CS411 Project Design

Team: Justin Wang, Eshaan Jalali, Jason Sandoval, Jonathan Xu, and Zhao Pinhan

Section 1 (10 points):

Select five functional requirements from your requirements document that your team will move forward with. Create 5 use case scenarios for these requirements. Document these scenarios the use case specification format discussed in class.

Answer:

***We argue that Registration and Login are core parts of our system and are not necessarily as simplistic due to HIPPA regulations and requirements with data and user PII.**

Use Case 1: Register & Verify Account

Primary Actors: Patient or Provider

Supporting Actors: Authentication Service, Email Service

Main Success Scenario:

1. User selects “Create Account.”
2. System prompts for name, email, and password.
3. User submits required information.
4. System creates a pending account and sends a verification email with a time-limited link.
5. User opens the verification link.
6. System verifies the email and activates the account.

Extensions (Alternate & Error):

- 2a. **Weak password provided:** System displays password requirements; user resubmits.
- 3a. **Email already in use:** System displays “account exists” and offers login or password reset.
- 4a. **Verification email not received:** User selects “Resend verification”; system sends a new link.
- 5a. **Expired or invalid link:** System displays error and offers to resend a new verification email.

Use Case 2: Log In to Account

Primary Actors: Patient or Provider

Supporting Actors: Authentication Service

Main Success Scenario:

1. User selects “Log In.”
2. System prompts for email and password.
3. User submits credentials.
4. System authenticates the user.
5. System opens the user dashboard.

Extensions (Alternate & Error):

- 3a. **Incorrect credentials:** System shows error and re-prompts for login.
- 3b. **Unverified email:** System blocks access, displays “Verify your email,” and offers “Resend verification.”
- 3c. **Forgot password:** User selects “Forgot Password”; system emails a time-limited reset link; user sets a new password and returns to Step 1.

Use Case 3: Search for Providers

Primary Actors: Patient

Supporting Actors: Providers

Main Success Scenario:

1. Call **Log In to Account**
2. Patient opens the provider search page.
3. System displays search inputs (i.e. specialty, location, insurance).
4. Patient enters specialty, selects geographic area, and chooses accepted insurance.
5. Patient submits the search.
6. System returns a list of matching providers within performance targets.
7. (Optional) Patient refines filters and views updated results.

Extensions (Alternate & Error):

- 6a. **No results:** System displays “No matching providers” and suggests relaxing filters.
- 7a. **Network error:** System displays a retry option; patient retries search.
- 7b. **Large result set:** System paginates and provides sort options (e.g., pg 1, pg 2).

Use Case 4: Consult AI Chatbot

Primary Actors: Patient

Supporting Actors: LLM Service

Main Success Scenario:

1. Call **Log In to Account**
2. Patient navigates to “AI Chatbot.”
3. System displays the chatbot interface with a clear disclaimer (no diagnosis/treatment).
4. Patient describes symptoms or care need in natural language.
5. System analyzes the input.
6. System recommends resources, possible causes, and next steps.
7. Patient selects a recommended specialty.
8. System redirects to the search page prefilled with the selected specialty and the patient’s location/insurance (if available).
9. Call **Search for Providers**

Extensions (Alternate & Error):

- 1a. **Patient not authenticated:** System prompts login/registration; upon success, returns to the chatbot (Step 1).
- 2a. **Patient acknowledges disclaimer requirement:** System requires confirmation before continuing.
- 3a. **Insufficient detail provided:** System prompts for clarifying information (e.g., duration, severity, body area).
- 5a. **Ambiguous recommendation:** System offers multiple specialty options with brief rationales.
- 5b. **Out-of-scope or urgent/emergency cues detected:** System displays safety guidance (e.g., “Call emergency services or visit the nearest ER”) and halts recommendations.
- 7a. **Missing location or insurance data:** System prompts patient to add or confirm these fields before redirecting to search.
- 2a. Error: **Service unavailable:** System shows a courteous error message and suggests using manual search (link to provider search page).

Use Case 5: Appointment Requesting

Primary Actors: Patients, Doctors

Supporting Actors: Clinic front-desk, Email notification vendor

Main Success Scenario:

1. The patient clicks a specific doctor card to view details.
2. The patient clicks **Request an Appointment**.
3. The system opens an appointment form with required fields: contact methods, preferred time window, and reason for visit.
4. The patient completes the form and clicks **Submit**.
5. The system validates the input.
6. The system creates an appointment request with status **Initiated**, showing an on-screen confirmation to the patient.
7. The system notifies the doctor through email notification.

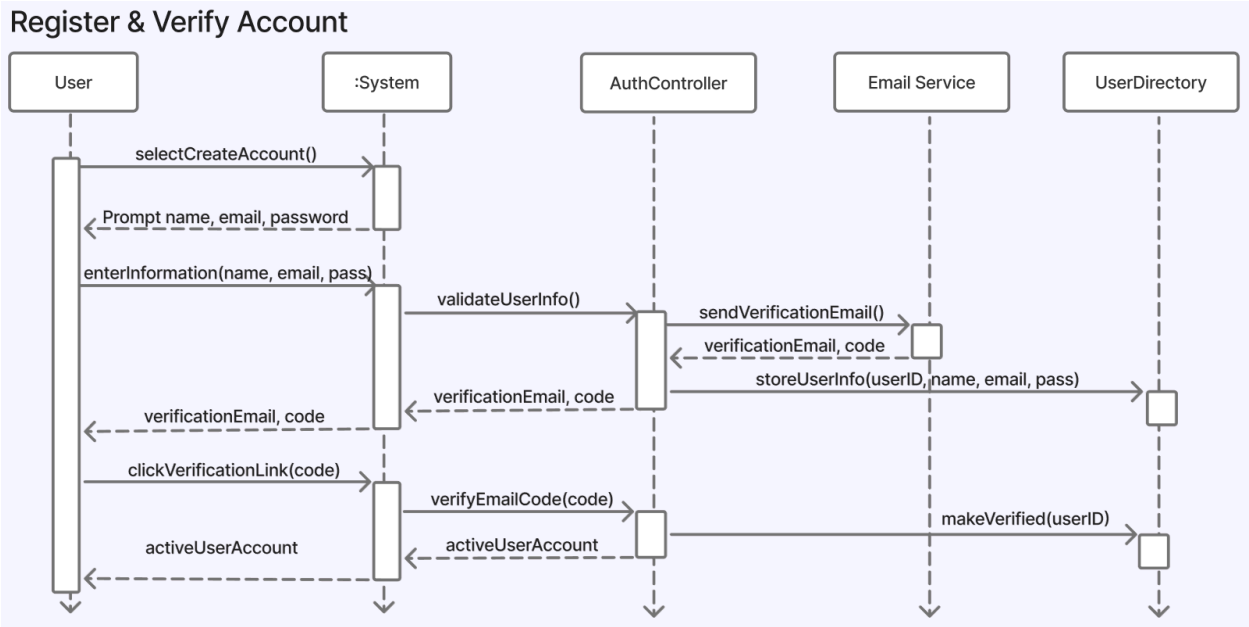
Extensions (Alternate & Error):

