Quinn Rentz

# Final Submission: MedAdherence

https://github.com/qbrentz/CMSC355-MedAdherenceProj
https://vcu.mediaspace.kaltura.com/media/Final+Submission+Video+Part/1_em63dav4

## Updates and Progress:

In the final weeks of this course I have been able to polish off a lot of the rough edges on the MedAdherence App. The Prescription, Patient, and Medication Log objects are nearly fully implemented in relation to the front end feature set. Prescriptions are able to be modified and created, Patients are able to change their usernames and emails, and Medication Logs are finally mostly functional. The Notifications feature is still not accessible, though I have begun the process of building those out. The backend functions are working for testing purposes, but accessing those, generating notifications for each patient, and storing them in the database for distribution is currently not functional. Below, I have outlined the progress I have made in each of the Class categories.

### Current State of Issues with Prescription Model, Services, and Controller:

The Prescription structure is arranged in a Many-to-One relation to the Patient Table by foreign key to the PatientID. When making calls to getPatient, this would cause an endless loop, wherein the Patient calls getPrescription, to display them in a list. This then calls back to the getPatient method, causing a stack overflow. Thankfully, I was able to leverage the JSon BackReference and ManageReference libraries in order to prevent this behavior. This functionality is also implemented in the relations between Prescription, Patient, and Medication Log. I had intended to alter the formatting of the “schedule” attribute of the Prescription model to allow for the Notifications model to check the submitted Log timestamp against the Prescription’s assigned “Schedule” value, but was unable to finalize this functionality before the final submission was due.

### Current State of Issues with Medication Log:

Medication Log has been integrated and is functional, albeit with some caveats. The medication logs store the timestamp of the log, as well as the patient and prescription ID’s, which act as foreign keys to those tables. These logs are visible through a designated History page in the front-end, which allow a user to see what medications they have taken, and when. When a medication is logged, the inventory of the prescription is decreased by one, and is reflected in the prescription display table as well as the database.

