sd Frontend

DrugModelController

DrugApiServer

sd A

DrgController

DrugDatabaseServer

Press 'Details' on a Drug Details(id)

HTTP GET request api/drug

Get Drugs

GetDrugAsync(Id)

All Drugs

Drug with Id or null

OK response with all drugs

alt

[Id != NULL &&
drug != NULL

Go to details
page for drug]

Client

MedicalRecordsModelController

AppDbContext

Database

FindAsync(medicalRecordId)

Query Medical Records by ID

DeleteConfirmed(medicalRecordId)

Medical Record entity or null

alt

[Medical Record not found]

Return (not found)

[Medical Record found]

Remove(medicalRecord)

SaveChangesAsync()

Apply deletion

rowsAffected

RedirectToAction("Index")

sd Desktop

User

Home Page
Doctors Tab

Doctor
Schedule
Page

ShiftsManager

ShiftsApiClient

sd Backend

ShiftsC

More Info Button Click

Doctor Id

Doctor Id

Request for shifts

Appointments for
doctor

List of shifts for doctor

List of shifts for doctor

Sequence diagram: Client → Service (DeleteDrug) → DbContext → Database → Console

Client: DeleteDrug(drugID)

alt [try block]

Service (DeleteDrug) → DbContext: FindAsync(drugID)

DbContext → Database: Query drug by ID

Database ⇠ DbContext: drug or null

DbContext ⇠ Service: drug or null

alt [drug not found]

Service ⇠ Client: return false

[drug found]

Service (DeleteDrug) → DbContext: Remove(drug)

Service (DeleteDrug) → DbContext: SaveChangesAsync()

DbContext → Database: Apply deletion

Database ⇠ DbContext: success

DbContext ⇠ Service: completed

Service ⇠ Client: return true

[exception thrown]

Service (DeleteDrug) → Console: Log error message

Service ⇠ Client: return false

sd WebApp

sd Backend

EquipmentsModelController

EquipmentApiService

EquipmentController

EquipmentDatabaseService

Press 'Details' on an
equipment
Details(id)

GetEquipmentAsync(id)

HTTP GET
Request
api/equipment

GetEquipments()

Equipment with ID 'id'
or null

OK Response
with all
equipments

All
Equipments

alt

[id != null &&
equipment != null]

Go to
Details
page for
equipment

Display
NotFound
Page

Create

EquipmentModelsController

1. User Submits Form

- Fills out data in the Create.cshtml view and clicks submit

2. Controller Receives Data

- EquipmentModelsController's Create action gets the form data

3. Validation Check

- System verifies all required fields are correct

4. API Call

- If valid, controller asks EquipmentApiService to save the equipment

5. Backend Processing

- API sends data to backend (save to database)

6. Result

- Success: Returns to equipment list

- Failure: Shows form again with error messages

1: FormSubmitted()

2: Create(equipmentData)

alt

3: [ModelState.IsValid]

4: AddEquipmentAsync (equipment)

5: API request

6.2: RedirectToAction("Index")