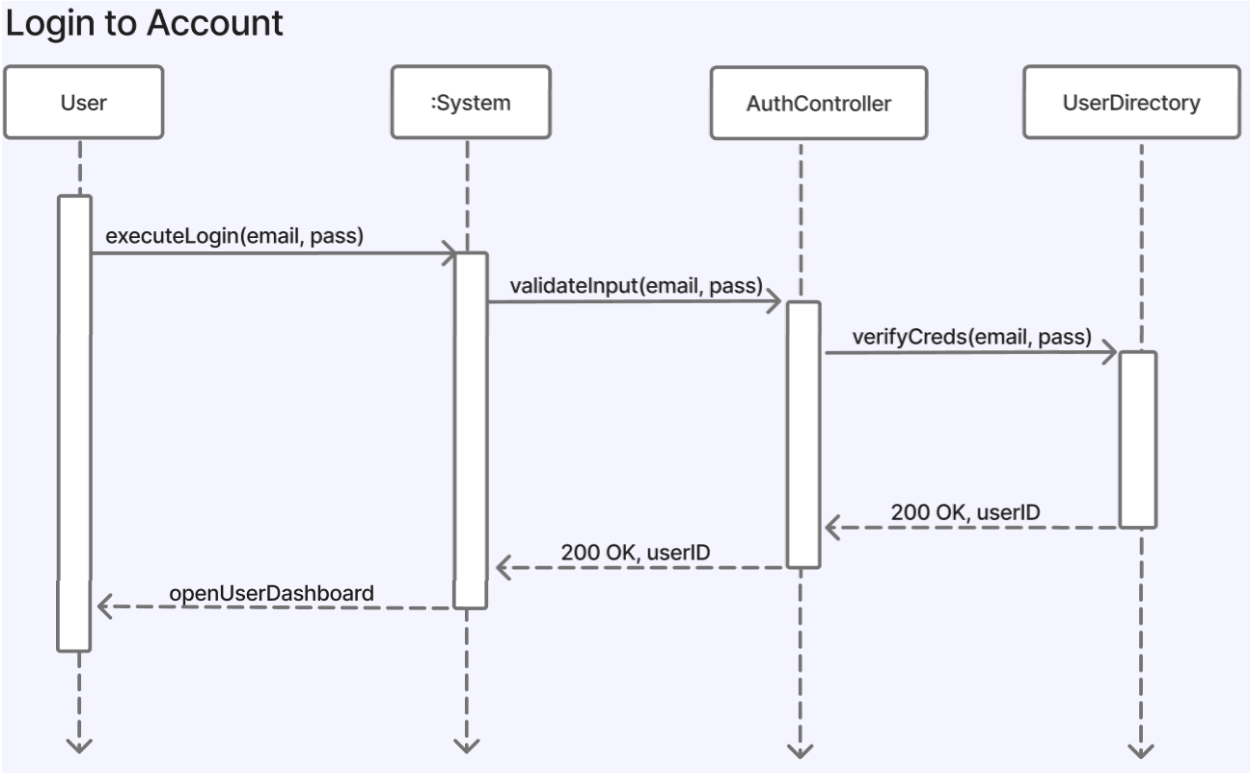
- 0. **Patient not authenticated**: The system redirects to the Login/Sign-up page and then returns to the same doctor details page.
- 1a. **Doctor no longer available**: The system shows “Doctor unavailable” banner and disables appointment actions. Offers **Return to Search** or **Similar Doctors** options.
- 1b. **Doctor not accepting new patients**: The system shows the **Join the Waitlist** option.
- 3a. **Time-zone mismatch detected**: The system clears the input. Pops with an error and asks the patient to enter another valid time-zone according to the Doctor's available time.
- 4a. **Required fields missing**: The system displays inline errors and prevents submitting the form.
- 4b. **Duplicate pending request**: The system warns and offers: view existing requests, continue anyway or cancel.
- 6a. **Patient cancels at confirmation screen**: The system offers **Discard** option.
- 7a. **Email bounce/SMTP failure**: The system informs the patient that the doctor has been alerted via an alternating channel.

Section 2 (10 points):

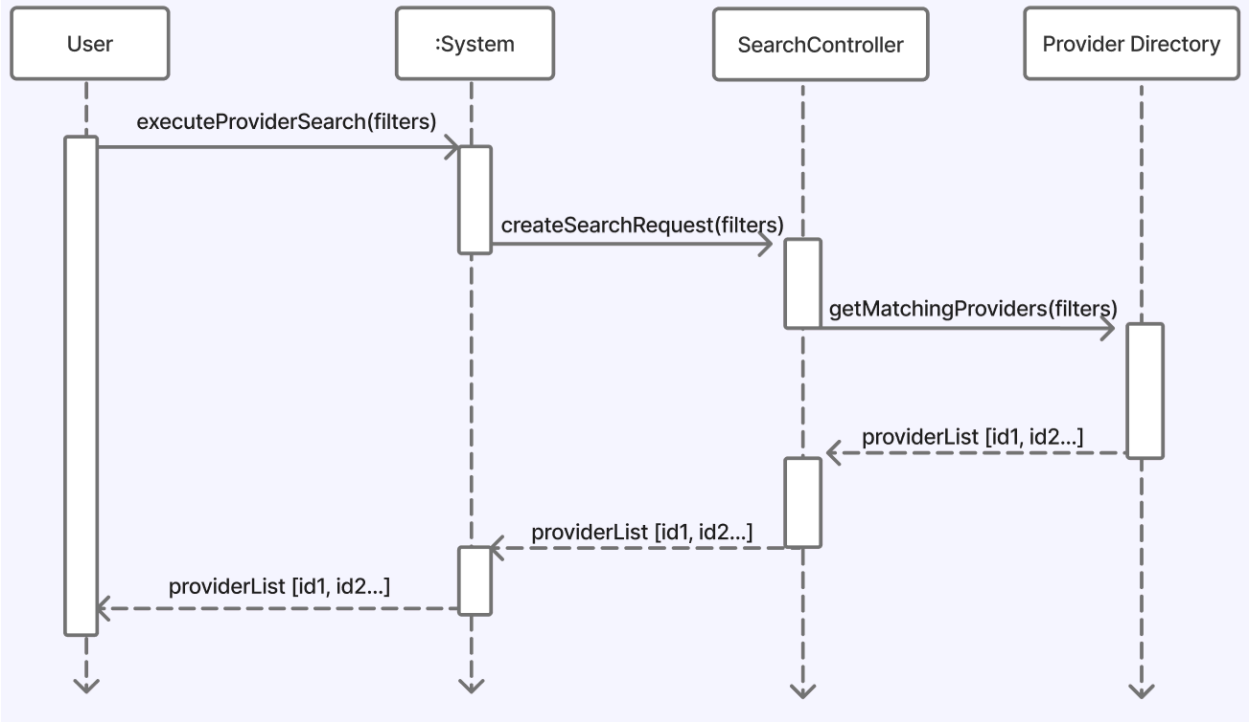
System Sequence Diagram (SSD): Create 5 System Sequence Diagrams that illustrate the main success scenarios of the selected use cases. The SSDs should clearly depict the interactions between the actor(s) and the system, showcasing the sequence of messages exchanged during the use case execution.

Answer:

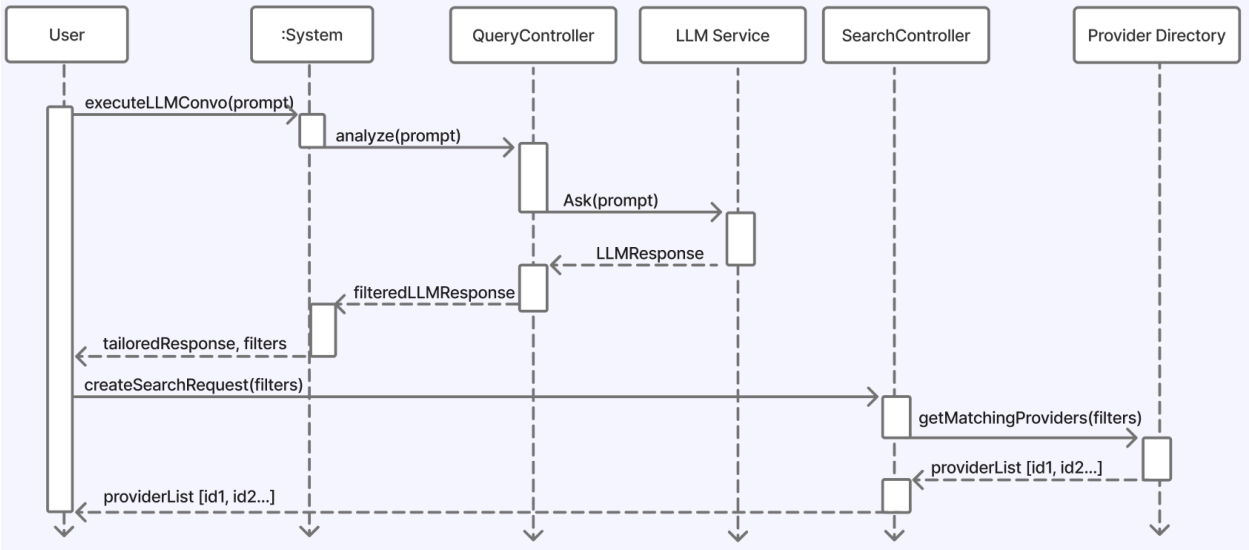


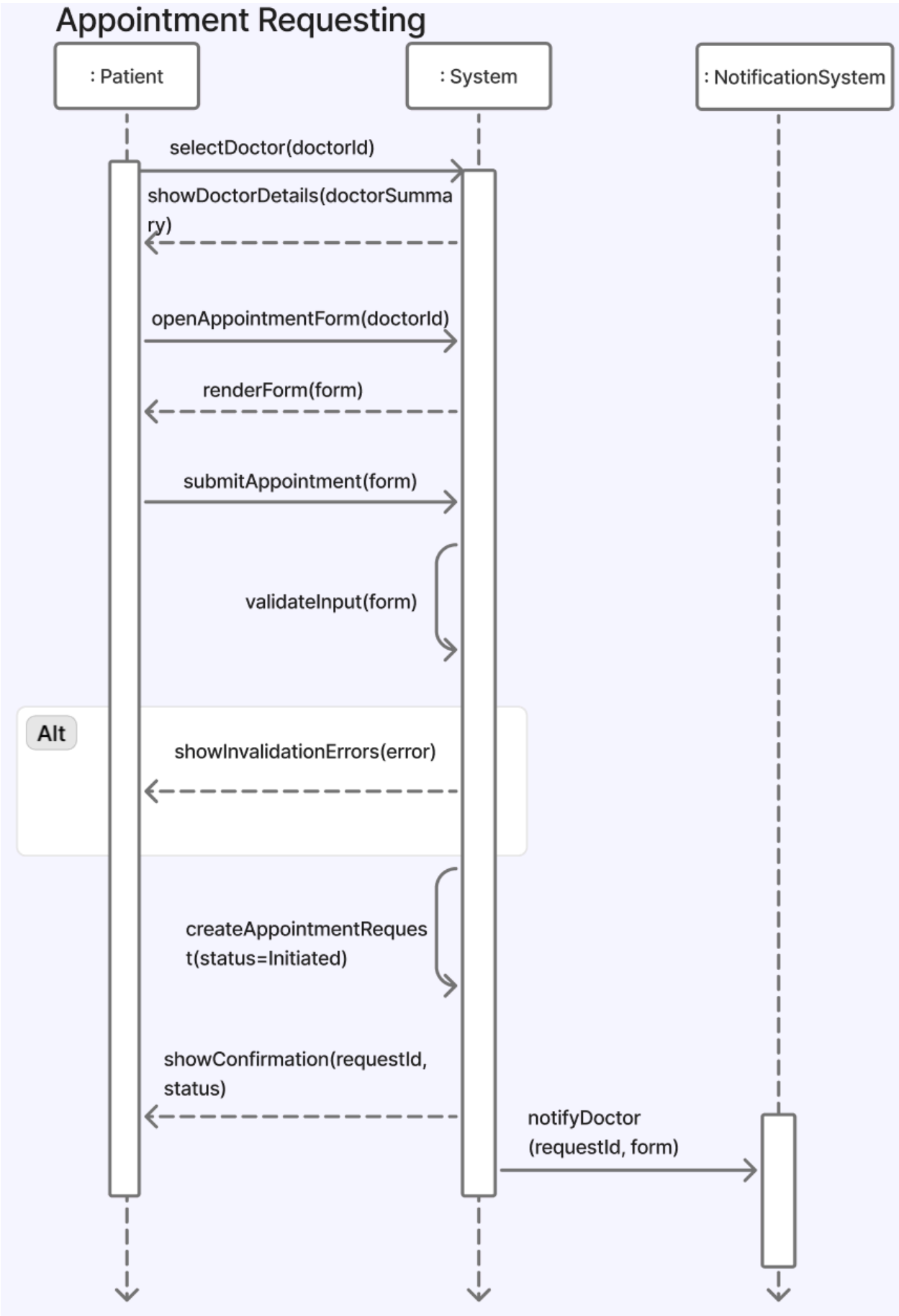


Search for Providers



Consult AI Chatbot

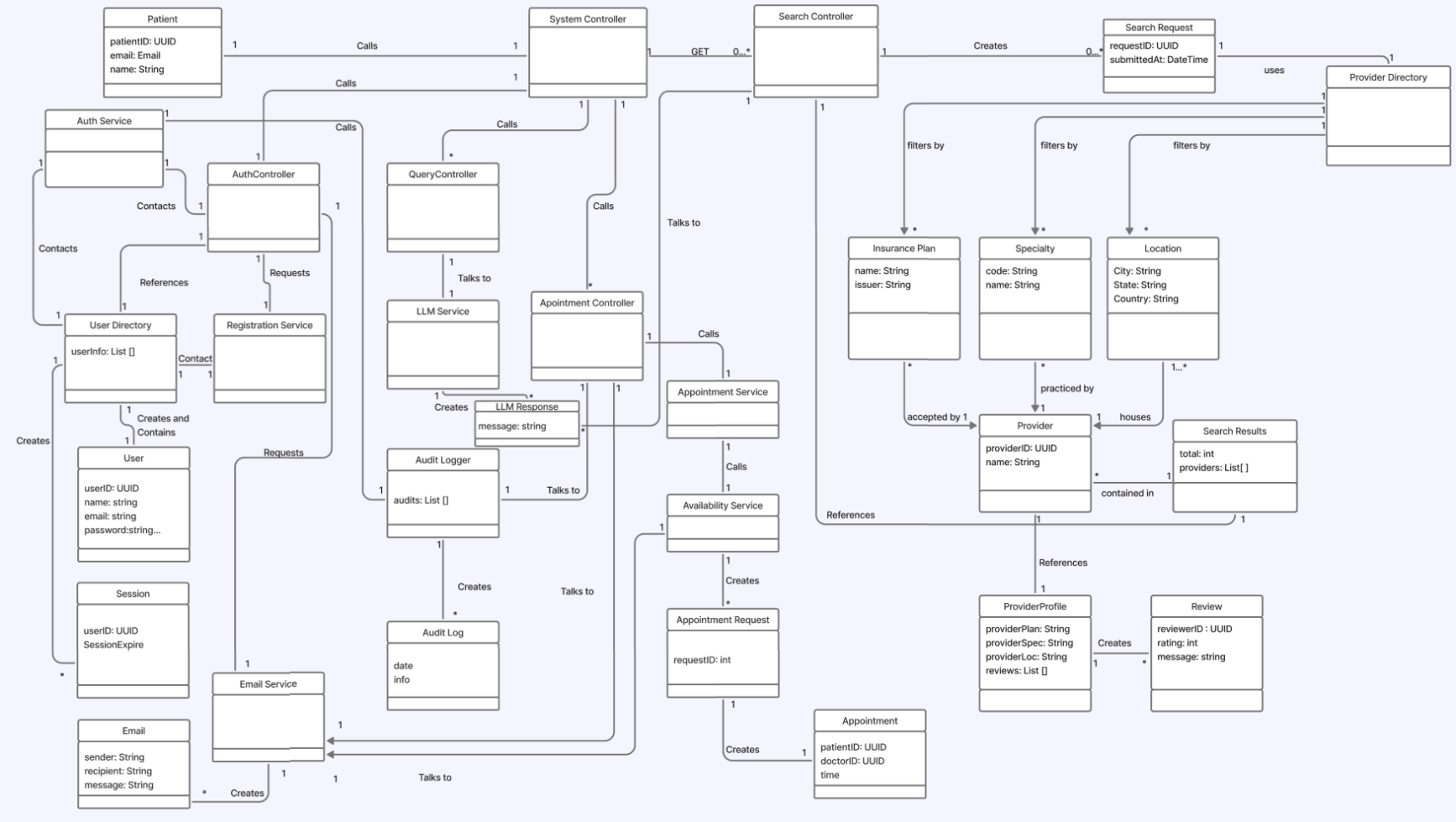



Appointment Requesting
: Patient
: System
: NotificationSystem
selectDoctor(doctorId)
showDoctorDetails(doctorSummary)
openAppointmentForm(doctorId)
renderForm(form)
submitAppointment(form)
validateInput(form)
Alt
showInvalidationErrors(error)
createAppointmentRequest(status=Initiated)
showConfirmation(requestId, status)
notifyDoctor (requestId, form)

Section 3 (10 points):

Domain Model: Develop one domain model that encompasses the fundamental concepts and associations relevant to the chosen use case scenarios. The domain model should include all significant concepts, their attributes along with data types, and associations. Each association within the domain model should be well-defined with a name and multiplicity.

Answer:



Section 4 (15 points):

Operation Contracts: Based on the SSDs, write different operation contracts for the main significant operations (at least five) using the template discussed in class. In the post-conditions section of the operation contract, explicitly describe any of the following: the creation of an object instance, the formation of an association, or the modification of an attribute.

Answer:

Name: selectCreateAccount(userEmail:String, userPass:String)

Responsibilities: Allow a new user (patient or provider) to create an account using an email and password, initialize verification status as unverified, and trigger a verification email.

Type: System

Cross References:

- Use Case 1 – Register & Verify Account

Notes:

- The system checks for duplicate accounts and stores a hashed password and email in the UserDirectory.

Exceptions:

- Email already exists → Error message “Account exists.”
- Weak password → Prompt requirements.
- Verification email delivery failed → Retry or show “Email could not be sent.”

Output:

- Account creation confirmation and verification email sent to user.

Pre-conditions:

- Email and password inputs are valid and well-formed.
- No existing account with the same email exists.

Post-conditions:

- The user is **authenticated** and **logged in**.
- A **session** is established.
- The user is taken to their dashboard.

Instance creation:

- UserAccount object is created by UserDirectory (via createUser(email, passHash) call).
- VerificationToken object is created by RegistrationService for email confirmation.

Associations formed:

- UserAccount is added to UserDirectory (UserDirectory → UserAccount).
- VerificationToken is linked to UserAccount for future email verification.

Attribute modification:

- UserAccount.isVerified := false
- UserAccount.createdAt := now
- UserAccount.password := hashed(userPass)

Operation Contract 2:

Name: executeLogin(email:String, password:String)

Responsibilities: Authenticate a registered and verified user (patient or provider) and grant access to their dashboard.

Type: System

Cross References:

- Use Case 2 – Log In to Account

Notes:

- Successful logins create a session token and log the event in the audit trail.

Exceptions:

- Invalid credentials → Error “Incorrect email or password.”
- Unverified email → Prompt to verify account.
- Account locked after too many failed attempts.

Output:

- Session token and redirect to user dashboard.

Pre-conditions:

- The user’s account exists and isVerified = true.

Post-conditions:

- A new account is created and stored.
- Account is marked **unverified**.
- A verification email is sent to the user.

Instance creation:

- A new Session object is created by AuthService through createSession(userID).
- An AuditLog record is created by AuditLogger containing userID, success, timestamp.

Associations formed:

- Session.userID is linked to the corresponding UserAccount.
- AuditLog.userID is linked to the same UserAccount.

Attribute modification:

- UserAccount.lastLogin := currentTimestamp.

- UserAccount.failedAttempts := 0 (reset after success).

Operation Contract 3:

Name: executeProviderSearch(specialty:String, location:String, insurancePlan:String)

Responsibilities: Execute a provider search based on user-selected filters (specialty, location, insurance) and return a list of matching providers that meet the criteria.

Type: System

Cross References:

- Use Case: Search for Providers

Notes:

- The operation records the search and returns matching verified providers that meet the specified filters.

Exceptions:

- If any filter input is invalid or missing, the system displays an error message.
- If no results are found, the system notifies the user and suggests relaxing filters.

Output:

- A list of providers with name, specialty, location, and insurance accepted.

Pre-conditions:

- The patient is logged in and known to the system.
- The selected specialty, insurance plan, and location exist in the system.

Post-conditions:

- The user is **authenticated** and **logged in**.
- A **session** is established.
- The user is taken to their dashboard.

Instance creation:

- A new SearchRequest object is created by SearchController (via createSearchRequest(filters)), containing the filters and timestamp.
- A new SearchResults object is created to store the matching providers.

Associations formed:

- SearchRequest is linked to the Patient (the one performing the search).
- SearchResults is linked to the SearchRequest through its requestId.
- Each Provider found in ProviderDirectory is associated with the SearchResults collection.

Attribute modification:

SearchRequest.status := "Completed".
SearchResults.count := numberOfMatchingProviders.

Operation Contract 4:

Name: executeLLMConvo(prompt:String)
Responsibilities: Allow a verified user to interact with the AI chatbot to receive general health guidance and provider recommendations (but no diagnoses).
Type: System
Cross References:

- Use Case 5 – Consult AI Chatbot

Notes:

- Uses LLM API for processing. Filters out PII and medical diagnoses. May prefill a provider search form based on recommended specialty.

Exceptions:

- AI service unavailable → Display "Chatbot temporarily unavailable."
- Inappropriate input (profanity, diagnostic queries) → Warn user and reject prompt.

Output:

- Structured response object containing recommended specialties, rationales, and safety disclaimer.

Pre-conditions:

- User is authenticated and verified.
- System has access to the LLM API.

Post-conditions:

- The prompt is processed and a **chat response** is returned.
- Recommendations and a **safety disclaimer** are provided.
- (Optional) A **prefilled provider search link** is provided based on a recommended specialty.

Instance creation:

- A new LLMResponse object is created by QueryController (via createResponse(data)) containing AI-generated recommendations, rationale, and disclaimer.
- (Optional) A new SearchRequest object is created by SearchController if the response includes a recommended specialty for prefill.

Associations formed:

- LLMResponse.user = UserAccount (the user who sent the prompt).

- (Optional) SearchRequest is linked to LLMResponse if prefill occurred.

Attribute modification:

- LLMResponse.timestamp := now.
- LLMResponse.status := "Completed".
- (Optional) SearchRequest.prefill := true if triggered from chatbot response.

Operation Contract 5:

Name: createAppointmentRequest(userId:String, time:DateTime, description:String, doctorId:String)

Responsibilities: Create a persistent appointment request object with status = “Initiated,” validate inputs, notify provider, and return a confirmation payload.

Type: System

Cross References:

- Use Case 4 – Book Appointment

Notes:

- Time normalized to UTC and doctor’s local time.
- Idempotency key prevents duplicates.
- All actions logged for audit.

Exceptions:

- Missing or invalid fields.
- Invalid time window (outside availability).
- Duplicate pending request.
- Unauthenticated patient.
- Network failure or notification error.

Output:

- CreateAppointmentRequestResults: requestId, status, doctorSummary, normalizedTime, nextActions, message.

