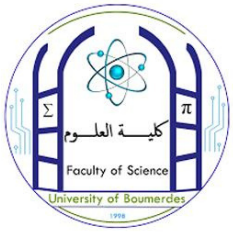
People's Democratic Republic of Algeria Ministry of Higher Education and Scientific Research M'hamed Bougara University – Boumerdès



Faculté des Sciences Département d'Informatique

Domain: Computer Science (Informatique)

Branch: Computer Science (Informatique)

Specialties: Computer Systems (Systèmes d'informatique)

Supervisor: IMINE Belkacem

Examiner: AZZOUG Wassila

Final dissertation submitted in partial fulfillment of the requirements for the Academic Bachelor'sDegree

Theme

Dental Clinic Management System with Online

Appointment Scheduling

Presented By:

FOUHAL Abdelmalek
DERDJAOUI Youcef
BOUDIAR Mohammed ElAmine
HAMRI Abdessamed

Table of Contents

General Introduction 5

Chapter 1: Presentation 6

1.1 Research Question (Problematic) 6

1.2 Project Description 6

1.3 Project Objectives 8

1.4 Tools and Technologies 8

1.5 Functional Requirements 9

1.6 Non-Functional Requirements 10

1.7 System Architecture 11

1.8 Conclusion 11

Chapter 2: Design 12

2.1 Introduction 12

2.2 System Actors 12

2.2.1 Definition..... 12

2.2.2 Actor Identification..... 12

2.2.2.1 Main Actors.....12

2.2.2.1 Secondary Actors.....13

2.3 Use Case Identification 13

2.4 Use Cases 14

2.5 Use Case Diagram..... 14

2.5.1 Description..... 14

2.5.2 Diagrams..... 15

2.6 Textual Description of Use Cases 16

2.7 Sequence Diagrams 19

2.8 Class Diagrams 24

2.8.1 Mobile Application Class Diagram..... 24

2.8.2 Desktop Application Class Diagram..... 25

2.9 Conclusion 26

Chapter 3: Implementation 27

3.1 Introduction 27

3.2 System interfaces representation 27

3.2.1 Mobile Applicatin (Client) Interfaces 27

3.2.2 Desktop (Clinic Staff) Interfaces 30

3.3 Conclusion 32

General conclusion..... 33

Webography 34

List of Acronyms

Acronym	Meaning
API	Application Programming Interface
REST	Representational State Transfer
JWT	JSON Web Token
UML	Unified Modeling Language
DB	Database
SQL	Structured Query Language
MySQL	My Structured Query Language
UI	User Interface
UX	User Experience
CRUD	Create, Read, Update, Delete
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
DCU	Use Case Diagram
IDE	Integrated Development Environment
SDK	Software Development Kit

List of Figures

Figure 2.1 : DCU of Basmatech (Desktop) 15

Figure 2.2 : DCU of Basmatech (Mobile App) 16

Figure 2.3 : Sequence Diagram of the Authentication Use Case 19

Figure 2.4 : Sequence Diagram of the Appointment Scheduling Use Case 20

Figure 2.5 : Sequence Diagram of the Payment Creation Use Case 21

Figure 2.6 : Sequence Diagram of Booking 22

Figure 2.7 : Sequence Diagram of Booking Confirmation 23

Figure 2.8 : Mobile Application Class Diagram 24

Figure 2.9 : Desktop Application Class Diagram 25

Figure 3.1 : Desktop Login Interface 27

Figure 3.2 : Mobile Login Interface 27
Figure 3.3 : Mobile Sign Up Interface 27
Figure 3.4 : Home Interface 28
Figure 3.5 : Home Interface (Arabic) 28
Figure 3.6 : Client Profile Interface 28
Figure 3.7 : Dentists List Interface 28
Figure 3.8 : Booking Selection Interface 29
Figure 3.9 : Days Selection Interface 29
Figure 3.10 : Time Slots Selection Interface 29
Figure 3.11 : Appointment Schedule Interface 29
Figure 3.12 : Language Selection Interface 30
Figure 3.13 : Home Dashboard Interface 30
Figure 3.14 : Schedule Interface 30
Figure 3.15 : Patient List Interface 31
Figure 3.16 : Patient Edit Interface 31
Figure 3.17 : Billing Interface 31
Figure 3.18 : Appointments Interface 31
Figure 3.19 : Prescriptions Interface 31
Figure 3.20 : Prescriptions Interface (Medications section) 31
Figure 3.21 : Settings Interface 31
Figure 3.22 : Prescription Template 31

List of Tables

Table 2.1 : Use Case Identification 13
Table 2.2 : Description of Authentication Use Case 16
Table 2.3 : Description of Manage Patients Use Case 16
Table 2.4 : Description of Manage Appointments Use Case 17
Table 2.5 : Description of Book Appointment Use Case 17
Table 2.6 : Description of Create Payment Record Use Case 17
Table 2.7 : Description of Manage Prescription Use Case 18
Table 2.8 : Description of Manage Treatments Use Case 18
Table 2.9 : Description of Generate Reports Use Case 18

Introduction:

Despite modern tools, clinics mostly book visits by phone or face-to-face. Because of this habit, delays pile up, check-ins get skipped, meanwhile paperwork drags on. Patients wait longer than needed whil e offices juggle messy files instead of smooth systems.

This work brings together a full online system made for scheduling healthcare visits. Built into it are tools that handle appointment setup along with patient access

- A ***mobile application*** for patients (Android & iOS)
- A ***dashboard*** built for doctors, running on desktop machines (Desktop)

Basmatech helps people find dentists . Booking visits happens instantly through the system. Medical files stay available online for quick lookups. On the provider side, doctors handle calendars directly. Acces patinets histories, Paper tasks shrink without extra effort. Time gets saved across daily operations.

Main objectives:

- Improve access to healthcare services for patients.
- Optimize appointment management and patient follow-up for doctors.
- Support the digital transformation of the healthcare sector.

Chapter 1 : Presentation

Most people still book dentist visits the old way. Phone lines jam with back-and-forth talking. Waiting takes ages just to check open slots. Seeing what time works right now? Rarely happens. Paper files vanish or get misfiled too easily. Most days, keeping up with appointments feels like chasing shadows - dentist after dentist knows that one too well. Juggling old records? Fillings here, a tooth pulled there, maybe an implant slipped in last month - all those details slip through cracks. Each visit piles on more notes. Mistakes creep in when hands are full just staying above water. Apart from that, patients and doctors lack a connected digital health file, making it harder to keep treatment on track. The absence of linked records interrupts follow-up care.

1.1- Research question (Problematic) :

Appointment scheduling, communicating with patients, and manual processes are common challenges for dental clinics, leading to scheduling conflicts, missed appointments, and lower service efficiency. Many existing systems are also not simple and easy to use for patients and clinic staff.What are the steps to be considered when designing a mobile dental clinic app to improve the efficiency of patient appointment scheduling, reduce missed appointments, and have a simple, easy-to-use interface for both patients and clinic staff?

1.2- Project Description :

Running Dental office gets easier when paperwork moves online. This tool handles everyday tasks so clinics keep track of visits without hassle. Instead of paper files, everything lives in one place where updates happen instantly. Staff find patient details quicker now. Communication flows smoother because messages do not get lost. Efficiency rises once forms shift from desks into secure screens. People feel more involved when appointments move at steady pace. A setup like this breaks down into several key pieces. One part handles tasks while another manages flow between sections. Each piece connects through shared pathways instead of working alone. Main features include control zones, data hubs, and response units. These parts operate together but keep separate roles. Structure follows function across every section. Components interact without merging completely clinic, Inside the clinic staff rely on a ***Desktop Application***, built just for them. From managing patient details to setting up visits, it handles day to day tasks. Instead of juggling separate tools, everything comes together here - treatment notes

, invoices, summaries. Doctors use it during consultations. Front desk workers keep schedules running smoothly. Admins pull data when needed. Far from flashy, it works quietly in the background. Most actions happen through this single hub. Efficiency shows up not in speed but in consistency. Behind every smooth appointment, this system plays a role.

App made just for patients - it lets them reach the system from anywhere. Right from their phone, they check private details, pick appointment times. One tap brings up past treatments, so nothing gets lost. Fewer calls come into the office because people handle things themselves now. Less busywork for workers means smoother days at the clinic.

From the app to storage, messages travel through hidden pathways. When users send info, checks happen before anything moves forward. Rules built into the system decide what passes and what gets stopped. Safety steps guard every piece during transfers. Information stays correct because updates follow strict patterns. One wrong step breaks nothing thanks to backup logic built in place. Smooth operation comes from quiet coordination behind scenes.

Storing everything from patient details to bills, the ***database*** acts like a backbone. From appointments to prescribed medicines - information lives here. Built on ***MySQL***, it keeps data neat through clear relationships between entries. Queries run fast since things are organized in rows and tables. Treatment history? That too gets saved exactly once, no duplicates.

Working as one, these pieces fit into a single setup where every element knows its job. While the desktop app manages tasks inside, patient access from outside comes through the mobile version, with the ***backend*** quietly moving data safely and steadily between them.

Component	Role
Desktop App	Clinic management
Mobile App	Patient access
Backend API	Data validation
Database	Data storage

1.3- Project Objectives :

One key aim here: ***build an online system*** making dental office tasks run smoother, more dependable. That overall purpose breaks down into smaller targets, each tackling a different piece.

Storing files on screens instead of shelves cuts down clutter while boosting reach. When notes live online, mistakes fade finding what matters feels smoother too.

One key goal sits around ***making booking easier***. Organized timing loops into place so double bookings fade out, waits shrink up, gaps closed own clinic gears turn smoother without hiccups gumming the works.

Communication gets easier when patients can reach clinic workers through a phone app. Instead of showing up just to set a meeting time, they handle it online. Remote access means fewer trips downtown. Tasks like rescheduling happen without stepping into the building. The tool lets people ***stay home while still connecting with care teams.*** Simple jobs take less effort now. Digital entry points replace some front desk interactions.

Right off the bat, streamlining admin duties is part of how the system works -
tasks like invoicing, compiling reports, and monitoring treatments happen automatically. Because of that shift, people at the clinic spend less time on paperwork, ***giving them room to pay closer attention to patients.***

Ending things off, one goal stands clear - ***keeping information accurate and safe*** througha single storage hub. This setup checks every move via a locked down control layer. Not just any approach, but one where rules are enforced behind the scenes. Every action gets reviewed before it sticks. Security here grows from structure, not slogans. What you enter meets barriers meant to protect, not impress.

1.4- Tools and Reasons :

Fresh tools shape the build, chosen because they fit what the job needs.
Each piece slots in place where it works best.

Desktop apps lean on ***JavaFX*** since its structure handles detailed visual layouts well. With tools for modern interface elements,
it builds programs that react quickly to user actions - key when running busy medical office systems.