3: Invalid

7.2: Return View(equipment)
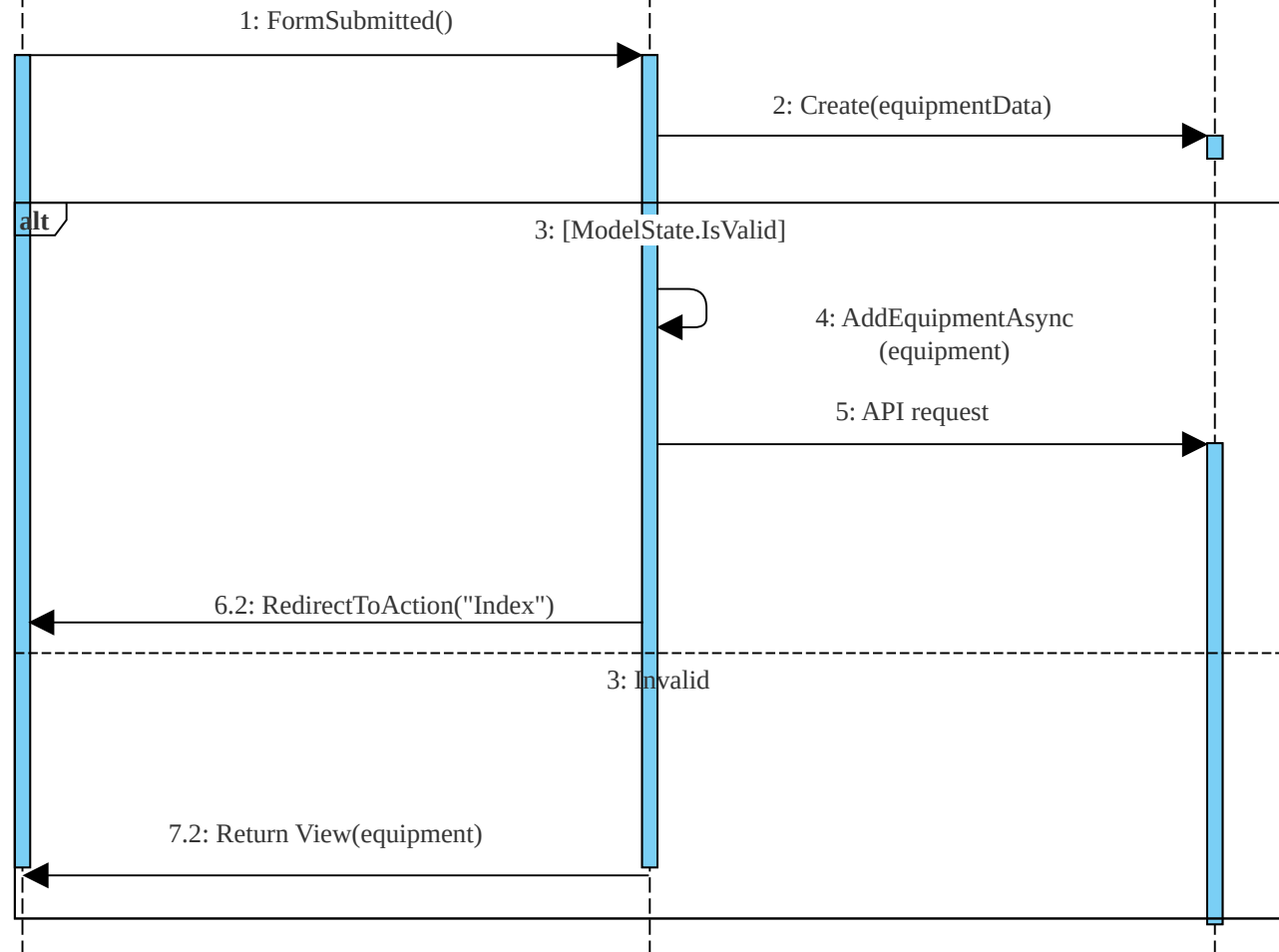
Add Drug: Use Case

Main scenaario:

The User completes the form
with:
- Name
- Administration
- Specification
- Supply