Pre-conditions:

- Authenticated and authorized patient.
- doctorId exists and is bookable.
- Database connection available.

Post-conditions:

- A new AppointmentRequest instance is created and saved in the system
- The appointment status is initialized as *Initiated*

- A notification is sent to the selected provider
- The patient receives a confirmation of the appointment request.

Instance creation:

- AppointmentRequest is created with fields: requestId, patientId=userID, doctorId, requestedTime (normalized), description, status="Initiated", createdAt.
- (Optional, per EmailService step) A Notification/EmailMessage is created (queued) to inform the provider (and patient).

Associations formed:

- AppointmentRequest.patient → Patient(userID).
- AppointmentRequest.doctor → Doctor(doctorID).
- (Optional) Notification.subject → AppointmentRequest; Notification.recipients → {Doctor(doctorID), Patient(userID)}.

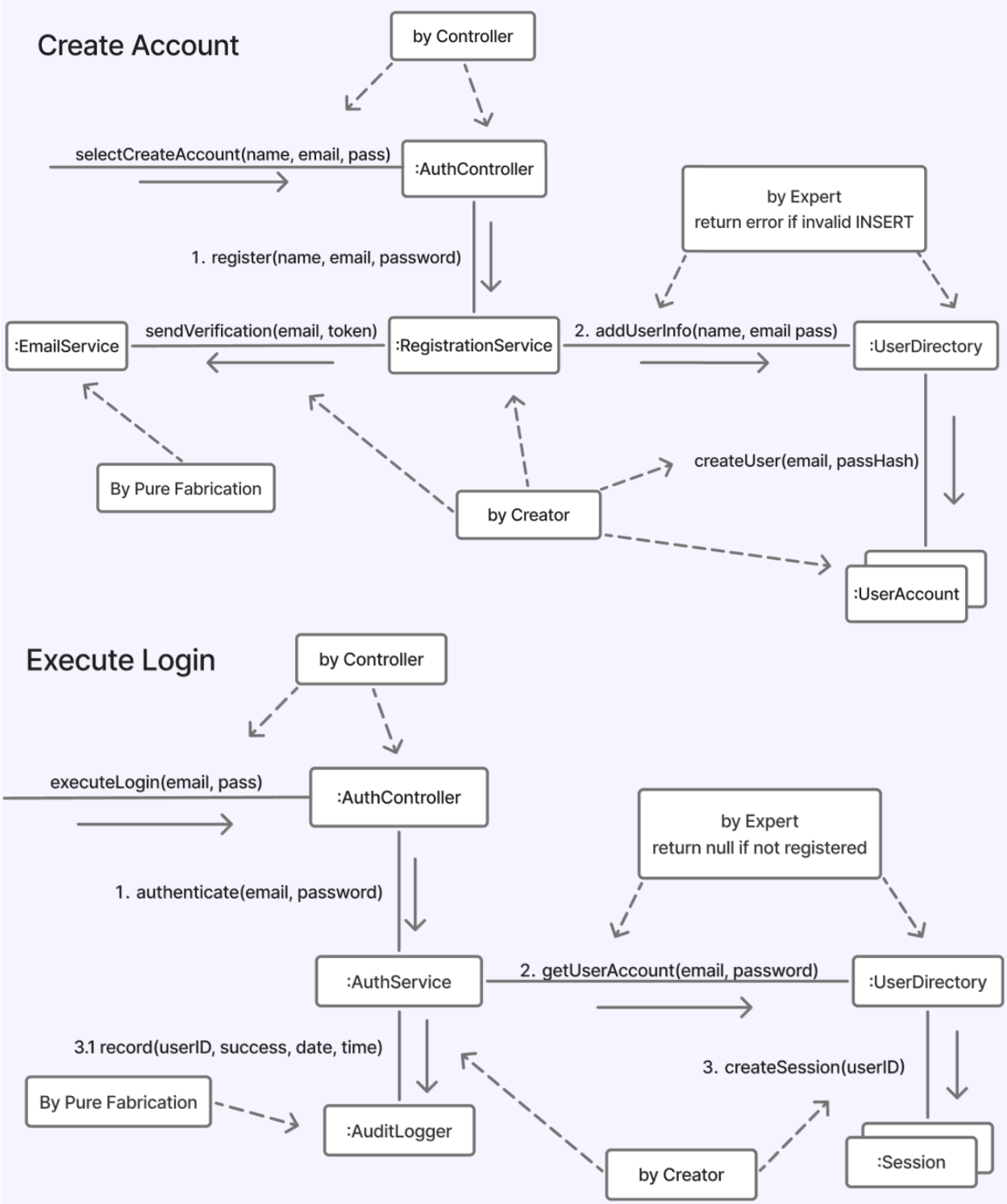
Attribute modification:

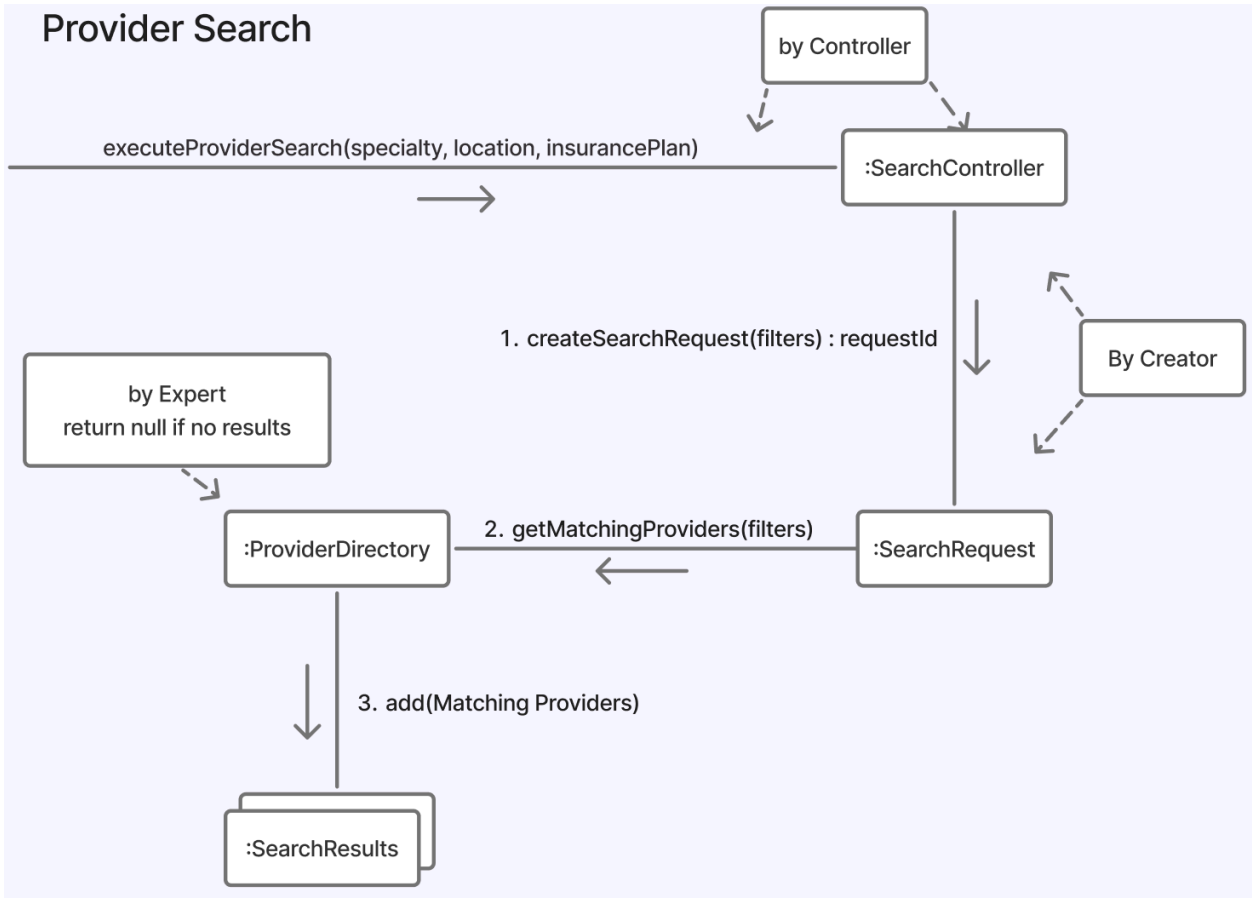
- AppointmentRequest.status := "Initiated" (on successful availability check).
- AppointmentRequest.requestedTime := normalized(time) (UTC/doctor local stored).
- AppointmentRequest.description := description.