One reason ***Flutter*** gets picked for Mobile app?
Its knack for working across different devices without extra effort. A
single set of code powers apps on various systems, which cuts down how long it takes to build them. Same look, same feel - no matter where you open it.

Back at square one, ***Java*** handles backend work thanks to steady performance, room to grow, yet a well-stocked toolkit nearby. When pushed, it builds tough apps on servers - juggling many tasks without buckling under pressure.

Out there, ***REST API*** design helps the app talk with the server behind it, Through HTTP rules, information moves clearly, safely, almost like notes passed on familiar paper. One thing leads to another - structure stays clean because standards guide every step. Security tags along without slowing things down too much. Data flows back and forth under agreed terms, nothing left guessing.

Built on a foundation of steady performance, ***MySQL*** steps into the role of database manager thanks to its broad adoption across industries. Reliance grows where structure meets speed - organized data fits neatly inside. Queries with layers find room to breathe here, handled without pause. Efficiency anchors every operation, making space for scalability down the line.

Running on your own machine, ***WAMP Server*** sets up space for MySQL while building apps. This setup makes checking and handling data easier by keeping things close at hand.

Builds get automated when ***Maven*** steps in, keeping every library in check. Project tasks become simpler because dependencies stay organized behind the scenes.

JasperReports a template engine steps in to build clean reports and medical slips, shaping how clinics handle printed records. What appears on paper comes through structured formatting behind the scenes.

A single one of these tools helps shape something workable, able to grow, and straightforward to keep running. What matters most shows up when they fit together smooth, steady, built to last without constant fixing.

1.5- Functional Requirements :

What a system does step by step comes down to its functional requirements. These actions answer exactly what users expect it to carry out. Each task fits a need, shaped by how people will interact with it. Operations emerge from real usage, not assumptions. Specifics matter because they guide design without guesswork.

Managing patient details should be possible for clinic workers using the system. Creating new files comes alongside editing existing ones, even removing them when needed. ***Booking visits*** falls under its functions too. Changes to scheduled times happen smoothly because double bookings get blocked automatically. Cancellations fit into this flow without causing overlaps.

Every now and then, a dentist needs to ***note down what was done during an appointment*** this setup makes that possible.

On top of that, ***handling bills*** fits right in,creating receipts and keeping tabs on money coming in.

Every patient needs a way into the app through ***registration and login***. Once inside, they can look over details tied to their account. ***Booking visits*** shows up as one of the main tasks available. Past bookings are stored so users can check what happened before. Getting around each part feels straightforward without extra steps.

Prescriptions can come from a system built to help doctors draft them quickly. Templates already set up make it possible to put together each one without starting over. Printing happens right after, smooth and ready. What matters is that everything fits standard needs while staying flexible behind the scenes.

1.6- Non-Functional Requirements :

How well a system works shows up in its non-functional traits.

Security begins by guarding private health details - strict entry checks keep strangers out. Access ties to permission levels, so each person sees only what they ought. A login acts like a key; without it, doors stay shut. Functionality opens piece by piece, never all at once. Who gets in depends on verified identity, nothing less.

Speed matters just as much. When users pull up records or set a meeting,***the response should feel instant***. A delay here breaks the flow, slows everything down.

Easy access matters most when people face screens daily. A smooth layout helps everyone move through steps without stopping to think. Clinic workers juggle many roles, so simplicity cuts down errors. Patients arrive stressed, often confused - clear paths guide them forward quietly. When design fades into the background, tasks get done. Fewer clicks mean less frustration building up behind quiet faces.

A system that works when it needs to - that kind of steady performance matters most. When things run smoothly, without constant breakdowns, trust builds naturally. Information stays accurate because errors aren't allowed to slip in unnoticed. Consistency protects the details, every single time.

Tokens guard the app's backdoor, built on ***JSON Web Token***. Once you log in right, the server fires off a key headed your way. That key rides along every next move you make online here. Only those who signed in get past the gate, thanks to this setup. Talks stay lightweight. No memory needed on the server side. Each exchange stands alone, verified fresh each time.

When you open the app again, ***it remembers who you*** are thanks to a small built in database saved right on your device. Instead of typing login details every single time, the system checks stored info like access codes safely tucked away locally.
This means logging in happens quietly behind the scenes. The whole process runs smoothly because security and convenience work together without slowing things down.

Still, it keeps things running smoothly without letting go of tight resource controls.
A smooth flow meets careful limits when users interact with the system. Not every method does both - this one holds steady on access even as navigation stays simple.

1.7- System Architecture :

Starting with layers, the setup splits tasks so each part stays clear and focused. One after another, functions are grouped by purpose, making management smoother without overlap.

Deep inside, a ***MySQL database*** holds every piece of app data. For daily tasks, the desktop software pulls information straight from it.

Behind the scenes, a bridge connects the app on your phone to stored information. When something is asked, checks happen first. Rules built into the system guide what comes next. Safety steps make sure only allowed
actions go through.

From the ***mobile app***, every request travels straight to the ***API*** - no reaching into the database on its own. Because of this setup, sensitive information stays protected.
Every change to data flows through a single checkpoint. Security gets stronger when paths are limited.

One reason this setup works well is how tasks get split into distinct parts, which keeps things tidy when updates happen or changes arrive later. Security gains a boost too - since nobody without permission can reach the data store directly.

1.8- Conclusion :

Right off the bat, this section laid out what the Dental Clinic Management System covers - goals, parts, tools used, and needs. Instead of old school ways, it builds a single digital space meant to update how dental offices run day to day Starting with linked desktop and mobile apps, connection to a central database happens through an API, making data handling smooth while boosting how users interact. Built on what this chapter outlines - both in actions expected and performance needs - the whole structure takes shape step by step.
Later on, we take a closer look at the setup through UML diagrams - these sketches map out how pieces connect and behave together. Visuals unfold the layout step by step showing flow instead of just naming parts.

Chapter 2 : Design

2.1- Introduction :

Here comes a look at how the Dental Clinic Management System was shaped. Who uses the system shows up early in this part, along with what they actually do with it. Each task gets explained in plain description before visuals take over. Motion of data appears through step by step sequence drawings. What holds everything together sits shown in class layouts, covering both computer and phone versions. Structure meets flow by the end without extra clutter. One part of the setup works on computers at the
clinic. This app runs things like scheduling and records right where care happens. Staff such as admins, front desk workers, and dentists use it every day. Inside it holds a private data store that keeps everything organized locally. Then there is another piece made for phones. Patients open this version to pick visit times or check their details.
Information moves between phone users and clinic database using REST API.

2.2- System Actors :

2.2.1- Definition :

An actor is an external entity that interacts with the system. It can be a human user or another software component. In this project, actors represent the different roles that use the dental clinic system and benefit from its services.

2.2.2- Actor Identification :

2.2.2.1- Main Actors :

• ***Administrator :***

Manage user accounts.

Configure the system preferences.

Update clinic profile information.

Manage medications.

Generate reports and view statistics.

• ***Receptionist :***

Register and update patients.

Manage appointments.

Create payment records.

Generate invoices.

View billable appointments.

• ***Dentist :***

View appointment calendar.

Mark appointments as completed.

Create and print prescriptions.

Record treatments.

Manage schedule and block time slots.

• ***Patient :***

Log in to the mobile patient portal.

Book an appointment after viewing availability.

Cancel own appointments.

View own appointments.

View own medical history and personal information.

2.2.2.2- Secondary Actors :

The system interacts with technical components. Information about patients sits inside the system alongside appointment logs, treatment records, prescription details, payment history, user profiles, and clinic settings. From the mobile app, data flows safely into view whenever needed thanks to a secure connection built on REST API.

2.3- Use Case Identification :

Actors	*Use Cases*
Administrator	Login to system, configure system, manage user accounts, create user, update user, delete user, set preferences, manage medications, update clinic profile, generate reports, view appointment statistics, view revenue report, view dentist performance.
Receptionist	Login to system, manage patients, register patient, update patient information, view patient profile, view medical history, manage appointments, schedule appointment, reschedule appointment, cancel appointment, view appointment calendar, manage billing, view billable appointments, create payment record, generate invoice.
Dentist	Login to system, view dashboard, manage appointments, view appointment calendar, mark appointment completed, cancel appointment, manage prescriptions, create prescription, add prescription items, print prescription, manage treatments, record treatment, manage treatment types, view treatment history, manage schedule, view availability, block time slots.
Patient	Login to system, validate credentials, book appointment, view availability, cancel own appointment, view own appointments, view personalinformation.

Table 2.1: Use Case Identification

2.4- Use Cases :

A) Authentication : This operation allows each user to access the system according to their role. The user enters their login credentials, and the system validates them before opening the appropriate interface.

B) Manage Patients : This operation allows the receptionist or administrator to register new patients, update patient information, consult patient profiles, and access their medical history.

C) Manage Appointments : This operation allows the receptionist and dentist to schedule, reschedule, cancel, and view appointments. It also checks dentist availability and prevents
time conflicts.

D) Manage Billing : This operation allows the receptionist to view billable appointments, create payment records, and generate invoices after completed appointments.

E) Manage Prescriptions : This operation allows the dentist to create prescriptions, add medication items, and print prescriptions for patients.

F) Manage Treatments : This operation allows the dentist to record treatments performed during appointments, manage treatment types, and view treatment history.

G) Manage Schedule : This operation allows the dentist to manage availability and block unavailable time slots.

H) Generate Reports : This operation allows the administrator to generate reports about revenue, appointment statistics, and dentist performance.

I) Patient Portal : This operation allows the patient to use the mobile application to book appointments, view availability, cancel own appointments, and consult personal medical information.

2.5- Use Case Diagram :

2.5.1-Description :

The global use case diagram represents the functions offered by the Dental Clinic Management System. The desktop part includes the main clinic actors: administrator, receptionist, and dentist. The patient portal part represents the mobile application used by the patient. Both parts belong to the same system, but they were separated in the original diagram for space reasons. In this document, they are combined into one global representation.

