Master Clinic/10

Project Master Clinic SDS 2.0CM_Id 04,29,2018
Master Clinic Statement of Work2.0_mc_04 29, 2018

Master Clinic

Software Design Specification (SDS)
Team 10 Master Clinic 2.0
IdDateVersion: 2.0

CM Identifier: Master Clinic2.0_mc_04 29, 2018

Revision History

Sl. No.	Prepared/ Modified by	E-mail	Version	Date	Approved by	Descriptions/ Remarks
1.	Abdelrahman Mohamed	abdomedono2014@gmail.com	2.0	5/8/2018	Hady Maher	System Architecture/ Design Models/ System Deployment/ Traceability to requirements..
2.	Reham Hamdy	rehamhy6@gmail.com	2.0	5/8/2018	Aisha Mousa	Introduction/ Data Models

Distribution list

Name	E-mail	Notes
Dr: Ahmed Hamdy		
TA: Ali El-Sedeek	alielseddeek@gmail.com	
TA: Dina El Reedy	dinaelreedy@gmail.com	

Table of Contents

List of Tables

Table of Figures

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to provide an overview of the design specifications employed to implement the requirements stated in the SRS document .The software design of **MC** (Master Clinic) includes the system architecture which describes the different relations between system components e.g. controllers and views. It also includes design models such as design patterns and diagrams illustrating the design elements and their interactions.The design uses a data model (er-diagram) and explains the System deployment plan .

1.2 Scope

The project is mainly concerned with building a centralized database system for a **MC** business, and building modules that allows three types of users (patients - nurses - doctors) to interact together in a fast and easy way. patients can make reservations with their doctors, check their medical file through a web application. Nurses confirm patients' reservations,update patients' information, delete patients ,create invoices and manage clinics. Doctors are concerned with patients' files , in terms of their creation ,update and deletion , alongside other administrative functions nurses enjoy like accessing patient's information and clinic management,doctors can also access nurses' information and perform all sort of data manipulation.

1.3 Table of Acronyms and Definitions

Term	Definition
Worker	anyone working in the clinic and not a doctor or nurse.
admin	in this project admin is the dentist himself
Super admin	clinic owners
Patient file	a file that contains all patient's prescriptions, photos and any other comments or data dentist noted about the patient like case description and progress.
SD	Sequential Diagram

1.4 References

MC_SR01_v1.0

MC _SE02 _v1.0

1.5 Overview of Document

Throughout the document we would discuss different parts of the project in details including system architecture, system different classes, system operations, system data model and system deployment. topics would be presented in the following order:

Section 2: Description of the system architecture
Section 3: Presents the layout of design models used in the system e.g. design patterns, class diagrams...etc.
Section 4: Data models of the system illustrated by an er-diagram
Section 5: The deployment plan of the system in terms of tiers.Finally Section 6 is a table to trace requirements to their corresponding design elements.

2. System Architecture

Since the project is website then it would be used be different users at the sametime. This means that a single point of control won't be suitable. So we use MVC architecture which consists of three separate layers. views layer which is the GUI that takes inputs and actions from users. Controllers layer which are the processing layer in which the application logic is running and have more than one controller that are able to function all together at the same time. Finally the Models layer which is the data layer that contains the application data and connects with the database. MVC architecture makes relation between different layers like the following views are connected to controllers and controllers are connected to models. We start with relation between controllers and models.

Relation between controllers and models

- Admin Controllers use all the models since admins are allowed to create, update or delete any other entity and themselves given the right permissions (only super admins can create, delete or update any other admins).
- Nurse Controllers use Nurse, Patient, Reservation, Receipt, Image, Comment and Prescription models.
- Patient Controllers use Patient, Reservation, Receipt, Image, Comment and Prescription models.

Relation between views and controllers

Since the navigation bar contains a link for logging out (If you are logged in if not it's replaced with a link to the login page) and a link for the home page this makes them every user (admin, nurse, patient or guest) views capable of accessing both the home controller and login controller. That's why no connections to these 2 controllers from their respective views would mentioned in the next section.

Admin

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient, nurse, admin and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon admin is logged in to make it easier to perform different functionalities

Nurse

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- rest of the views: can access patient and profile controllers due to the links that includes all these views in the side navigation bar. these links would available in every single view as soon nurse is logged in to make it easier to perform different functionalities

Patient

- login page: can only access forgot password controller
- send reset email page: can only access forgot password controller
- reset password page: can only access reset password controller
- home page: can access patient controller
- rest of the views: can access patient and profile controllers due to the links that includes all these views in the navigation bar. these links would available in every single view as soon patient is logged in to make it easier to perform different functionalities

Guest

- index page: accesses patient’s home and login controllers as a normal guest is a logged out patient

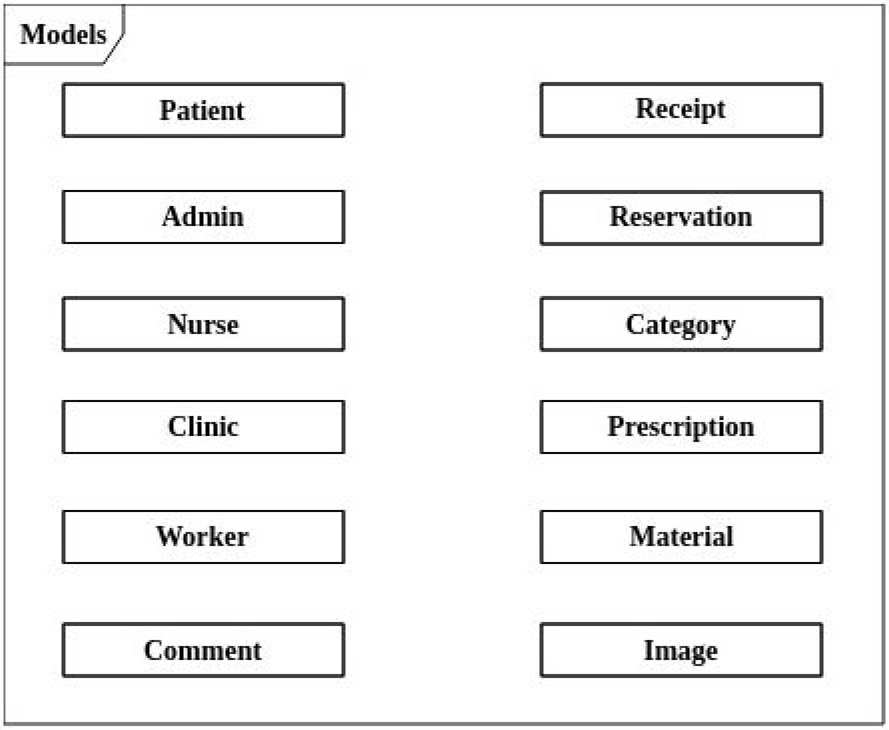


Figure 1: Architecture Model Diagram

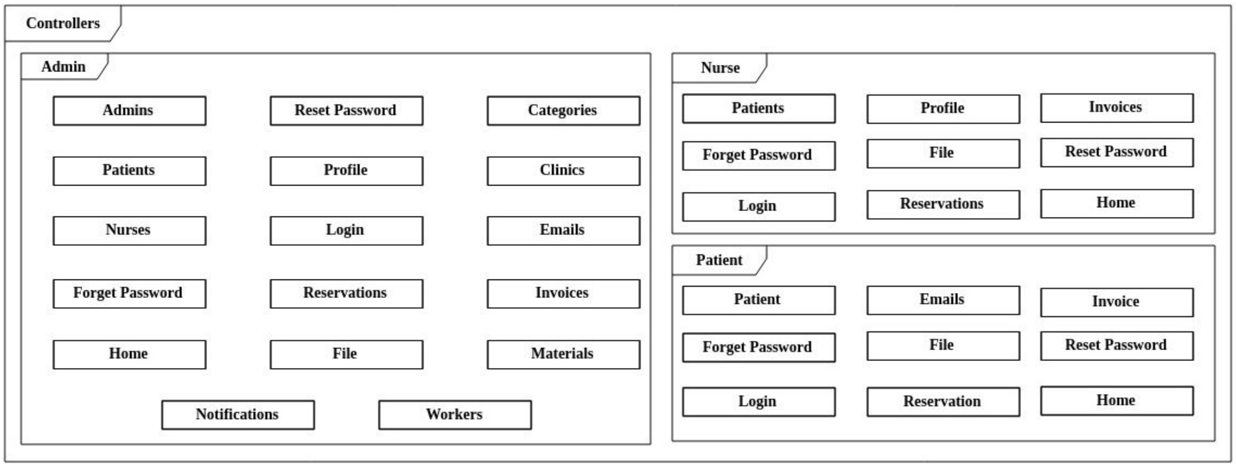


Figure 2: Architecture Controllers Diagram

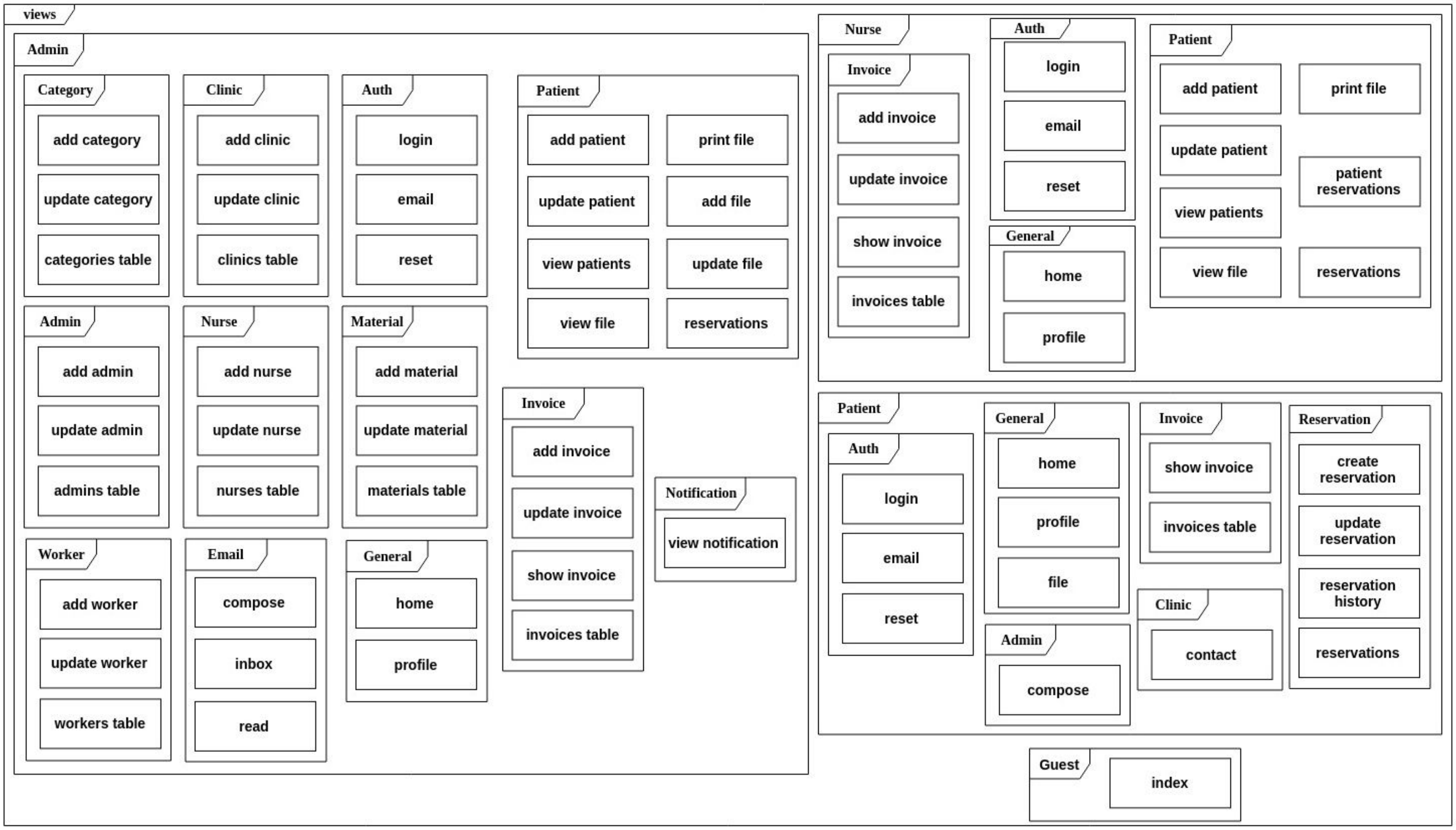


Figure 3: Architecture View Diagram

3. Design Models

3.1 Design Patterns Description

3.1.1 *MVC Architectural design pattern*

MVC architecture which consists of three separate layers. views layer which is the GUI that takes inputs and actions from users. Controllers layer which are the processing layer in which the application logic is running and have more than one controller that are able to function all together at the same time. Finally the Models layer which is the data layer that contains the application data and connects with the database. MVC architecture makes relation between different layers like the following views are connected to controllers and controllers are connected to models.

3.1.2 *Factory design pattern*

Factory creational design pattern which define an interface for creating an object, but let subclasses decide which class to instantiate. Models layer of MVC architectural design pattern is using factory to create different models for creating different object with the basic models functions and implement its own functions.

3.2 Class Diagrams

Class diagram is meant to show system different classes, their attributes and operation (methods). So we have five classes Patient, Admin, Nurse, User and Clinic. There is also interfaces which are some relevant functions packed together in one structure and we have one interface Model which contains functions to interact with database. We start with class admin

User

User is an abstract class that handles system users with basic data and functions.

properties

User class have not properties as it's only to handle basic users functions. Every user would implement his own properties

Methods

1. login: this methods is for handling user login it takes user's email and password as arguments to check if the user already exist and authorize his access

Admin

Admin class is a user class that inherits User class and represents admin user.

properties

1. id: integer that represents admin id number which is used to identify the admin thus no two admins would have the same id
2. name: string that represents admin's name (usually first name)

3. email: string that represents admin's email address and must be unique as it's used for logging into the system
4. password: string that represents admin's password and is used to authorize logging in
5. image: string represents path admin's profile picture in the storage files
6. mobile: string that represents admin's mobile number
7. status: boolean variable that indicates account status inactive (false) and active (true)
8. role: string that represents the role of the admin with two valid roles normal and super
9. about: small text bio about the doctor for displaying in patient view
10. startDay: day of the week doctor start working
11. endDay: day of the week doctor ends working
12. startTime: time doctor start taking appointments
13. endTime: time doctor finish his appointment in the day

Methods

1. createPaitent: method for creating patients which takes an array of patient's data create the patient if the data is valid and then return the new patient
2. updatePatient: method for updating patients which takes an array of patient's data update the patient if the data is valid
3. deletePatient: method for deleting patients that takes the patient argument then delete him
4. listPatients: method that returns a list of all available patients
5. viewFile: method that returns a specific patient file given his id
6. addPatientFile: method that take new appointment data and add it to the file
7. updatePatientFile: method that takes appointment number and new appointment data to update the appointment
8. deleteFile: method that takes patient id and delete his file
9. createNurse: method for creating nurses which takes an array of nurse's data create the nurse if the data is valid and then return the new nurse
10. updateNurse: method for updating nurses which takes an array of nurse's data update the nurse if the data is valid
11. deleteNurse: method for deleting nurses that takes the nurse as argument then delete her
12. listNurses: method that returns a list of all available nurses
13. createClinic: method for creating clinics which takes an array of clinic's data create the clinic if the data is valid and then return the new clinic
14. updateClinic: method for updating clinics which takes an array of clinic's data update the clinic if the data is valid
15. deleteClinic: method for deleting clinics that takes the clinic as argument then delete it with all data associated with it (materials, nurses, workers, ...)
16. listClinics: method that returns a list of all available clinics
17. createAdmin: method for creating admins which takes an array of admin's data and string role (to verify permissions only super admin can create both super and normal admins) create the admin if the data is valid and then return the new admin
18. updateAdmin: method for updating admins which takes an array of admin's data and role (only super admin can update other admins) update the patient if the data is valid
19. deleteAdmin: method for deleting admins that takes the admin and role (only super admin can delete other admins) as argument then delete him

20. listAdmins: method that takes role (only super admin can list all admins) as an argument returns a list of all available admins
21. createCategory: method for creating categories which takes an array of the category's data create the category if the data is valid and then return the new category
22. updateCategory: method for updating categories which takes an array of the category's data update the category if the data is valid
23. deleteCategory: method for deleting categories that takes the category as argument then delete it and all its materials
24. listCategory: methods that returns a list of all available categories
25. createMaterial: method for creating materials which takes an array of the material's data create the material if the data is valid and then return the new material
26. updateMaterial: method for updating materials which takes an array of the material's data update the material if the data is valid
27. deleteMaterial: method for deleting materials that takes the material as argument then delete it
28. listMaterial: method that returns a list of all available materials
29. createWorker: method for creating workers which takes an array of the worker's data create the worker if the data is valid and then return the new worker
30. updateWorker: method for updating workers which takes an array of the worker's data update the worker if the data is valid
31. deleteWorker: method for deleting workers that takes the worker as argument then delete him
32. listWorker: method that returns a list of all available workers
33. compose: method for creating emails which takes an array of the email's data create the email if the data is valid and then send it
34. inbox: method for show email inbox that contains all email received by the admin
35. showEmail: method for showing an individual email for reading or deleting it provided its id
36. deleteEmail: method for deleting emails given its id
37. createInvoice: method for creating invoices which takes an array of the invoice's data create the invoice if the data is valid and then return the new invoice
38. updateInvoice: method for updating invoices which takes an array of the invoice's data update the invoice if the data is valid
39. showInvoice: method for showing an individual invoice for reading, printing or deleting it provided its id
40. deleteInvoice: method for deleting invoices that takes the invoice as argument then delete it
41. listInvoice: method that returns a list of all available invoices
42. viewNotifications: method that takes notification as argument and displays them
43. viewReservation: method for showing an individual reservation provided its id
44. listReservation: method that returns a list of all available reservations

Nurse

Nurse class is a user class that inherits User class and represents nurse user.

properties

1. id: integer that represents nurse id number which is used to identify the nurse thus no two nurses would have the same id
2. name: string that represents nurse's name (usually first name)

3. email: string that represents nurse’s email address and must be unique as it’s used for logging into the system
4. password: string that represents nurse’s password and is used to authorize logging in
5. image: string represents path nurse’s profile picture in the storage files
6. mobile: string that represents nurse’s mobile number
7. status: boolean variable that indicates account status inactive (false) and active (true)
8. gender: nurse gender (male/female)
9. dateOfBirth: nurse date of birth
10. salary: nurse monthly salary
11. startDay: day of the week nurse start working
12. endDay: day of the week nurse ends working
13. startTime: time nurse start working in a working day
14. endTime: time nurse finish her in a working day
15. clinicId: id of the clinic is working in

Methods

1. createPaitent: methods for creating patients which takes an array of patient’s data create the patient if the data is valid and then return the new patient
2. updatePatient: methods for updating patients which takes an array of patient’s data update the patient if the data is valid
3. deletePatient: method for deleting patients that takes the patient argument then delete him
4. listPatients: method that returns a list of all available patients
5. viewFile: method that returns a specific patient file given his id
6. deleteFile: method that takes patient id and delete his file
7. createInvoice: method for creating invoices which takes an array of the invoice’s data create the invoice if the data is valid and then return the new invoice
8. updateInvoice: method for updating invoices which takes an array of the invoice’s data update the invoice if the data is valid
9. showInvoice: method for showing an individual invoice for reading, printing or deleting it provided its id
10. deleteInvoice: method for deleting invoices that takes the invoice as argument then delete it
11. listInvoice: method that returns a list of all available invoices
12. viewReservation: method for showing an individual reservation provided its id
13. confirm: method for confirming reservations so doctor receive them and patient can take the appointment
14. listReservation: method that returns a list of all available reservations
15. deleteReservation: method for deleting reservations that takes the reservation as argument then delete it

Patient

Patient class is a user class that inherits User class and represents patient user.

properties

1. id: integer that represents nurse id number which is used to identify the nurse thus no two nurses would have the same id

2. name: string that represents nurse’s name (usually first name)
3. email: string that represents nurse’s email address and must be unique as it’s used for logging into the system
4. password: string that represents nurse’s password and is used to authorize logging in
5. image: string represents path nurse’s profile picture in the storage files
6. mobile: string that represents nurse’s mobile number
7. status: boolean variable that indicates account status inactive (false) and active (true)
8. gender: patient gender (male/female)
9. dateOfBirth: patient date of birth

Methods

1. compose: method for creating emails which takes an array of the email’s data create the email if the data is valid and then send it
2. showInvoice: method for showing an individual invoice for reading, printing or deleting it provided its id
3. listInvoice: method that returns a list of all available invoices
4. createReservation: method for creating reservations which takes an array of the reservation’s data create the reservation if the data is valid and then return the new reservation
5. updateReservation: method for updating reservations which takes an array of the reservation’s data update the invoice if the data is valid
6. deleteReservation: method for deleting reservations that takes the reservation as argument then delete it
7. listReservation: method that returns a list of all available reservations

Clinic

Clinic class represents clinics which unlike previous classes is not a user. That’s why it doesn’t inherit User class

properties

1. id: integer that represents clinic id number which is used to identify the clinic thus no two clinics would have the same id
2. address: string that represents clinic address
3. name: string that represents clinic’s name
4. email: string that represents clinic’s email address which is used for contacting with patients
5. telephone: string that represents clinic’s telephone number
6. openingTime: the time the clinic opens
7. closingTime: the time the clinic closes

Methods

Clinic is not a user class so it doesn’t have any methods at all.

Category

Category class represents categories of materials used in the clinics

properties

1. id: integer that represents category id number which is used to identify the category thus no two categories would have the same id
2. name: string that represents category's name

Methods

Category is not a user class so it doesn't have any methods at all.

Material

Material class represents materials used in the clinics

properties

1. id: integer that represents material id number which is used to identify the material thus no two materials would have the same id
2. name: string that represents material's name
3. clinic_id: id of the clinic material belongs to
4. num: integer that represents number of available units of the material
5. min_num: integer that represents minimum number of units of the material
6. cost: double that represents price of the individual unit of the material
7. category_id: id of the category material belongs to

Methods

Material is not a user class so it doesn't have any methods at all.

Worker

Worker class represents workers working in the clinics that are not nurses

properties

1. id: integer that represents worker id number which is used to identify the worker thus no two workers would have the same id
2. name: string that represents worker's name
3. salary: double that represents salary given to the worker on a monthly bases
4. mobile: string that represents worker phone number
5. date_of_birth: worker's date of birth
6. date_of_start: date that worker start working in the clinic
7. clinic_id: id of the clinic material belongs to
8. position: string that represents job title of the worker

Methods

Worker is not a user class so it doesn't have any methods at all.

Receipt

Receipt class represents receipts created to keep track of patient payment as an info for the patient himself and for the doctor to keep track of the statistics of his clinics

properties

1. id: integer that represents receipt id number which is used to identify the receipt thus no two receipts would have the same id
2. clinic_id: id of the clinic receipt belongs to
3. nurse_id: id of the nurse created the receipt (must be null if admin_id is available)
4. admin_id: id of the admin created the receipt (must be null if nurse_id is available)
5. day: day at which the receipt was created
6. tax: double that represents amount tax added to the receipt which is optional
7. discount: double that represents amount of discount added to the receipt if any
8. total_price: double that represents total price must be paid

Methods

Receipt is not a user class so it doesn't have any methods at all.

Reservation

Reservation class represents reservations created by patients to make an appointment

properties

1. id: integer that represents reservation id number which is used to identify the reservation thus no two reservations would have the same id
2. clinic_id: id of the clinic reservation belongs to
3. nurse_id: id of the nurse created the confirmed the reservation
4. admin_id: id of the admin received the reservation
5. time: time at which the receipt was created
6. tax: double that represents amount tax added to the reservation which is optional
7. attend: boolean if true then patient did attend his appointment if false then he didn't
8. reject: boolean if true then the appointment is rejected if false then the appointment is no rejected yet

Methods

Reservation is not a user class so it doesn't have any methods at all.

