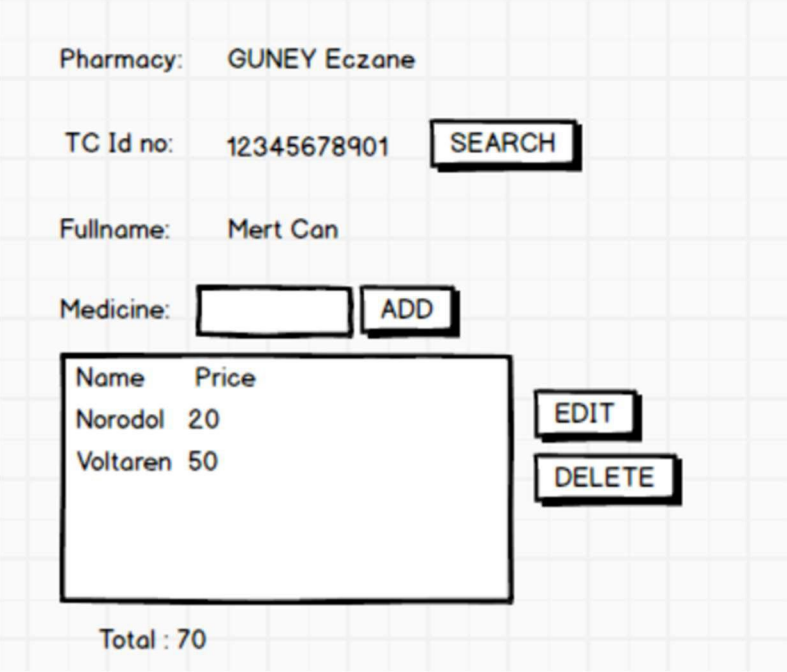
Create a Prescription Management System For Pharmacies

Create a prescription management system for Saglik Bakanligi for pharmacies. Implement Below are use cases for functional and non-functional requirements.

SUBMIT PRESCRIPTION FOR PATIENT



- Will be used by pharmacies to enter prescriptions
- Pharmacies will authenticate by username/password to call 'Create Prescription' service.
- The medicines will be a no-authentication lookup which is explained in next use case

MEDICINE LOOKUP SERVICE

- There is no medicine lookup service by Saglik Bakanligi
- Medicine lists are published weekly at https://www.titck.gov.tr/dinamikmodul/43 .
- You will write a no-authentication REST API that will enable querying medicine name
- i.e if you search any medicine that ASP word in it, it will return

```
{"medicationNames":
["ASPIRIN","CASPOBIEM","CASPOPOL","CORASPIN","SIGMASPORIN","VASPARIN"]
}
```

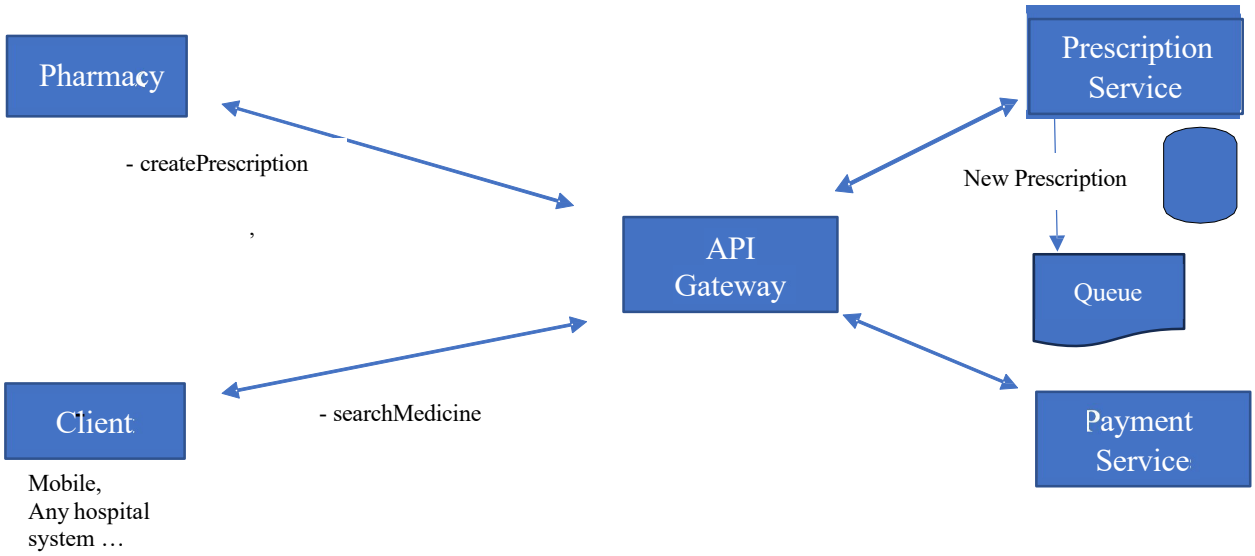
- Automatically download latest file and write a parsing program to upload this to your Database.
- Doing this via a mockup service like https://run.mocky.io IS NOT ACCEPTABLE, data must reside in your database

PAYMENT SERVICE

- A separate payment service will be developed that will browse all prescriptions submitted at current date and will email a report to each pharmacy of the total amount due to pharmacy. This will run at 1:00 every night

> To: GUNEY Eczane
> You have submitted 10 prescriptions today
> Total amount is 1560 TL

NON-FUNCTIONAL REQUIREMENTS



- Project can be built on any service oriented framework as long as requirements are met.
- Simple UI implementation per mock-ups given above is required
- All business use cases above must be available via REST webservices
- The project will be deployed to provider Azure
- Per deployment diagram above, Prescription and Payment services will be deployed separately. APIs will be reached via an API gateway
- All REST services must be versionable and support pagination when needed
- You can choose any development environment you like as long as they support REST services and is deployable in cloud. (Use Visual Studio)
- You can make assumptions as long as you document them
- DEPLOYMENT
  - create a data model for each data source and use a data service fromAzure SQL Server, Azure Cosmos Table
  - For API and UI layer hosting of app services, use a cloud service Azure App Services

    - For API gateway, use a cloud service Azure API management
    - For queue service, use Azure Message Queues
    - For scheduling services (or workflow to send payment email), use a cloud service Azure App Logic
    - For email send, you can use a gmail account see https://kinsta.com/blog/gmail-smtp-server/

DELIVERABLES

- Public Github repository link
- A README document in your GITHUB repository that has
    - Final deployed urls of the application
    - your design, assumptions, and issues you encountered.
    - Data models (i.e an ER)
    - Include a link to a short video presenting your project.