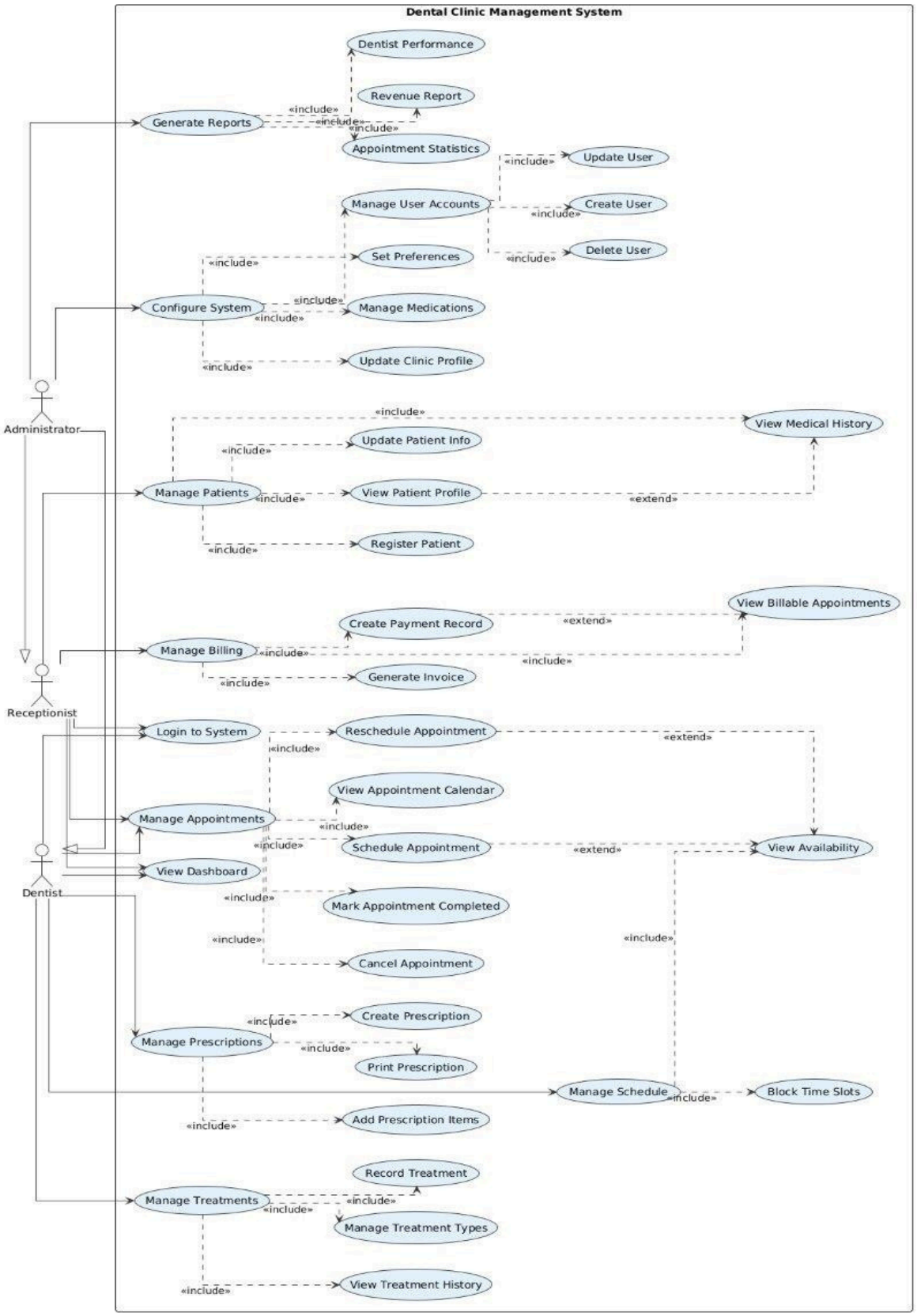


Figure 2.1: DCU of Basmatech (Desktop)

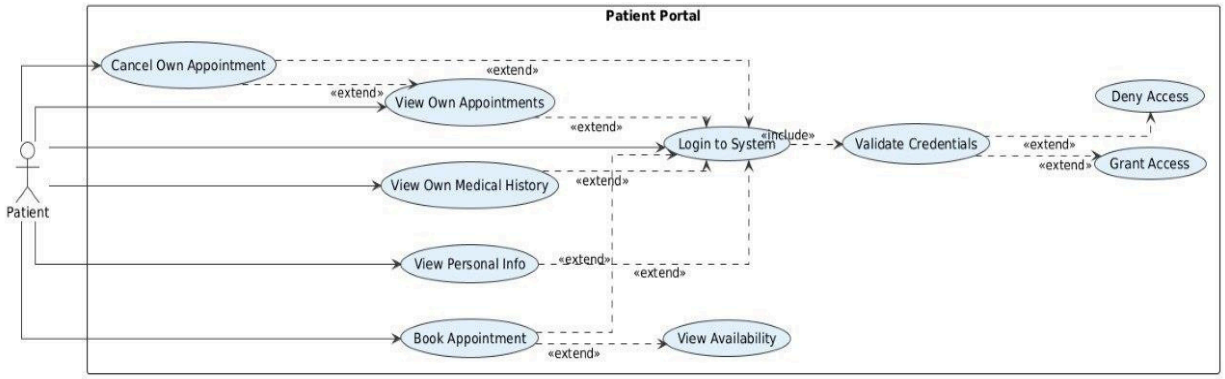


Figure 2.2: DCU of Basmatech (Mobile App)

2.6- Textual Description of Use Cases :

Use Case No. 1: Authentication

Main Actor	Administrator / Receptionist / Dentist / Patient
Objective	Allow the user to access the system securely.
Precondition	The user must have an existing account.
Flow	• The user opens the desktop or mobile application. • The system displays the login form. • The user enters the email or username and password. • The system verifies the credentials. • If the credentials are valid, the system opens the corresponding interface. • If the credentials are invalid, the system displays an error message.

Table 2.2: Description of Authentication Use Case

Use Case No. 2: Manage Patients

Main Actor	Receptionist / Administrator
Objective	Register and manage patient information.
Precondition	The user must be authenticated.
Flow	• The receptionist opens the patient management section. • The system displays the patient list. • The receptionist can register a new patient. • The receptionist can update patient information. • The system saves the changes in the database. • The user can view the patient profile and medical history.

Table 2.3: Description of Manage Patients Use Case

Use Case No. 3: Manage Appointments

Main Actor	Receptionist / Dentist
Objective	Manage clinic appointments.
Precondition	The user must be authenticated.
Flow	• The user opens the appointment management section. • The system displays the appointment calendar. • The user selects a patient, dentist, date, and time. • The system checks dentist availability and blocked slots. • If the selected time is available, the appointment is saved. • If there is a conflict, the system displays an error message.

Table 2.4: Description of Manage Appointements Use Case

Use Case No. 4: Book Appointment

Main Actor	Patient
Objective	Allow the patient to book an appointment from the mobile application.
Precondition	The patient must be authenticated.
Flow	• The patient opens the patient portal. • The patient views available slots. • The patient selects the desired appointment time. • The mobile application sends the request through the REST API. • The system verifies availability in the database. • The appointmentis createdand confirmationis displayed.

Table 2.5: Description of Book Appointment Use Case

Use Case No. 5: Create Payment Record

Main Actor	Receptionist
Objective	Record a payment for a completed appointment.
Precondition	The appointment must be completed and not already paid.
Flow	• The receptionist opens the billing section. • The system displays billable appointments. • The receptionist selects an appointment and enters payment details. • The system checks if the appointment is valid and not already paid. • If the payment is valid, it is saved in the database. • The systemdisplaysa paymentconfirmation.

Table 2.6: Description of Create Payment Record Use Case

Use Case No. 6: Manage Prescriptions

Main Actor	Dentist
Objective	Create and print prescriptions for patients.
Precondition	The dentist must be authenticated.
Flow	• The dentist opens the prescription section. • The dentist selects the patient. • The dentist creates a prescription. • The dentist adds prescription items such as medication, dosage, and instructions. • The system saves the prescription. • The prescription can beprinted.

Table 2.7: Description of Manage Prescription Use Case

Use Case No. 7: Manage Treatments

Main Actor	Dentist
Objective	Record treatments and consult treatment history.
Precondition	The dentist must be authenticated.
Flow	• The dentist opens the treatment section. • The dentist selects the related appointment or patient. • The dentist records the treatment details. • The system stores the treatment record. • The dentist canconsult thetreatmenthistory.

Table 2.8: Description of Manage Treatments Use Case

Use Case No. 8: Generate Reports :

Main Actor	Administrator
Objective	Generate reports for clinic management.
Precondition	The administrator must be authenticated.
Flow	• The administrator opens the reports section. • The system displays report options. • The administrator selects the type of report. • The system retrieves the required data from the database. • The report is generated and displayed.

Table 2.9: Description of Generate Reports Use Case

2.7-Sequence Diagrams

Sequence Diagram No. 1: Authentication

One should someone try to enter the app, up pops a login screen right away. They type their name and passcode into separate boxes. Next, the check-in tool looks through stored records trying to match that person. Finding nothing means back comes a note saying the details are wrong. When the user is found, verification of the typed password happens through the password hasher. Should the password fail validation, access gets denied outright. A correct match that ties to a deactivated profile triggers a notice about disabled status. Only when every condition passes does the dashboard appear, shaped by what role the person has.