Prescription

Prescription class represents prescriptions given by the doctor to the patient in an appointment

properties

1. id: integer that represents prescription id number which is used to identify the prescription thus no two prescriptions would have the same id
2. name: text that represents content of the prescription
3. patient_id: id of the patient given that prescription
4. admin_id: id of the admin that gave the prescription

Methods
Prescription is not a user class so it doesn't have any methods at all.

Image

Image class represents patient photos taken during an appointment

properties

1. id: integer that represents image id number which is used to identify the image thus no two images would have the same id
2. image: string that represents photo path in the storage
3. caption: string that represents photo caption
4. patient_id: id of the patient
5. admin_id: id of the admin

Methods
Image is not a user class so it doesn't have any methods at all.

Comment

Comment class represents notes taken by the doctor during in an appointment

properties

1. id: integer that represents comment id number which is used to identify the comment thus no two comments would have the same id
2. content: text that represents content of the notes
3. patient_id: id of the patient
4. admin_id: id of the admin that took the notes

Methods
Comment is not a user class so it doesn't have any methods at all.

Model

This is an interface used to interact with database. It provide functions to make secure connections with database and perform different operation on it.

Methods

1. find: method that gets row by id from its table
2. update: method that takes an array of data and update a row in database table
3. insert: method that takes an array of data and creates a new row in a database table
4. delete: method that deletes a row by its id

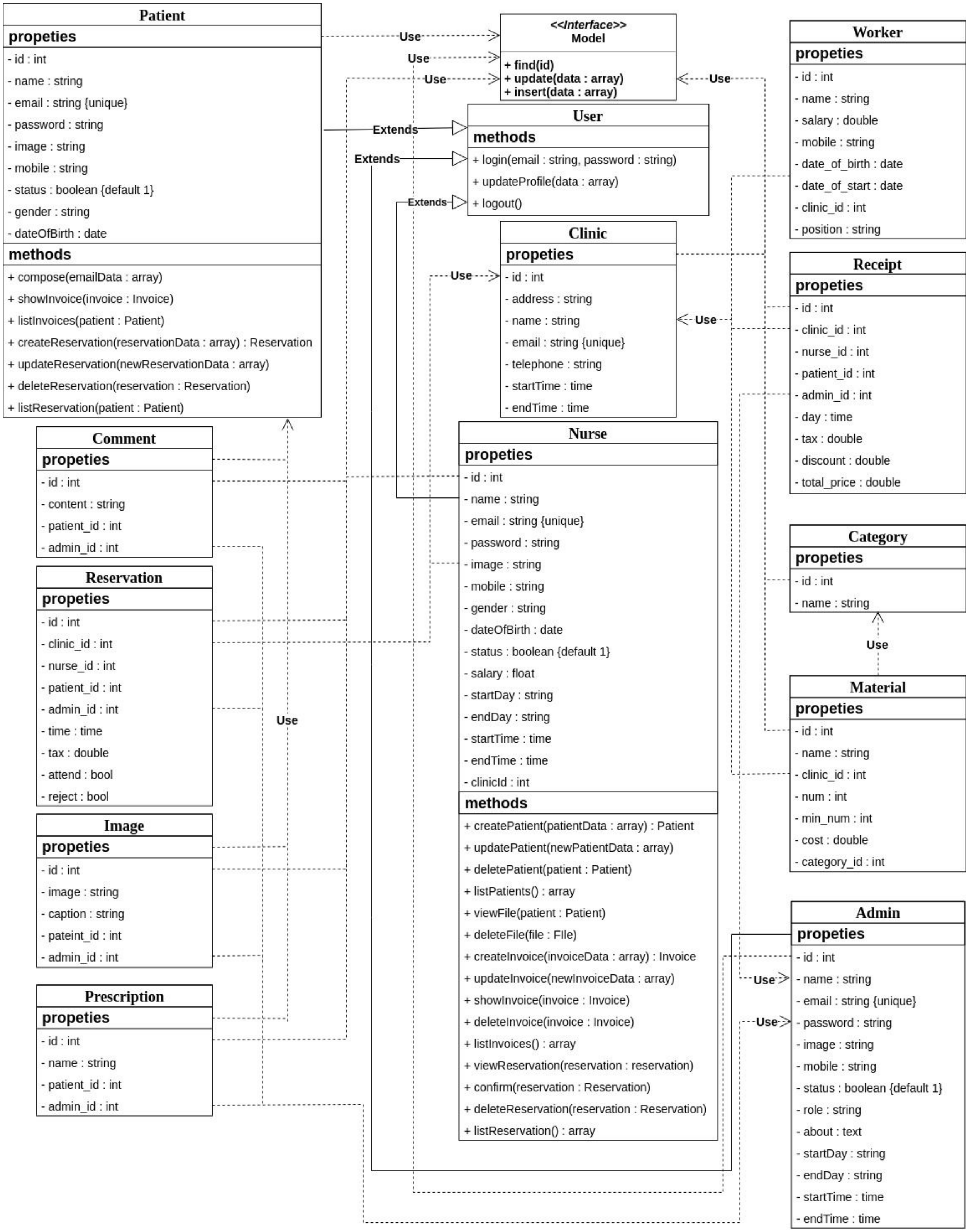
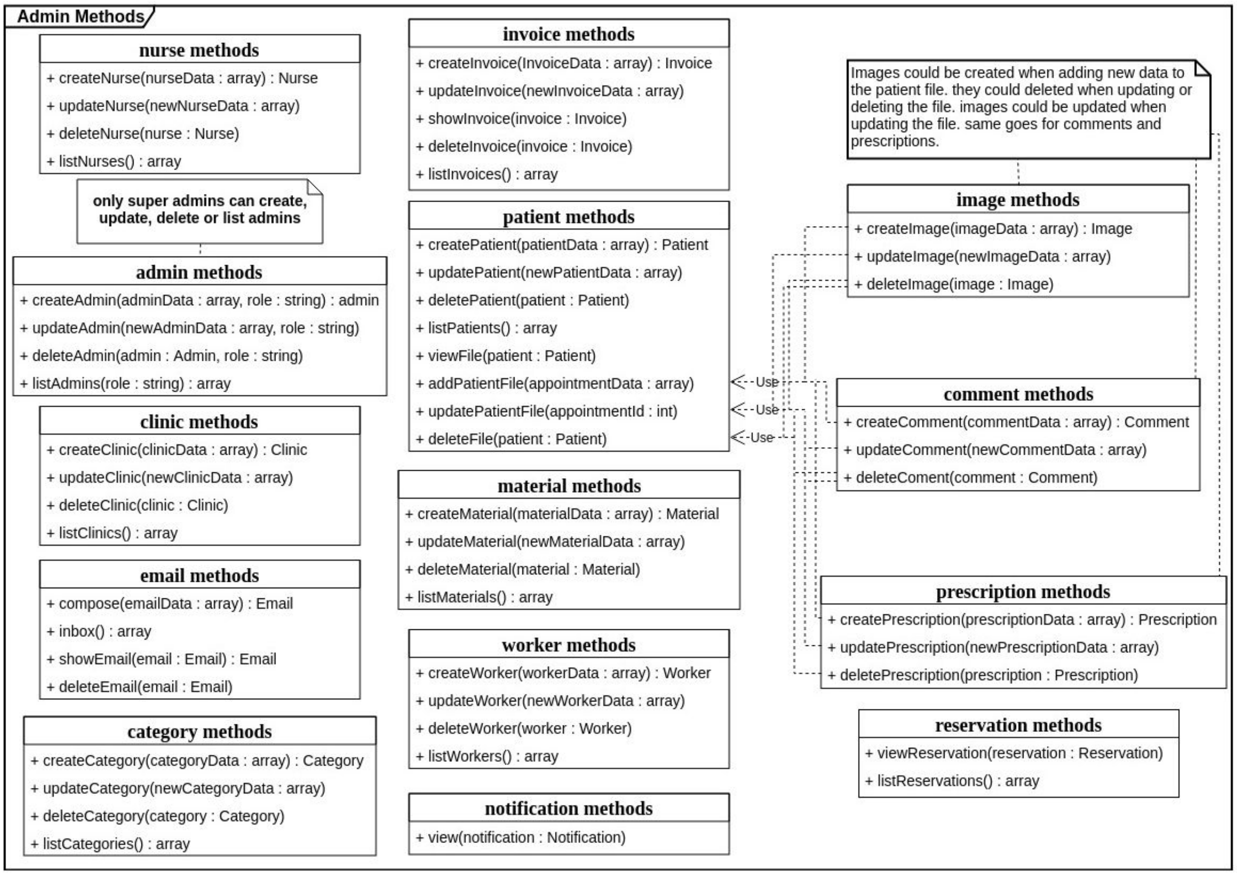


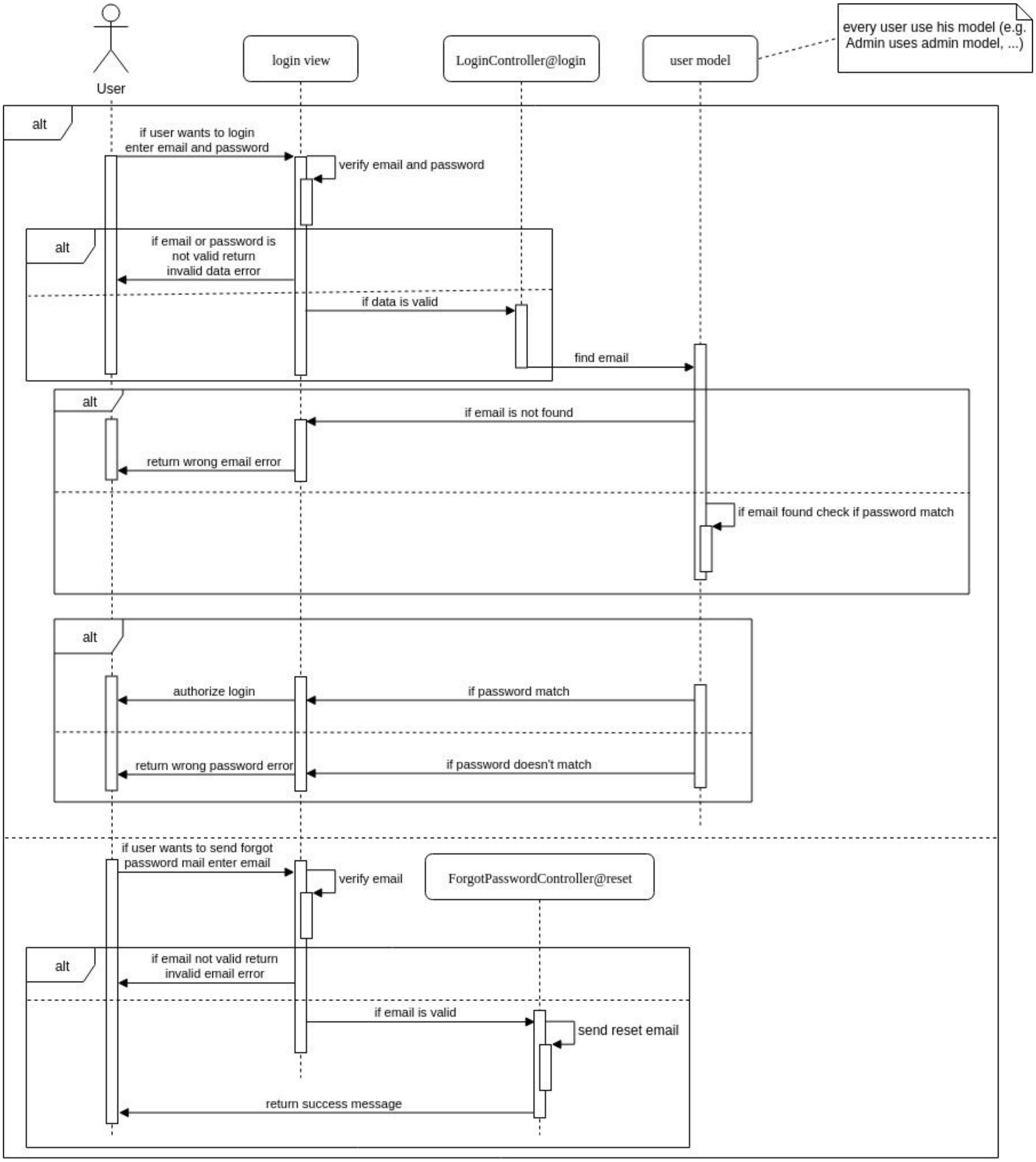
Figure 4: Class Diagram



Admin Methods
nurse methods
+ createNurse(nurseData : array) : Nurse
+ updateNurse(newNurseData : array)
+ deleteNurse(nurse : Nurse)
+ listNurses() : array
only super admins can create, update, delete or list admins
admin methods
+ createAdmin(adminData : array, role : string) : admin
+ updateAdmin(newAdminData : array, role : string)
+ deleteAdmin(admin : Admin, role : string)
+ listAdmins(role : string) : array
clinic methods
+ createClinic(clinicData : array) : Clinic
+ updateClinic(newClinicData : array)
+ deleteClinic(clinic : Clinic)
+ listClinics() : array
email methods
+ compose(emailData : array) : Email
+ inbox() : array
+ showEmail(email : Email) : Email
+ deleteEmail(email : Email)
category methods
+ createCategory(categoryData : array) : Category
+ updateCategory(newCategoryData : array)
+ deleteCategory(category : Category)
+ listCategories() : array
invoice methods
+ createInvoice(InvoiceData : array) : Invoice
+ updateInvoice(newInvoiceData : array)
+ showInvoice(invoice : Invoice)
+ deleteInvoice(invoice : Invoice)
+ listInvoices() : array
patient methods
+ createPatient(patientData : array) : Patient
+ updatePatient(newPatientData : array)
+ deletePatient(patient : Patient)
+ listPatients() : array
+ viewFile(patient : Patient)
+ addPatientFile(appointmentData : array)
+ updatePatientFile(appointmentId : int)
+ deleteFile(patient : Patient)
material methods
+ createMaterial(materialData : array) : Material
+ updateMaterial(newMaterialData : array)
+ deleteMaterial(material : Material)
+ listMaterials() : array
worker methods
+ createWorker(workerData : array) : Worker
+ updateWorker(newWorkerData : array)
+ deleteWorker(worker : Worker)
+ listWorkers() : array
notification methods
+ view(notification : Notification)
Images could be created when adding new data to the patient file. they could deleted when updating or deleting the file. images could be updated when updating the file. same goes for comments and prescriptions.
image methods
+ createImage(imageData : array) : Image
+ updateImage(newImageData : array)
+ deleteImage(image : Image)
Use
Use
Use
comment methods
+ createComment(commentData : array) : Comment
+ updateComment(newCommentData : array)
+ deleteComent(comment : Comment)
prescription methods
+ createPrescription(prescriptionData : array) : Prescription
+ updatePrescription(newPrescriptionData : array)
+ deletePrescription(prescription : Prescription)
reservation methods
+ viewReservation(reservation : Reservation)
+ listReservations() : array

Figure 5: Class Diagram Admin Methods

3.3 Interaction Diagrams



User
login view
LoginController@login
user model
every user use his model (e.g. Admin uses admin model, ...)
alt
if user wants to login enter email and password
verify email and password
alt
if email or password is not valid return invalid data error
if data is valid
find email
alt
if email is not found
return wrong email error
if email found check if password match
alt
authorize login
if password match
return wrong password error
if password doesn't match
if user wants to send forgot password mail enter email
verify email
ForgotPasswordController@reset
alt
if email not valid return invalid email error
if email is valid
send reset email
return success message

Figure 6: SD1 login sequence diagram

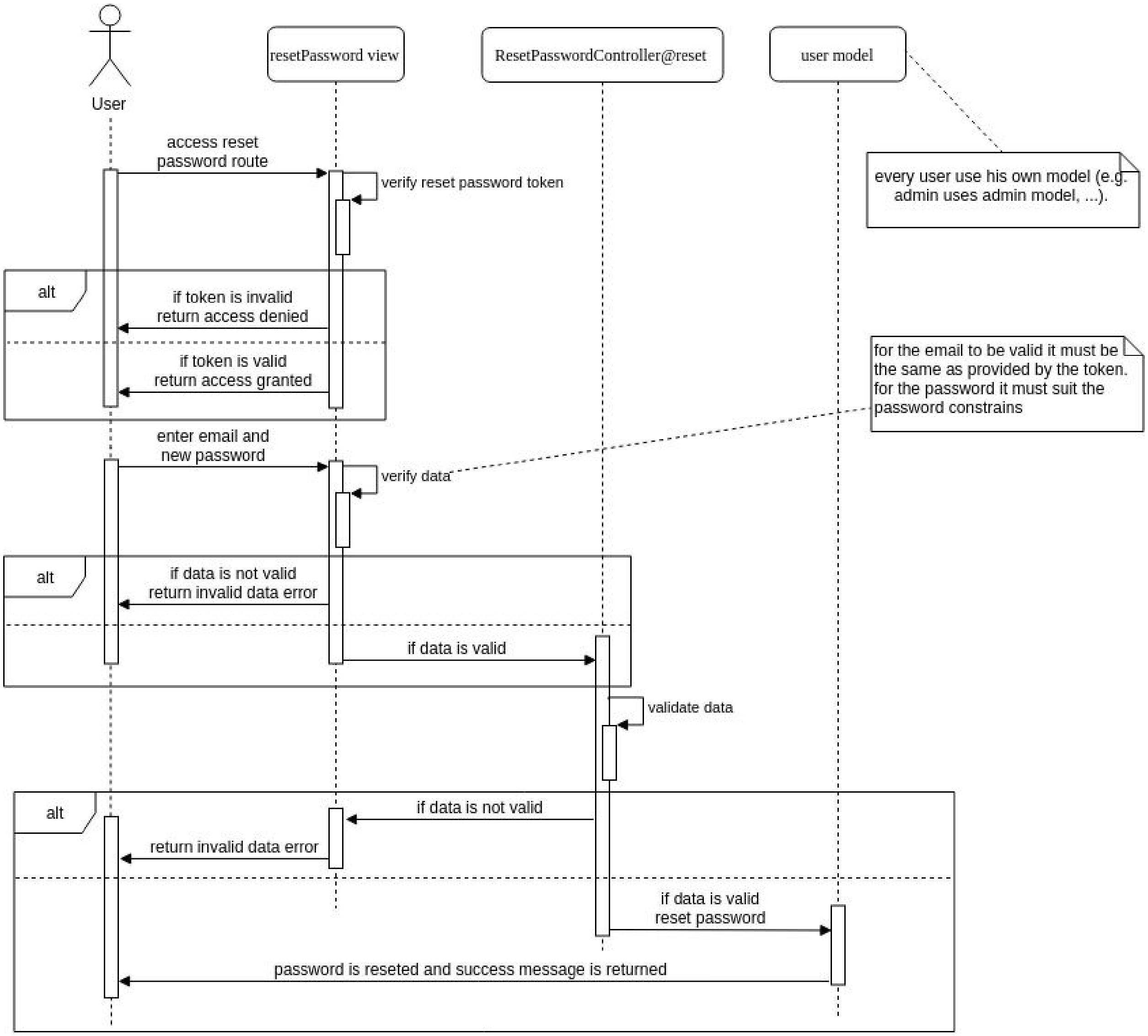
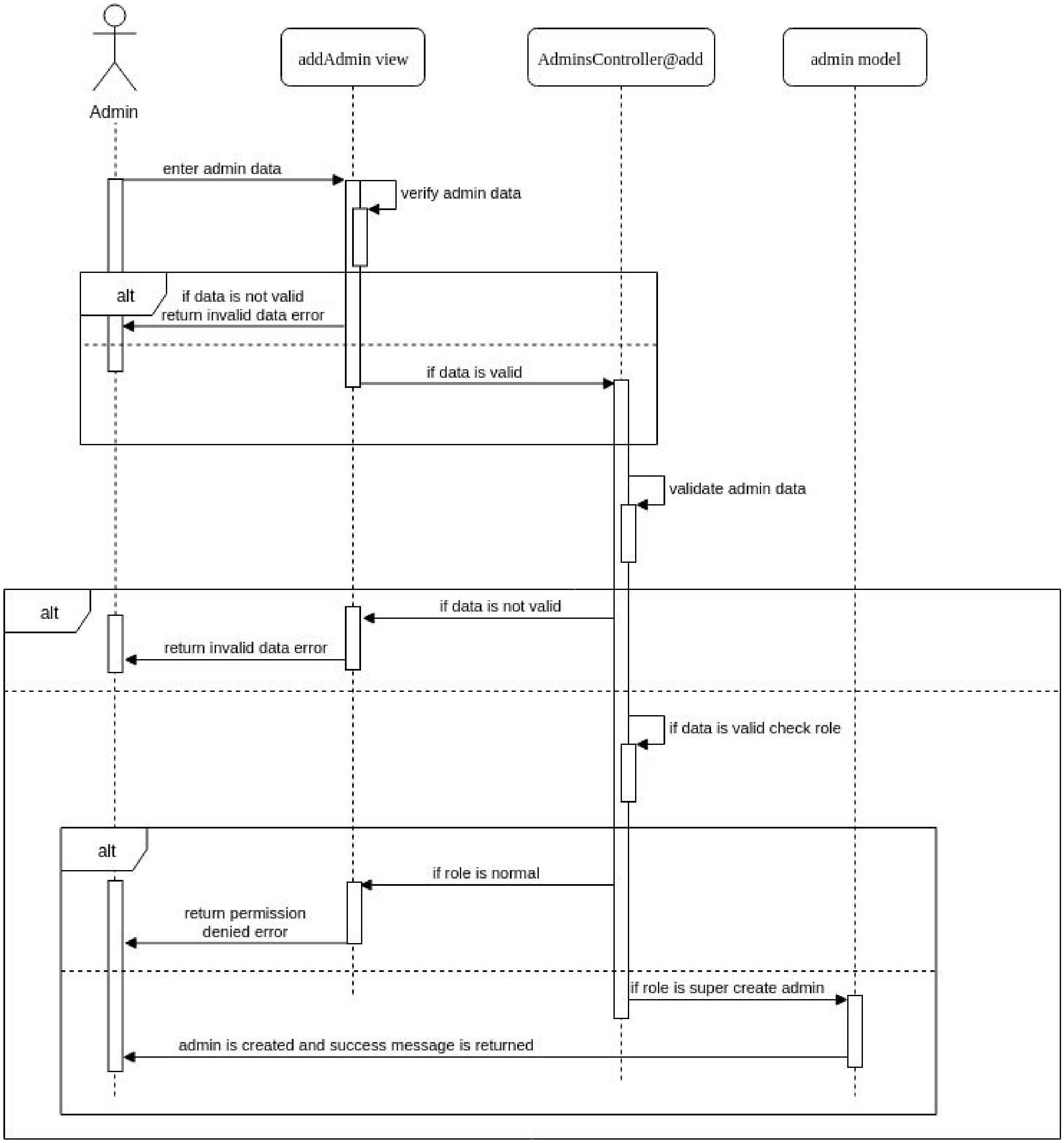

User
resetPassword view
ResetPasswordController@reset
user model
every user use his own model (e.g. admin uses admin model, ...).
access reset password route
verify reset password token
alt
if token is invalid return access denied
if token is valid return access granted
for the email to be valid it must be the same as provided by the token. for the password it must suit the password constrains
enter email and new password
verify data
alt
if data is not valid return invalid data error
if data is valid
validate data
alt
if data is not valid
return invalid data error
if data is valid reset password
password is reseted and success message is returned

Figure 7: SD2 reset password sequence diagram



Admin
addAdmin view
AdminsController@add
admin model
enter admin data
verify admin data
alt
if data is not valid
return invalid data error
if data is valid
validate admin data
alt
if data is not valid
return invalid data error
if data is valid check role
alt
if role is normal
return permission denied error
if role is super create admin
admin is created and success message is returned

