**Product Design**

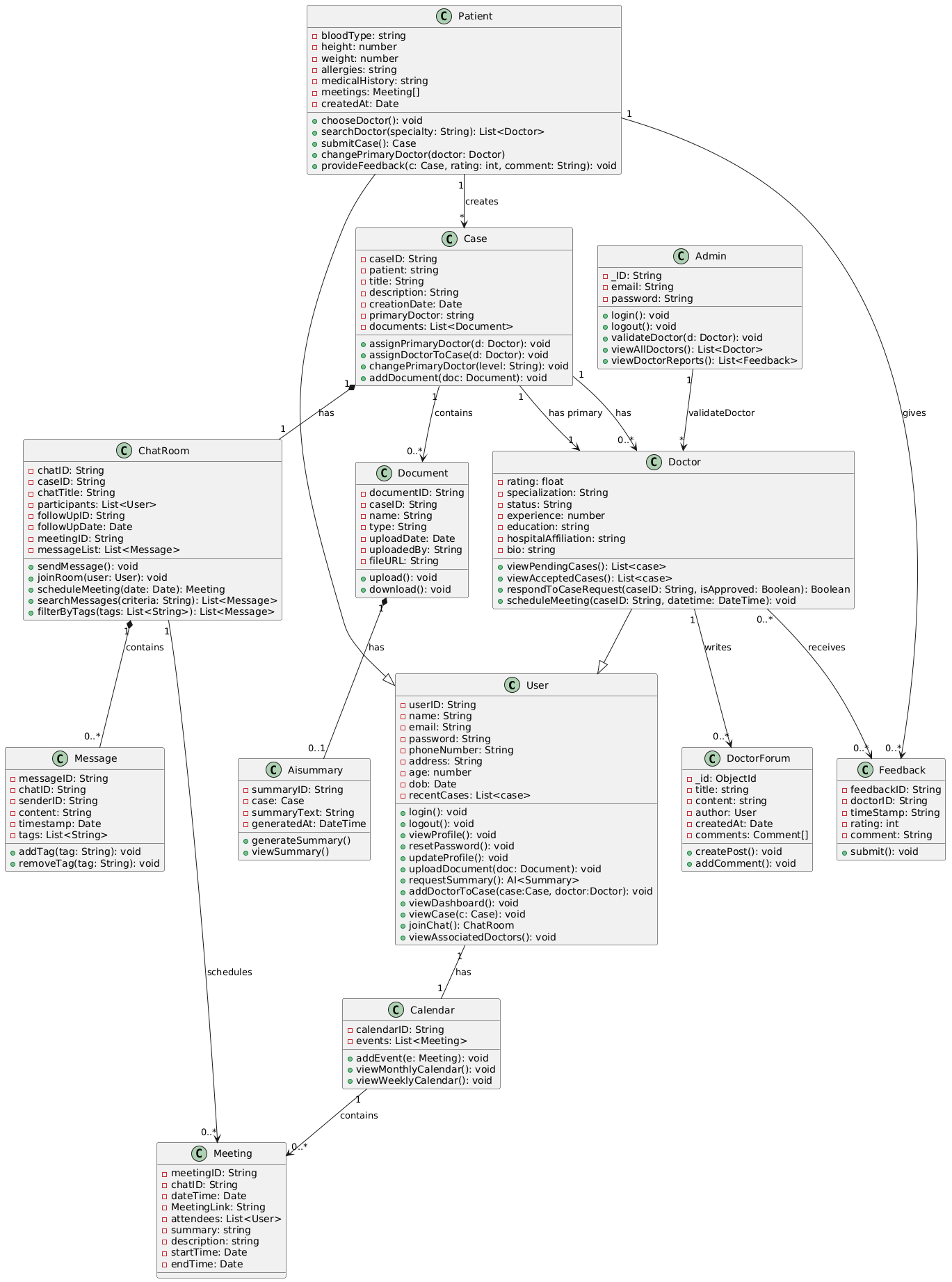
**Team 12 ICanCare Project**

**Team Members:**

* **Navya Shrivastava**
* **Chekka Yogeswari**
* **Rohitha Sarvani**
* **Krrish Tomar**
* **Manas Agrawal**

Note: This is a “living document”, meaning its content and format will grow with the implementation of the project. Use it to capture key project concepts and to document important design decisions.