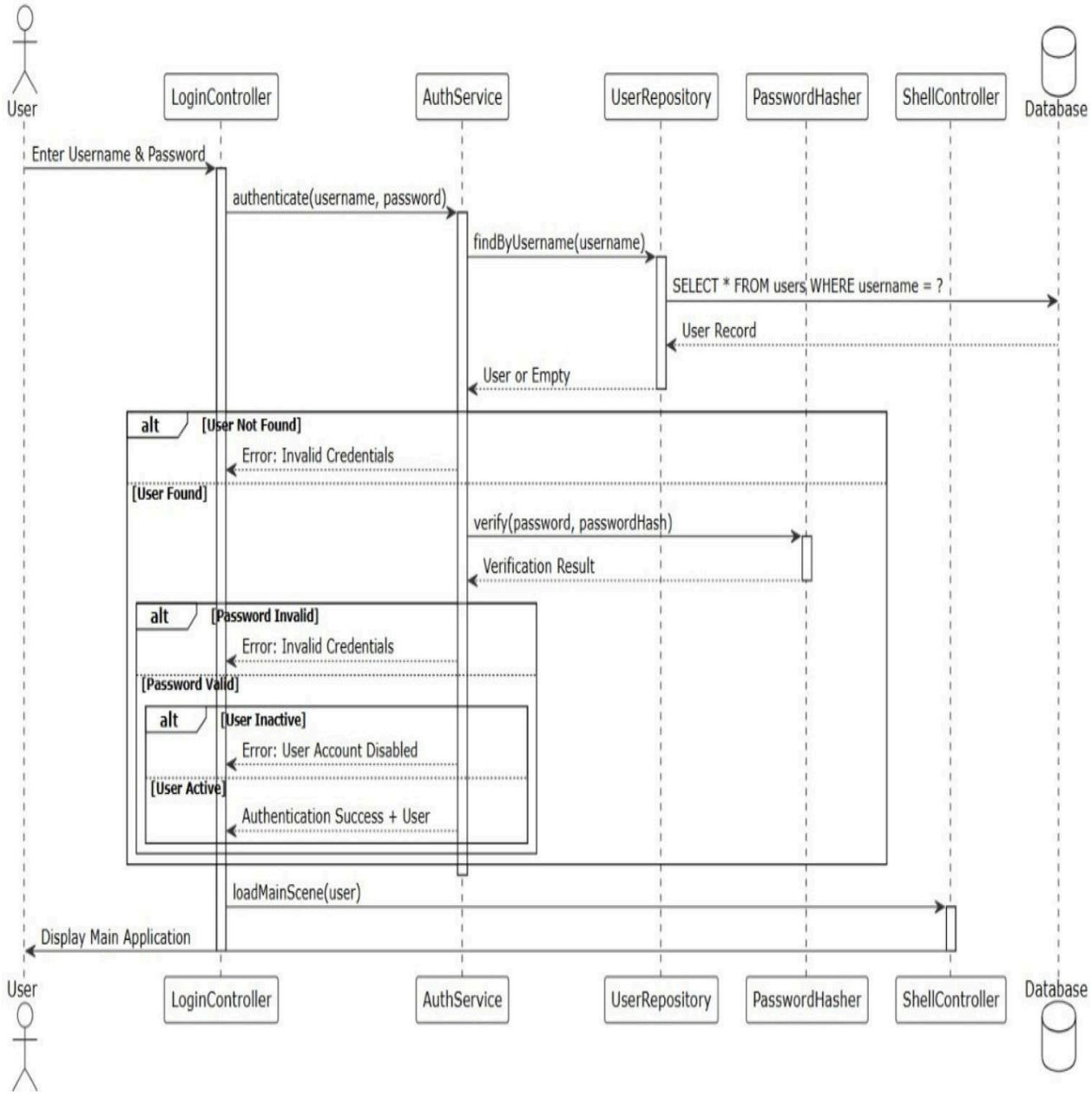


Figure 2.3: Sequence Diagram of the Authentication Use Case

Sequence Diagram No. 2: Appointment Scheduling

Most times, a person picks a date for a visit. Before that shows up on screen, the system asks what patients exist - also who among the dentists is free. Data pulls happen through separate channels, each one hitting its own storage spot in the background. Once names appear, someone types in new booking details. That data moves forward only after checking time chunks already taken. Availability gets confirmed by scanning those blocks carefully. Something goes wrong? The system shows an error. No problem found? It stores the appointment and gives back a note
saying it's done.

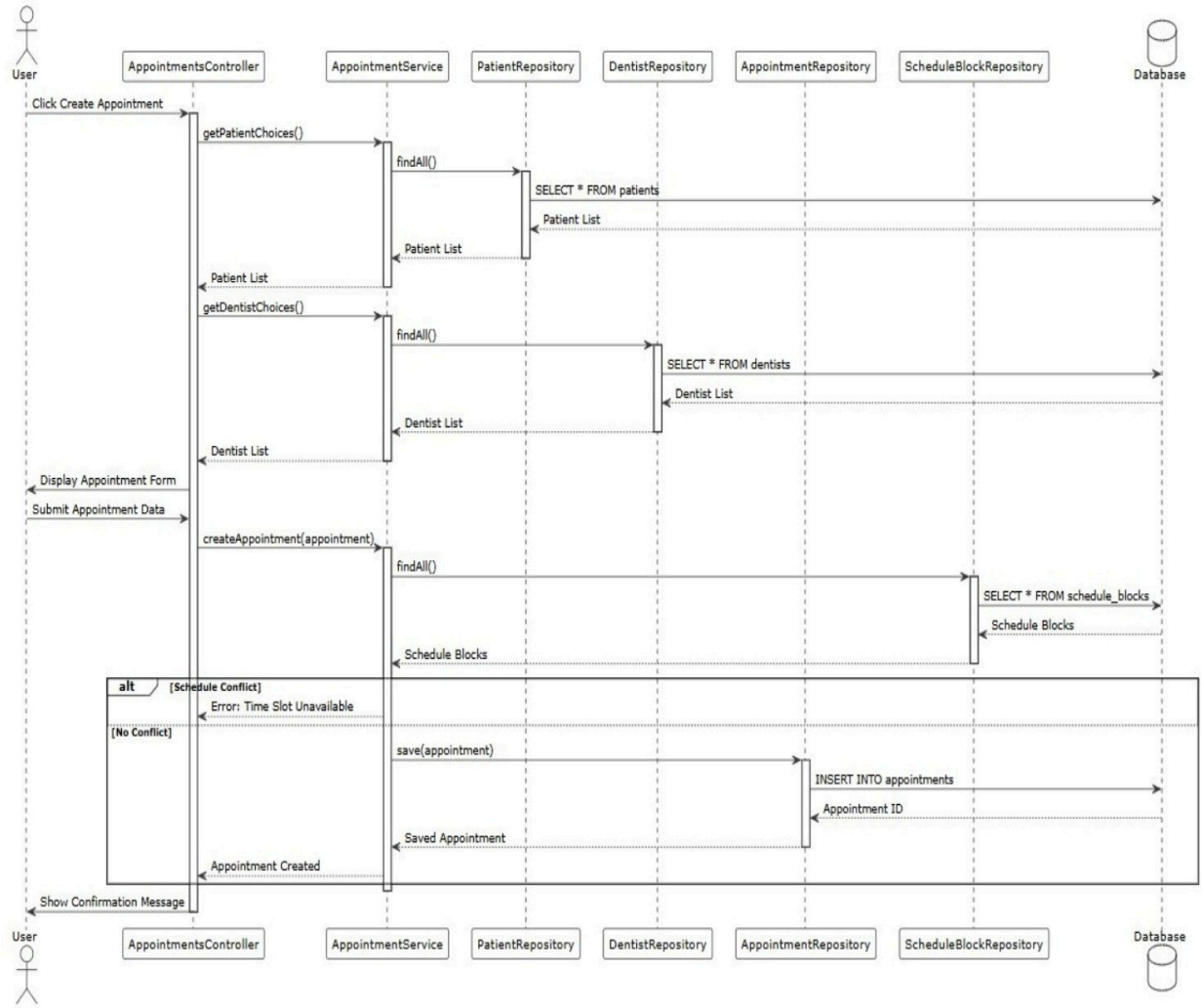


Figure 2.4: Sequence Diagram of the Appointment Scheduling Use Case

Sequence Diagram No. 3: Payment Creation

Should the receptionist decide to make a payment, the billing controller reaches out to the billing service to fetch appointments ready for billing. Completed visits showing no prior payment get pulled by the payment repository. Once an appointment gets picked along with payment info, validation begins. Is the visit confirmed as real? That check runs first. Then comes another layer - looking up whether that same slot already has a recorded transaction. When appointments aren't valid or were paid before, the screen shows a problem. If everything checks out, money details go into storage and a message appears saying it's saved.



Figure 2.5: Sequence Diagram of the Payment Creation Use Case

Sequence Diagram No.4: Booking Availability in Mobile App

Opening the app brings up the home screen right away, showing shortcuts for common tasks. From there, picking “Book Visit” leads into the scheduling section without delay. That page checks with the server to find open time slots nearby. Whenever openings show up, they appear on screen in a clear list below. No times at all? Then a note pops up saying nothing is free just now, along with a polite notice about keeping an eye out later.2.8 Class Diagrams

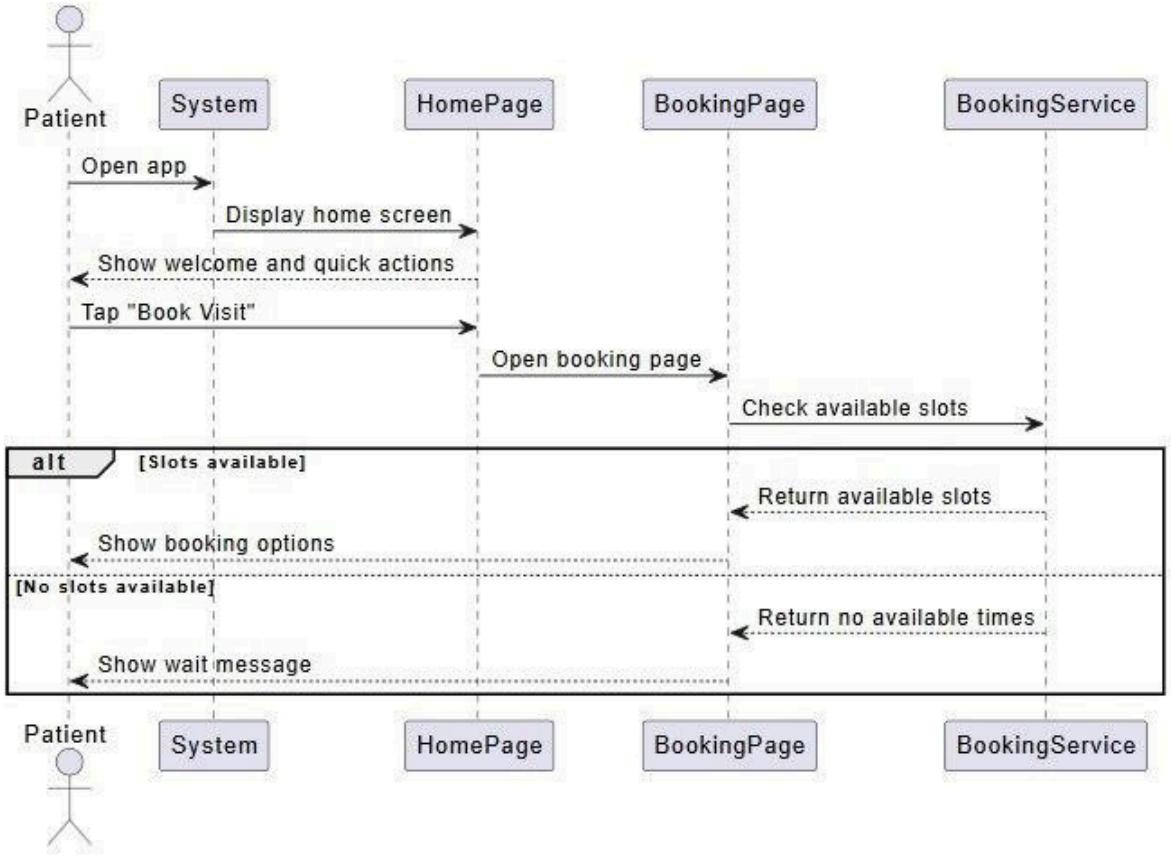


Figure 2.6: Sequence Diagram of Booking Availability

Sequence Diagram No.5: Appointment Booking Confirmation

Starting at the app's booking area, the patient picks when they'd like to come in. A form appears once they arrive there, guiding their choices. They fill out day and hour, then send it forward instead of waiting. The moment it arrives, the service looks up whether that slot sits empty. Only if free does it lock in the visit, storing details quietly. Right after, words pop up saying everything went through. No room means a note comes back - time taken, try again later. Pick something new becomes the only path ahead.