Figure 8: SD3 create admin sequence diagram

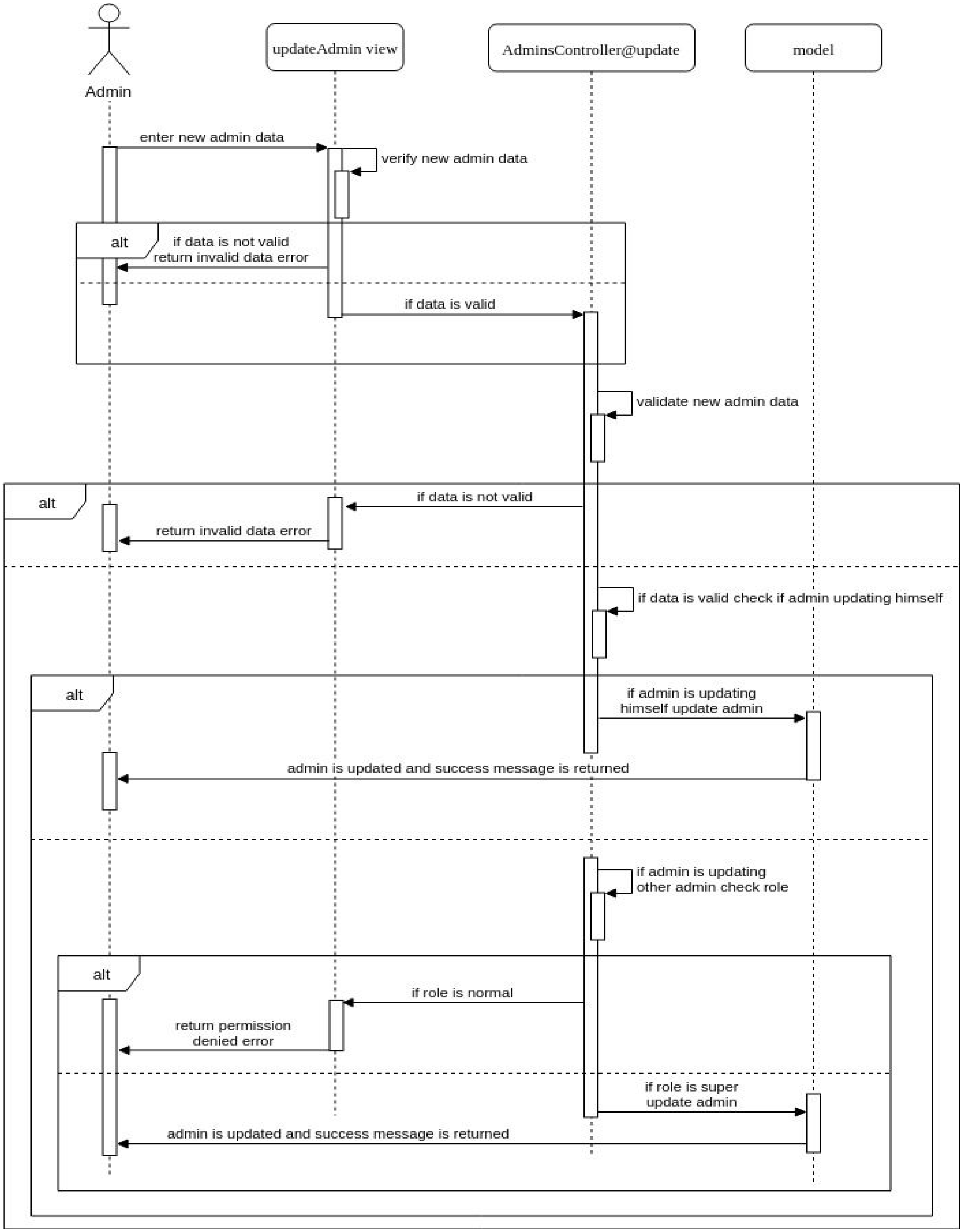


Figure 9: SD4 update admin sequence diagram

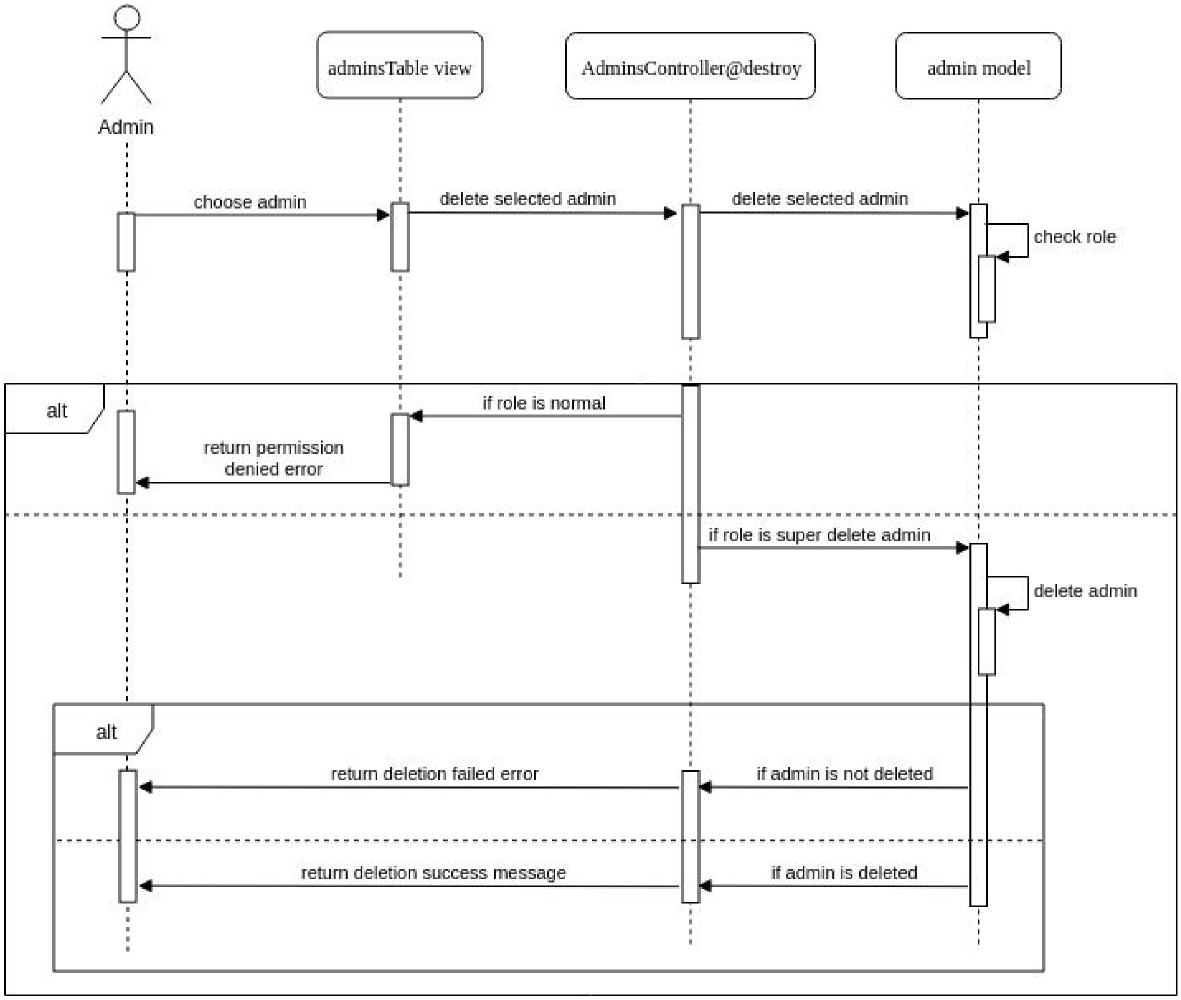


Figure 10: SD5 delete admin sequence diagram

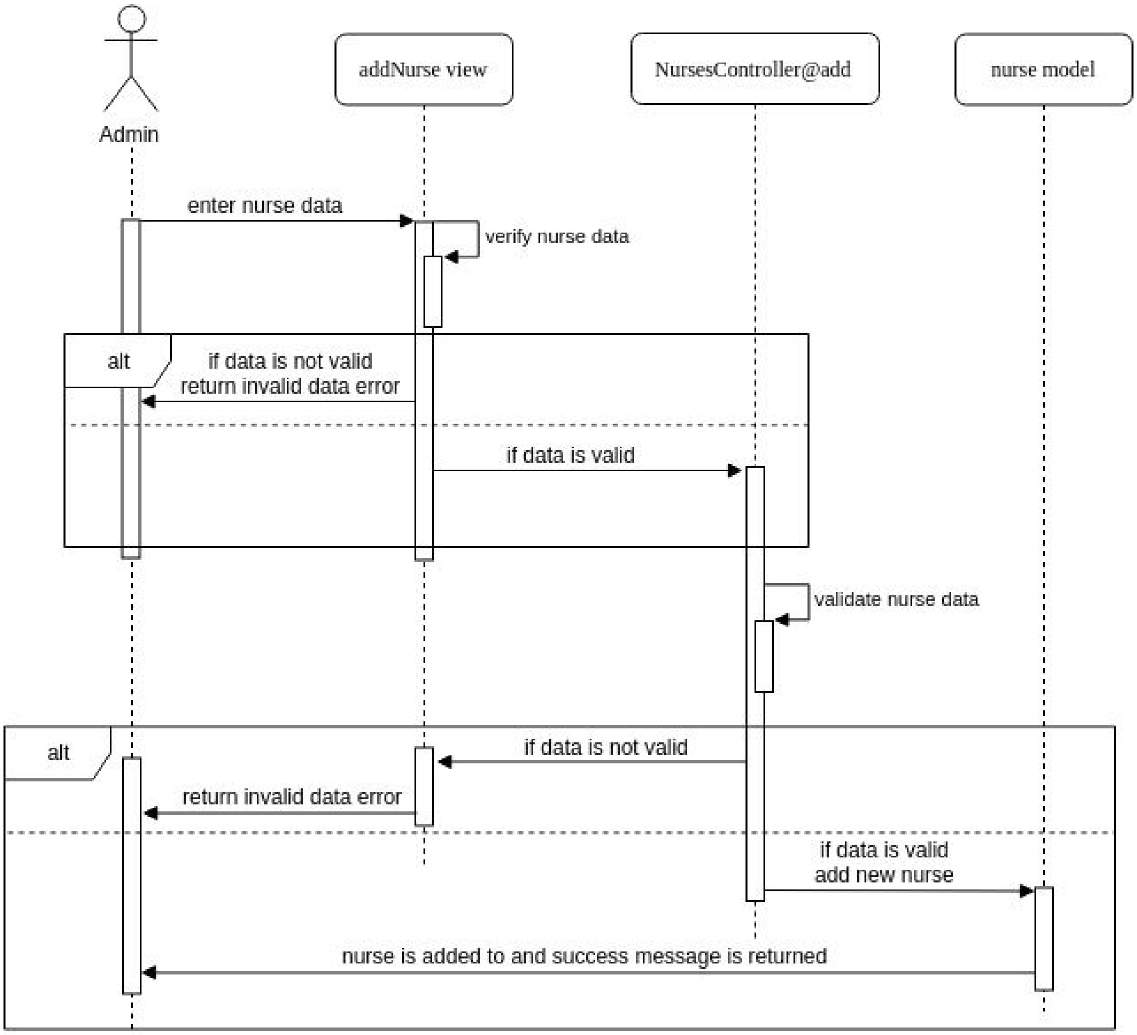


Figure 11: SD6 create nurse sequence diagram

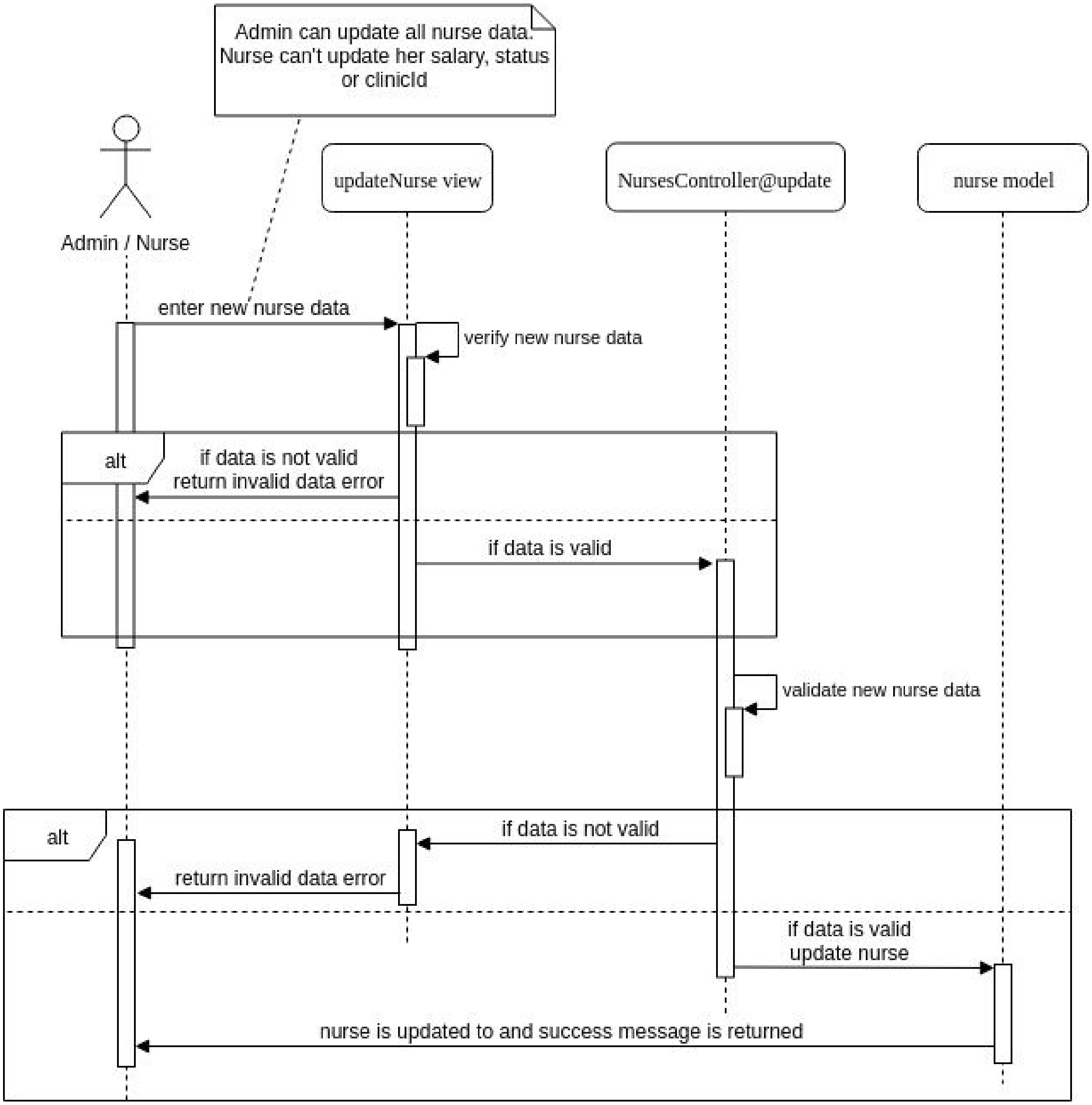


Figure 12: SD7 update nurse sequence diagram

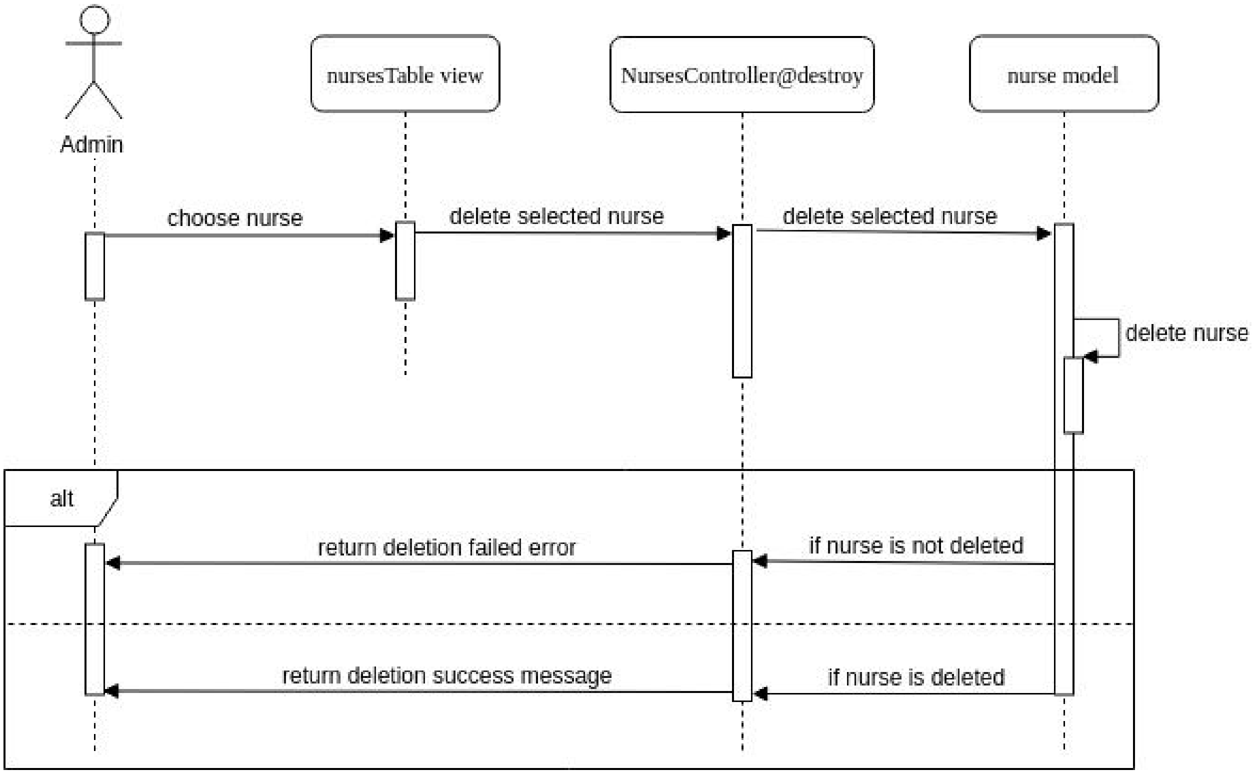


Figure 13: SD8 delete nurse sequence diagram

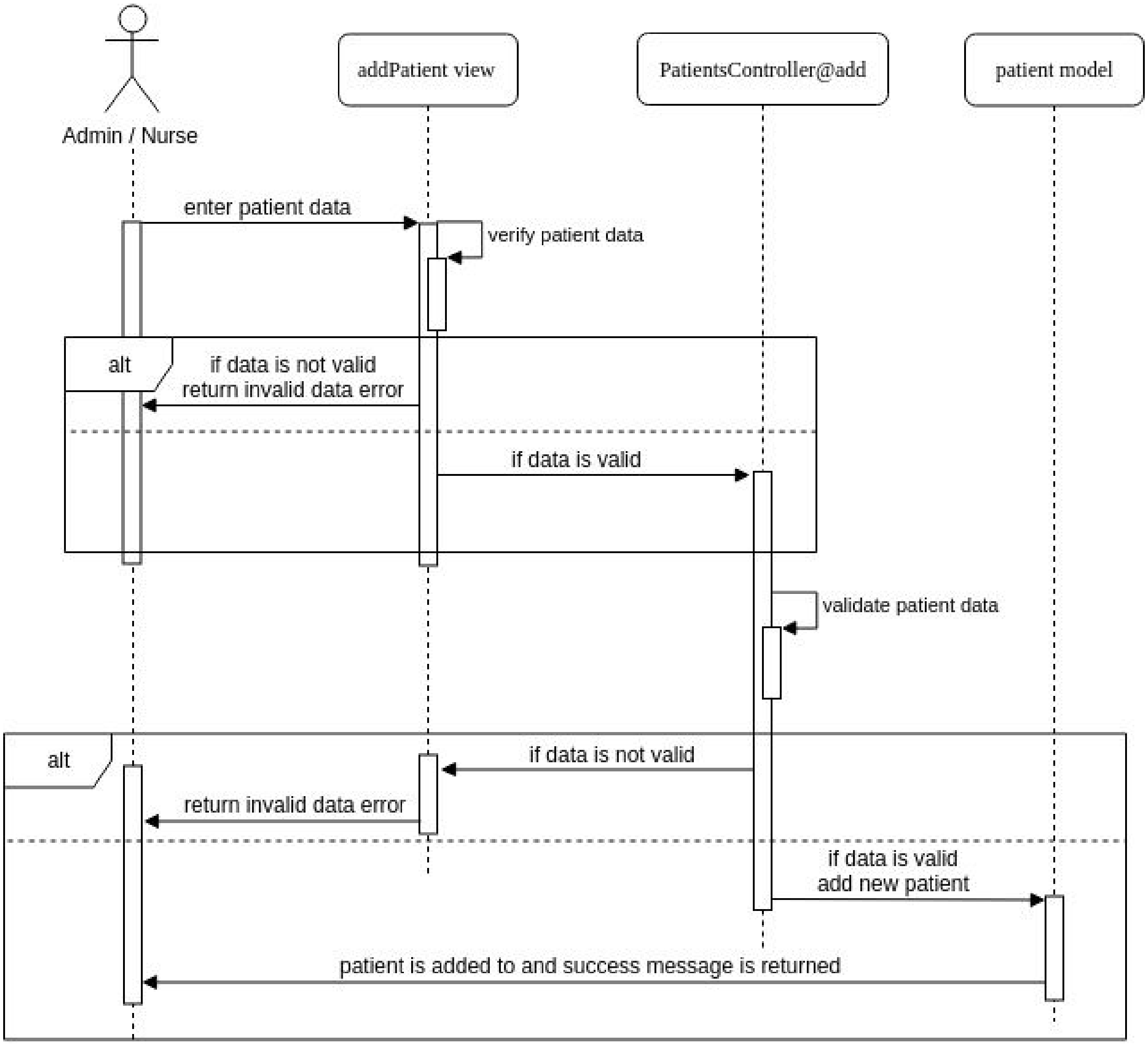


Figure 14: SD9 create patient sequence diagram

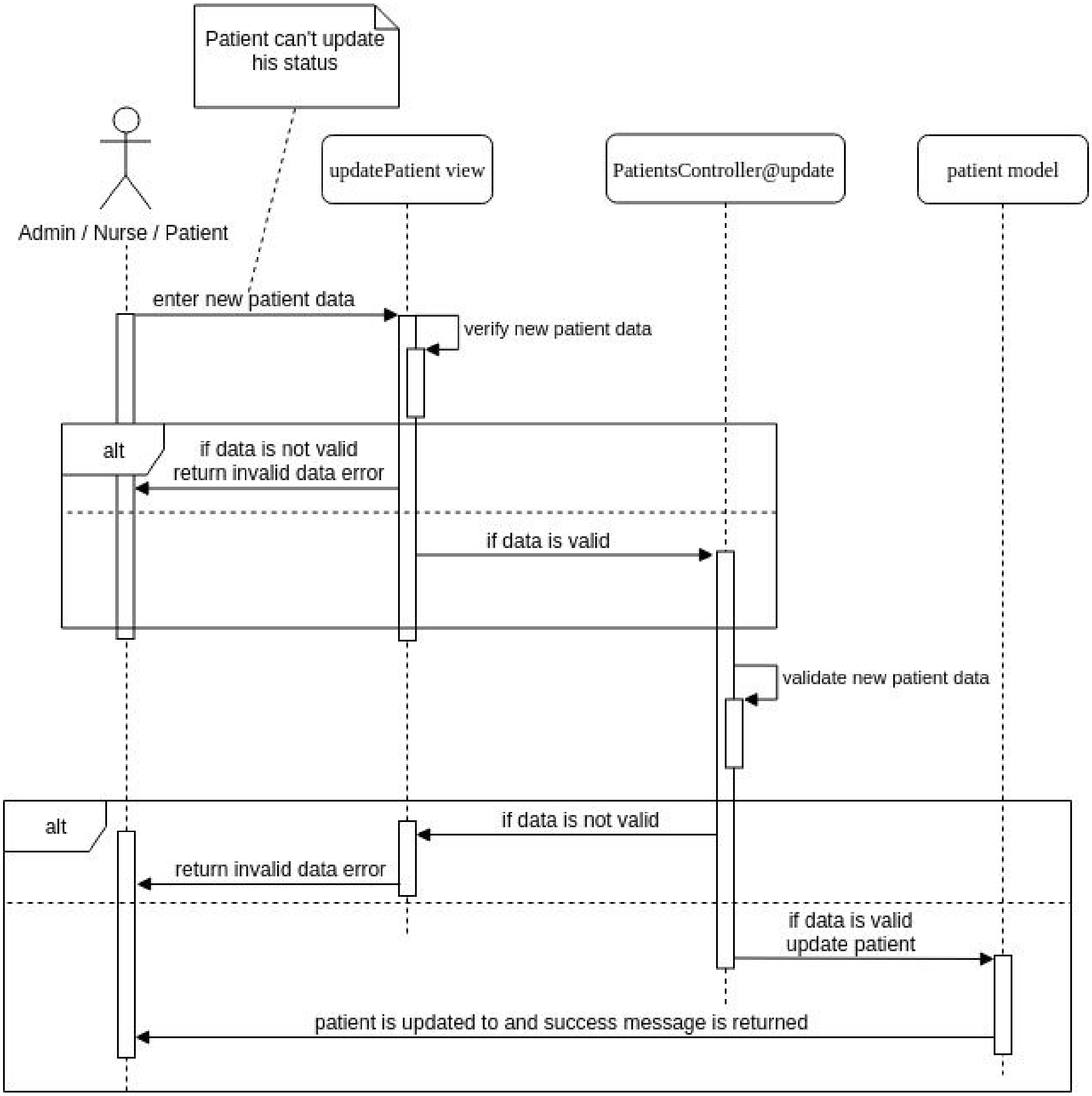


Figure 15: SD10 update patient sequence diagram

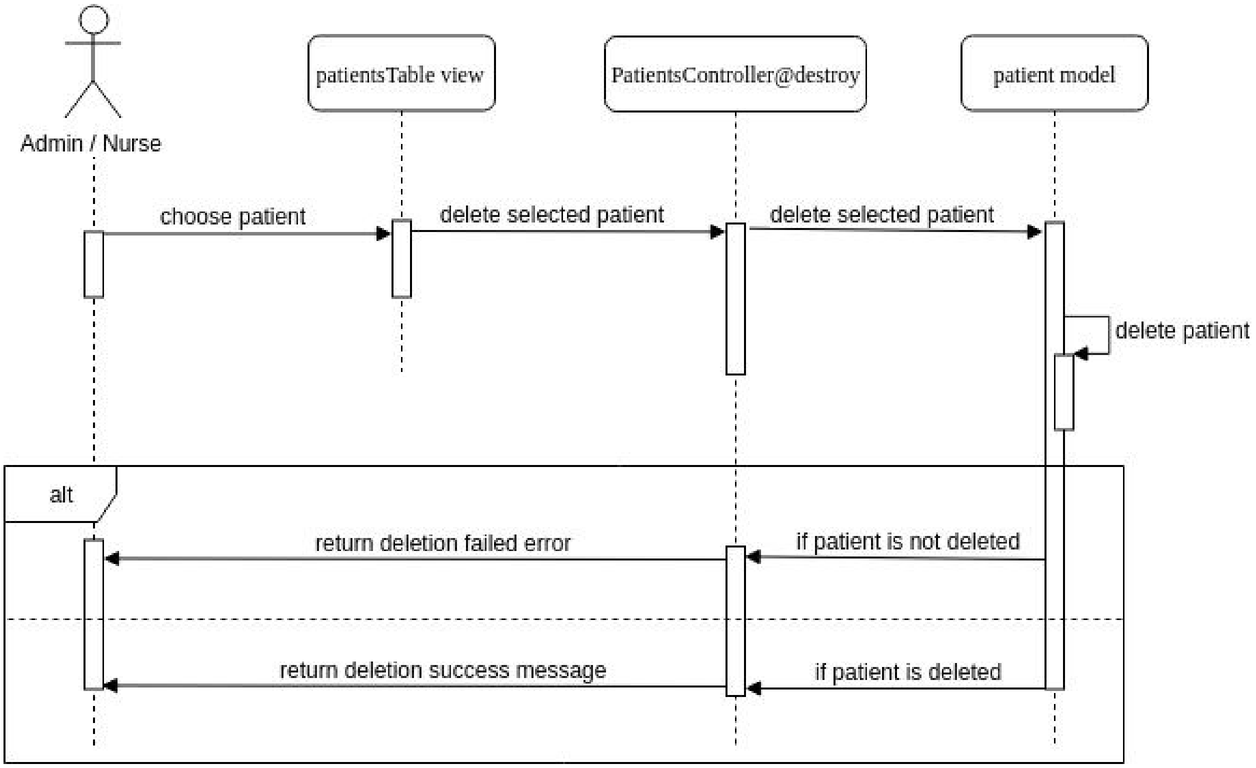


Figure 16: SD11 delete patient sequence diagram

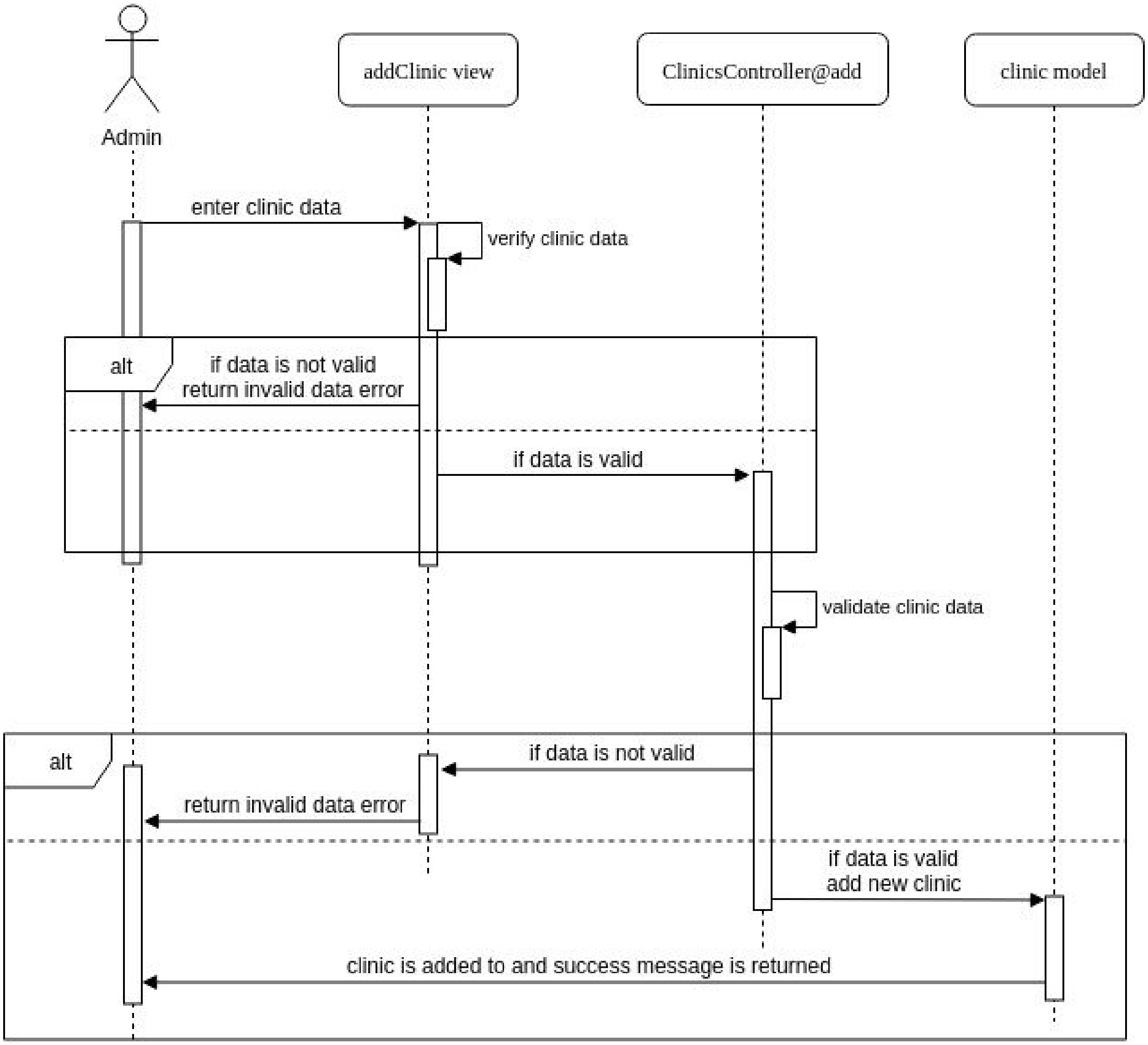