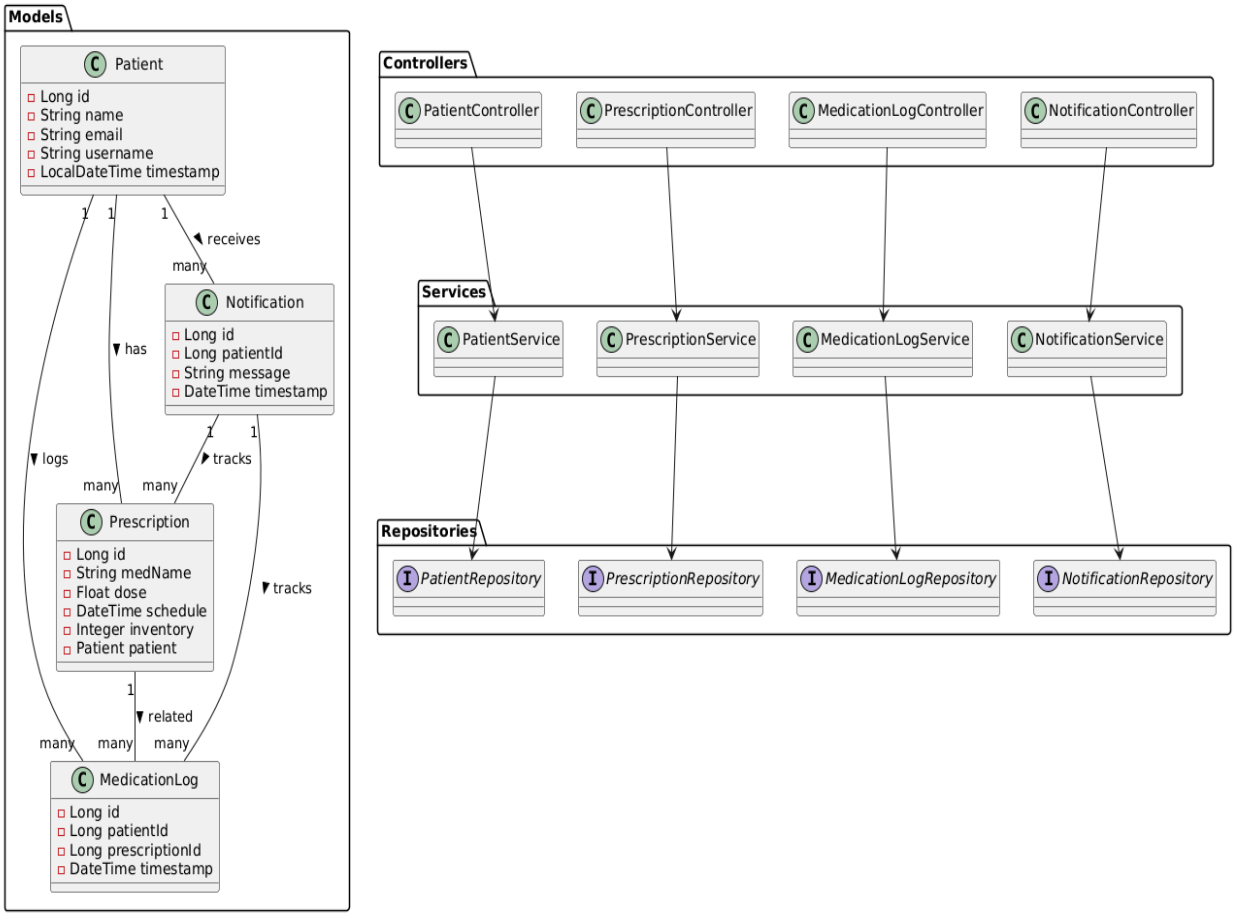
Quinn Rentz

### Continued Lack of Notifications:

Due to the lack of support or presence from my team, I was unable to even begin to implement the proposed notification features. As of Final Submission, the notifications feature is not implemented.

## UML Diagrams:

The diagrams below reflect the intended relations for the Models, Services and Controllers for each Class:



Quinn Rentz

# Previous Test Case planning details:

## 1. Overview

The medadherenceproject is an application designed to manage patient medication adherence. The application includes entities such as Patient, Prescription, MedicationLog, and Notification. This document outlines the test cases for the following layers:

- Controllers: REST API endpoints.
- Services: Business logic.
- Models: ensure that the models behave as expected.

---

## 2. Test Categories

### 2.1 Controller Tests

Controller tests ensure that the REST API endpoints behave as expected. These tests use Mockito to mock service layers and validate HTTP responses.

#### 2.1.1 PatientController

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PC-001 | Get all patients | Returns a list of all patients with HTTP 200 status. |
| TC-PC-002 | Add a new patient | Returns the created patient with HTTP 200 status. |
| TC-PC-003 | Get a patient by ID (exists) | Returns the patient with HTTP 200 status. |
| TC-PC-004 | Get a patient by ID (does not exist) | Returns HTTP 404 status. |
| TC-PC-005 | Delete a patient by ID (exists) | Returns success message with HTTP 200 status. |
| TC-PC-006 | Delete a patient by ID (does not exist) | Returns HTTP 404 status. |

#### 2.1.2 PrescriptionController

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PRC-001 | Get all prescriptions | Returns a list of all prescriptions with HTTP 200 status. |

| | | |
|---|---|---|
| TC-PRC-002 | Add a new prescription | Returns the created prescription with HTTP 200 status. |
| TC-PRC-003 | Get a prescription by ID (exists) | Returns the prescription with HTTP 200 status. |
| TC-PRC-004 | Get a prescription by ID (does not exist) | Returns HTTP 404 status. |
| TC-PRC-005 | Delete a prescription by ID (exists) | Returns success message with HTTP 200 status. |
| TC-PRC-006 | Delete a prescription by ID (does not exist) | Returns HTTP 404 status. |

### 2.1.3 MedicationLogController

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-MLC-001 | Get all medication logs | Returns a list of all medication logs with HTTP 200 status. |
| TC-MLC-002 | Add a new medication log | Returns the created medication log with HTTP 200 status. |
| TC-MLC-003 | Get a medication log by ID (exists) | Returns the medication log with HTTP 200 status. |
| TC-MLC-004 | Get a medication log by ID (does not exist) | Returns HTTP 404 status. |
| TC-MLC-005 | Delete a medication log by ID (exists) | Returns success message with HTTP 200 status. |
| TC-MLC-006 | Delete a medication log by ID (does not exist) | Returns HTTP 404 status. |

### 2.1.4 NotificationController

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-NC-001 | Get all notifications | Returns a list of all notifications with HTTP 200 status. |
| TC-NC-002 | Add a new notification | Returns the created notification with HTTP 200 status. |
| TC-NC-003 | Get a notification by ID (exists) | Returns the notification with HTTP 200 status. |
| TC-NC-004 | Get a notification by ID (does not exist) | Returns HTTP 404 status. |
| TC-NC-005 | Delete a notification by ID (exists) | Returns success message with HTTP 200 status. |

| | | |
|---|---|---|
| TC-NC-006 | Delete a notification by ID (does not exist) | Returns HTTP 404 status. |

---

## 2.2 Service Tests

Service tests validate the business logic of the application. These tests use Mockito to mock repository layers.