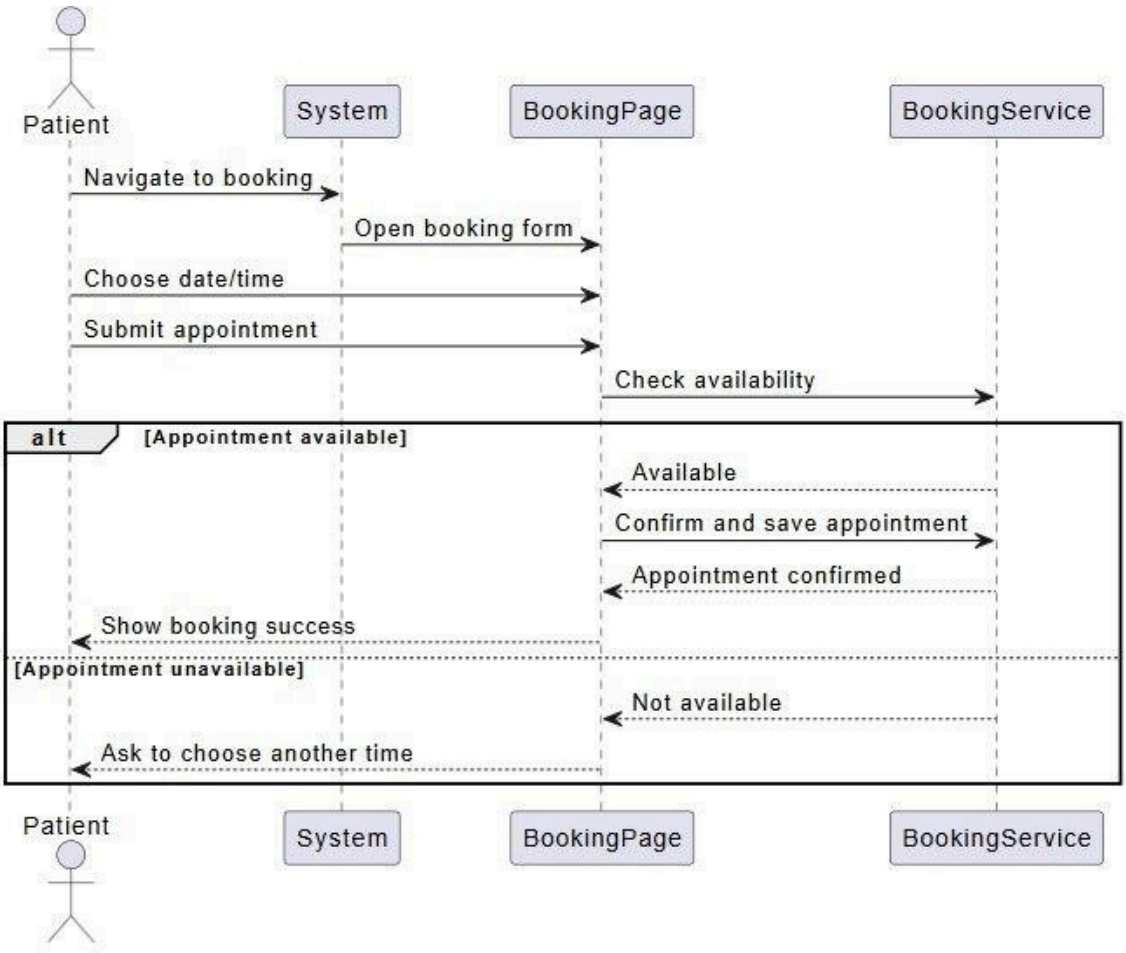


Figure 2.7: Sequence Diagram of Booking Confirmation

2.8- Class Diagrams :

2.8.1- Mobile Application Class Diagram :

Picture how the app fits together through its core pieces. Inside lives Patient, Appointment, Dentist, Treatment, AvailabilitySlot, along with AuthResponse. Appointments connect one dentist to one treatment, carrying also a current status. Booking happens when open times match personal needs. Those openings - called availability slots show exactly when dentists accept visits. Authentication gives back an answer shaped like AuthResponse. This piece holds both user details and access key. Each part links cleanly, forming quiet order beneath actions taken. Structure stays clear even as choices grow.

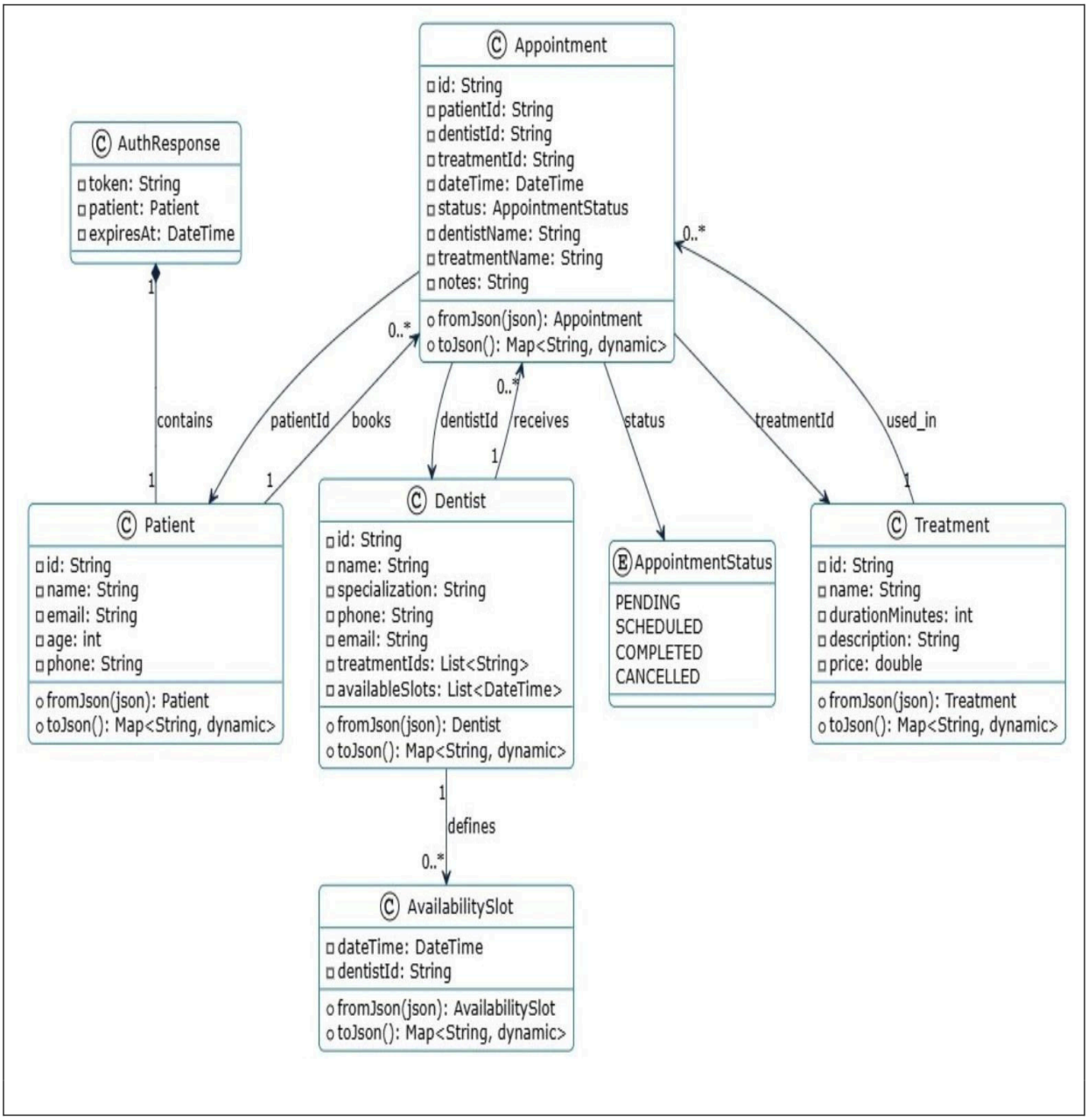


Figure 2.8: Mobile Application Class Diagram

2.8.2- Desktop Application Class Diagram :

The desktop class diagram represents the internal clinic management system. Classes cover users, roles, patients, dentists, appointments, treatments, prescriptions, meds, payments, schedule slots, clinic details. Right at the heart sits the appointment - links together patient info, dentist data, treatment kinds, payment status, health notes.The prescription classes allow the dentist to create prescriptions and add medication items. The payment class records billing information related to completed appointments.

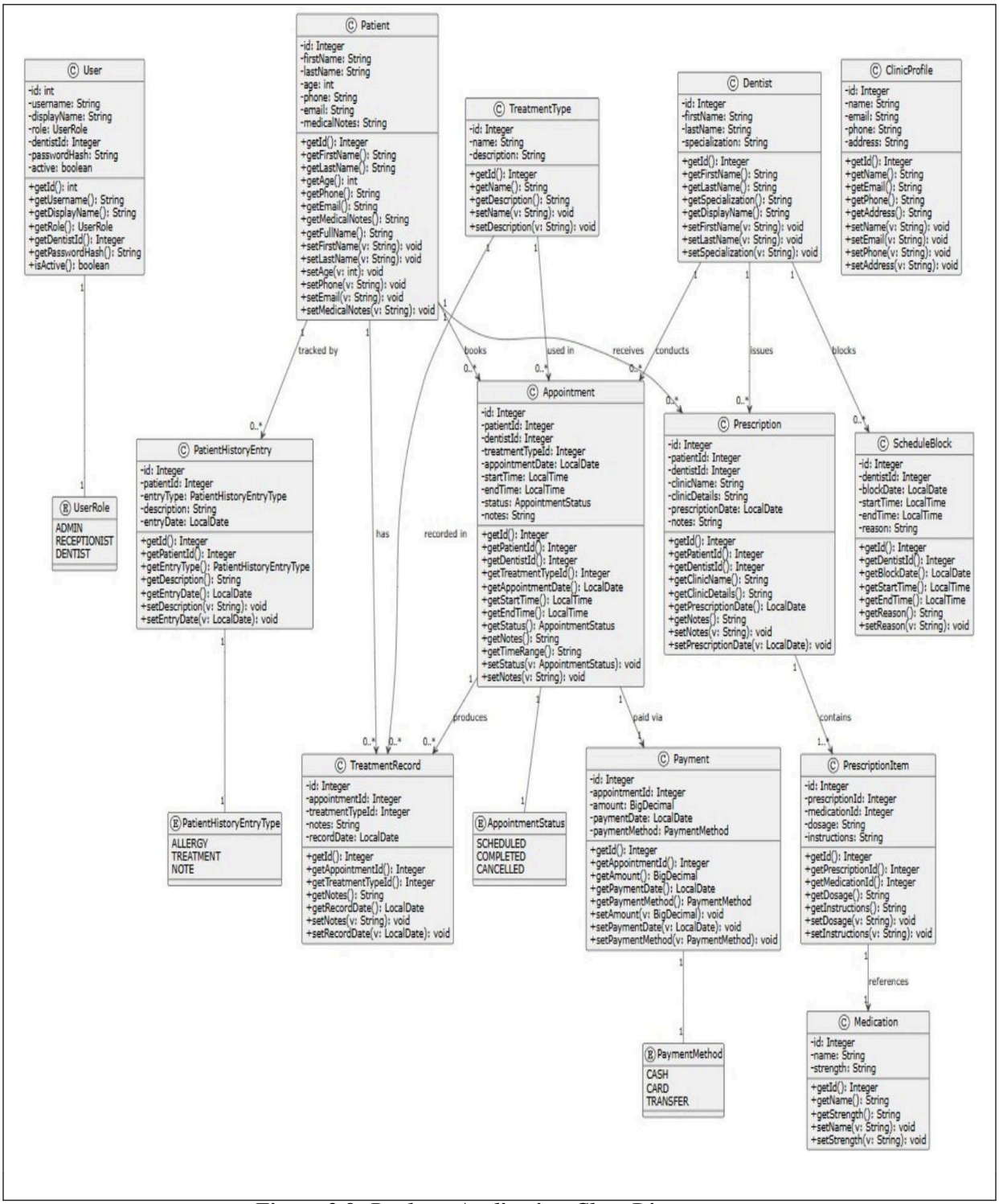


Figure 2.9: Desktop Application Class Diagram

2.9- Conclusion:

Here comes the breakdown of what happened in this chapter. Actors were spotted first, then major actions the system handles got pinned down. Instead of just listing tasks, real-life flows took shape through written scenario charts. To show timing and order, sequences walked through login steps, booking visits, plus handling payments. What holds everything together appeared next - classes mapping out mobile and desktop sides. Structure settled, groundwork now sits ready for actual coding to begin.

Chapter 3 : Implementation

3.1- Introduction:

This Chapter walks through how the Basmatech Dental clinic Software looks and works. Different views appear depending on whether you're using a computer or phone. One screen handles logging in securely. Booking visits comes next, followed by tools to update personal details. Managing who sees what happens alongside setting dates for care. Bills get sorted right after, without delays. Writing prescriptions fits into the flow naturally. Each piece connects, yet stands clear on its own.

Shown here are screen images from different user experiences inside the app. Each role - admin, dentist, front desk worker, or patient - navigates in distinct ways. What appears on display shifts based on who is using it. Functionality follows responsibility, nothing more. Views adapt silently as logins change. Access defines what shows up, every time.

3.2-System interfaces representation :

3.2.1- Mobile Application (Client) Interfaces:

a) Authentification Interfaces :

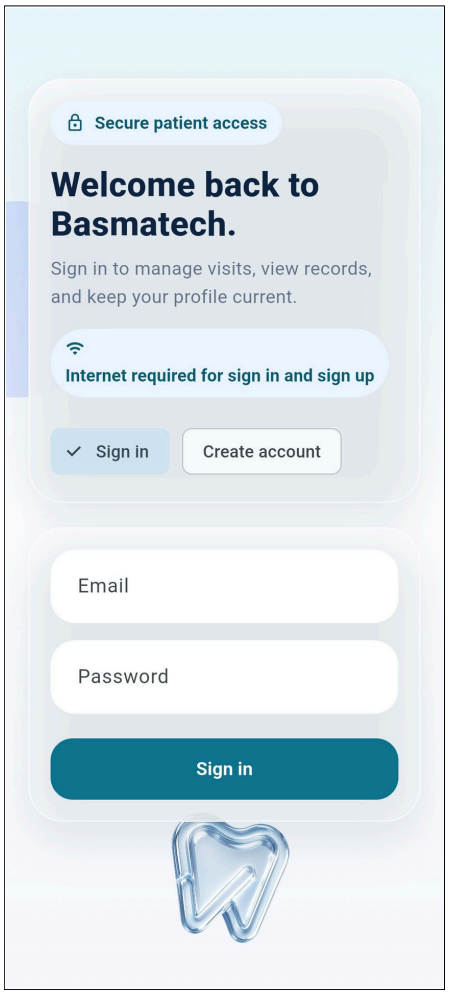


Figure 3.2 : Mobile Sign-in Interface

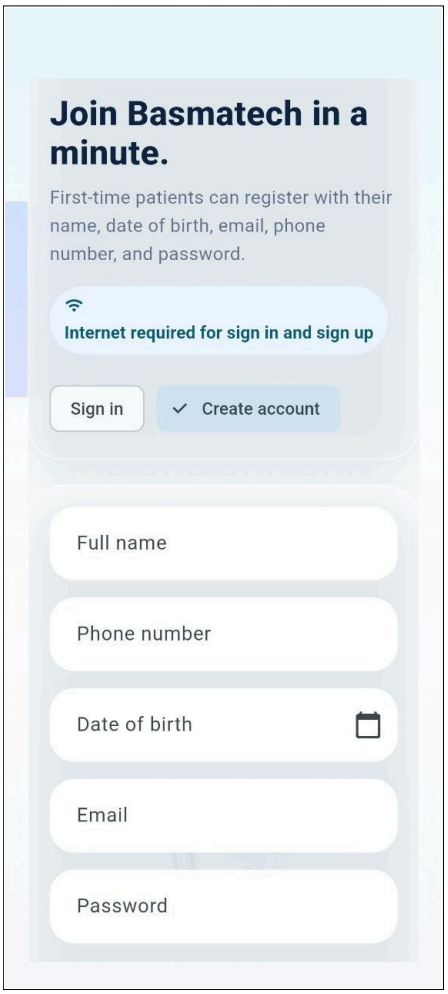


Figure 3.3 : Mobile Sign-up Interface

b) Home Interfaces :

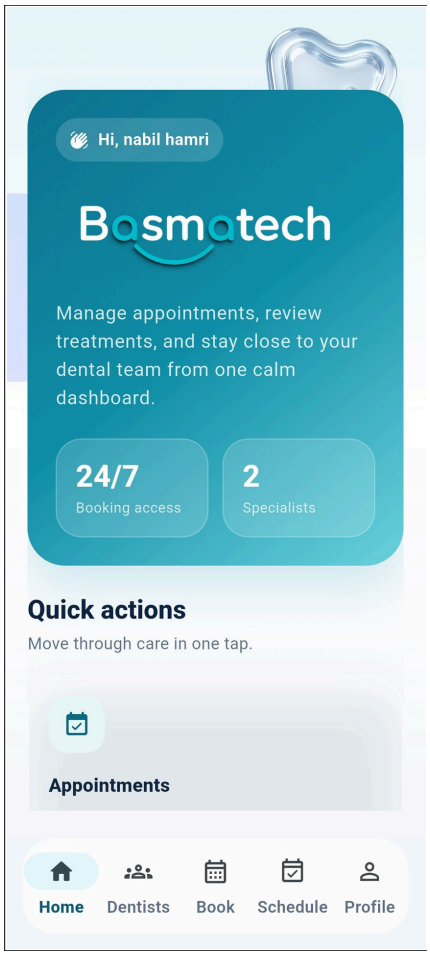


Figure 3.4 : Home Interface



Figure 3.5: Home Interface (Arabic)

c) Profile and Dentists Interfaces:

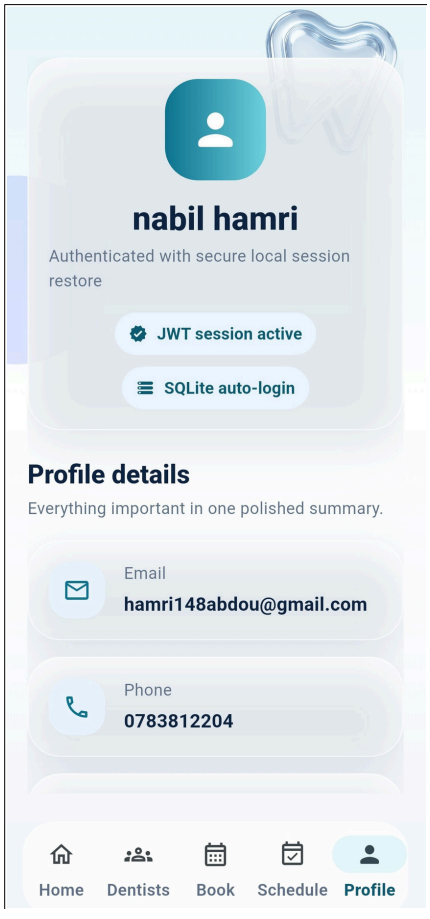


Figure 3.6: Client Profile Interface

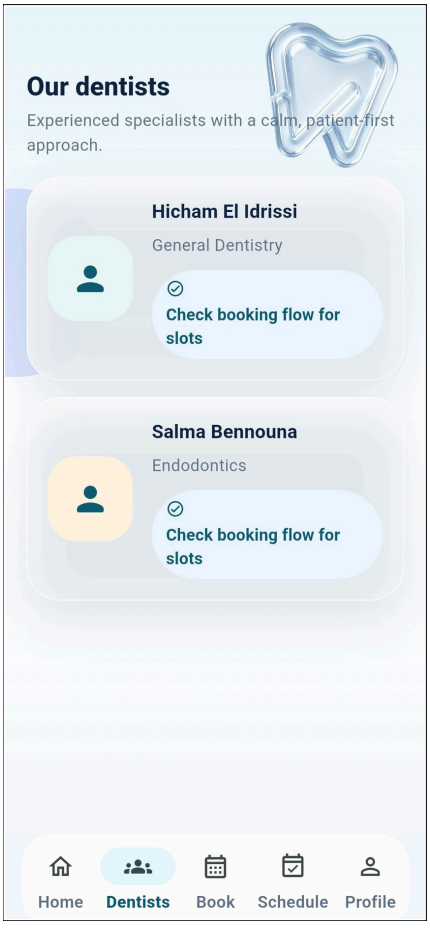


Figure 3.7: Dentists List Interface

d) Book Appointement Interfaces:

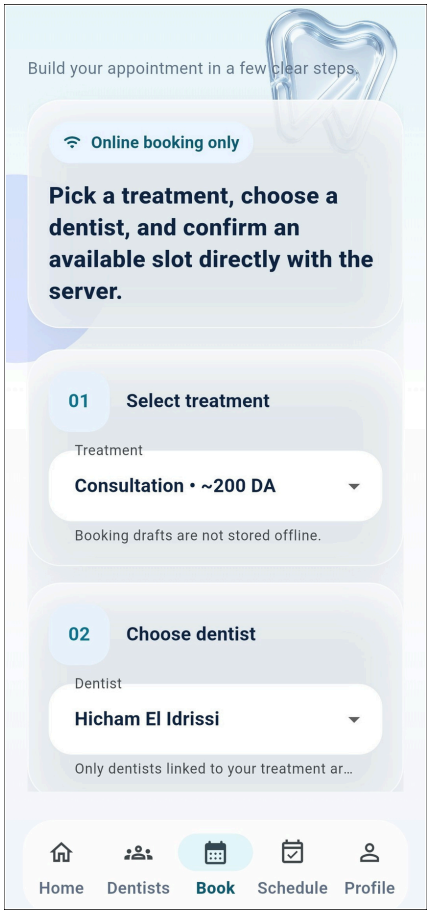


Figure 3.8 : Booking selection Interface

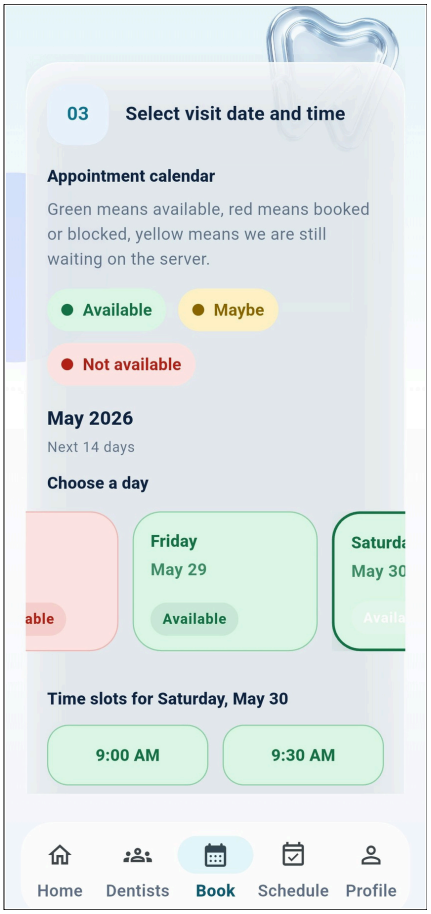


Figure 3.9 : Days selection Interface

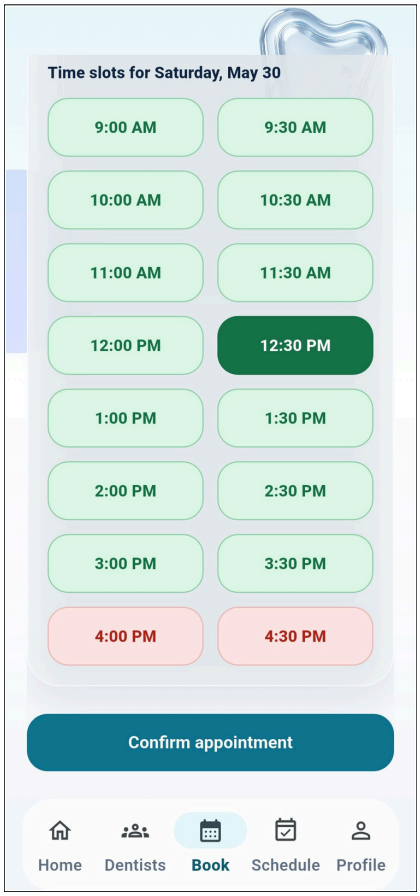


Figure 3.10: Time Slots selection Interface

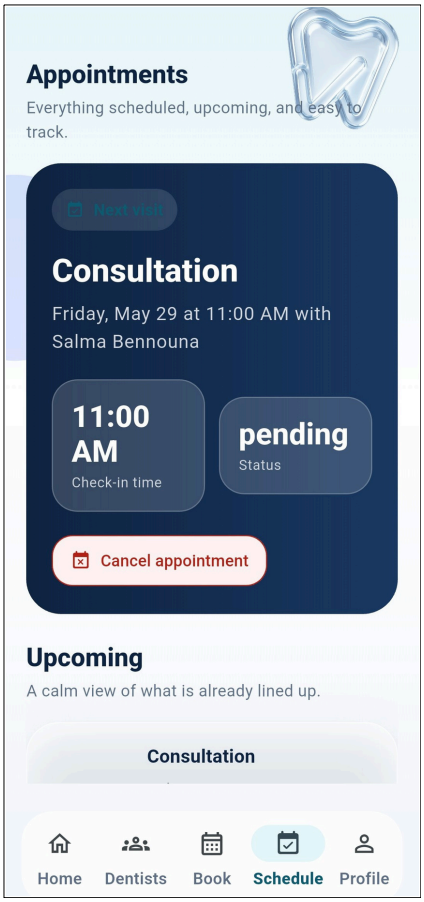


Figure 3.11: Appointement Schedule Interface

e) Other Interfaces :



Figure 3.12: Language selection Interface

3.2.2 - Desktop (Clinic Staff) Interfaces :

a) Authentification Interfaces:

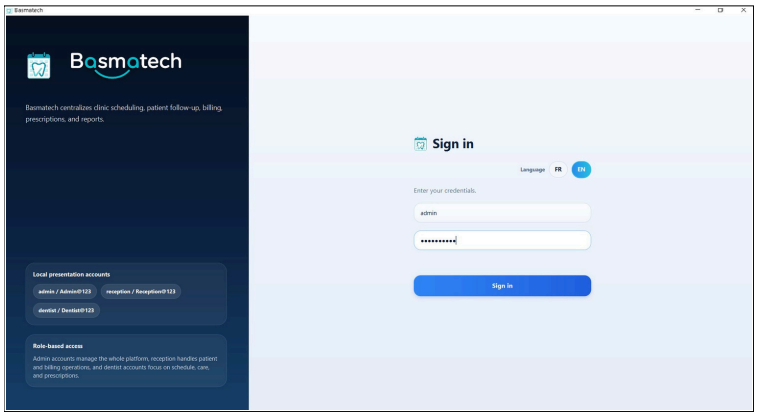


Figure 3.1: Desktop Login Interface

b) Dashboard and Scheduling Interfaces:

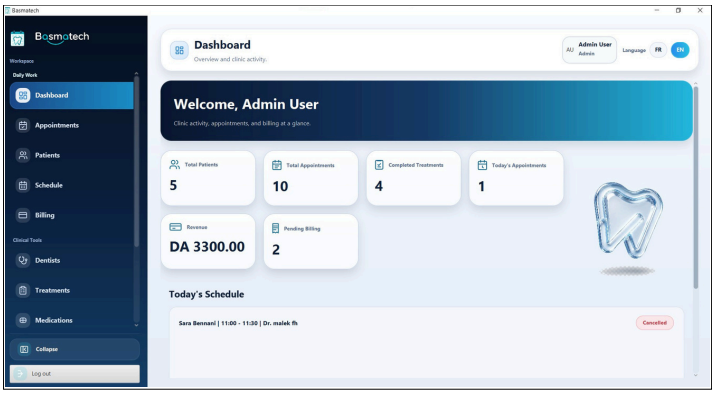


Figure 3.13: Home Dashboard Interface

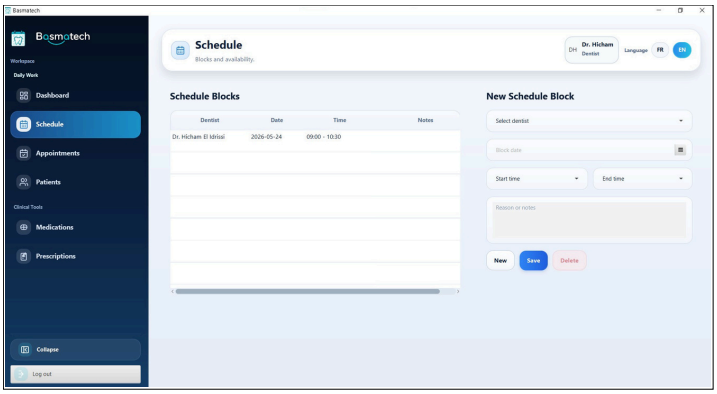


Figure 3.14: Schedule Blocks Interface

c) Patient Management Interfaces:

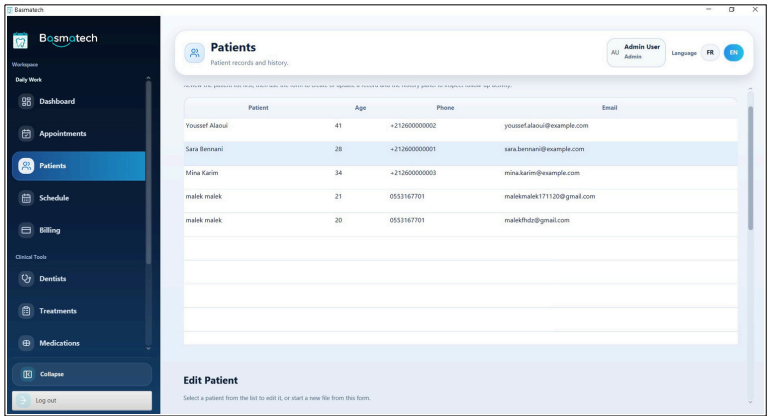


Figure 3.15 : Patient List Interface

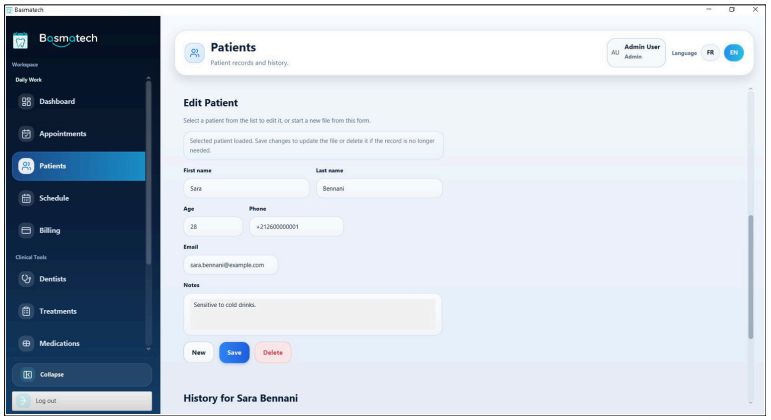


Figure 3.16 : Patient Edit Interface

d) Billing and Appoinements Interfaces :