Figure 17: SD12 create clinic sequence diagram

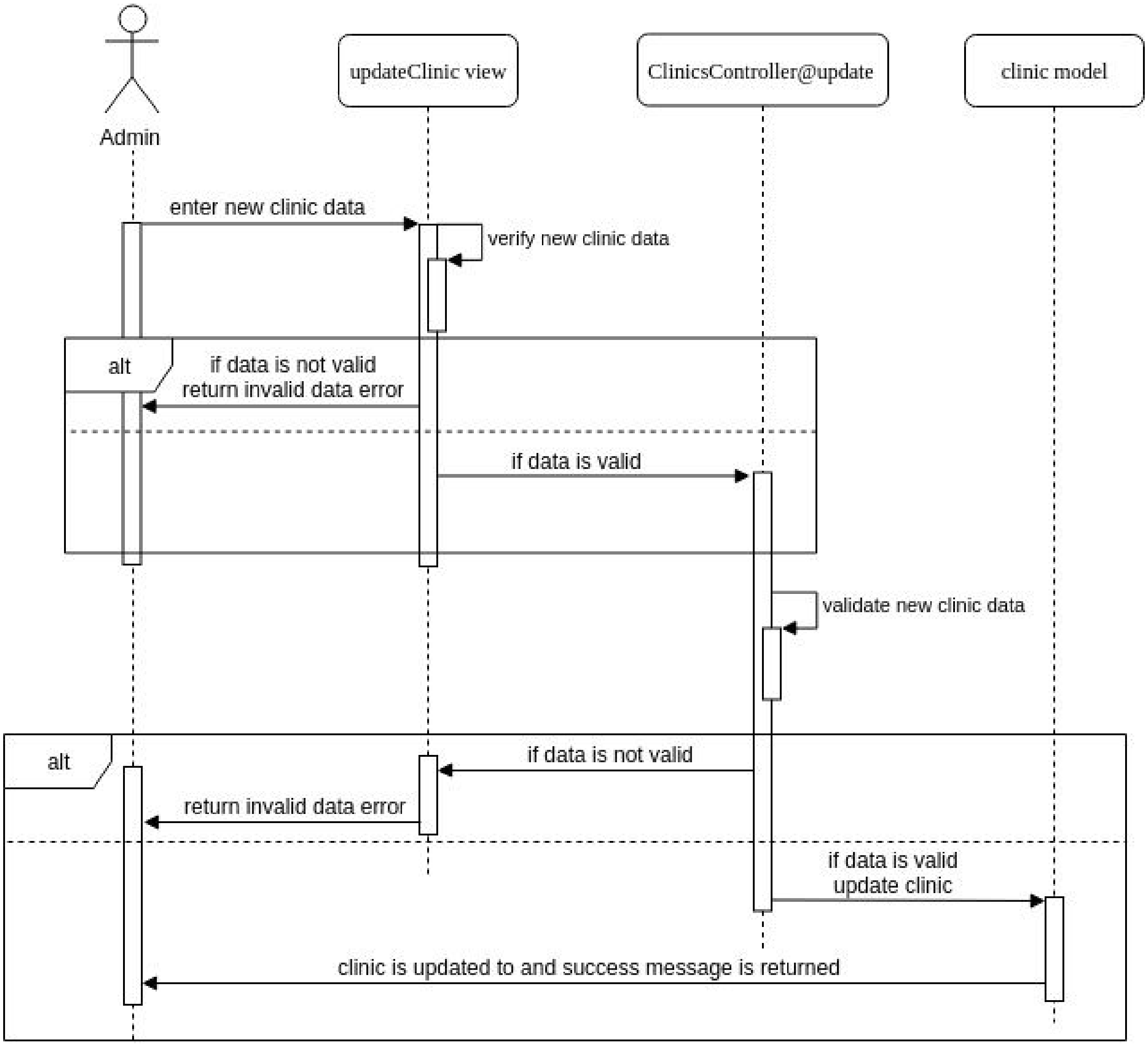


Figure 18: SD13 update clinic sequence diagram

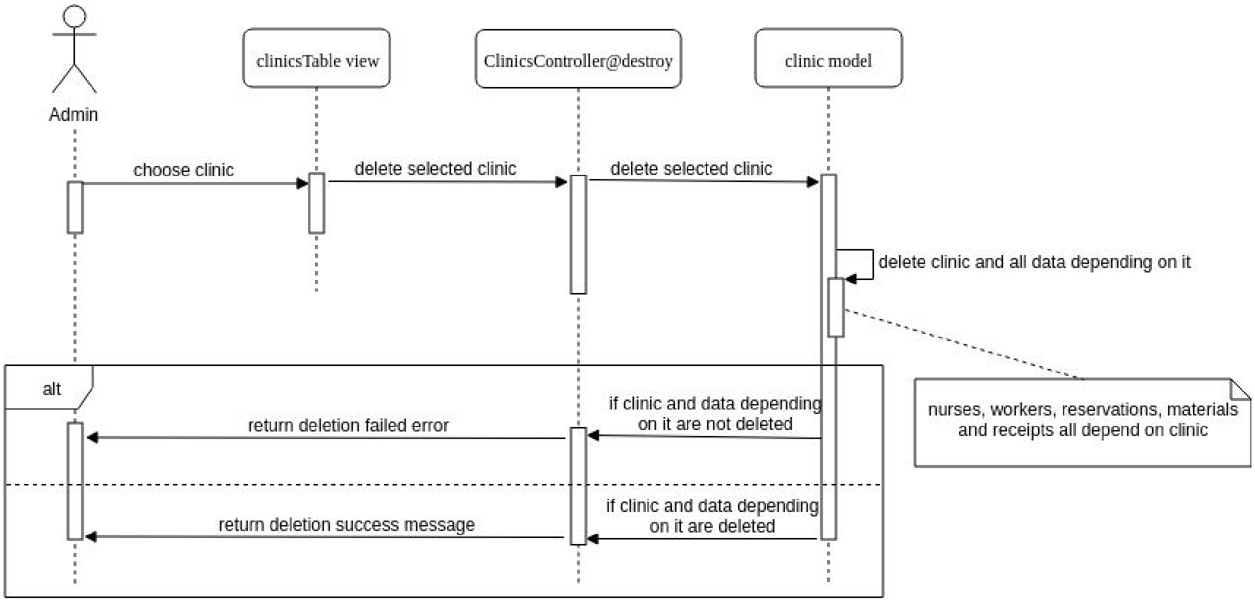

Admin
clinicsTable view
ClinicsController@destroy
clinic model
choose clinic
delete selected clinic
delete selected clinic
delete clinic and all data depending on it
nurses, workers, reservations, materials and receipts all depend on clinic
alt
return deletion failed error
if clinic and data depending on it are not deleted
return deletion success message
if clinic and data depending on it are deleted

Figure 19: SD14 delete clinic sequence diagram

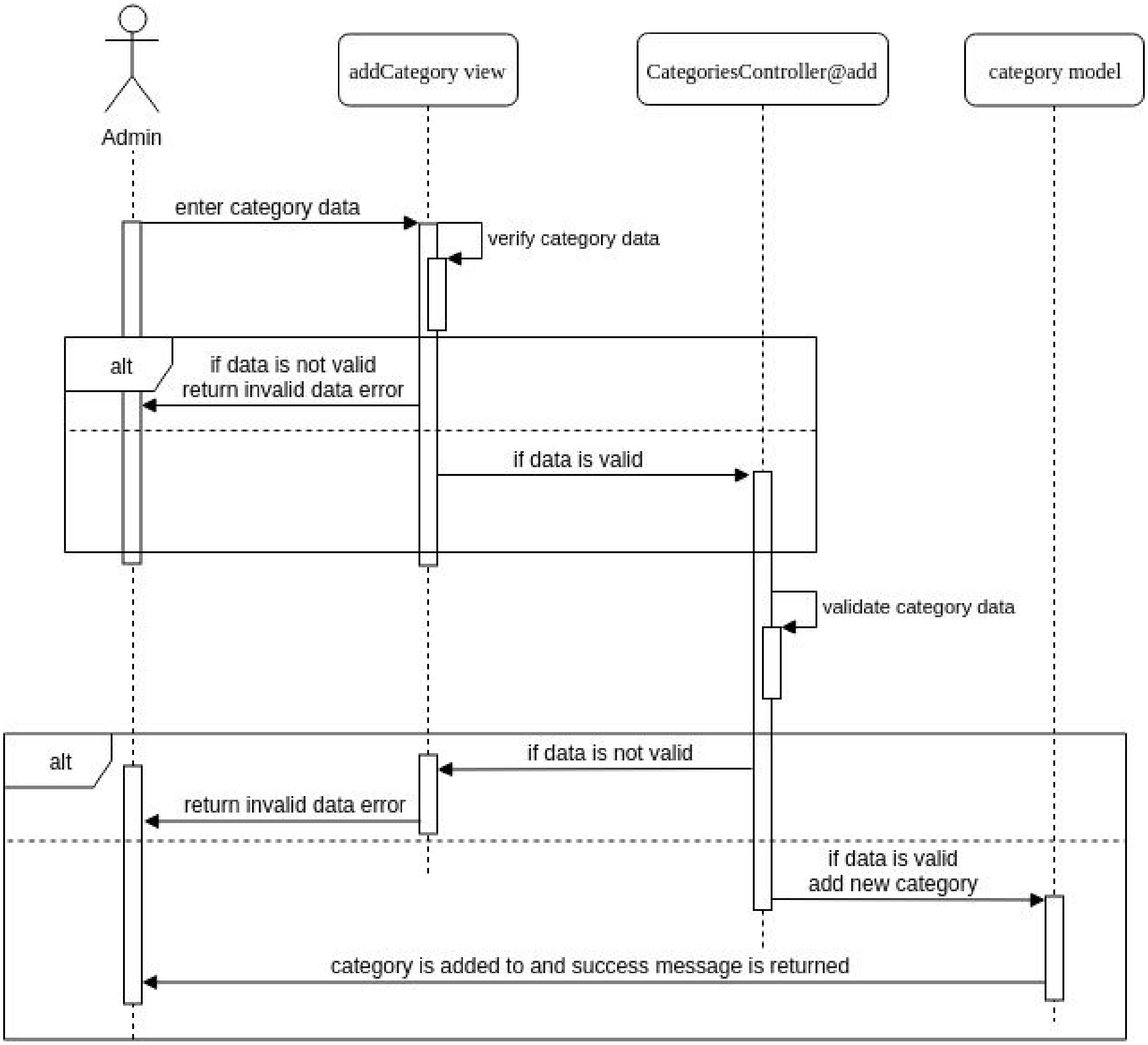


Figure 20: SD15 create category sequence diagram

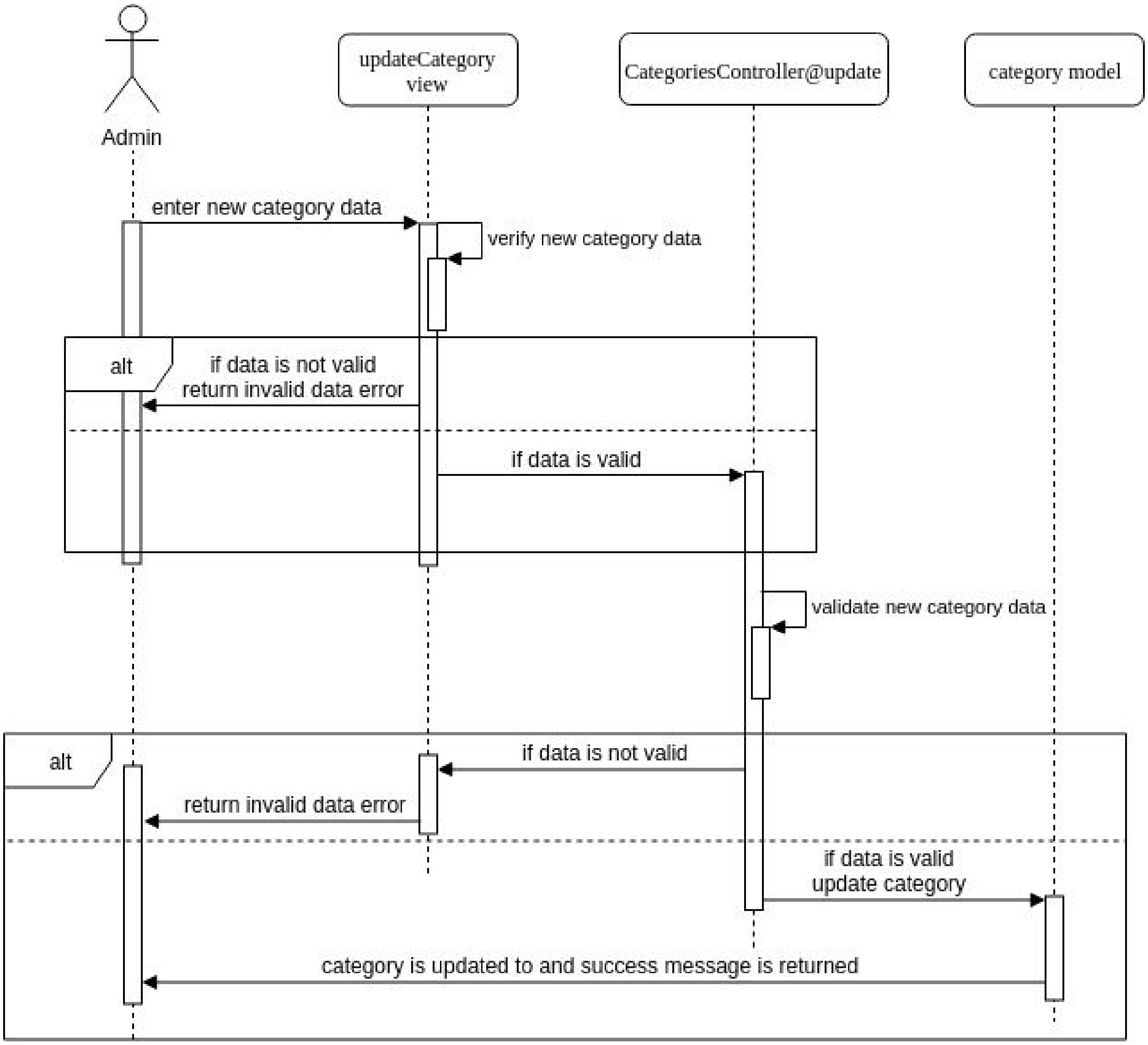
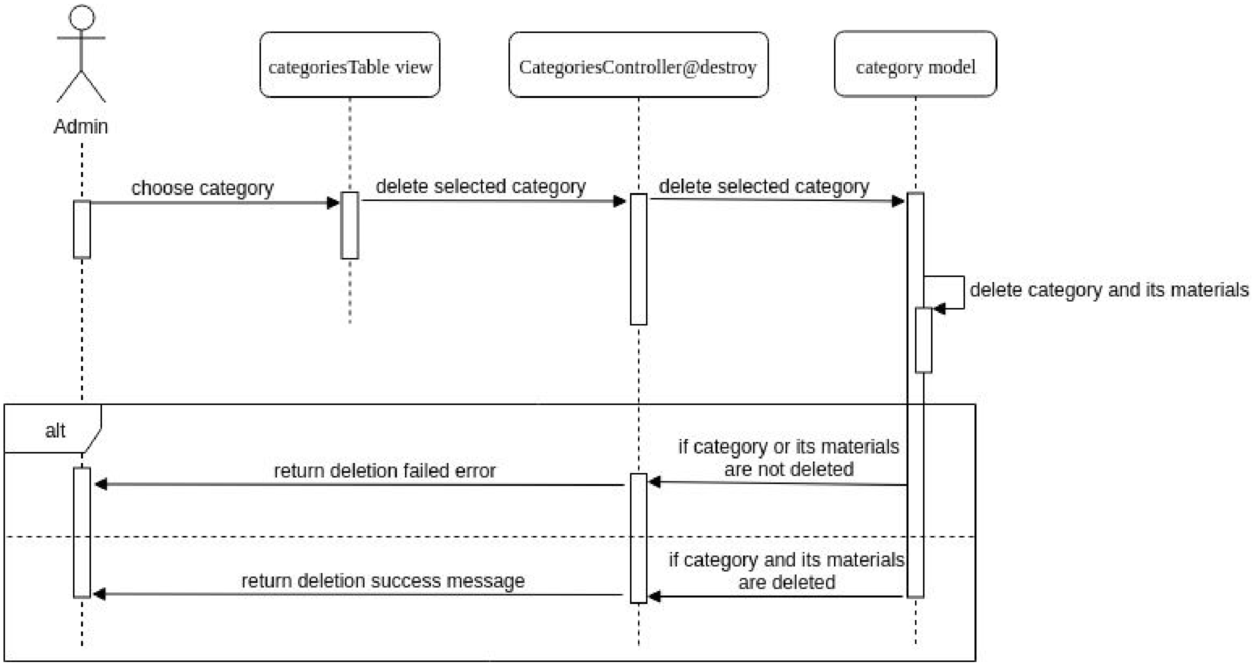