# Design Model

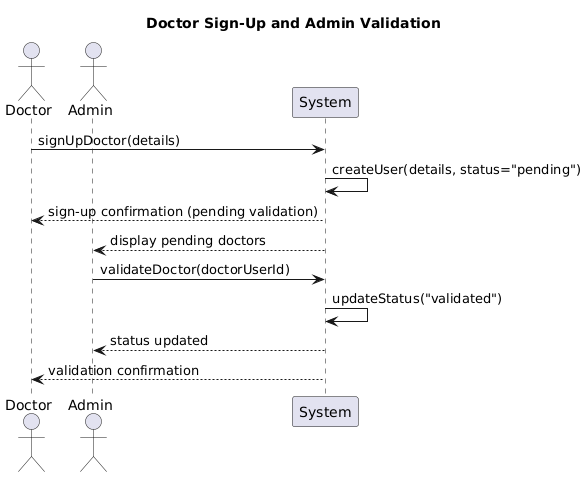


­

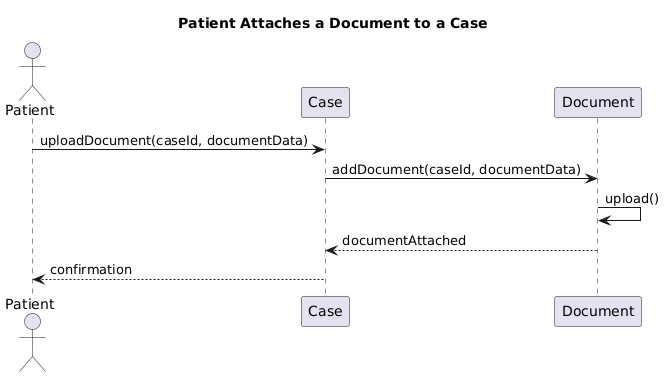
**Description of the Classes**

|  |  |
| --- | --- |
| <Class No. 1>  User | Class state   * userID: A unique identifier for each user. * name: The user’s full name. * email: The contact email address. * password: User's encrypted password for authentication * phoneNumber: Contact phone number. * address: User's physical address. * Age: User’s age * Dob: date of birth of user   Class behavior   * login(): Authenticates and initiates a user session. * logout(): Terminates the session. * viewProfile(): Displays user details. * resetPassword(): Allows user to change their password * updateProfile(): Modifies user profile information. * uploadDocument(): Uploads medical or supporting documents. * requestSummary(): getting AI summary of uploaded document. * addDoctorToCase(case : Case, doctor : Doctor): Associates an additional doctor with a case. * viewDashboard(): Displays an overview of case statuses and updates. * viewCases(): Retrieves a list of history entries detailing past case events * joinChatRoom(chat : ChatRoom): Joins a discussion chat room related to a case. * viewAssociatedDoctors(): View all the doctors in the application |
| <Class No. 2>  Patient | Class state   * Inherits the basic user attributes from User (userID, name, email). * Maintains associations with cases, documents, and history entries relevant to the patient. * Bloodtype: patients’s blood type * Height: patient’s height * Weight: patient’s weight * Allergies: patient’s allergies if any * MedicalHIstory: any past medical record of patient * createdAt: date on which account was made   Class behavior   * chooseDoctor(): Selects a doctor for the case. * searchDoctor(criteria : String): Searches for doctors matching specific criteria. * changePrimaryDoctor(): Allows changing the primary doctor of the case. * submitCase(): Creates and submits a new case. * provideFeedback(doctorId: String, rating: int, comment: String): Submits rating and comments for a doctor. |
| <Class No. 3>  Doctor | Class state   * Inherits the basic user attributes from User. * Associated with multiple cases either as the case manager or as a board member. * May be linked to chat rooms for real-time discussion on cases. * rating: Average rating based on patient feedback. * specialization: Doctor's medical specialty area * status: Approval status (pending, approved, rejected)’ * experience: doctor’s experience in years * education: doctor’s education * hospitalAffiliation: hospital the doctor is affiliated at   Class behavior   * viewPendingCases(): Lists case requests awaiting acceptance * viewAcceptedCases(): Lists all cases assigned to the doctor * respondToCaseRequest(caseId: String, isApproved: Boolean): Accepts or rejects a case request. * scheduleMeeting(caseId: String, dateTime: DateTime): Schedules a consultation or board meeting. |
| <Class No. 4>  Admin | Class state   * \_ID: admin id * Email: admin’s email * Password: password for admin login   Class behavior   * Login(): login as admin * Logout(): admin logout * validateDoctor(doctor : Doctor): Checks and validates a doctor’s credentials. * viewAllDoctors(): Lists all registered doctors. * viewDoctorFeedbacks(): Reviews patient feedback about doctors. |
| <Class No. 5>  Case | Class state   * caseID: Unique identifier for the case. * patient: The patient who submitted the case. * title: Brief title describing the case. * description: Detailed description of the medical issue. * creationDate: When the case was created. * Primary Doctor: The doctor assigned to manage the case. * board: A list of doctors who form the case review board. * documents: Collection of medical documents related to the case * Linked to a dedicated ChatRoom for discussions and an AICaseSummary for automated summaries.   Class behavior   * assign PrimaryDoctor(doctor : Doctor): Sets or changes the primary doctor * addDoctor(doctor : Doctor): Adds a new doctor to the case board. * scheduleFollowUp(date : Date): Schedules a follow-up appointment or review meeting in the chat. * changePrimaryDoctor(doctorId: String): Sets or changes the primary doctor. * addDocument(document: Document): Associates a document with the case |