The Application validates
the fields

sends the *DrugModel* object
to DrugsAp/Service

If valid and saved successfuilly:
- displays a confirmation
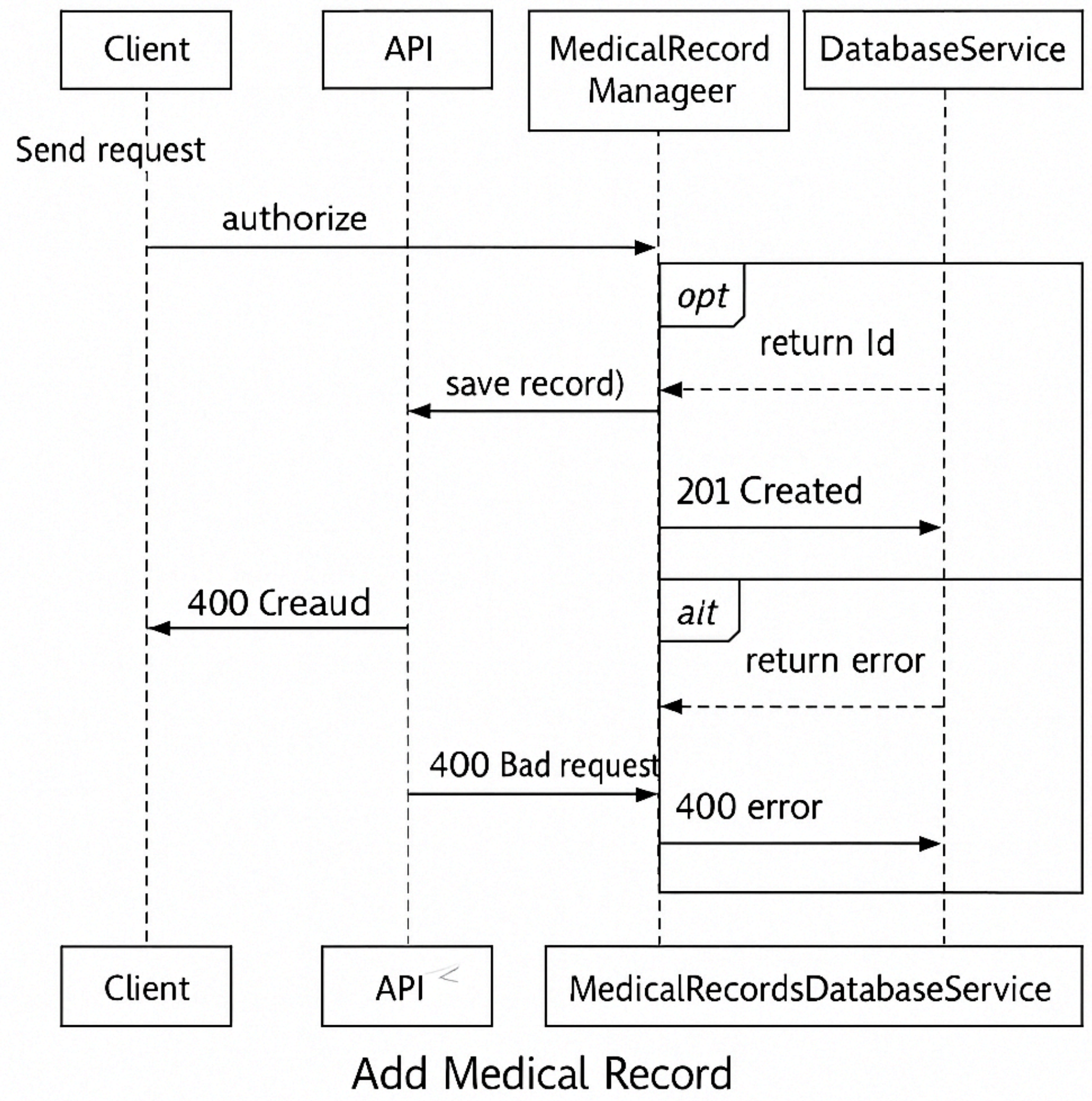