Figure 21: SD16 update category sequence diagram



Admin
categoriesTable view
CategoriesController@destroy
category model
choose category
delete selected category
delete selected category
delete category and its materials
alt
return deletion failed error
if category or its materials are not deleted
return deletion success message
if category and its materials are deleted

Figure 22: SD17 delete category sequence diagram

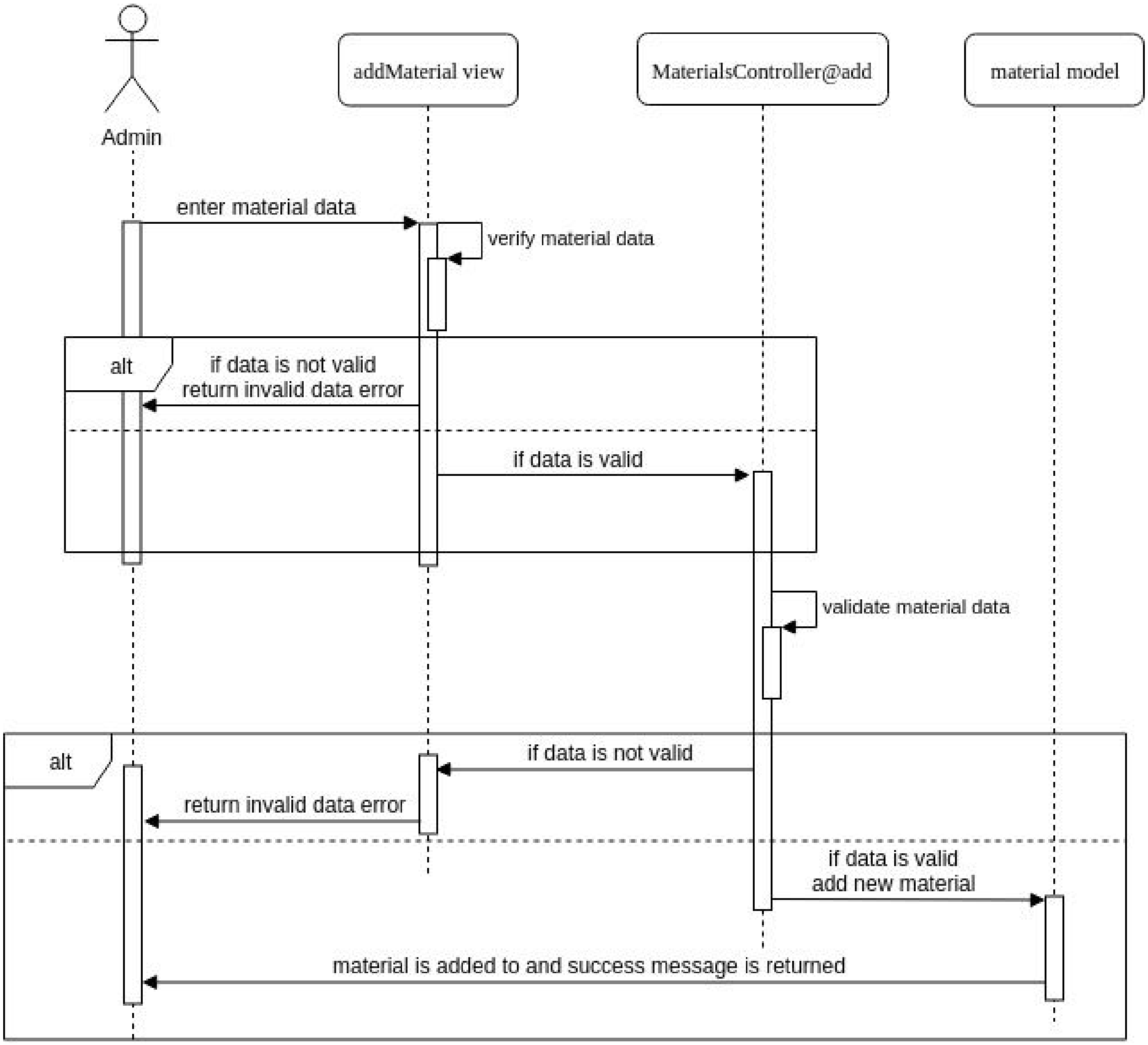


Figure 23: SD18 create material sequence diagram

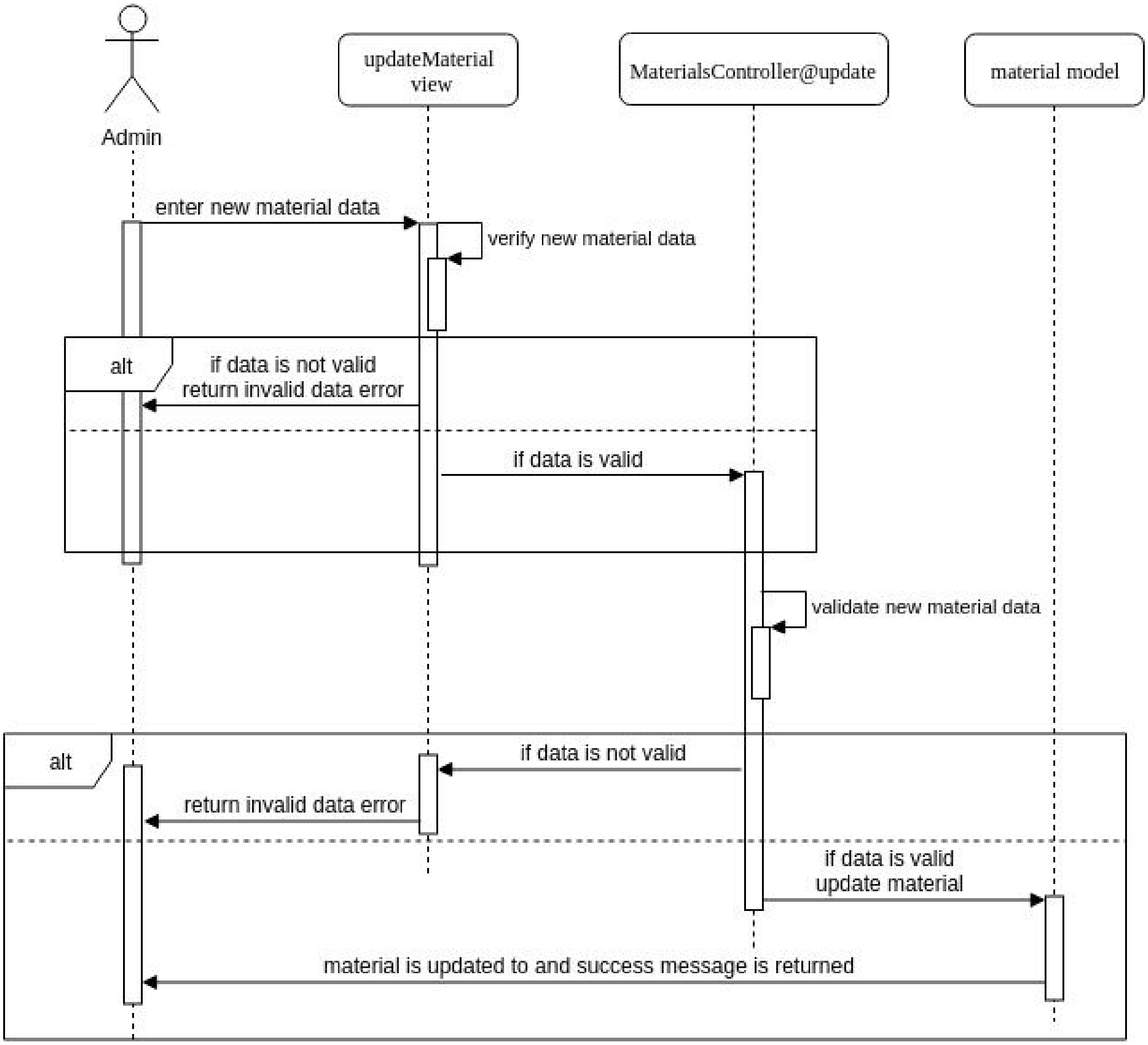


Figure 24: SD19 update material sequence diagram

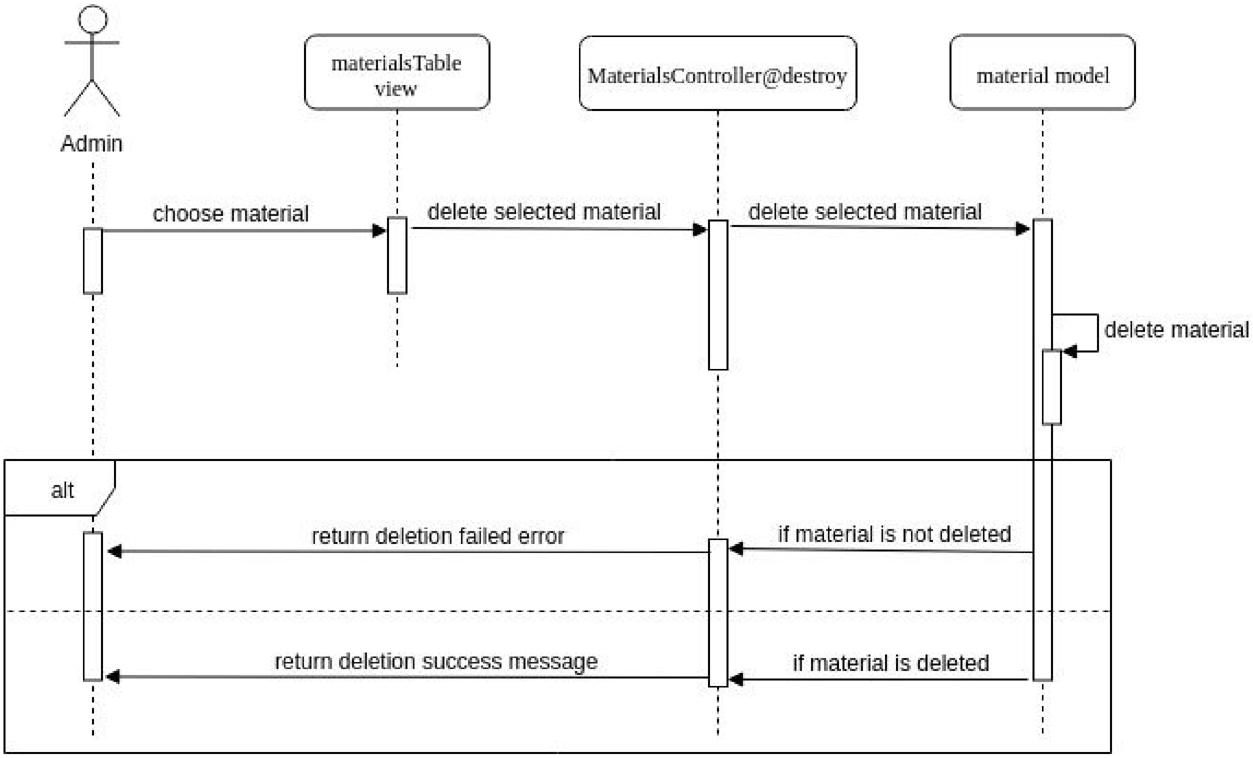


Figure 25: SD20 delete material sequence diagram

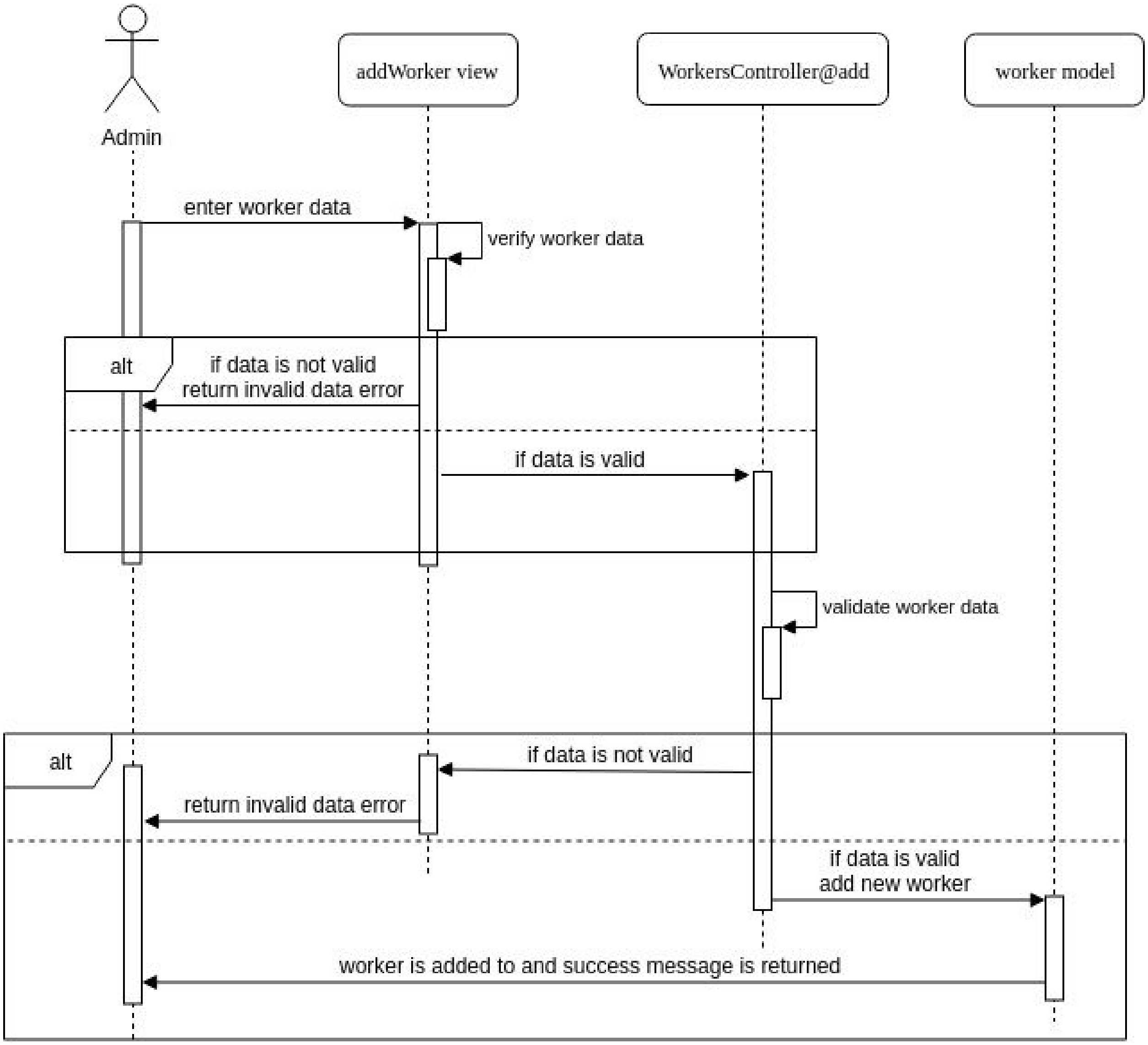


Figure 26: SD21 create worker sequence diagram

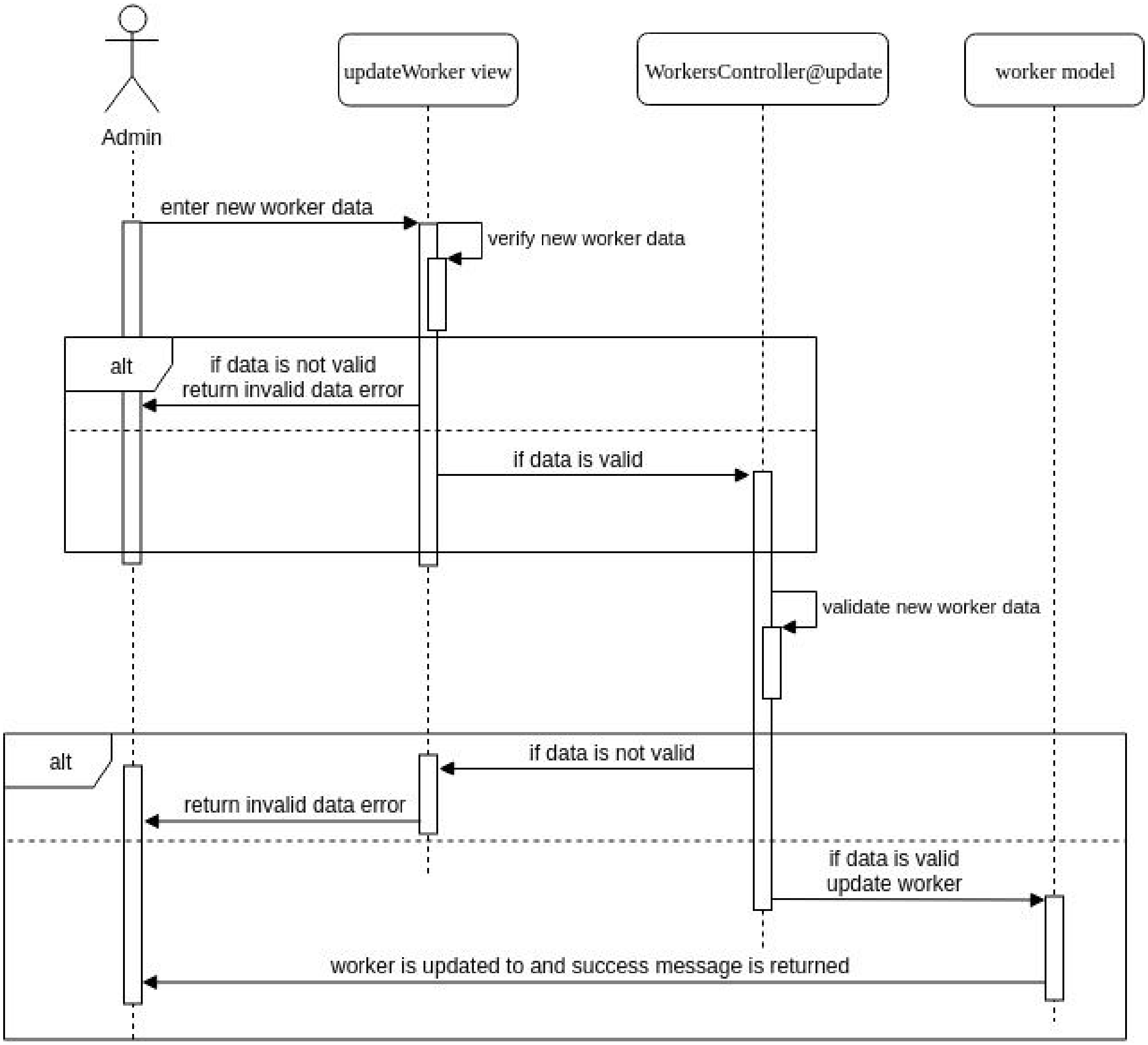


Figure 27: SD22 update worker sequence diagram

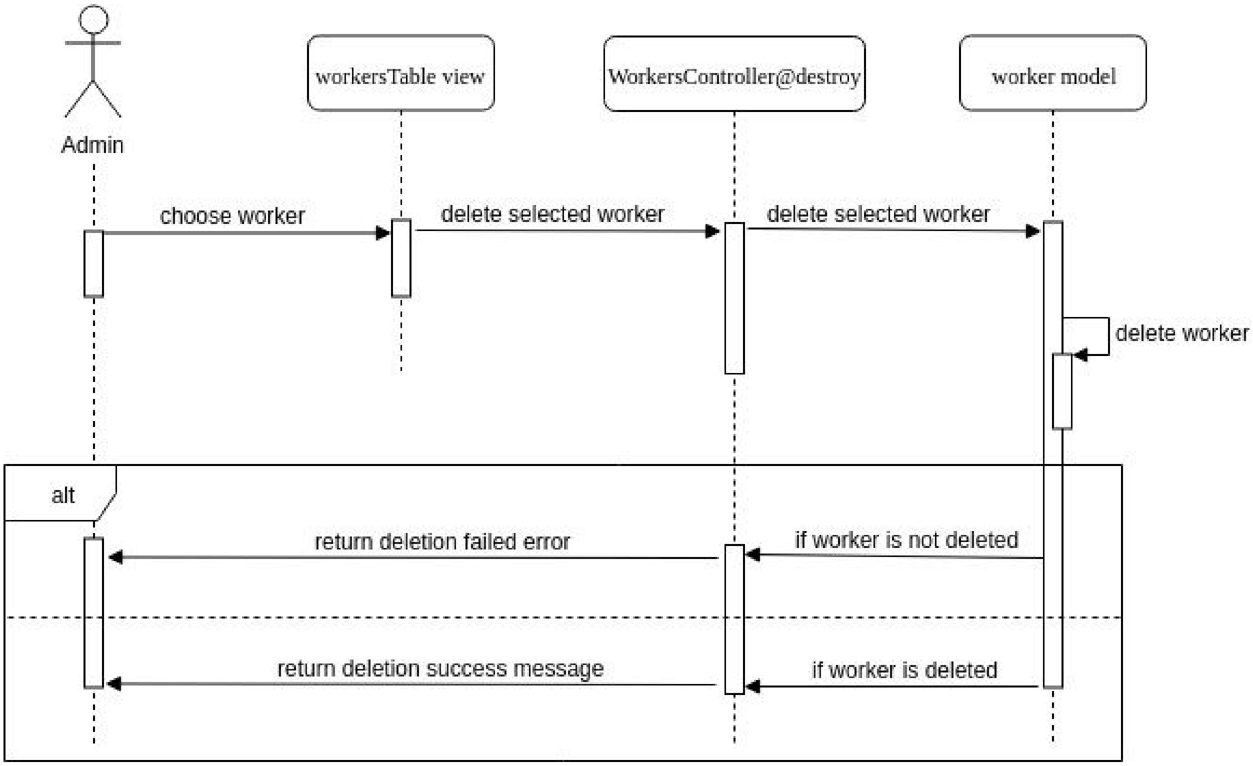


Figure 28: SD23 delete worker sequence diagram

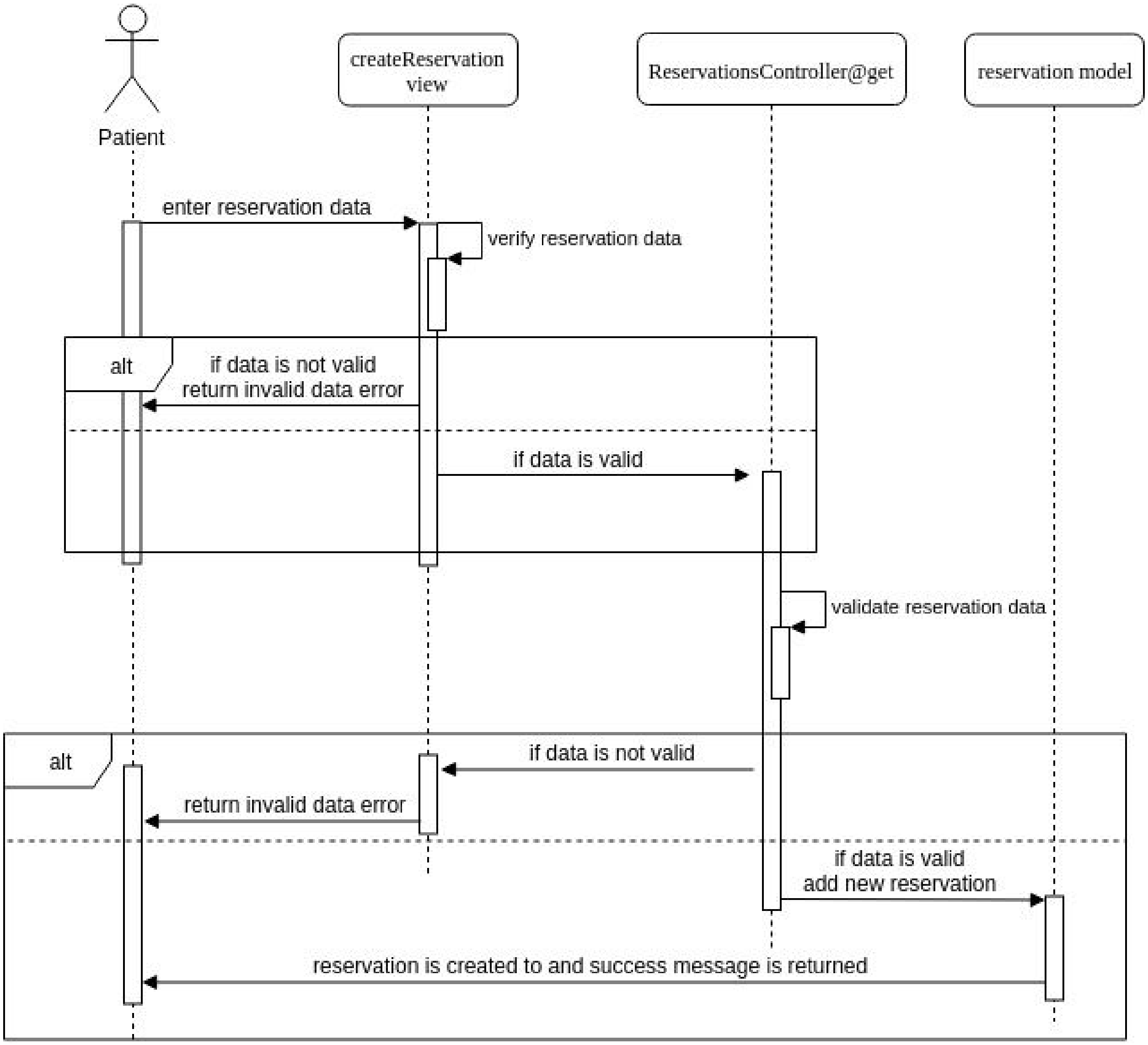


Figure 29: SD24 create reservation sequence diagram

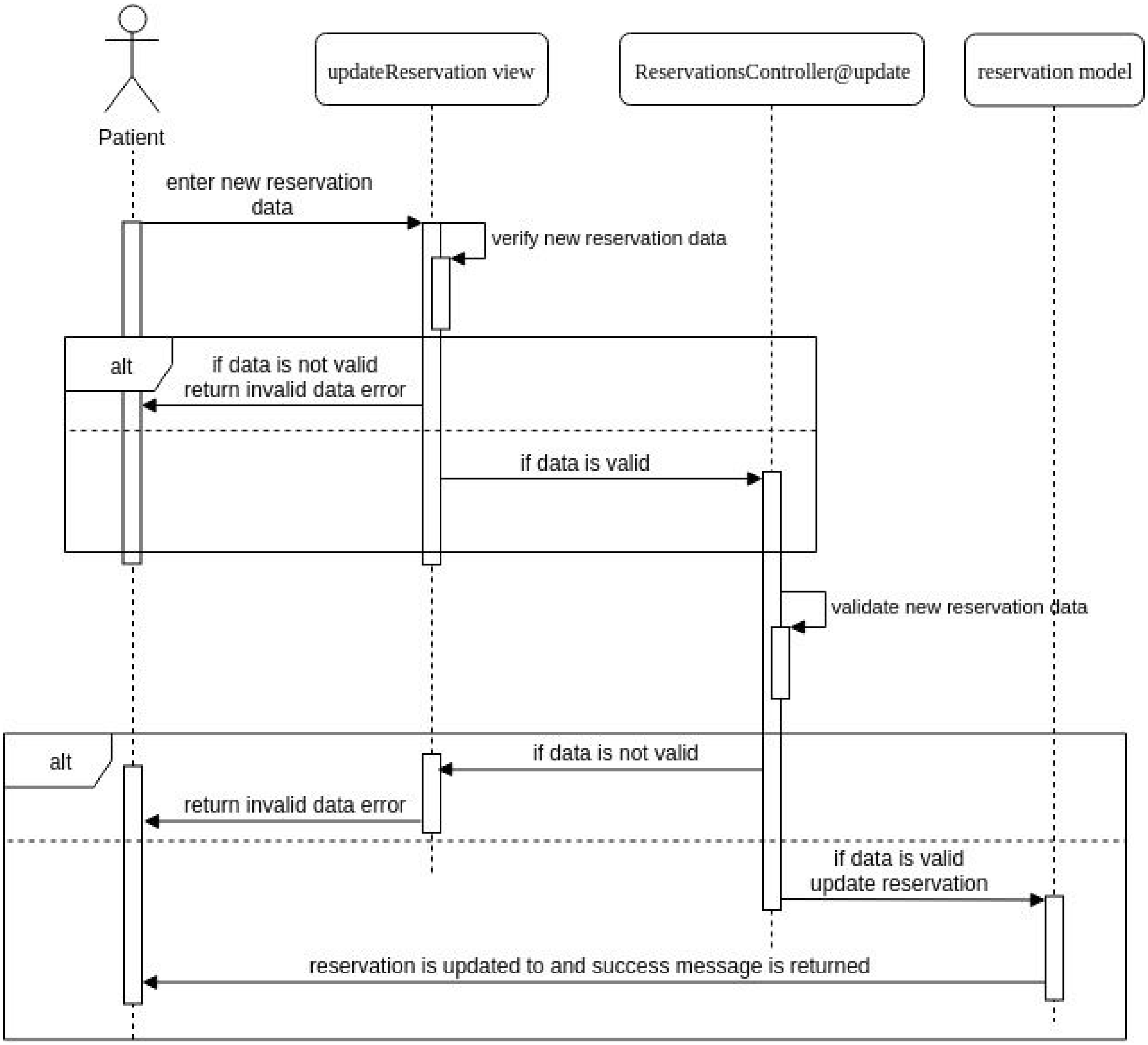
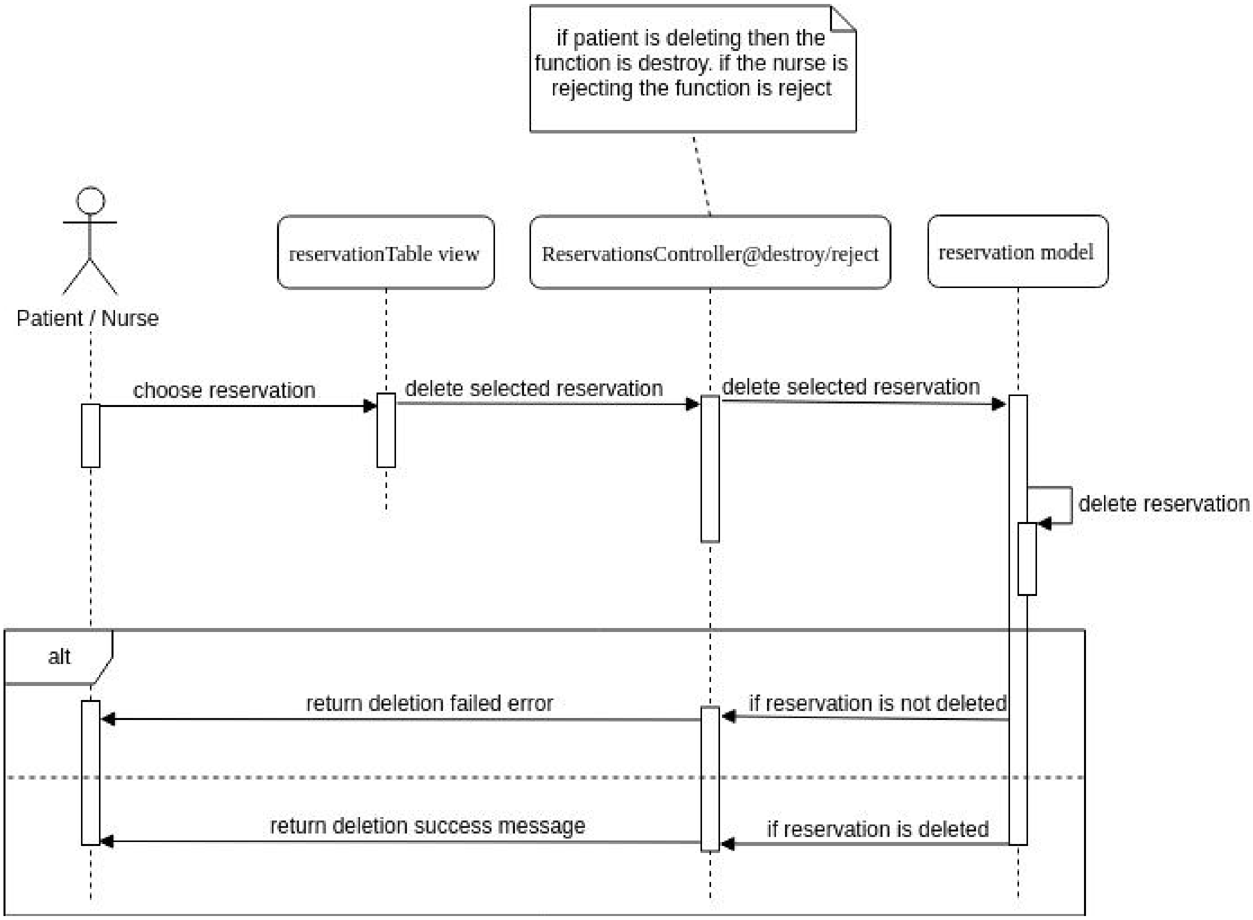
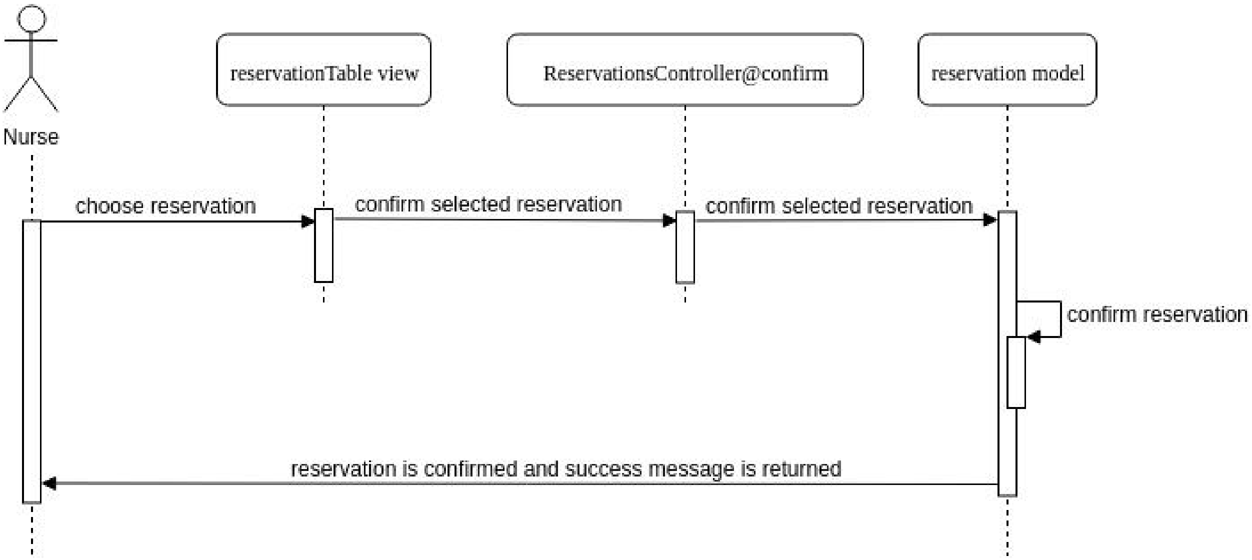

Patient
updateReservation view
ReservationsController@update
reservation model
enter new reservation data
verify new reservation data
alt
if data is not valid
return invalid data error
if data is valid
validate new reservation data
alt
if data is not valid
return invalid data error
if data is valid
update reservation
reservation is updated to and success message is returned

Figure 30: SD25 update reservation sequence diagram



if patient is deleting then the function is destroy. if the nurse is rejecting the function is reject
Patient / Nurse
reservationTable view
ReservationsController@destroy/reject
reservation model
choose reservation
delete selected reservation
delete selected reservation
delete reservation
alt
return deletion failed error
if reservation is not deleted
return deletion success message
if reservation is deleted

Figure 31: SD26 delete reservation sequence diagram



Nurse
reservationTable view
ReservationsController@confirm
reservation model
choose reservation
confirm selected reservation
confirm selected reservation
confirm reservation
reservation is confirmed and success message is returned