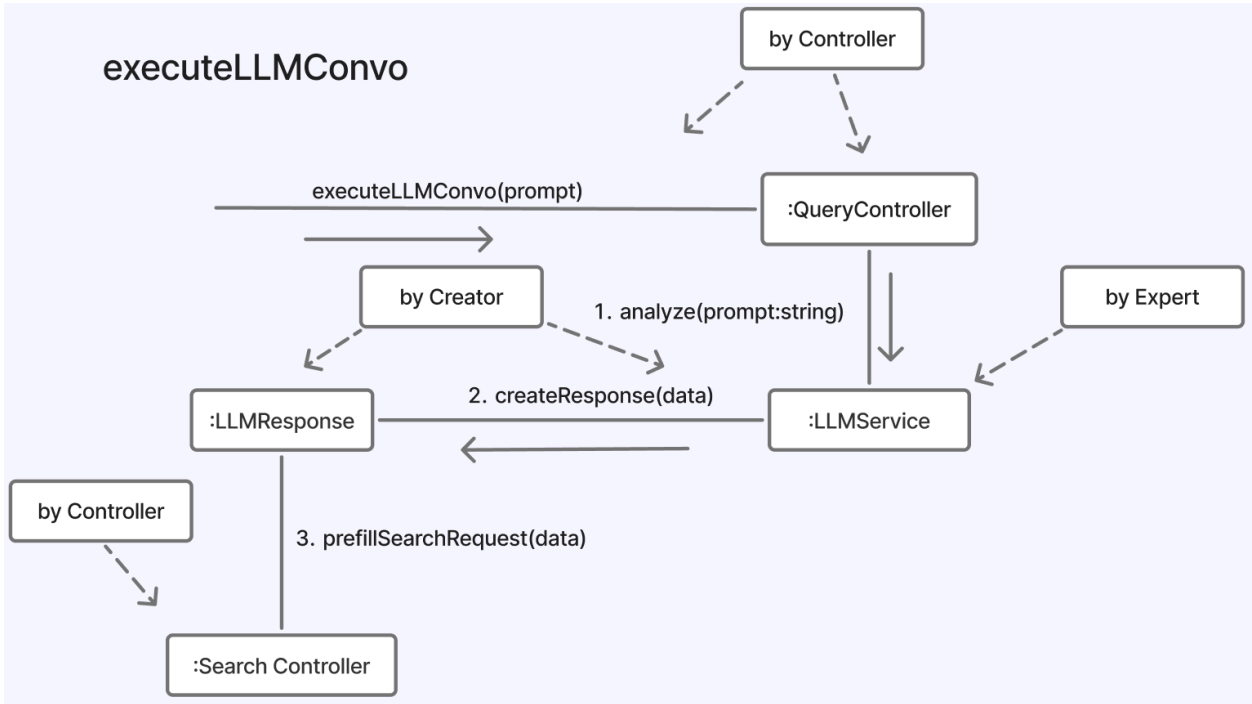
Section 5 (20 points):

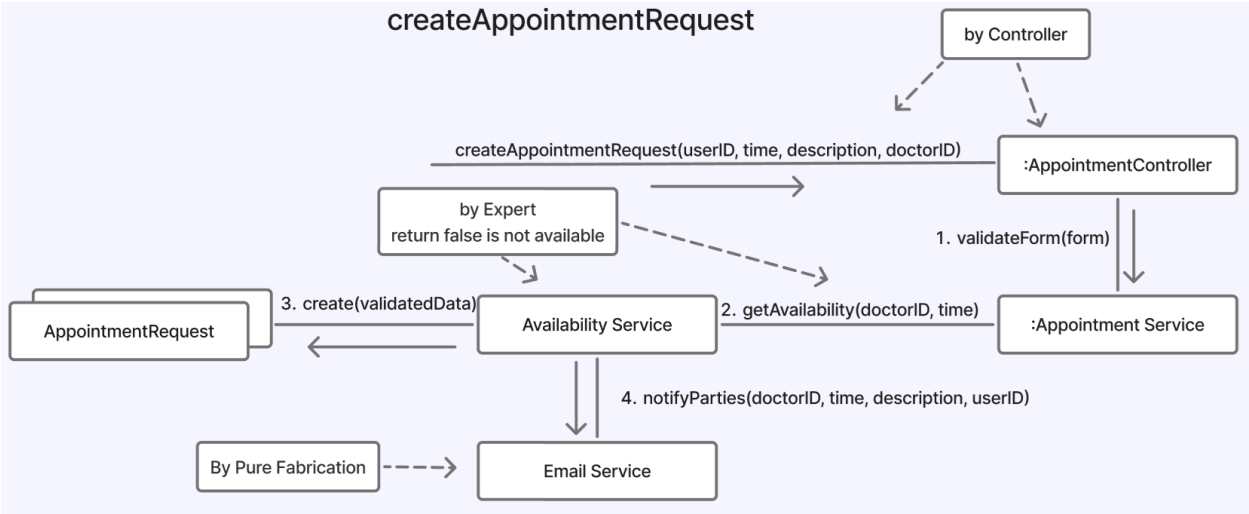
UML Interaction Diagram: Utilize the operation contracts developed in the previous step as a reference. Create a UML interaction diagrams (at least five) that visualizes the interactions between objects during the execution of the selected operation. Annotate the messages in the diagram with GRASP patterns such as Expert, Creator, etc., where applicable.

Answer:



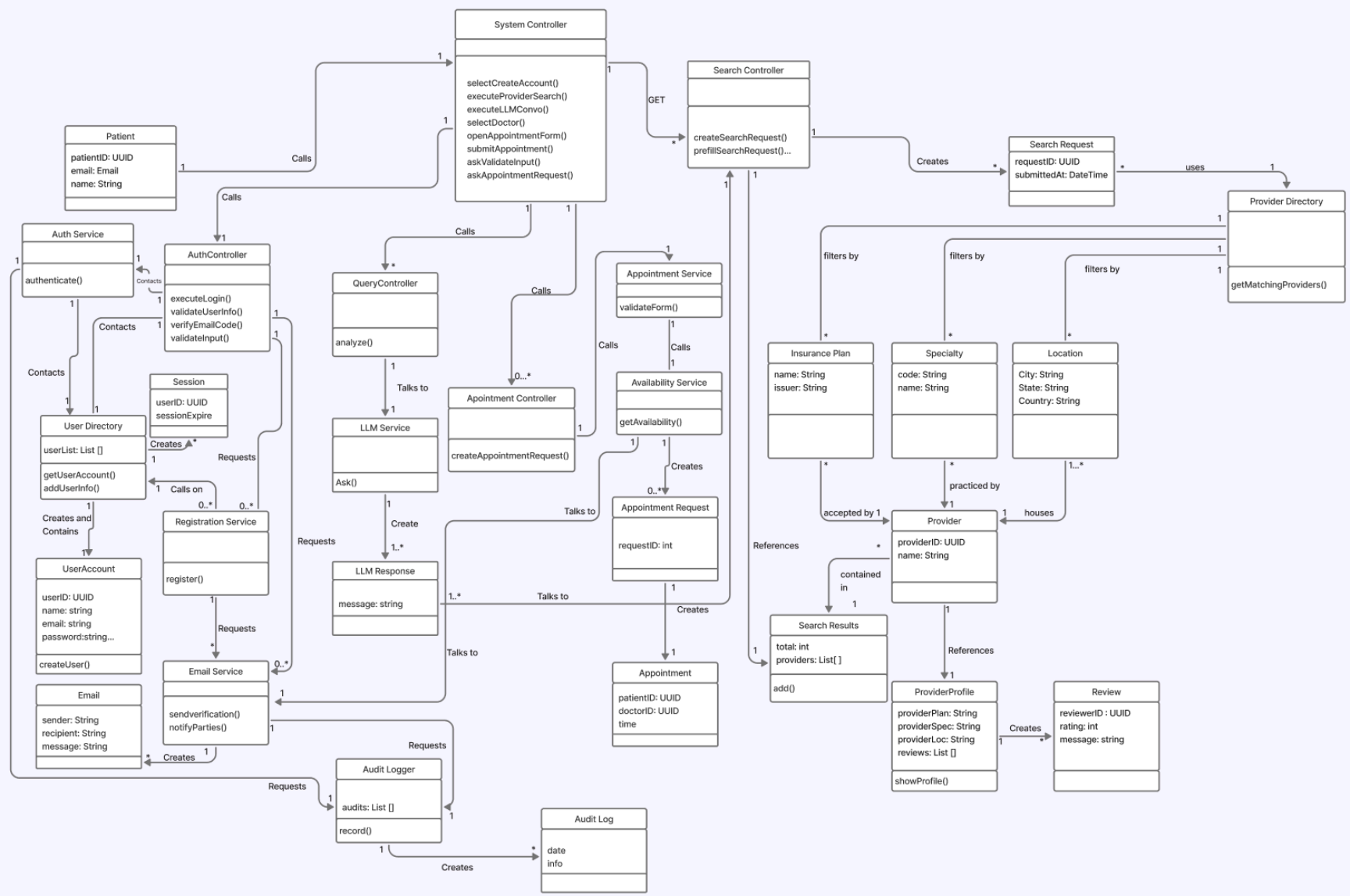

Provider Search
by Controller
executeProviderSearch(specialty, location, insurancePlan)
:SearchController
1. createSearchRequest(filters) : requestId
By Creator
by Expert
return null if no results
:ProviderDirectory
2. getMatchingProviders(filters)
:SearchRequest
3. add(Matching Providers)
:SearchResults


executeLLMConvo
by Controller
executeLLMConvo(prompt)
:QueryController
by Creator
1. analyze(prompt:string)
by Expert
:LLMResponse
2. createResponse(data)
:LLMService
by Controller
3. prefillSearchRequest(data)
:Search Controller


createAppointmentRequest
by Controller
createAppointmentRequest(userID, time, description, doctorID)
:AppointmentController
by Expert
return false is not available
1. validateForm(form)
AppointmentRequest
3. create(validatedData)
Availability Service
2. getAvailability(doctorID, time)
:Appointment Service
4. notifyParties(doctorID, time, description, userID)
By Pure Fabrication
Email Service

Section 6 (20 points):

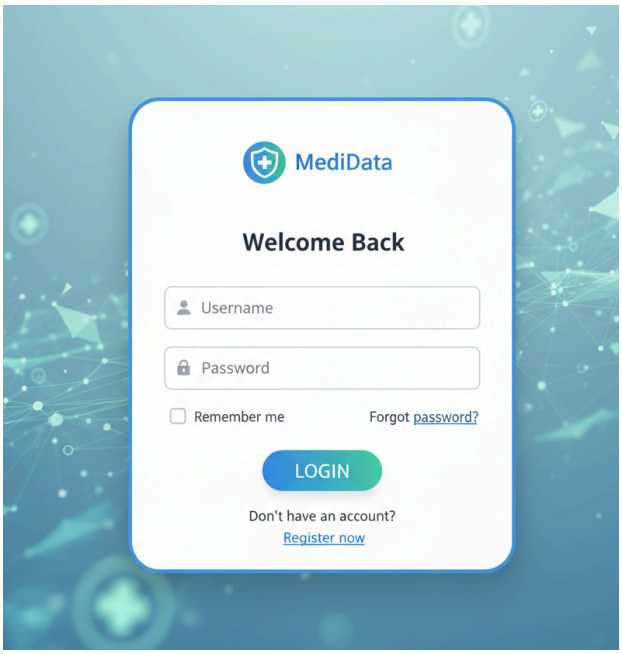
Bounded by the domain model, operation contracts and the interaction diagrams, create the final class diagram (one class diagram) that you will move forward with for this system.