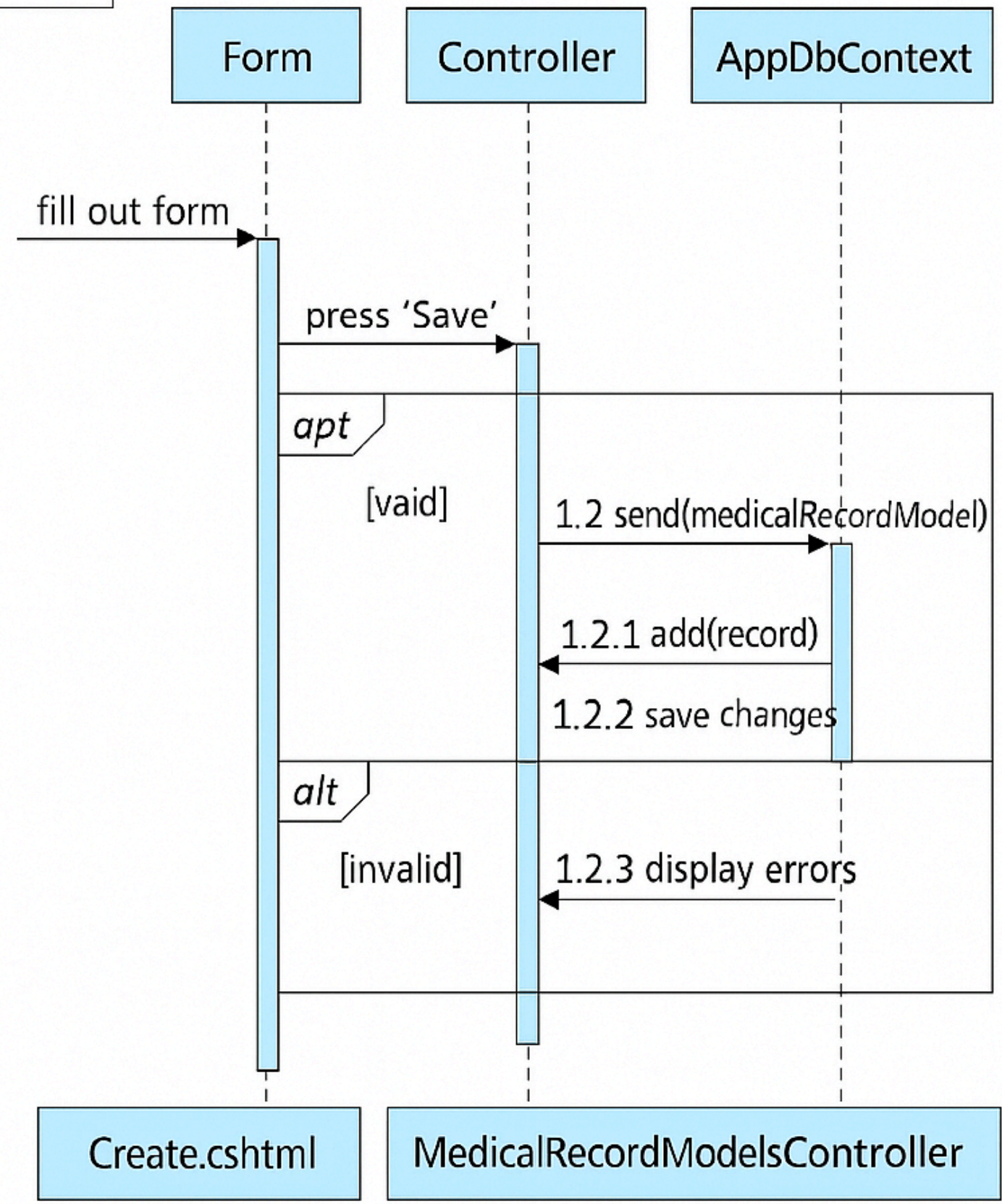 message
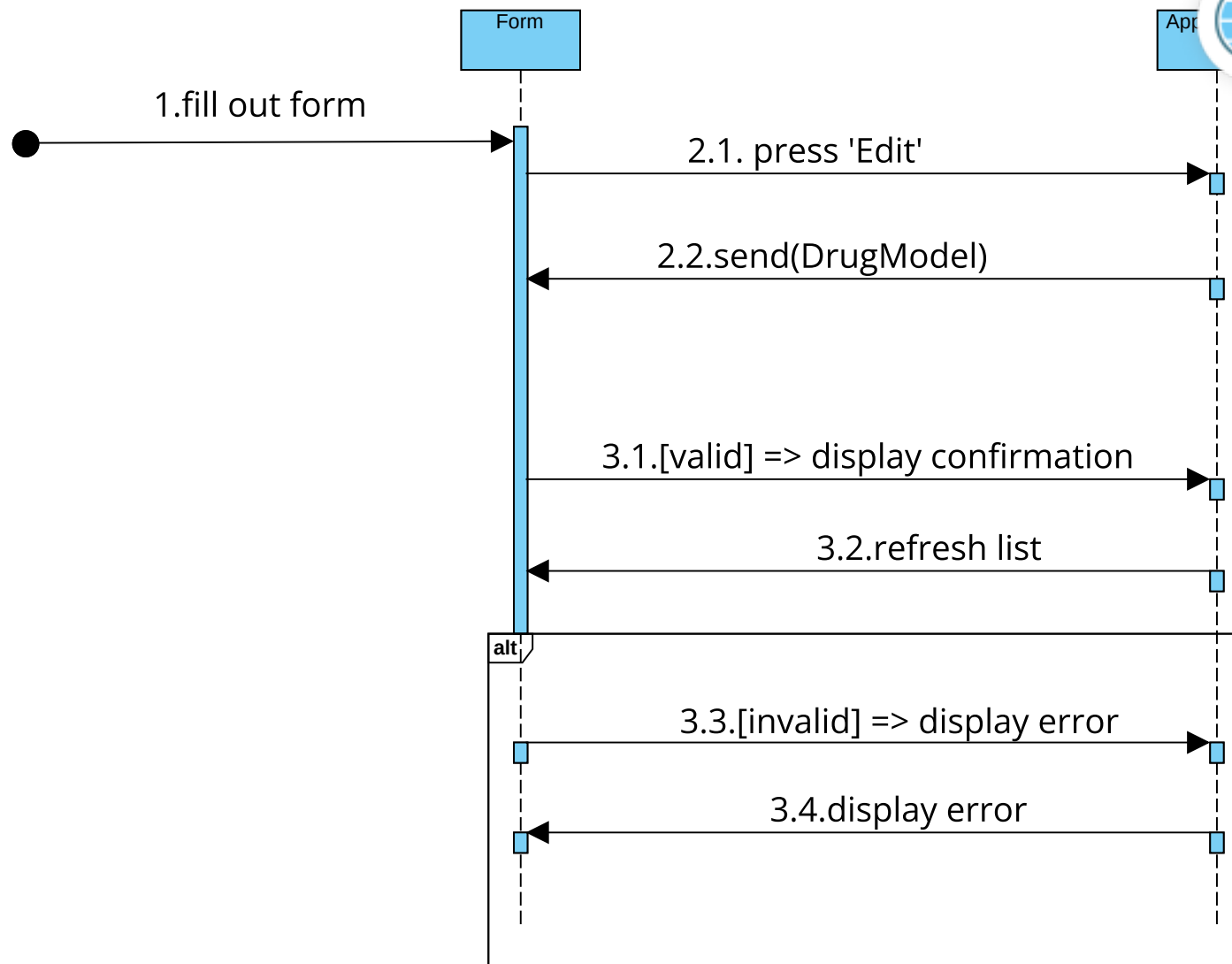- refreshes the drug list