Figure 32: SD27 confirm reservation sequence diagram

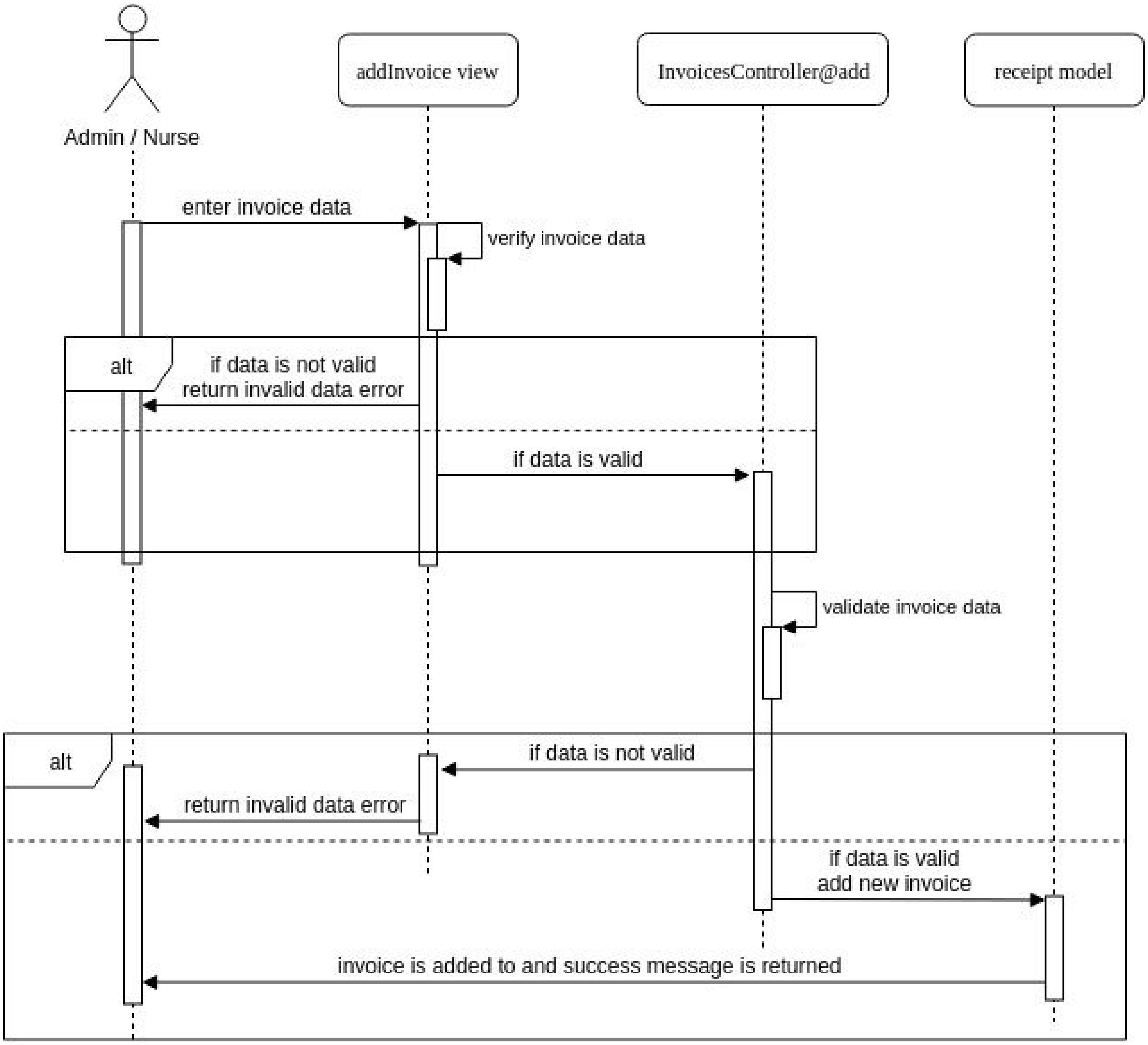


Figure 33: SD28 create invoice sequence diagram

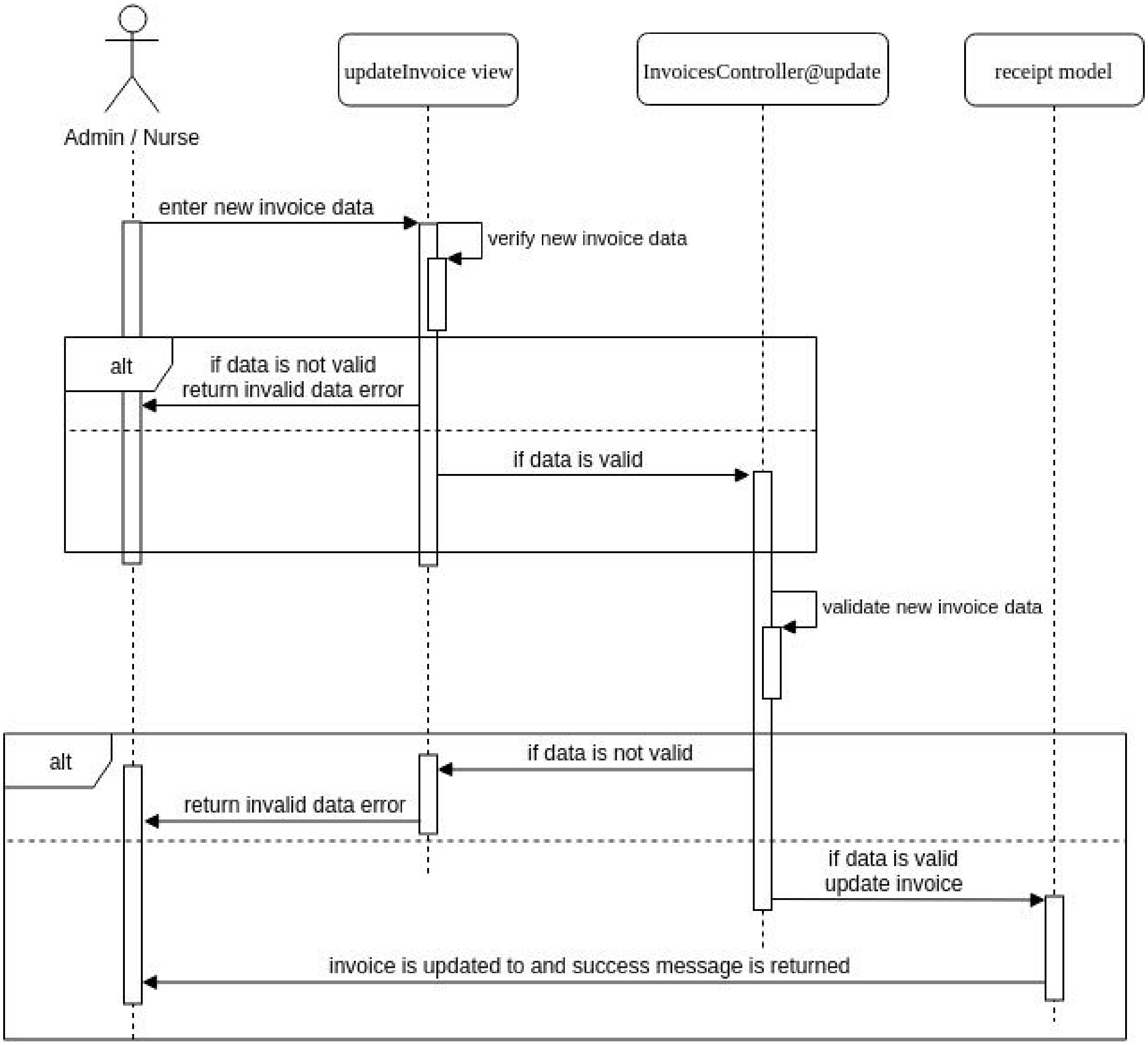


Figure 34: SD29 update invoice sequence diagram

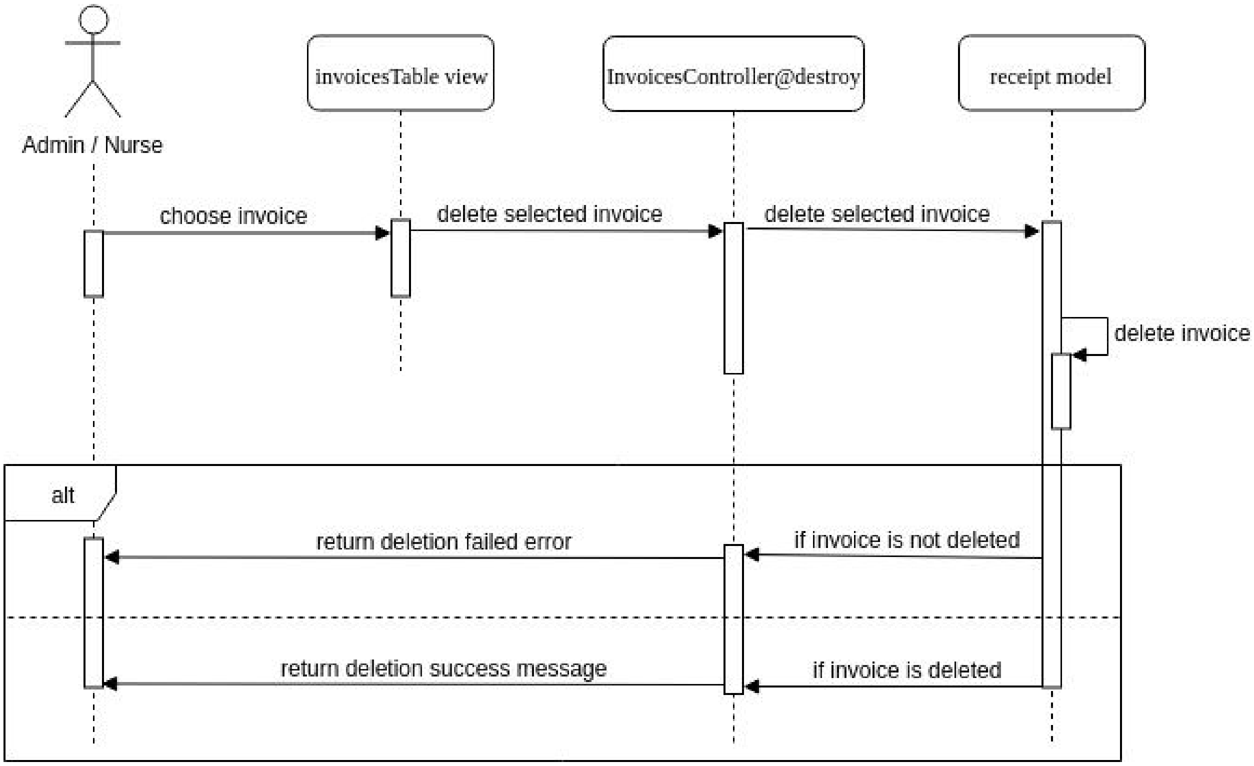


Figure 35: SD30 delete invoice sequence diagram

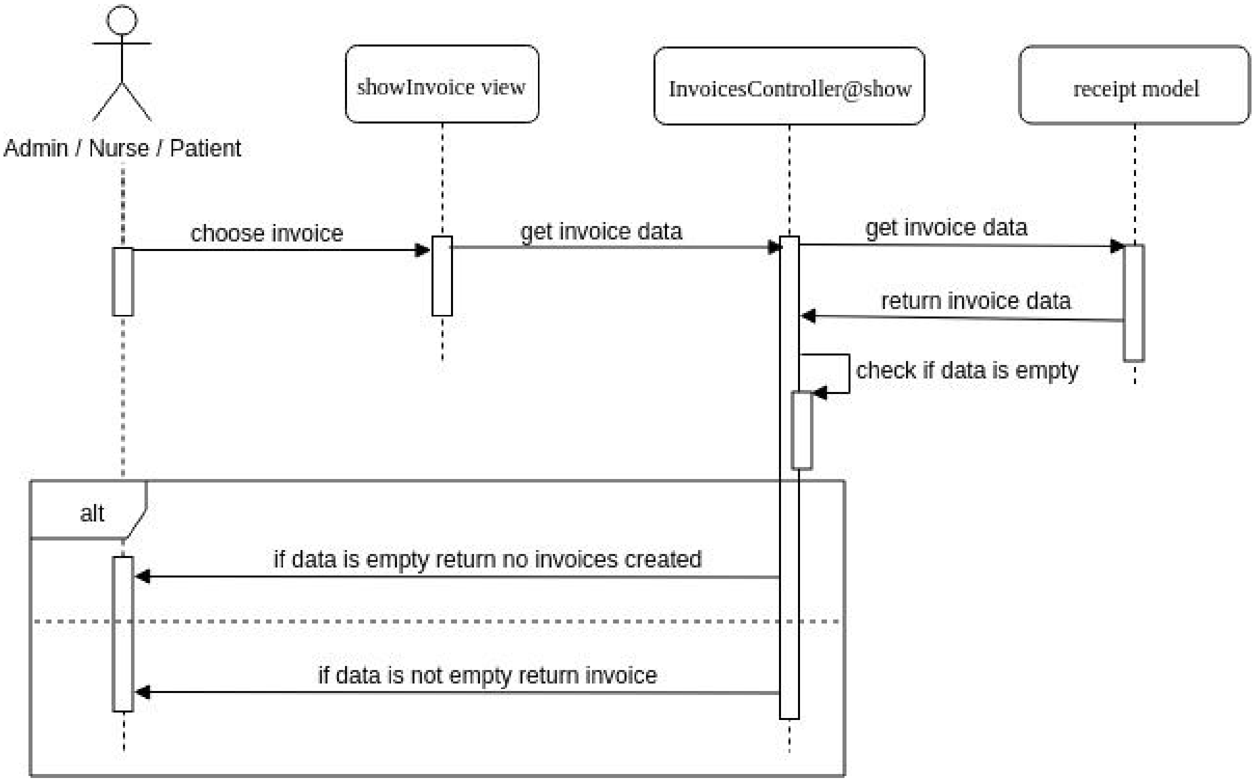


Figure 36: SD31 show invoice sequence diagram

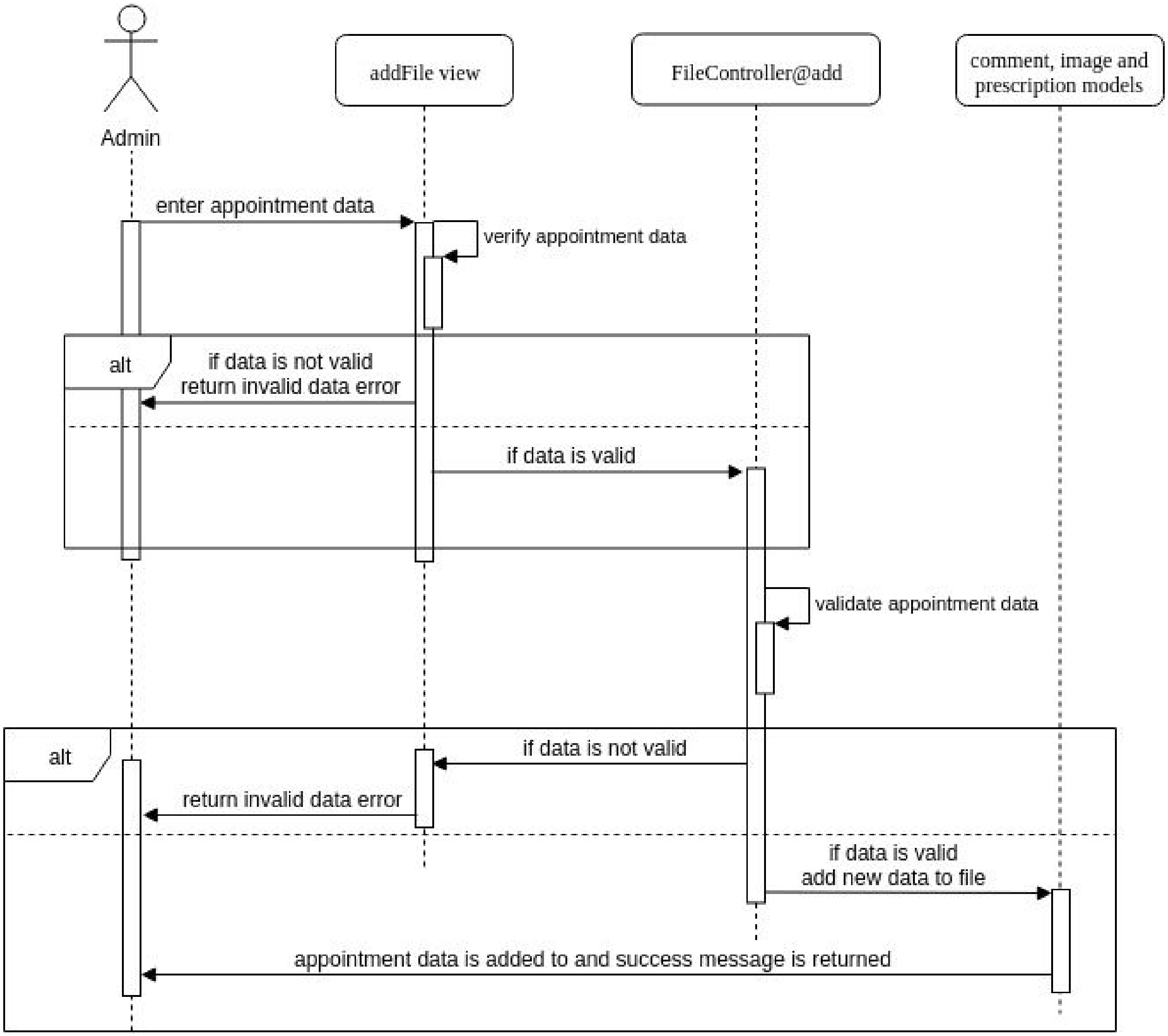
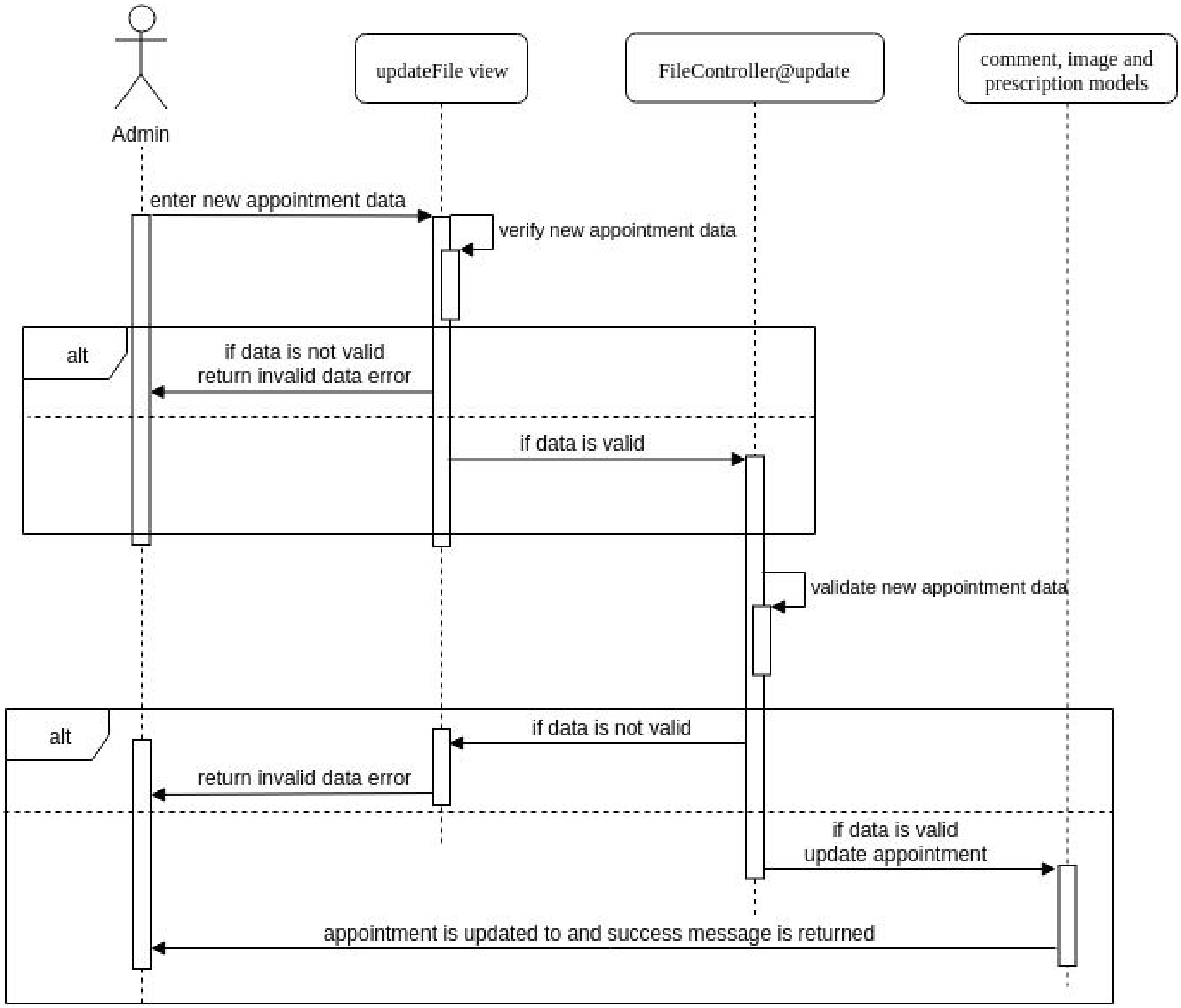


Figure 37: SD32 create file sequence diagram



Admin
updateFile view
FileController@update
comment, image and prescription models
enter new appointment data
verify new appointment data
alt
if data is not valid
return invalid data error
if data is valid
validate new appointment data
alt
if data is not valid
return invalid data error
if data is valid
update appointment
appointment is updated to and success message is returned

Figure 38: SD33 update file sequence diagram