| <Class No. 6>  Chat Room | Class state   * chatID: Unique identifier for the chat room. * caseID: The case associated with the chat room. * participants: A list of users (patients, doctors, admins) engaged in the conversation. * FollowUp details: Includes followUpID, followUpDate, and followUpStatus for tracking follow-up activities. * Meeting details: Contains meetingID, meetingDate, and meetingAgenda for scheduled meetings. * messages: Collection of all messages in the chatroom.   Class behavior   * sendMessage(msg : String): Sends a message to all participants in the chat room. * joinRoom(user : User): Adds a user to the chat room. * scheduleMeeting(date : Date, agenda : String): Organizes a new meeting within the chat room. * searchMessages(criteria: String): Finds messages matching search terms. * filterByTags(tags: List<String>): Filters messages by assigned tags. |
| <Class No. 7>  Message | Class state   * messageId: Unique identifier for the message. * chatroomId: The chat room containing this message. * senderId: User who sent the message. * content: The actual message text. * timestamp: When the message was sent. * tags: List of categorization tags for filtering.   Class behavior   * addTag(tag: String): Assigns a new tag to the message. * removeTag(tag: String): Removes a tag from the message. |
| <Class No .8>  Meeting | Class state   * meetingId: Unique identifier for the meeting * chatId: The case associated with the meeting.   dateTime: When the meeting is scheduled to occur.   * duration: Length of the meeting in minutes. * attendees: List of users invited to attend. * meetingLink: URL for the virtual meeting. * Summary: title of the meet * Description: description of the meet * StartTime: the start time of the meet * endTime: the end time of the meet |
| <Class No. 9>  AI Summary | Class state   * summaryID: Unique identifier for the summary. * case: The associated documentfor which the summary is generated. * summaryText: The actual summary text generated by the AI. * generatedAt: Timestamp marking when the summary was produced.   Class behavior   * generateSummary(): Triggers the AI process to create a summary. * viewSummary(): Retrieves the generated summary text for display. |
| <Class No. 10>  Feedback | Class state   * feedbackId: Unique identifier for the feedback. * timestamp:When the feedback was submitted * doctorId: The doctor being rated * comment: Detailed feedback text. * rating: Numerical score (typically 1-5).   Class behavior   * submit(): Records the feedback in the system. |
| <Class No. 11>  Documents | Class state     * documentId: Unique identifier for the document. * caseId: The case associated with the document. * name: Document name or title. * type: Document file type or category. * uploadDate: When the document was uploaded. * uploadedBy: User who uploaded the document * fileUrl: Link to access the actual file   class behavior   * upload(): Stores the document in the system. * download(): Retrieves the document file. |
| <Class No. 12>  Calendar | Class state   * userId: The user associated with this calendar. * events: List of meetings scheduled for the user.   Class behavior   * addEvent(meeting: Meeting): Adds a meeting to the calendar. * viewMonthlyCalendar(): Displays events for the month. * viewWeeklyCalendar(): Displays events for the week |
| <Class No. 13>  DoctorForum | Class state   * \_id: id of the post * Title: title of the post * Content: content of the post * Author: author of the post * createdAt: time at which the post was created * comments: comments on the post   Class Behavior   * createPost(): creates of a post * addComment(): adds comment on a post |