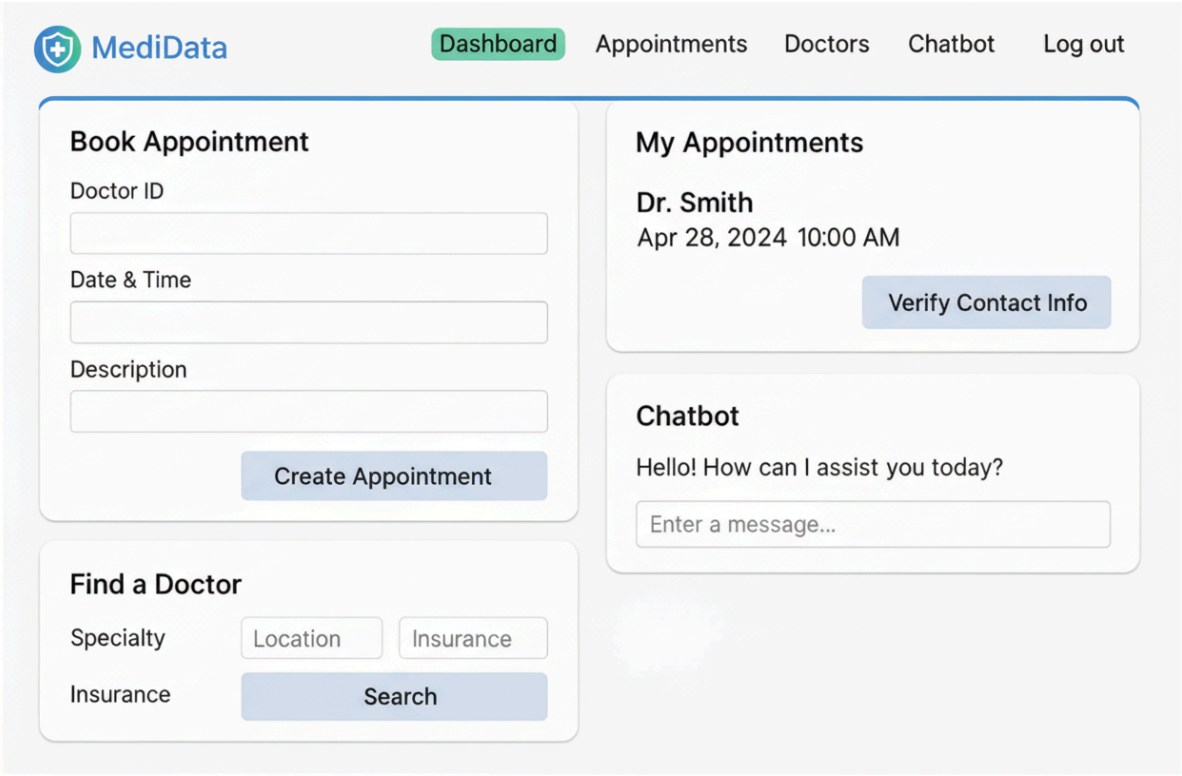


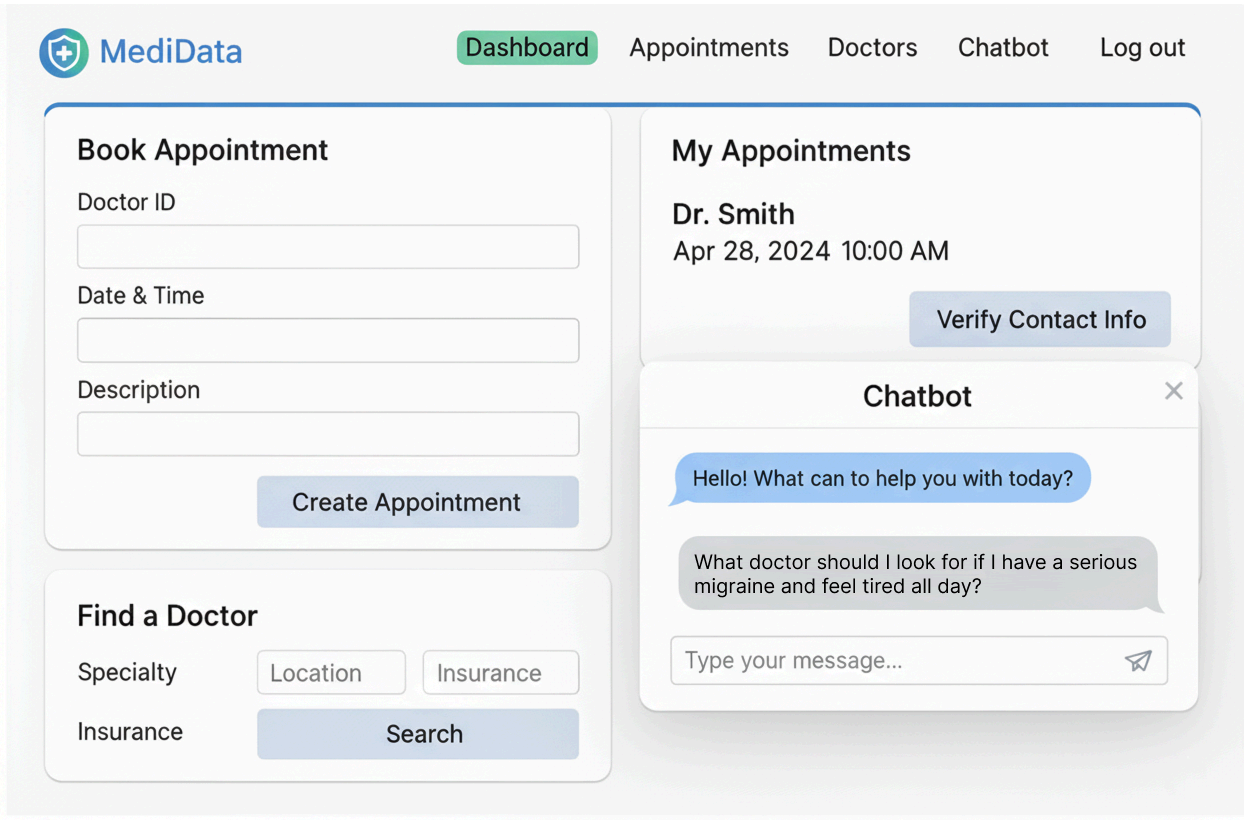
Section 7 (15 points):

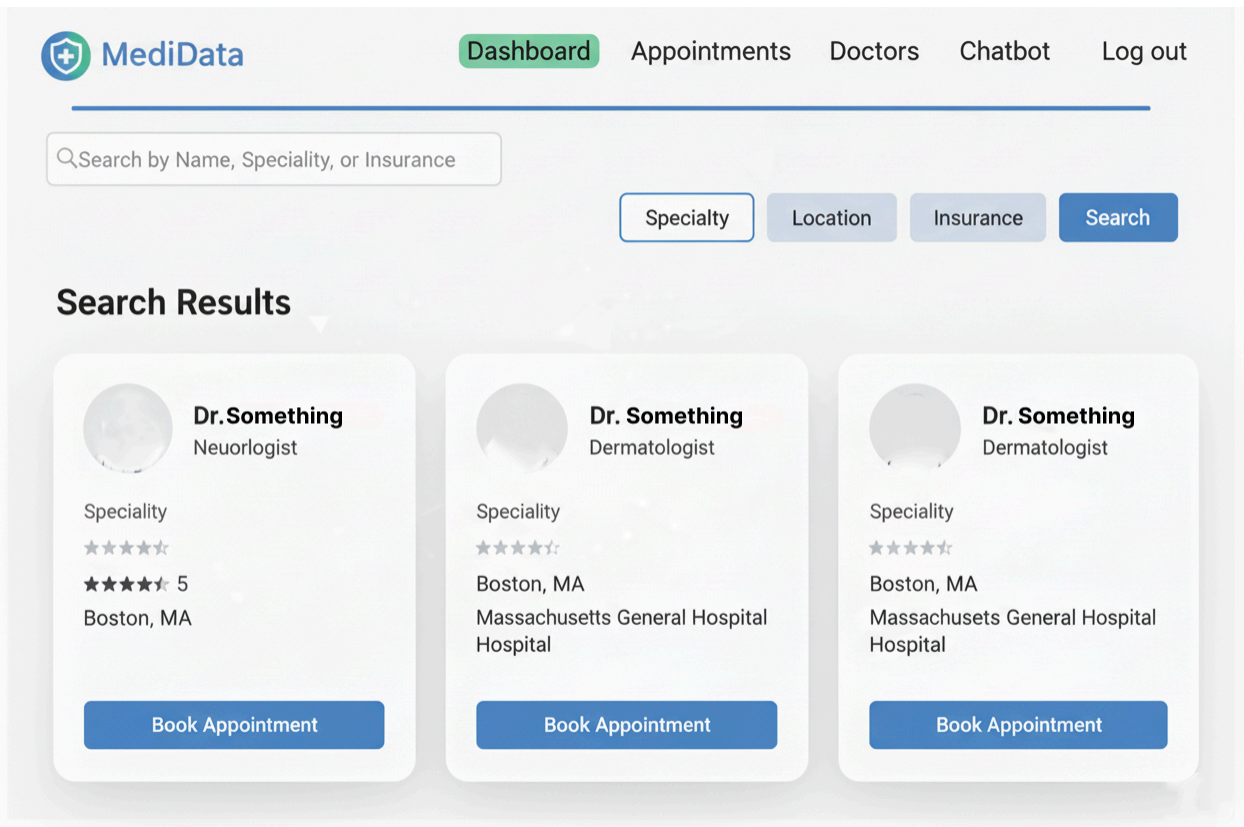
Include (refined) User Interface (UI) prototype screens of your application. You need to include at least two different screens. Interacting with these (two or more) screens should cover the five selected user cases.

Answer:






MediData
Dashboard
Appointments
Doctors
Chatbot
Log out
Book Appointment
Doctor ID
Date & Time
Description
Create Appointment
My Appointments
Dr. Smith
Apr 28, 2024 10:00 AM
Verify Contact Info
Chatbot
Hello! What can to help you with today?
What doctor should I look for if I have a serious migraine and feel tired all day?
Type your message...
Find a Doctor
Specialty
Location
Insurance
Insurance
Search


MediData
Dashboard
Appointments
Doctors
Chatbot
Log out
Search by Name, Speciality, or Insurance
Specialty
Location
Insurance
Search
Search Results
Dr.Something
Neuorlogist
Speciality
5
Boston, MA
Book Appointment
Dr. Something
Dermatologist
Speciality
Boston, MA
Massachusetts General Hospital
Hospital
Book Appointment
Dr. Something
Dermatologist
Speciality
Boston, MA
Massachusets General Hospital
Hospital
Book Appointment