### 2.2.1 PatientService

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PS-001 | Get all patients | Returns a list of all patients. |
| TC-PS-002 | Add a new patient | Saves and returns the created patient. |
| TC-PS-003 | Get a patient by ID (exists) | Returns the patient. |
| TC-PS-004 | Get a patient by ID (does not exist) | Returns null. |
| TC-PS-005 | Update a patient by ID (exists) | Updates and returns the patient. |
| TC-PS-006 | Update a patient by ID (does not exist) | Returns null. |
| TC-PS-007 | Delete a patient by ID (exists) | Deletes the patient and returns true. |
| TC-PS-008 | Delete a patient by ID (does not exist) | Returns false. |

### 2.2.2 PrescriptionService

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PRS-001 | Get all prescriptions | Returns a list of all prescriptions. |
| TC-PRS-002 | Add a new prescription | Saves and returns the created prescription. |
| TC-PRS-003 | Get a prescription by ID (exists) | Returns the prescription. |
| TC-PRS-004 | Get a prescription by ID (does not exist) | Returns null. |
| TC-PRS-005 | Update a prescription by ID (exists) | Updates and returns the prescription. |
| TC-PRS-006 | Update a prescription by ID (does not exist) | Returns null. |
| TC-PRS-007 | Delete a prescription by ID (exists) | Deletes the prescription and returns true. |

| | | |
|---|---|---|
| TC-PRS-008 | Delete a prescription by ID (does not exist) | Returns false. |

### 2.2.3 MedicationLogService

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-MLS-001 | Get all medication logs | Returns a list of all medication logs. |
| TC-MLS-002 | Add a new medication log | Saves and returns the created medication log. |
| TC-MLS-003 | Get a medication log by ID (exists) | Returns the medication log. |
| TC-MLS-004 | Get a medication log by ID (does not exist) | Returns null. |
| TC-MLS-005 | Delete a medication log by ID (exists) | Deletes the medication log and returns true. |
| TC-MLS-006 | Delete a medication log by ID (does not exist) | Returns false. |

### 2.2.4 NotificationService

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-NS-001 | Get all notifications | Returns a list of all notifications. |
| TC-NS-002 | Add a new notification | Saves and returns the created notification. |
| TC-NS-003 | Get a notification by ID (exists) | Returns the notification. |
| TC-NS-004 | Get a notification by ID (does not exist) | Returns null. |
| TC-NS-005 | Delete a notification by ID (exists) | Deletes the notification and returns a success message. |
| TC-NS-006 | Delete a notification by ID (does not exist) | Returns a "not found" message. |

---

## 3. Model Tests

The Model Tests focus on validating the behavior of the entities in the medadherenceproject. These tests ensure that the models behave as expected, including field validation, relationships, and default values.

### 3.1 Patient Model Tests

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PM-001 | Create a patient with valid data | Successfully creates a patient object. |
| TC-PM-002 | Create a patient with missing required fields | Throws a validation error for missing fields (name, email). |
| TC-PM-003 | Create a patient with duplicate email | Throws a unique constraint violation error. |
| TC-PM-004 | Verify default create_time value | create_time is automatically set to the current timestamp. |

---

### 3.2 Prescription Model Tests

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-PRM-001 | Create a prescription with valid data | Successfully creates a prescription object. |
| TC-PRM-002 | Create a prescription with missing required fields | Throws a validation error for missing fields (medication_name, start_date). |
| TC-PRM-003 | Verify relationship with Patient | Prescription is correctly associated with a Patient. |

---

### 3.3 MedicationLog Model Tests

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-MLM-001 | Create a medication log with valid data | Successfully creates a medication log object. |
| TC-MLM-002 | Create a medication log with missing required fields | Throws a validation error for missing fields (prescription_id). |
| TC-MLM-003 | Verify relationship with Prescription | Medication log is correctly associated with a Prescription. |

---

### 3.4 Notification Model Tests

| Test Case ID | Test Description | Expected Outcome |
|---|---|---|
| TC-NM-001 | Create a notification with valid data | Successfully creates a notification object. |
| TC-NM-002 | Create a notification with missing required fields | Throws a validation error for missing fields (message, patient_id). |
| TC-NM-003 | Verify relationship with Patient | Notification is correctly associated with a Patient. |

---

## 4. Tools and Frameworks

- VSCode: For writing and running software and tests.
- Mockito: For mocking dependencies in service and controller tests.
- Spring Boot Test: For integration testing.
- Hibernate Validator: For validating entity constraints.
- H2 Database: For testing relationships and cascading behavior.
- PostgreSQL: For Database management.