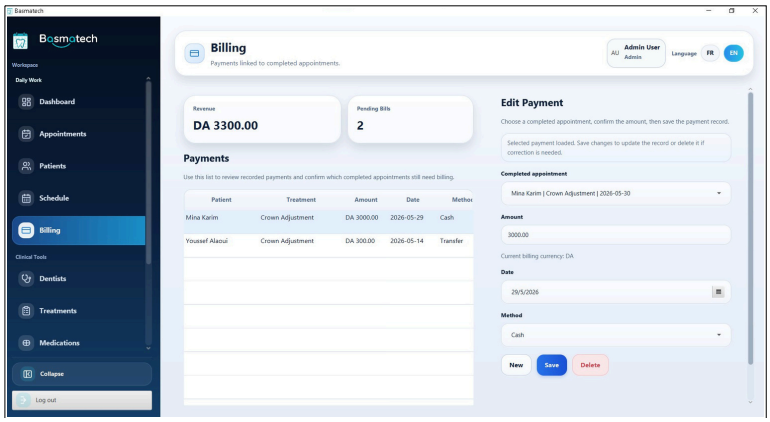


Figure 3.17: Billing Interface

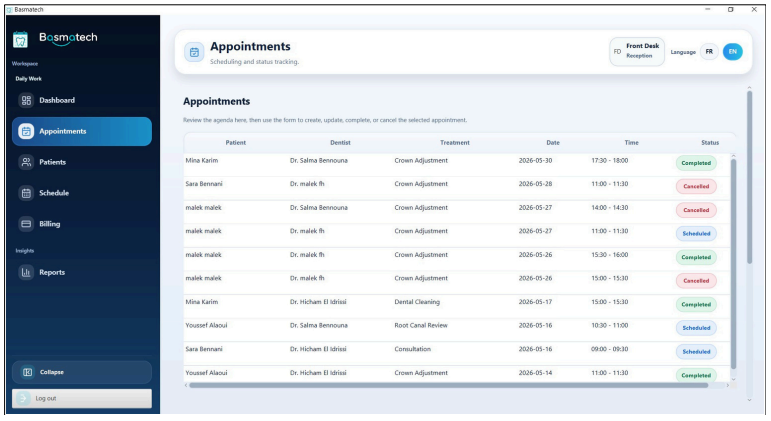


Figure 3.18: Appoinments Interface

e) Prescription Interfaces:

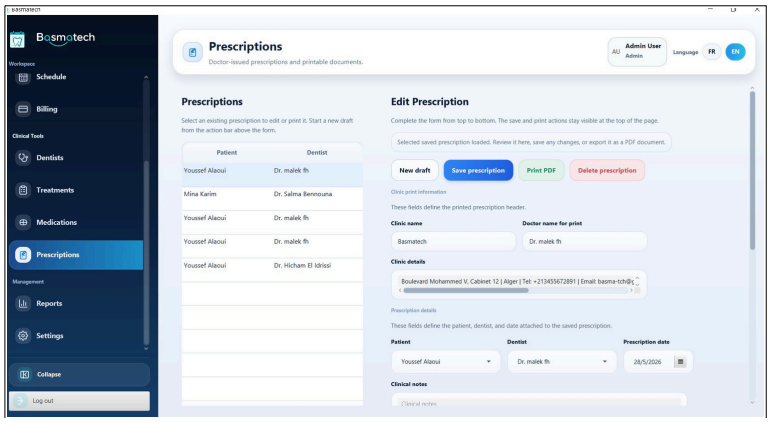


Figure 3.19: Prescriptions Interface

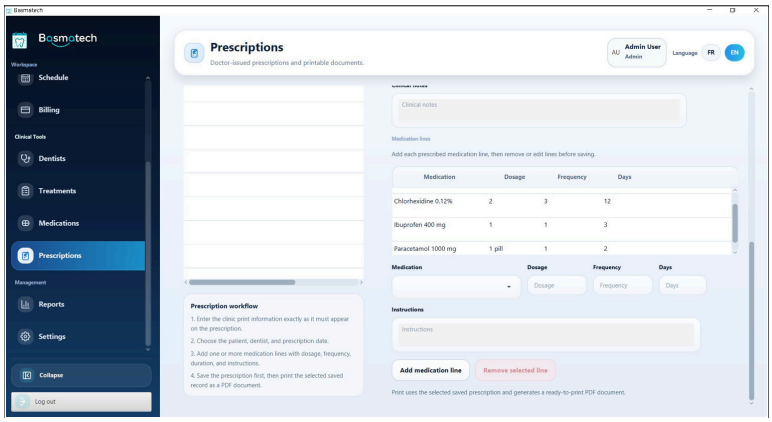


Figure 3.20 : Prescriptios Interface (Medications section)

f) Settings Interface (Configuration) and Prescription Template:

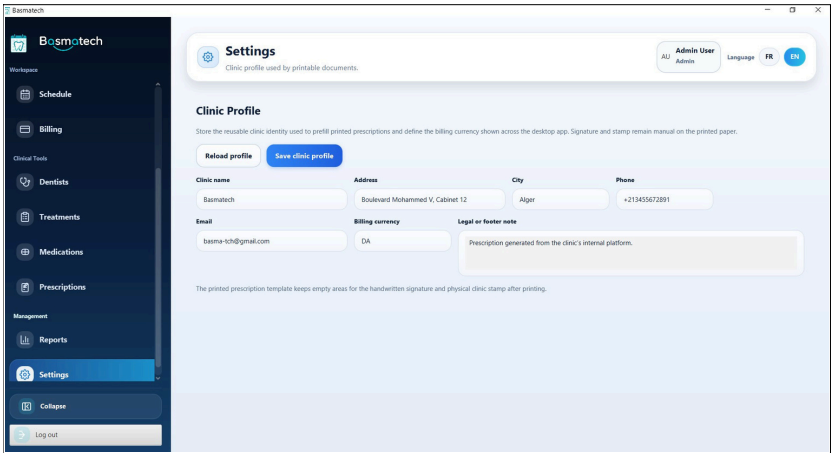


Figure 3.21 : Settings Interface (Configuration)

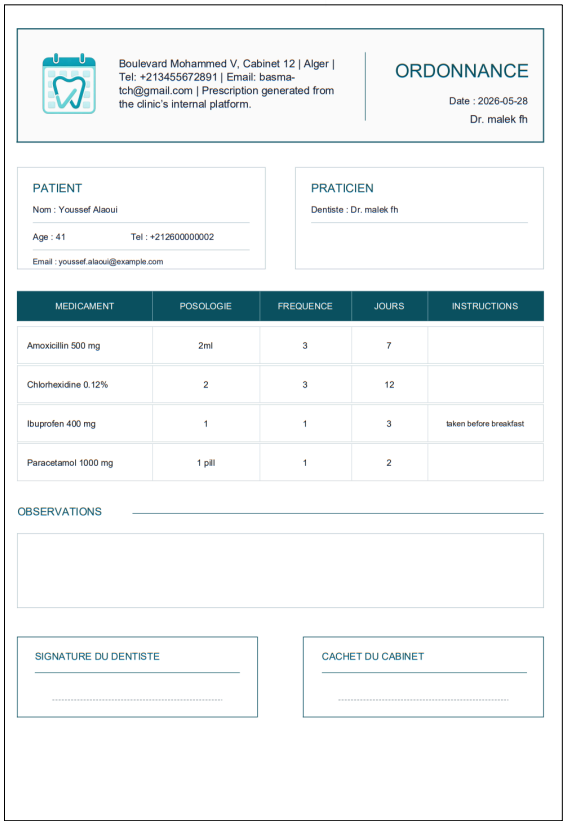


Figure 3.22 : Prescription Template

3.3- Conclusion :

In this chapter , comes a look at how the Basmatech Dental Clinic Software appears on screen. Shown through images, its core features come alive - login screens appear alongside tools for booking visits. One moment you see patient records and give prescriptions, next up sits payment handling. Function by function, desktop meets phone design in clear views. Moving between tasks like setting appointments or updating personal details feels laid out step by step. Each view gives space to actions that matter day to day.

From here, people using the system see clear paths to work with the app without confusion. Each role finds its own space, shaped by tasks that fit naturally. Through these views, access becomes straightforward, guided by layout rather than instruction.

General conclusion :

The appearance of information technologies has a huge influence on the manner in which organizations run their day-to-day operations and healthcare institutions are not an exception. In this context, the objective of this project was to design and implement a **Dental Clinic Management System** that can improve the efficiency of clinic operations while providing a better experience for patients and medical staff.

The proposed solution is a mix of a **desktop application** for clinic staff and a **mobile application** for patients. The system allows appointment scheduling, patient record management, treatment monitoring, billing, generation of prescriptions and administrative reporting in a single environment. With the platform, the centralization of information and automation of routine tasks help to reduce the manual workload, organize the data better and ensure faster access to important information. Several software engineering concepts were used during the realization of this project such as requirements analysis, system modeling using UML diagrams, database design and application development using client-server architecture. These stages enabled the creation of a consistent, secure and maintainable system that meets the needs identified in dental clinics.
The results obtained show that digital solutions can greatly contribute to the improvement of healthcare service management. The system provides an easy way of scheduling appointments, maintaining accurate patient records and improving communication between patients and clinic staff, therefore improving productivity and quality of service.
In terms of future improvements, additional functionalities such as automatic appointment reminders, online payment services, cloud synchronization, telemedicine features, and advanced statistical analysis could be integrated to further enhance the capabilities of the platform.

In conclusion, the project met its objectives and is an important step towards the digitalization of dental clinic management, providing a modern, efficient and user-centric solution that fits the demands of the current healthcare environments.

Webography

[1] Object Management Group. (2017). OMG Unified Modeling Language Specification, Version 2.5.1.
https://www.omg.org/spec/UML
Official UML specification. Used as a modeling reference for the project's use case, class, and sequence diagrams.

[2] Fielding, R. T. (2000). Architectural Styles and the Design of Network-Based Software Architectures.
https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
Foundational REST architecture reference. Used for the mobile API communication model.

[3] OpenJFX. OpenJFX Documentation.
https://openjfx.io/openjfx-docs/
Official JavaFX documentation. Used for JavaFX application structure, FXML-based interfaces, and Maven execution.

[4] Flutter. Flutter Architectural Overview.
https://docs.flutter.dev/resources/architectural-overview
Official Flutter architecture documentation. Used for widget-based mobile UI structure and cross-platform development.

[5] Oracle. MySQL 8.4 Reference Manual.
https://dev.mysql.com/doc/refman/8.4/en/
Official MySQL documentation. Used for relational database structure, SQL storage, and data management.

[6] SQLite. Appropriate Uses for SQLite.
https://www.sqlite.org/whentouse.html
Official SQLite documentation. Used for local mobile storage and cache justification.

[7] Jaspersoft Community. JasperReports Library Tutorial.
https://community.jaspersoft.com/knowledgebase/faq/jasperreports-library-tutorial/
JasperReports documentation. Used for JRXML report templates and PDF prescription generation.

[8] OWASP Foundation. Application Security Verification Standard.
https://owasp.org/www-project-application-security-verification-standard/
Security verification standard. Used for authentication and role-based access control considerations.

[9] OWASP Cheat Sheet Series. Password Storage Cheat Sheet.
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
Password security reference. Used for password hashing and avoiding plain-text password storage.

[10] PlantUML. PlantUML Language Reference Guide.
https://plantuml.com/guide
Official PlantUML documentation. Used for generating UML diagrams, including use case, class, and sequence diagrams from textual descriptions.