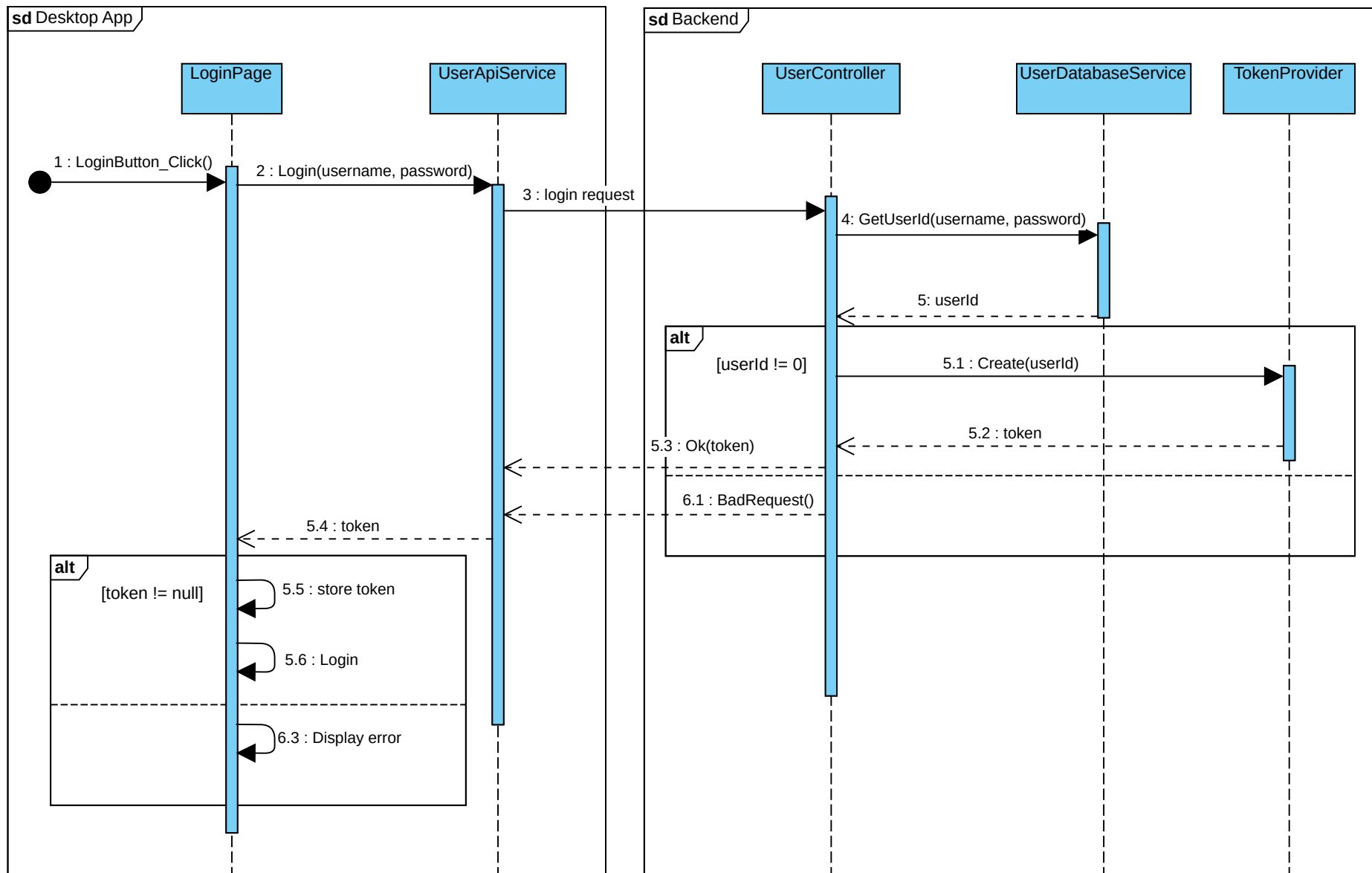Otherwise:
- displays an error message

– Form

Form

Application

fill out form

presss 'Save'

1.2 send(*DrugModel*)

opt

[valid]
display confirmation

1.2.2 refresh list

alt

[invalid]
display error

1.2.3 display error

Add Medical Record

Edit Drug Use Case

Form

App

1.fill out form

2.1. press 'Edit'

2.2.send(DrugModel)

3.1.[valid] => display confirmation

3.2.refresh list

alt

3.3.[invalid] => display error

3.4.display error

sd Desktop App

sd Backend

LoginPage | UserApiService | UserController | UserDatabaseService | TokenProvider

1 : LoginButton_Click()

2 : Login(username, password)

3 : login request

4: GetUserId(username, password)

5: userId

alt

[userId != 0]

5.1 : Create(userId)

5.2 : token

5.3 : Ok(token)

6.1 : BadRequest()

5.4 : token

alt

[token != null]

5.5 : store token

5.6 : Login

6.3 : Display error

Add Appointment: Use Case

— **Main scenario**

The User specifies appointment
details on the Add Appointment
Page and presses the Save
button.

The system validates
appointment details.

If details are valid, the system
saves the appointment to the
database.

The system displays
a confirmation message

— **Alternate path:**

If the appointment, details are
incomplete or invalid, the system
displays an error message

User	Add Appointment Page	System	Database

1:save()

1; validateAppointment()

[valid]

val.[d]

1:2: saveToDatabase()

1.3: displayEror()

1.3: displayConfirmation)