Medicine Dashboard
Documentation

Project Description:

Medicine Dashboard is a software application designed to provide medicines for patients. The system includes modules for product, requests, patients, and admin management. These modules allow the admin to perform various tasks, such as adding new products to the catalog, processing requests, managing customer accounts, and monitoring site activity, all via a user-friendly interface. Overall, the system helps to streamline the management of an online pharmacy, improve the shopping experience for patients, and make the admin's job easier and more efficient.

Main functions:

- Medicine management: This includes adding new medicines to the catalog, removing medicines, updating medicines details, and organizing medicines into categories.
- Request management: This includes processing requests, updating request status, and viewing request details.
- Patient management: This includes managing patient accounts, viewing patient details, and tracking patient requests.
- Category management: This includes adding new categories to the catalog, removing categories, and updating the details of existing categories.
- Admin management: This includes managing admin accounts, setting up permissions, and monitoring site activity.

OCL constrains:

We used few OCL expressions to define the constraints and relationships between the classes in the class diagram:

Medicine and Category:

- The categoryId of a product must refer to an existing category:
  - self.categoryId = Category.allInstances()->exists(c: Category | c.categoryId = self.categoryId)
- The categoryId of a category must be unique:
  - Category.allInstances()->forAll(c1, c2 | c1 <> c2 implies c1.categoryId <> c2.categoryId)

Request and Medicine:

- A request must have at least one medicine:
  - self.products->size() > 0
- A medicine must have a quantity greater than zero:
  - self.quantity > 0

Request and Patient:

- The patientId of a request must refer to an existing patient:
  - self.patientId = Patient.allInstances()->exists(p: Patient | p.patientId = self.patientId)
- A patient can only view their own requests:
  - Patient.allInstances()->forAll(p: Patient | p.viewRequestHistory() = Request.allInstances()->select(r: Request | r.patientId = p.patientId))

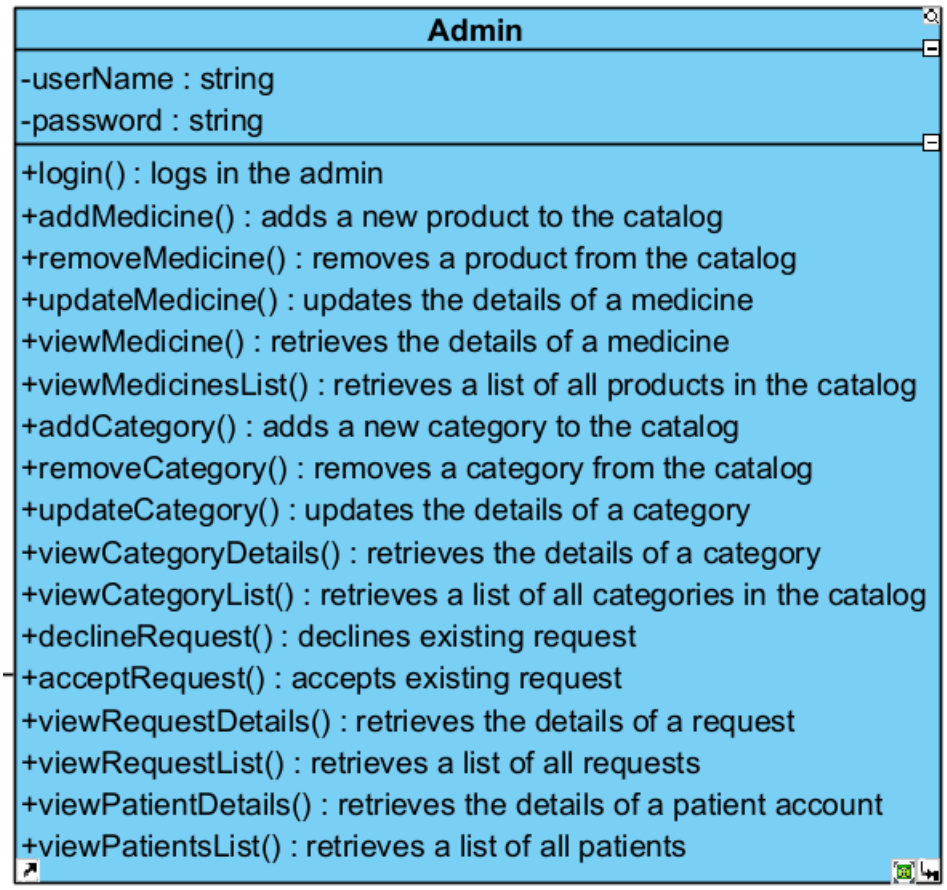
Admin and Patient:

- The patientId of a customer must be unique:
  - Patient.allInstances()->forAll(p1, p2 | p1 <> p2 implies p1.customerId <> p2.customerId)
- An admin can only view patient details if they are logged in:
  - self.login() implies patient.allInstances()->forAll(p: Patient | self.viewPatientDetails(p.patientId) <> null)

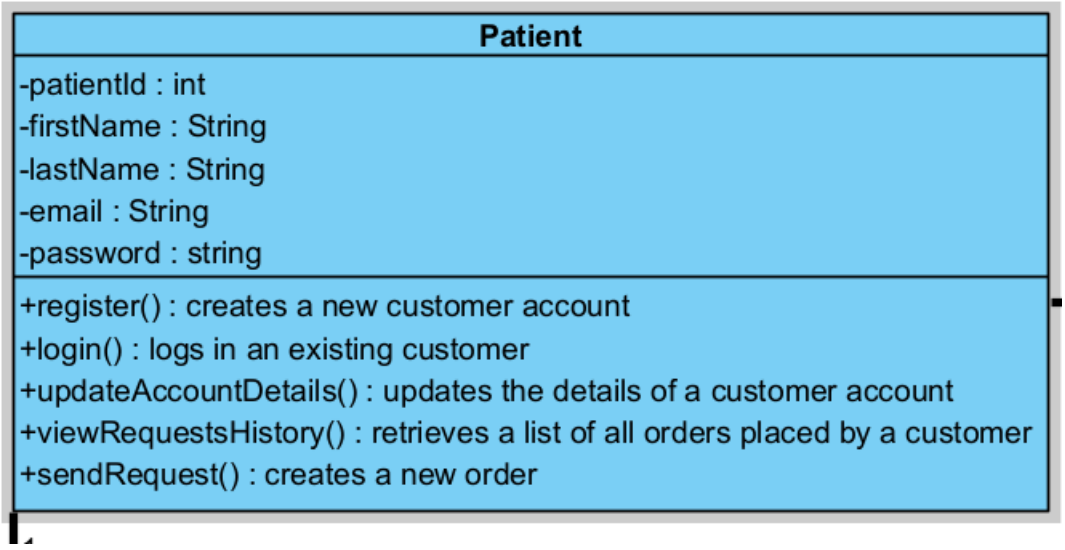
UML Diagrams:

Class Diagram:

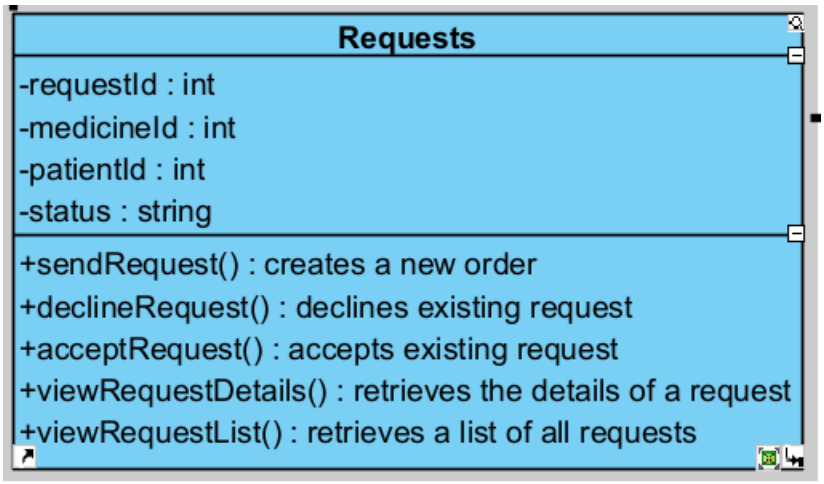
- Admin class



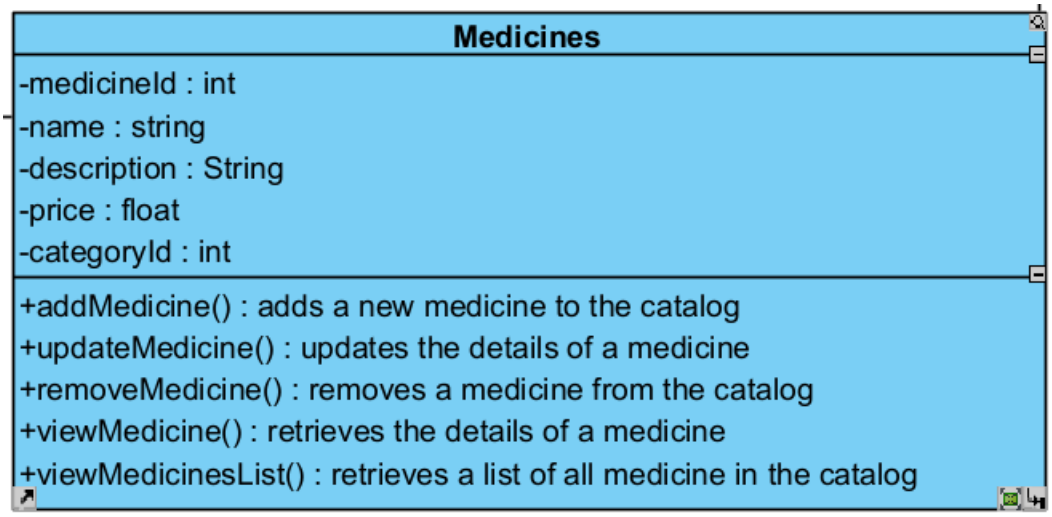
- Patient:



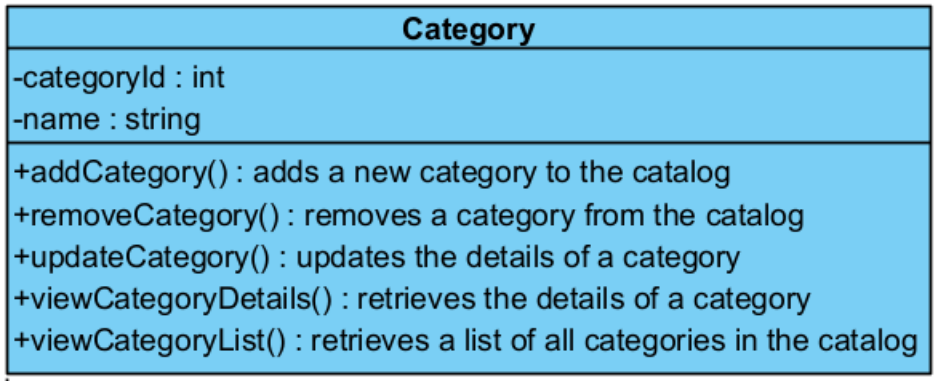
- Requests:



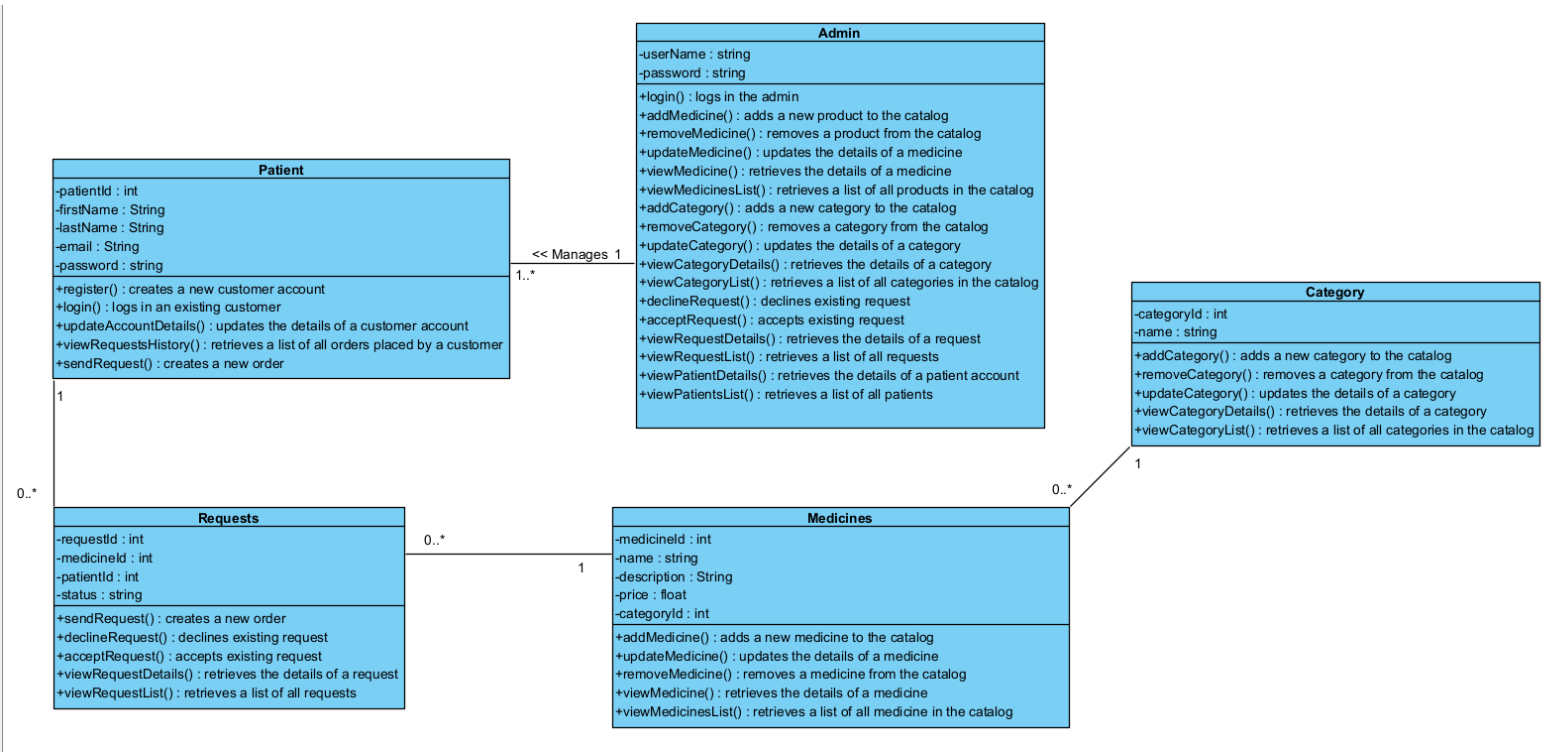
- Medicines:



- Categories:

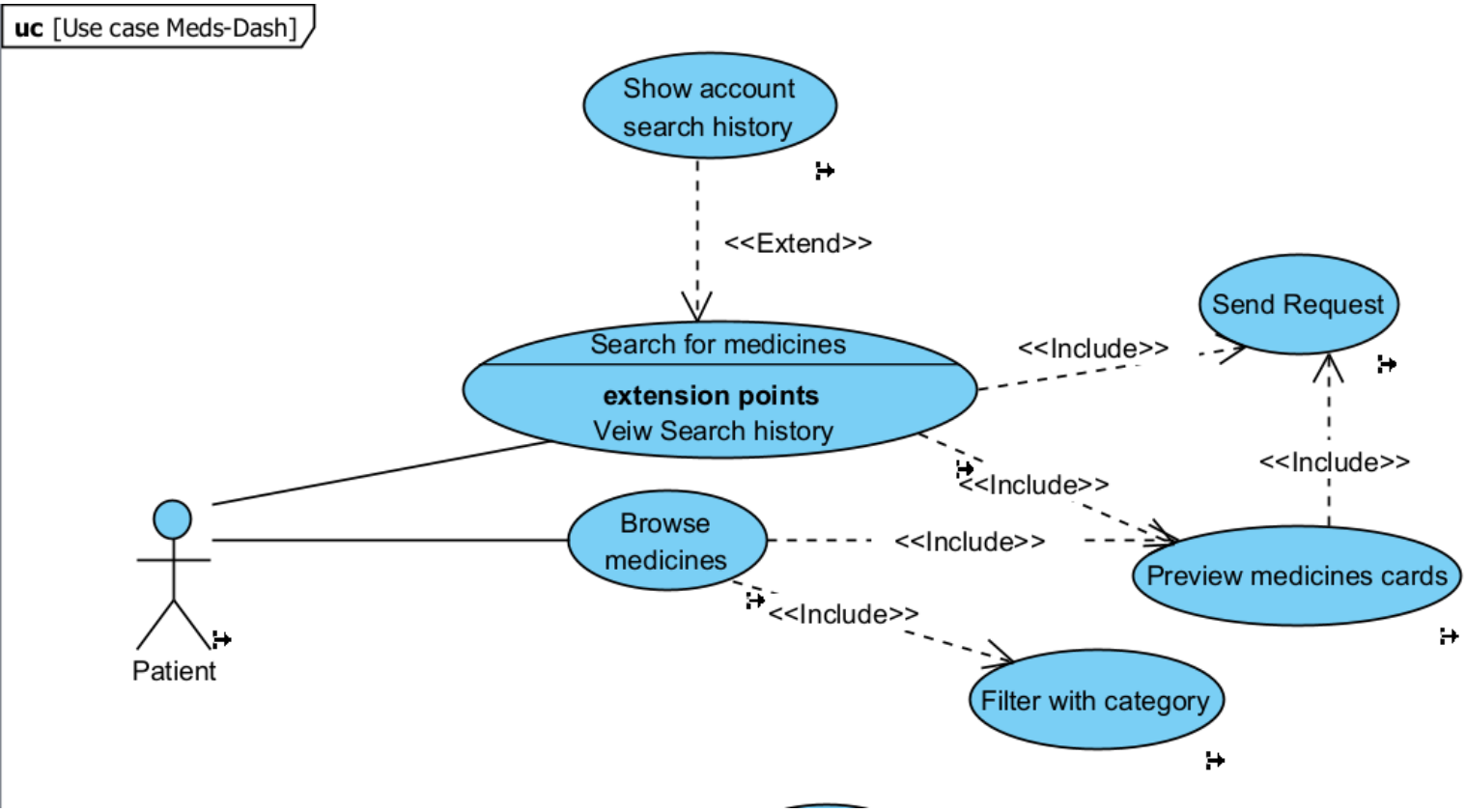


- Relations between classes:

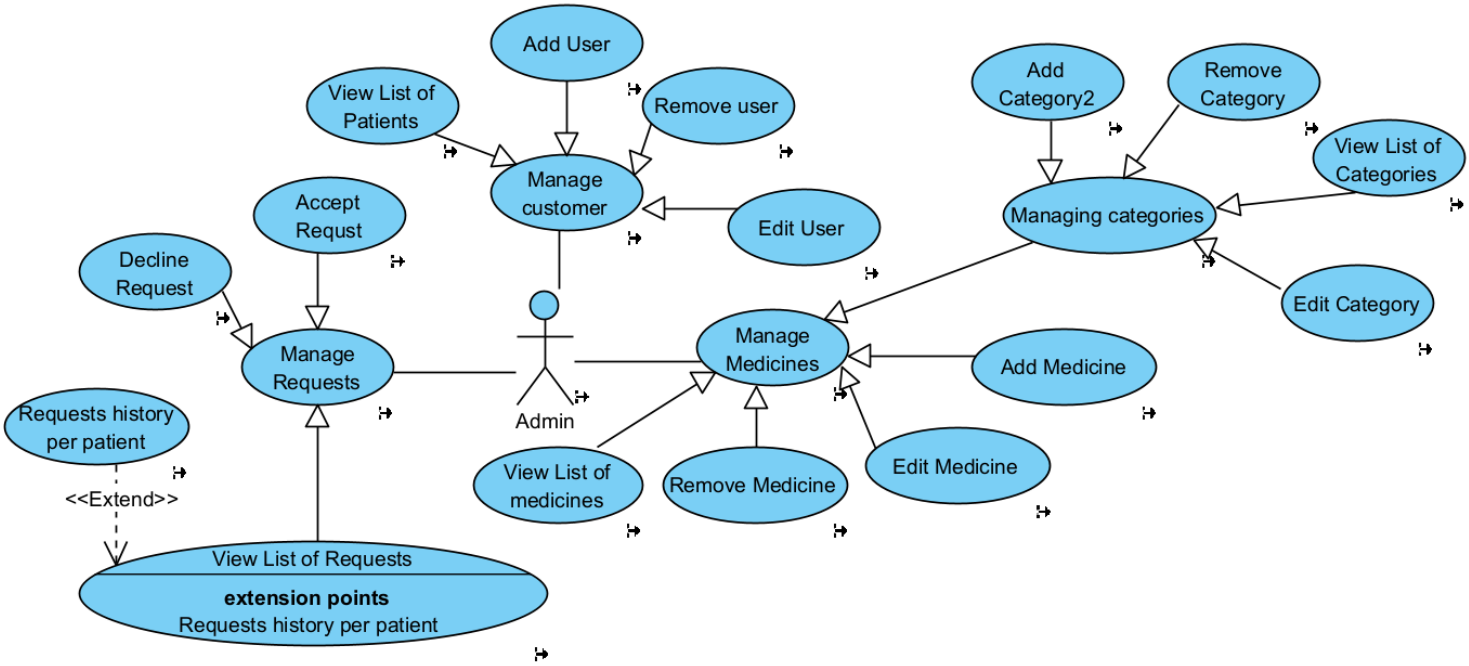


Use case Diagram:

- Patient:



- Admin:



Collaborators:

Tasneem Mahmoud
Yomna Mohamed
Menna Mohamed
Ahmed Fawzy
Alaa Erfan
Fatima Al-Zahraa