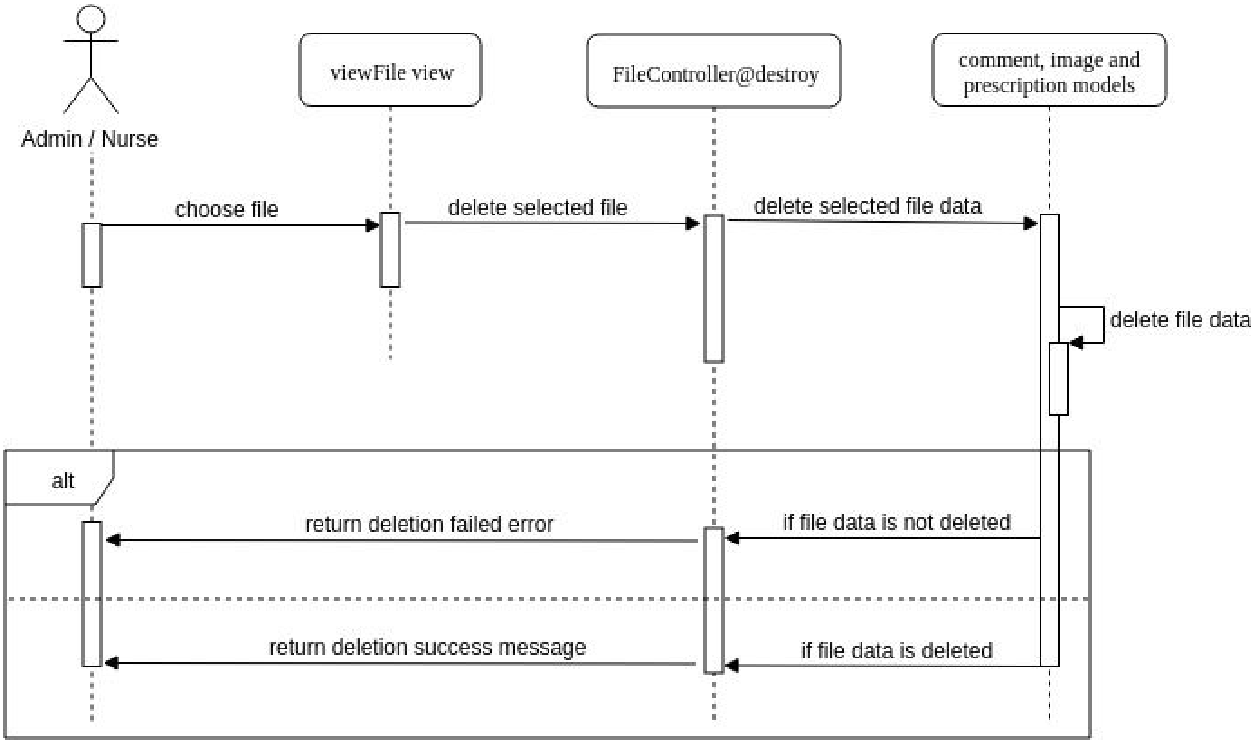

Admin / Nurse
viewFile view
FileController@destroy
comment, image and prescription models
choose file
delete selected file
delete selected file data
delete file data
alt
return deletion failed error
if file data is not deleted
return deletion success message
if file data is deleted

Figure 39: SD34 view file sequence diagram

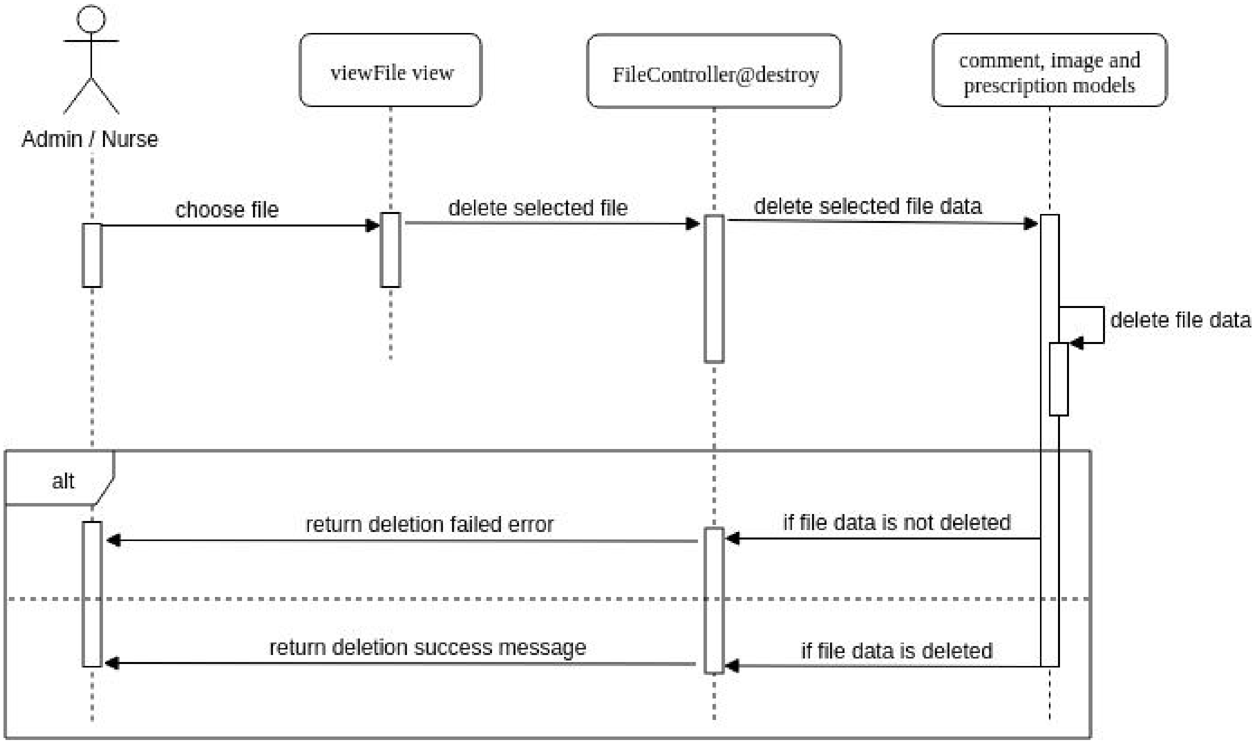

Admin / Nurse
viewFile view
FileController@destroy
comment, image and prescription models
choose file
delete selected file
delete selected file data
delete file data
alt
return deletion failed error
if file data is not deleted
return deletion success message
if file data is deleted

Figure 40: SD35 delete file sequence diagram

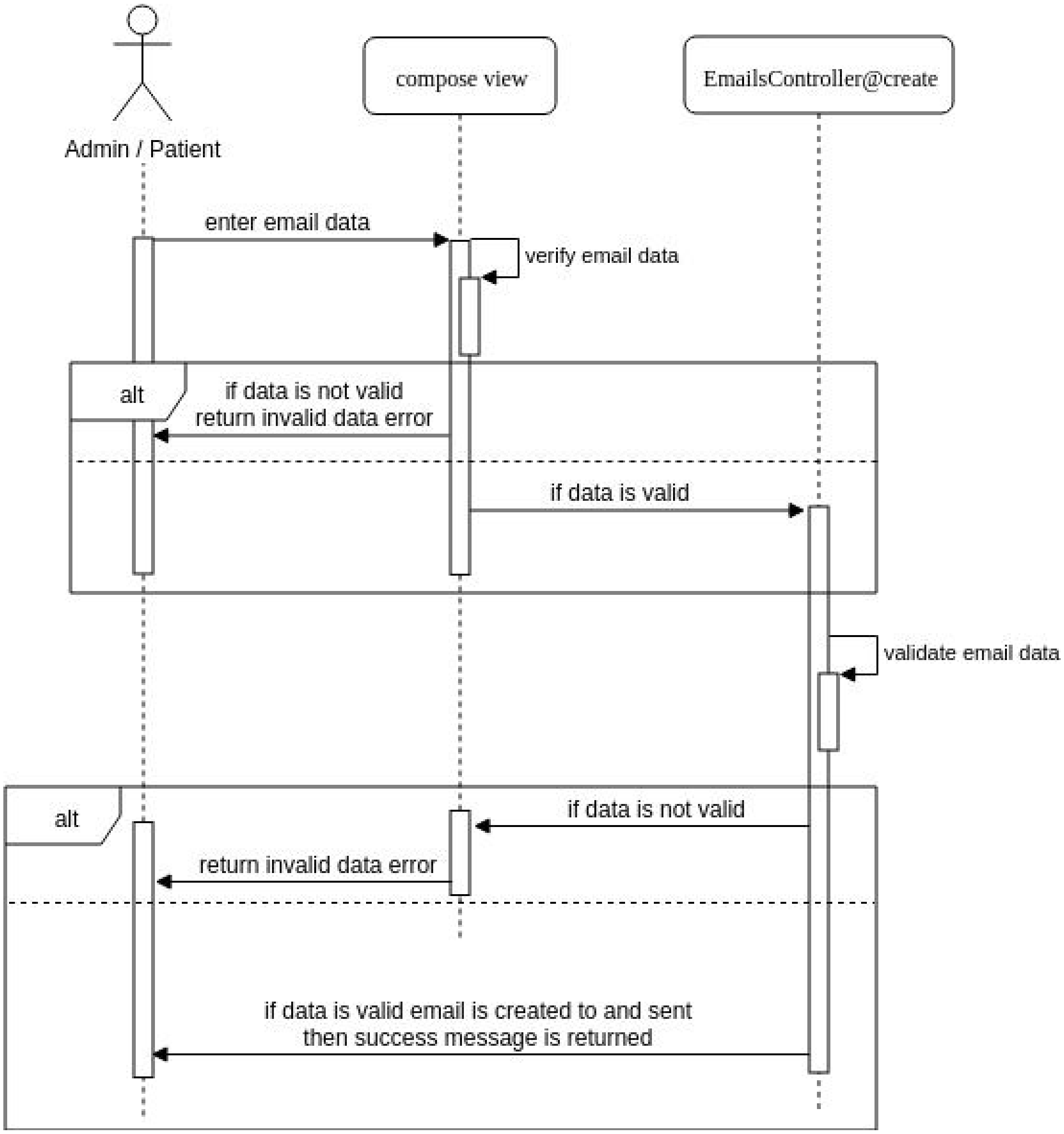


Figure 41: SD36 compose email sequence diagram

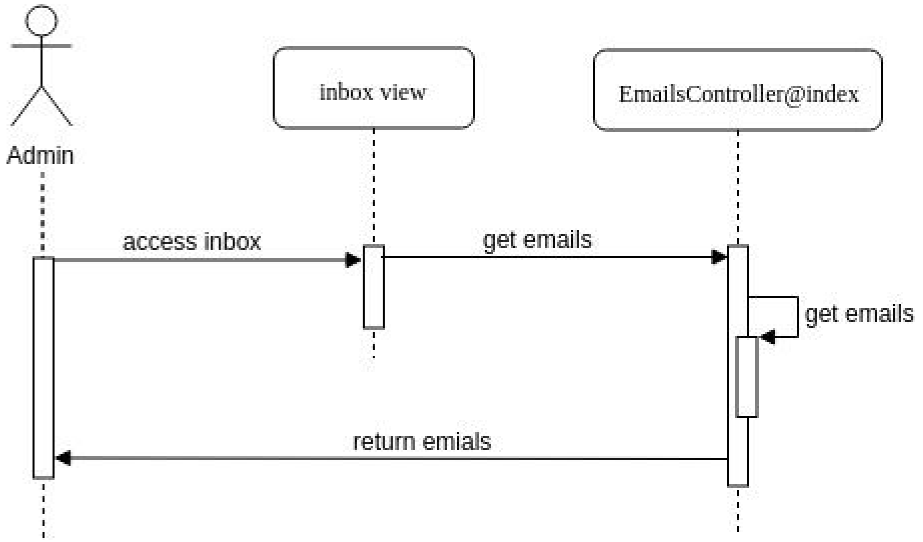


Figure 42: SD37 view emails inbox sequence diagram

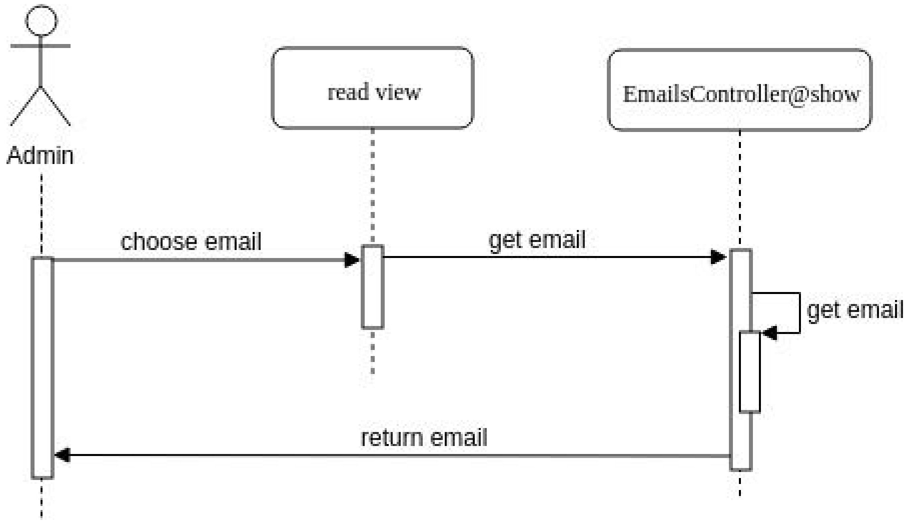
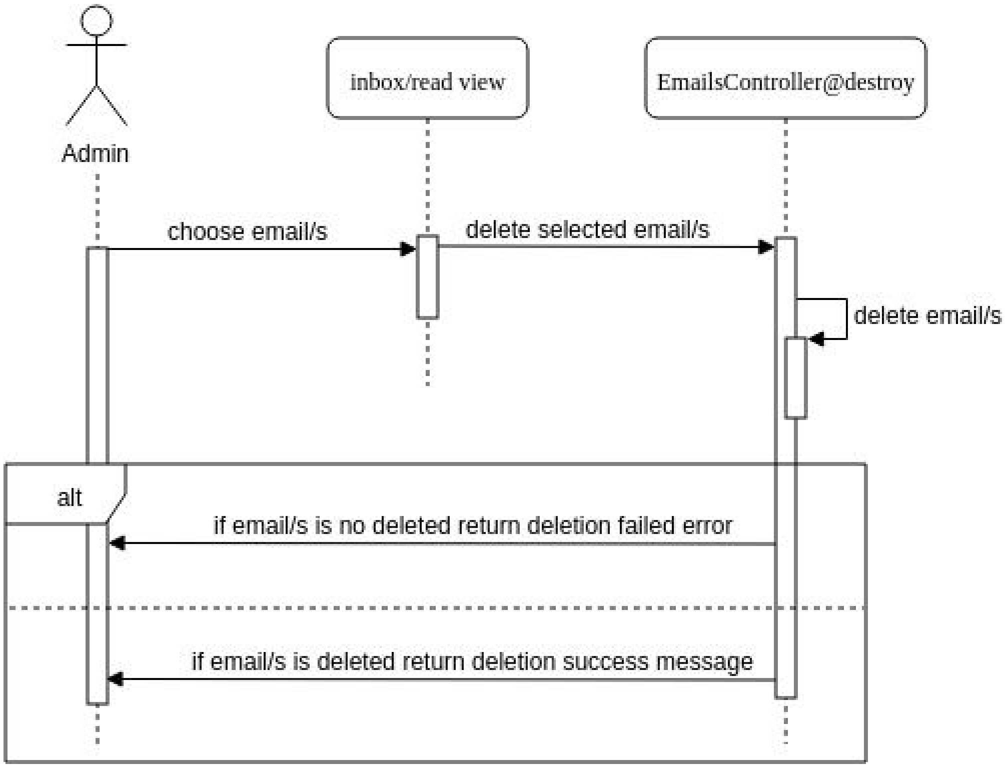