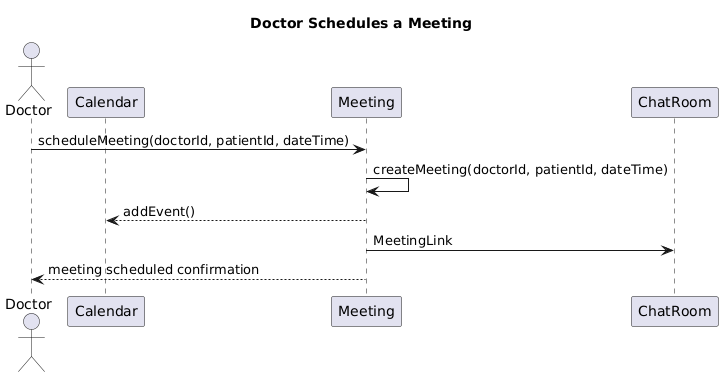
# Sequence Diagram(s)



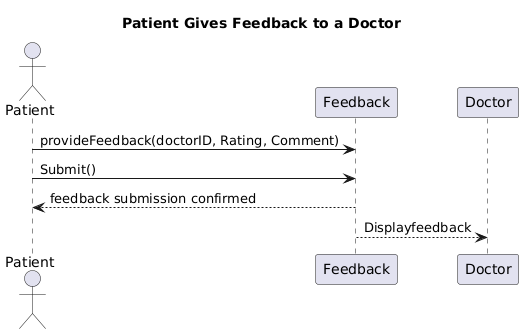
UseCase 1



UseCase 2



UseCase 3



UseCase 4

# Design Rationale

1. User Role Management

* Alternatives Considered:
  + Single User Class with Role Attributes:
    - *Pros:* Simpler class hierarchy, less initial code duplication.
    - *Cons:* Increased risk of role-specific logic intermingling, harder to enforce distinct behaviors.
  + Abstract User Base with Subclasses (Patient, Doctor, Admin):
    - *Pros:* Clear separation of responsibilities, easier to maintain and extend role-specific functionality.
    - *Cons:* Slightly more complex class hierarchy.
* Final Decision: Adopt the abstract base class User with separate subclasses for Patient, Doctor, and Admin.

2. Case Discussion and Communication

* Alternatives Considered:
  + Standalone Chat Module:
    - Pros: Modular, could be reused for different types of communications.
    - Cons: Integration with case functionalities (e.g., follow-up scheduling, board meetings) would require extra glue code and coordination.
  + Integrated ChatRoom Class with Extended Functionalities:
    - Pros: Initially, a single ChatRoom handled every type of discussion. Later, we refined the approach to support two distinct types of chatrooms for doctors: one dedicated to patient-doctor interactions and another exclusively for doctor-doctor communications. This separation enhances clarity and security in communication channels while still directly supporting real-time discussions, follow-up scheduling, and board meeting functionalities within the case context.
    - Cons: Increases the complexity of the ChatRoom class by having multiple types of chatrooms, which requires additional logic for routing messages and managing access controls.
* Final Decision: Implement an integrated ChatRoom system that distinguishes between patient-doctor and doctor-doctor chatrooms, alongside additional operations for follow-up and board meeting scheduling.