Figure 43: SD38 view an email sequence diagram



Admin
inbox/read view
EmailsController@destroy
choose email/s
delete selected email/s
delete email/s
alt
if email/s is no deleted return deletion failed error
if email/s is deleted return deletion success message

Figure 44: SD39 delete email sequence diagram

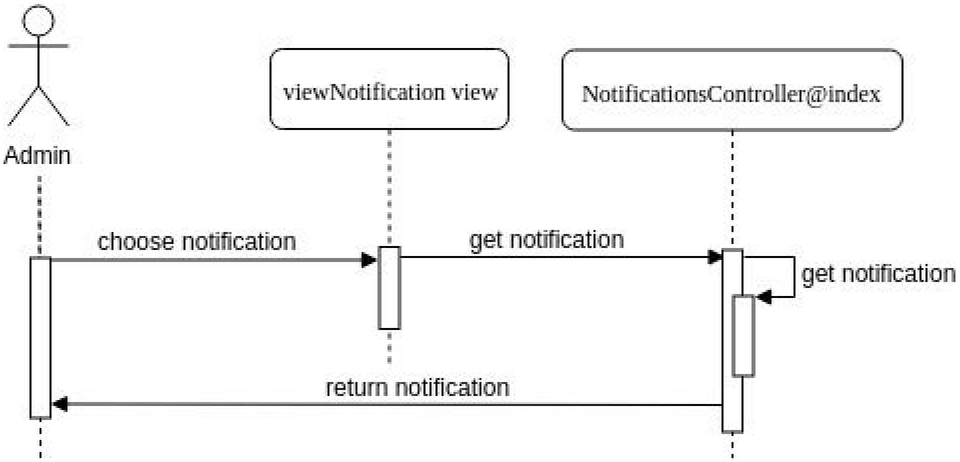


Figure 45: SD40 view notifications sequence diagram

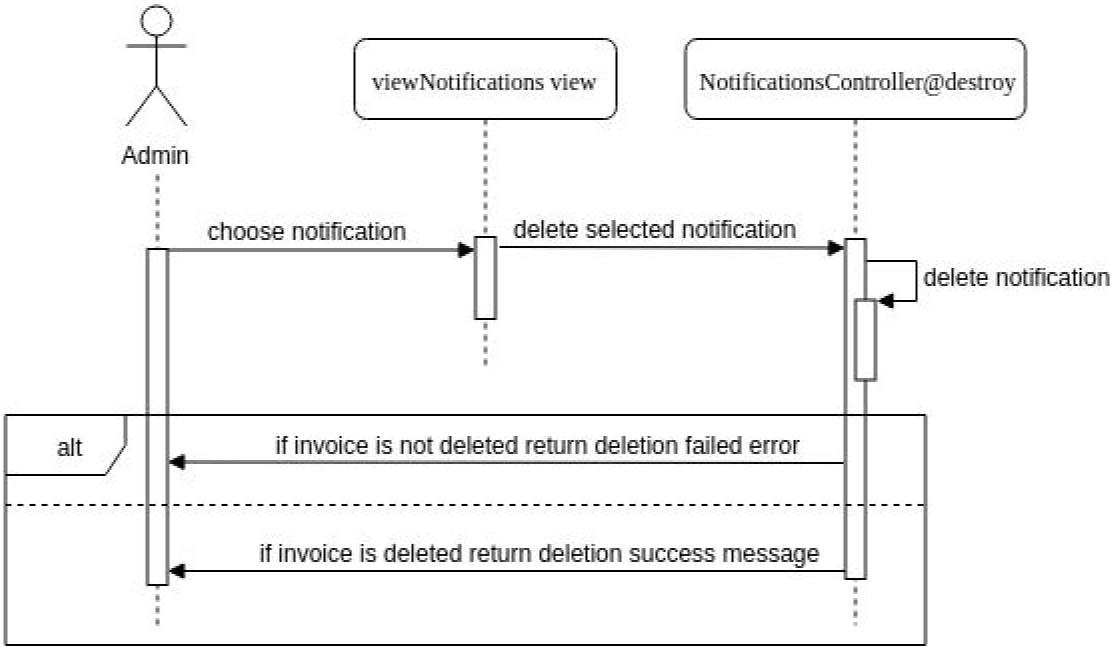
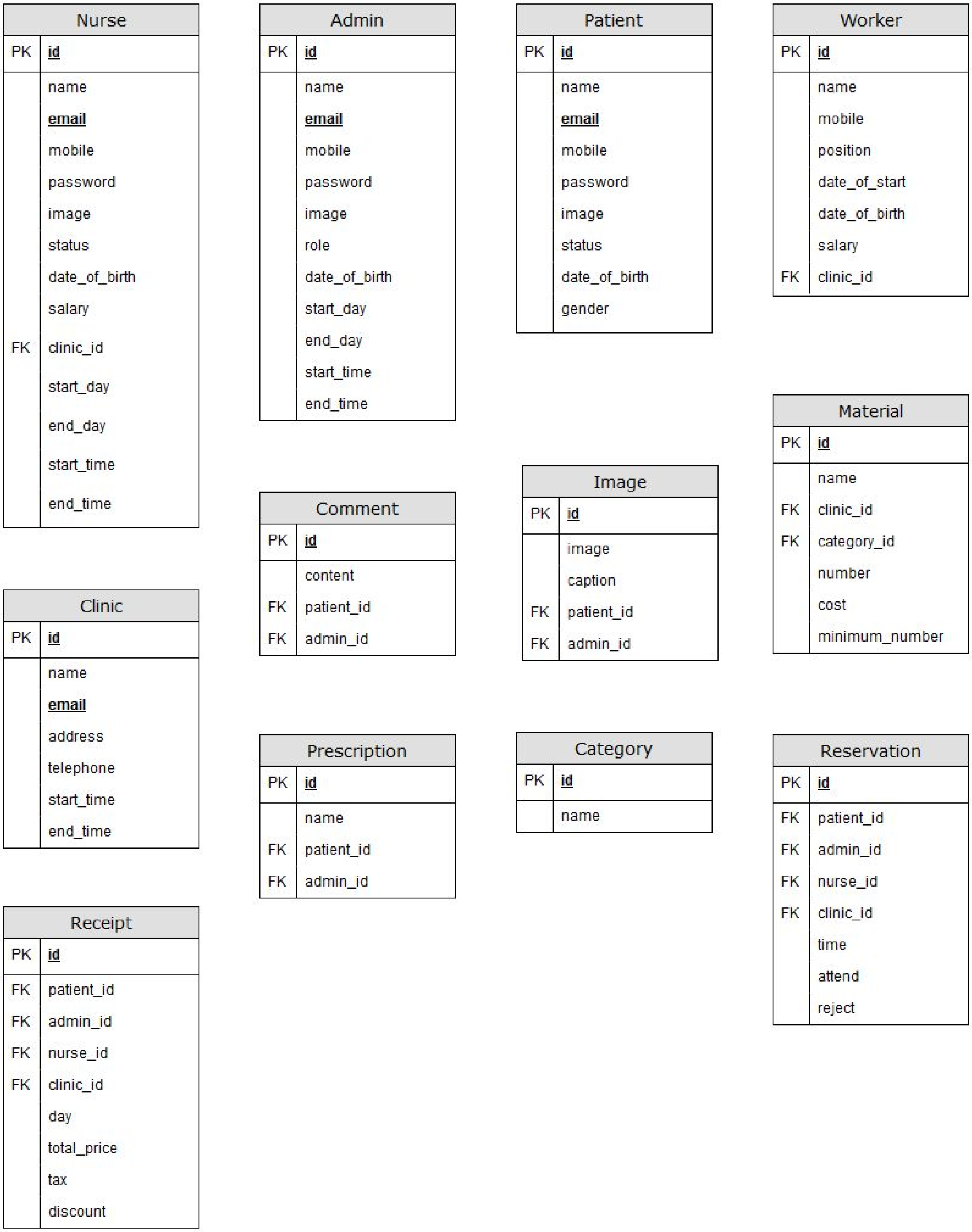

Admin
viewNotifications view
NotificationsController@destroy
choose notification
delete selected notification
delete notification
alt
if invoice is not deleted return deletion failed error
if invoice is deleted return deletion success message

Figure 46: SD41 delete notification sequence diagram

4. Data Models



	Nurse
PK	id
	name
	email
	mobile
	password
	image
	status
	date_of_birth
	salary
FK	clinic_id
	start_day
	end_day
	start_time
	end_time

	Admin
PK	id
	name
	email
	mobile
	password
	image
	role
	date_of_birth
	start_day
	end_day
	start_time
	end_time

	Patient
PK	id
	name
	email
	mobile
	password
	image
	status
	date_of_birth
	gender

	Worker
PK	id
	name
	mobile
	position
	date_of_start
	date_of_birth
	salary
FK	clinic_id

	Material
PK	id
	name
FK	clinic_id
FK	category_id
	number
	cost
	minimum_number

	Comment
PK	id
	content
FK	patient_id
FK	admin_id

	Image
PK	id
	image
	caption
FK	patient_id
FK	admin_id

	Clinic
PK	id
	name
	email
	address
	telephone
	start_time
	end_time

	Prescription
PK	id
	name
FK	patient_id
FK	admin_id

	Category
PK	id
	name

	Reservation
PK	id
FK	patient_id
FK	admin_id
FK	nurse_id
FK	clinic_id
	time
	attend
	reject

	Receipt
PK	id
FK	patient_id
FK	admin_id
FK	nurse_id
FK	clinic_id
	day
	total_price
	tax
	discount

Figure 47: er-entities

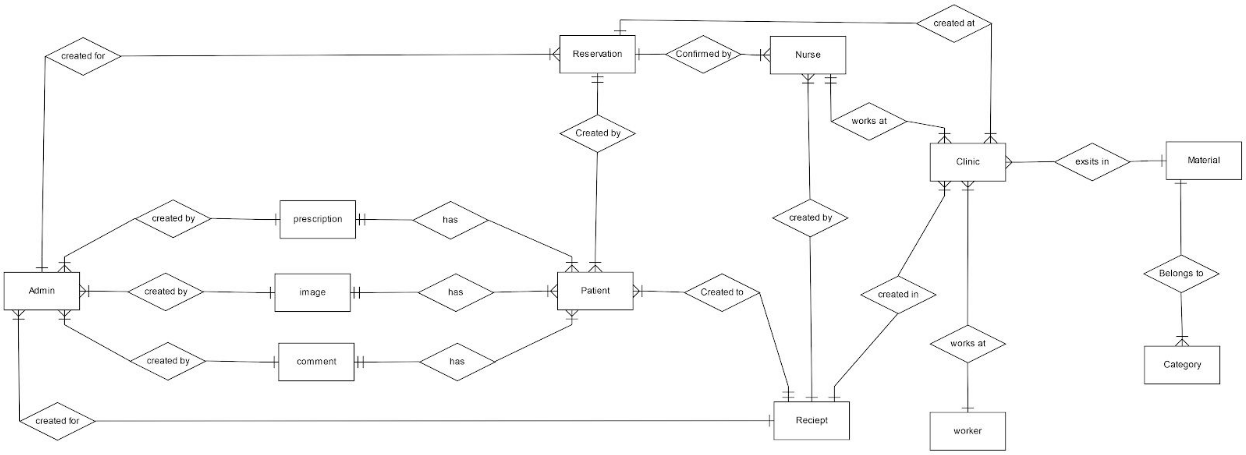
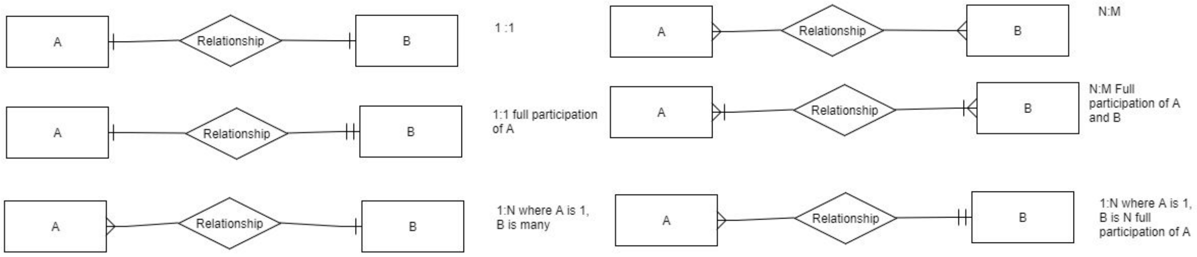


Figure 48: er-relations



A
Relationship
B
1 :1
A
Relationship
B
N:M
A
Relationship
B
1:1 full participation of A
A
Relationship
B
N:M Full participation of A and B
A
Relationship
B
1:N where A is 1, B is many
A
Relationship
B
1:N where A is 1, B is N full participation of A

Figure 49: er-map

Entities

- **Nurse** : assists doctor in running the clinic .
- **Patient** : the main client the system aims to serve .
- **Admin** : a doctor in the clinic (could be an owner or just an assisting doctor) .
- **Clinic** : the place where admins and nurses work .
- **Receipt** : a written statement acknowledging that patients have paid for their treatment.
- **Prescription** : drug description written by doctors for their patients.
- **Comment** : any observations doctors have of their patient's medical condition, they are added to patients' files .
- **Image** : a visual description of a patient's medical condition which is also added to patents' files.
- **Material** : medical substances used by doctors in the treatment and are required to be monitored.
- **Reservation** : a request to have an appointment with a doctor made by patients and approved by nurses.
- **Category** : the classification of materials.
- **Worker** : a person who works at a clinic who isn't a doctor or a nurse but is important to the clinic records.

Relationships

- (**Reservation-Patient**) created by : a list of all reservations made by the patient.

- (**Reservation-Nurse**) confirmed by : every reservation made by a patient has to be confirmed by a nurse
- (**Reservation-Clinic**) created at : every reservation is assigned a clinic id to specify the location of the appointment .
- (**Reservation- Admin**) created for : patients can request a certain doctor while reserving an appointment and so a reservation has information on the doctor it is assigned for .
- (**Nurse-Clinic**) works at : every nurse is assigned to a certain clinic (only one) .
- (**Nurse-Receipt**) created by : nurses create receipts for patients after every treatment session to manage the finances of the client and inform their client of their treatment expenses.
- (**Receipt-Patient**) created to : the receipt holds information about the client it is assigned to.
- (**Receipt-Admin**) created for : the receipt holds information about the doctor who did the treatment.
- (**Receipt-Clinic**) created in : every clinic manages its record of receipts independent of other clinics.
- (**Clinic-Worker**) works at : every clinic employs a number of workers and keeps track of their information.
- (**Clinic-Material**) exists in : clinics keeps record of their available materials to track their quantity and cost.
- (**Material-Category**) belongs to : materials that serve a common purpose are classified together under one category to make search and enumeration easy processes.
- (**Comment-Patient**) has : patients' files contain comments and observations made by doctors on their medical condition.
- (**Image-Patient**) has : patients' files may contain images that visually aids the doctor to understand the issues of their patients.
- (**Prescription-Patient**) has : drug prescription given to patients which they can check in their files.
- (**Comment-Admin**) created by : doctors record their remarks on the medical conditions of their patients in their files.
- (**Image-Admin**) created by : doctors add images essential to the description of their patient's case to their files.
- (**Prescription-Admin**) created by : doctors can write drug prescription to their patients for medical purposes.

5. System Deployment

The following diagram is the deployment diagram which is shows the hardware deployment and integrity of the system. Following the two tier client server model the system would be divided into two tiers client tier and server tier.

Client tier

This tier is deployed in the users side. It contains views that interact with controllers on server through a browser. This tier sends data and different requests to controllers on the server side.

Server tier

This tier is deployed in the server side. It contains controllers, models and database. This tier receives data and requests from the client side. The connection protocol between the two tiers is a normal interprocess communication (**IPC**).

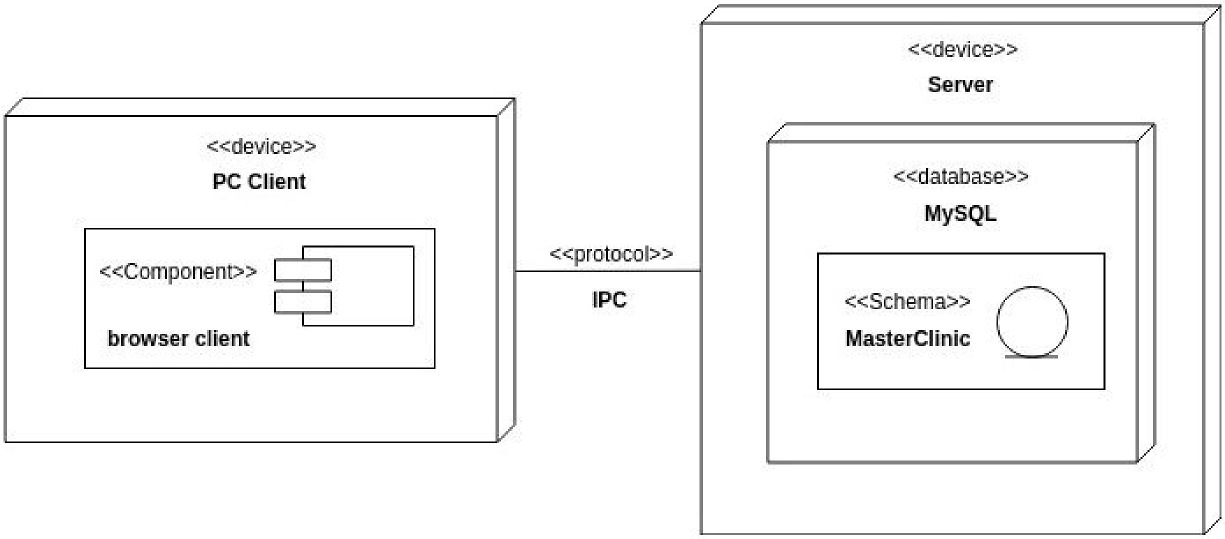
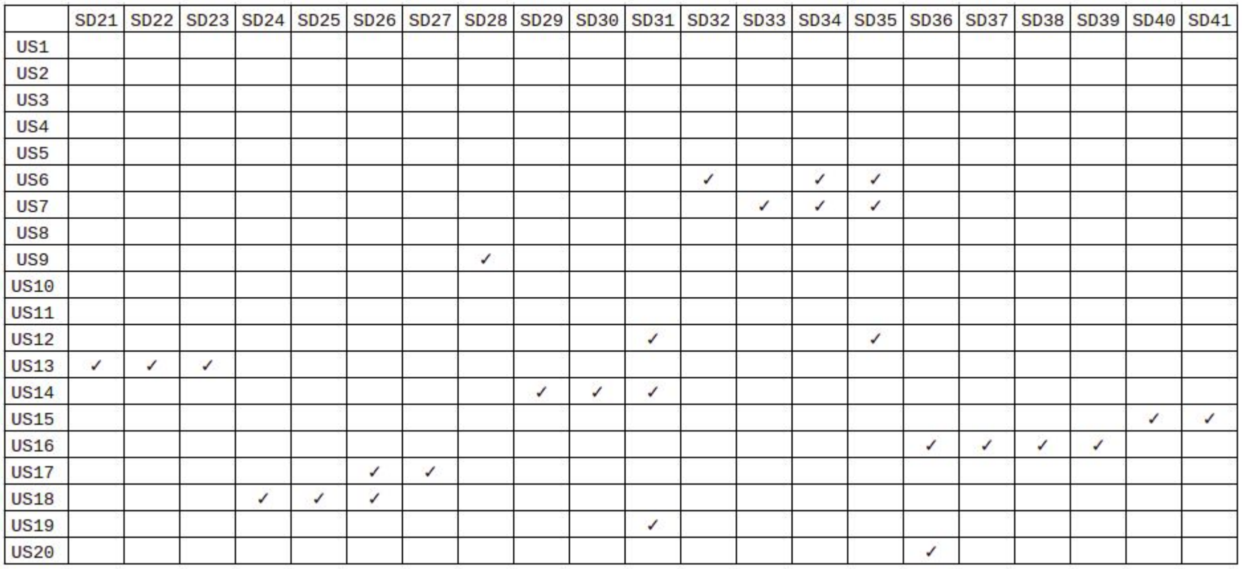


Figure 50: Deployment Diagram

6. Traceability to Requirements

	SD1	SD2	SD3	SD4	SD5	SD6	SD7	SD8	SD9	SD10	SD11	SD12	SD13	SD14	SD15	SD16	SD17	SD18	SD19	SD20
US1	✓	✓	✓			✓	✓	✓				✓								
US2	✓	✓	✓									✓	✓	✓						
US3	✓	✓	✓						✓	✓	✓	✓								
US4	✓	✓	✓	✓								✓								
US5	✓	✓	✓		✓							✓								
US6	✓	✓	✓						✓			✓								
US7	✓	✓	✓						✓			✓								
US8	✓	✓	✓									✓			✓	✓	✓	✓	✓	✓
US9	✓	✓	✓			✓			✓			✓								
US10	✓	✓	✓									✓			✓	✓	✓	✓	✓	✓
US11	✓	✓	✓			✓			✓			✓								
US12	✓	✓	✓			✓			✓			✓								
US13	✓	✓	✓				✓					✓								
US14	✓	✓	✓			✓			✓			✓								
US15	✓	✓	✓									✓						✓		
US16	✓	✓	✓						✓			✓								
US17	✓	✓	✓			✓			✓			✓								
US18	✓	✓	✓			✓			✓			✓								
US19	✓	✓	✓			✓			✓			✓								
US20	✓	✓	✓						✓			✓								



	SD21	SD22	SD23	SD24	SD25	SD26	SD27	SD28	SD29	SD30	SD31	SD32	SD33	SD34	SD35	SD36	SD37	SD38	SD39	SD40	SD41
US1																					
US2																					
US3																					
US4																					
US5																					
US6												✓		✓	✓						
US7													✓	✓	✓						
US8																					
US9								✓													
US10																					
US11																					
US12											✓				✓						
US13	✓	✓	✓																		
US14									✓	✓	✓										
US15																				✓	✓
US16																✓	✓	✓	✓		
US17						✓	✓														
US18				✓	✓	✓															
US19											✓										
US20																✓					